Chapter 4
Adaptive RBF Neural Network Control

Abstract This chapter introduces several online adaptive RBF neural network control methods, including adaptive control based on neural approximation, adaptive control based on neural approximation with unknown parameter, and a direct robust adaptive control. For above control laws, the adaptive law is designed based on the Lyapunov stability theory, the closed-loop system stability can be achieved.

Keywords RBF neural network• Adaptive control• Neural approximation• Lyapunov stability

Note that using the gradient descent method to design the neural network weights adjustment law, neural network parameters are selected by experience, only local optimization can be guaranteed, closed-loop system stability cannot be guaranteed, and closed-loop system control is easy to diverge. To solve this problem, there has been online adaptive neural network control method, the adaptive law is designed based on the Lyapunov stability theory, and the closed-loop system stability can be achieved.

4.1 Adaptive Control Based on Neural Approximation

4.1.1 Problem Description

Consider a second-order nonlinear system

$$\ddot{x} = f(x, \dot{x}) + g(x, \dot{x})u \tag{4.1}$$

where f is unknown nonlinear function, g is known nonlinear function, and $u \in R^n$ and $y \in R^n$ are input and output.

J. Liu, *Radial Basis Function (RBF) Neural Network Control for Mechanical Systems: Design, Analysis and Matlab Simulation*, DOI 10.1007/978-3-642-34816-7_4,

The Eq. (4.1) can also be written as

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x_1, x_2) + g(x_1, x_2)u \\ y &= x_1. \end{aligned} \tag{4.2}$$

We assume the ideal position signal is y_d. Let

$$e = y_\mathrm{d} - y = y_\mathrm{d} - x_1, \quad \boldsymbol{E} = (\, e \quad \dot{e}\,)^\mathrm{T}.$$

We design the control law as

$$u^* = \frac{1}{g(\boldsymbol{x})}\left[-f(\boldsymbol{x}) + \ddot{y}_\mathrm{d} + \boldsymbol{K}^\mathrm{T}\boldsymbol{E}\right] \tag{4.3}$$

substituting (4.3) into (4.1), we can get the closed control system as

$$\ddot{e} + k_\mathrm{p}e + k_\mathrm{d}\dot{e} = 0. \tag{4.4}$$

We design $\boldsymbol{K} = (k_\mathrm{p}, k_\mathrm{d})^\mathrm{T}$ so that all the roots of the polynomial $s^2 + k_\mathrm{d}s + k_\mathrm{p} = 0$ are in the left part of the complex plane. Then we have $t \to \infty$, $e(t) \to 0$ and $\dot{e}(t) \to 0$.

From (4.3), we know if the function $f(\boldsymbol{x})$ is unknown, the control law will not be realized.

4.1.2 Adaptive RBF Controller Design

4.1.2.1 RBF Neural Network Design

In this section, we use RBF to design $\hat{f}(\boldsymbol{x})$ to approximate $f(\boldsymbol{x})$. The algorithm of RBF is described as

$$h_j = g\left(\left\|\boldsymbol{x} - \boldsymbol{c}_{ij}\right\|^2 \Big/ b_j^2\right)$$

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

where $\boldsymbol{x}$ is the input vector, i denotes input neural nets number in the input layer, j denotes hidden neural nets number in the hidden layer, $\boldsymbol{h} = [h_1, h_2, \ldots, h_n]^\mathrm{T}$ denotes the output of hidden layer, $\boldsymbol{W}$ is weight value, ε is approximation error, and $\varepsilon \le \varepsilon_\mathrm{N}$.

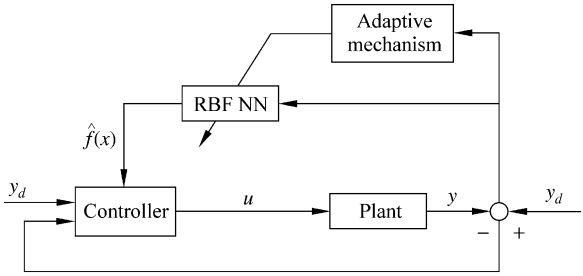


Fig. 4.1 Block diagram of the control scheme

We use RBF to approximate f, the input vector is chosen as $\boldsymbol{x} = [e \quad \dot{e}]^{\mathrm{T}}$, the output of RBF is

$$\hat{f}(x) = \hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}). \tag{4.5}$$

4.1.2.2 Control Law and Adaptive Law Design

The fuzzy system approximation algorithm was applied to design indirect adaptive fuzzy controller [1]. Now we used RBF to replace fuzzy system to design RBF adaptive controller.

If we use RBF neural network to represent the unknown nonlinear function f, the control law becomes

$$u = \frac{1}{g(\boldsymbol{x})}\left[-\hat{f}(\boldsymbol{x}) + \ddot{y}_{\mathrm{d}} + \boldsymbol{K}^{\mathrm{T}}\boldsymbol{E}\right] \tag{4.6}$$

$$\hat{f}(\boldsymbol{x}) = \hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) \tag{4.7}$$

where $\boldsymbol{h}(\boldsymbol{x})$ is Gaussian function and $\hat{\boldsymbol{W}}$ is the estimated parameter for $\boldsymbol{W}$.

Figure 4.1 shows the closed-loop neural-based adaptive control scheme.

We choose the adaptive law as

$$\dot{\hat{\boldsymbol{W}}} = -\gamma\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}\boldsymbol{h}(\boldsymbol{x}) \tag{4.8}$$

4.1.2.3 Stability Analysis

Submitting the control law (4.6) into (4.1), the closed-loop system is expressed as

$$\ddot{e} = -\boldsymbol{K}^{\mathrm{T}}\boldsymbol{E} + \left[\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\right]. \tag{4.9}$$

Let

$$\boldsymbol{\Lambda} = \begin{bmatrix} 0 & 1 \\ -k_{\mathrm{p}} & -k_{\mathrm{d}} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{4.10}$$

Now, (4.9) can be rewritten as

$$\dot{\boldsymbol{E}} = \boldsymbol{\Lambda E} + B\left[\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\right]. \tag{4.11}$$

The optimal weight value is

$$W^* = \arg\min_{\mathbf{W}\in\Omega} \left[\sup\left|\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\right|\right]. \tag{4.12}$$

Define the modeling error as

$$\omega = \hat{f}(\boldsymbol{x}|\boldsymbol{W}^*) - f(\boldsymbol{x}). \tag{4.13}$$

Then Eq. (4.11) becomes

$$\dot{\boldsymbol{E}} = \boldsymbol{\Lambda E} + B\left\{\left[\hat{f}(x|) - \hat{f}(\boldsymbol{x}|\boldsymbol{W}^*)\right] + \omega\right\}. \tag{4.14}$$

Submitting (4.7) into (4.14), we can get closed equation as

$$\dot{\boldsymbol{E}} = \boldsymbol{\Lambda E} + B\left[(\hat{\boldsymbol{W}} - \boldsymbol{W}^*)^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) + \omega\right] \tag{4.15}$$

Choose a Lyapunov function as

$$V = \frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{E} + \frac{1}{2\gamma}(\hat{\boldsymbol{W}} - \boldsymbol{W}^*)^{\mathrm{T}}(\hat{\boldsymbol{W}} - \boldsymbol{W}^*) \tag{4.16}$$

where γ is positive constant. $\hat{\boldsymbol{W}} - \boldsymbol{W}^*$ denotes the parameter estimation error, and the matrix $\boldsymbol{P}$ is symmetric and positive definite and satisfies the following Lyapunov equation:

$$\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{P} + \boldsymbol{P}\Lambda = -\boldsymbol{Q}. \tag{4.17}$$

With $\boldsymbol{Q} \geq 0$, $\boldsymbol{\Lambda}$ is given by (4.10).

Choose $V_1 = \frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{E}, V_2 = \frac{1}{2\gamma}(\hat{\boldsymbol{W}} - \boldsymbol{W}^*)^{\mathrm{T}}(\hat{\boldsymbol{W}} - \boldsymbol{W}^*)$, and let $\boldsymbol{M} = B\left[(\hat{\boldsymbol{W}} - \boldsymbol{W}^*)^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) + \omega\right]$, then (4.15) becomes

$$\dot{\boldsymbol{E}} = \boldsymbol{\Lambda E} + \boldsymbol{M}.$$

Then

$$\begin{aligned}\dot{V}_1 &= \frac{1}{2}\dot{\boldsymbol{E}}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{E} + \frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\dot{\boldsymbol{E}} = \frac{1}{2}(\boldsymbol{E}^{\mathrm{T}}\boldsymbol{\Lambda}^{\mathrm{T}} + \boldsymbol{M}^{\mathrm{T}})\boldsymbol{P}\boldsymbol{E} + \frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}(\boldsymbol{\Lambda}\boldsymbol{E} + M) \\ &= \frac{1}{2}\boldsymbol{E}^{\mathrm{T}}(\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{P} + \boldsymbol{P}\Lambda)\boldsymbol{E} + \frac{1}{2}\boldsymbol{M}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{E} + \frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{M} \\ &= -\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E} + \frac{1}{2}(\boldsymbol{M}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{E} + \boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{M}) = -\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E} + \boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{M}.\end{aligned}$$

Submitting $\boldsymbol{M}$ into above, noting that $\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}(\hat{\boldsymbol{W}} - \boldsymbol{W}^*)^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) = (\hat{\boldsymbol{W}} - \boldsymbol{W}^*)^{\mathrm{T}}\ [\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\ \boldsymbol{h}(\boldsymbol{x})]$, we get

$$\begin{aligned}\dot{V}_1 &= -\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E} + \boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\left(\hat{\mathbf{W}} - \boldsymbol{W}^*\right)^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\omega \\ &= -\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E} + \left(\hat{\boldsymbol{W}} - \boldsymbol{W}^*\right)^{\mathrm{T}}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\omega\end{aligned}$$

$$\dot{V}_2 = \frac{1}{\gamma}(\hat{\boldsymbol{W}} - \boldsymbol{W}^*)^{\mathrm{T}}\dot{\hat{\boldsymbol{W}}}.$$

Then the derivative V becomes

$$\dot{V} = \dot{V}_1 + \dot{V}_2 = -\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E} + \boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\omega + \frac{1}{\gamma}\left(\hat{\boldsymbol{W}} - \boldsymbol{W}^*\right)^{\mathrm{T}}\left[\dot{\hat{\boldsymbol{W}}} + \gamma\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\boldsymbol{h}(\boldsymbol{x})\right].$$

Submitting the adaptive law (4.8) into above, we have

$$\dot{V} = -\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E} + \boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\omega.$$

Since $-\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E} \leq 0$, consider the adaptive fuzzy control system convergence analysis in the book [1]; if we can make the approximation error ω very small by using RBF, we can get $\dot{V} \leq 0$.

4.1.3 Simulation Examples

4.1.3.1 First Simulation Example

Consider a linear plant as follows:

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

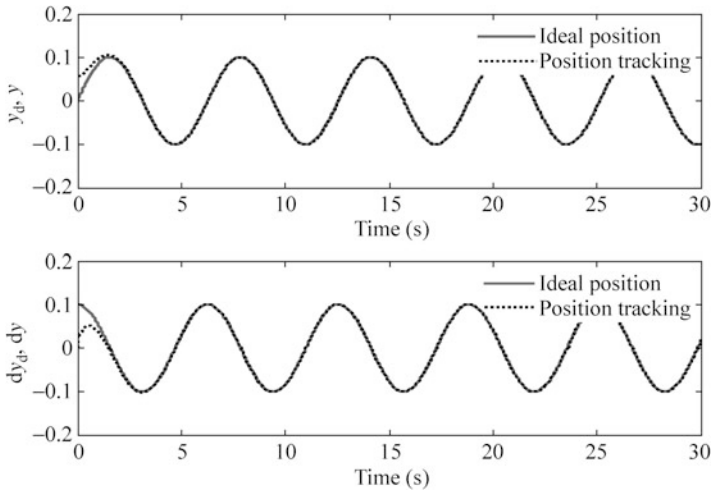


Fig. 4.2 Position and speed tracking

where x_1 and x_2 are position and speed, respectively; u is control input; $f(\boldsymbol{x}) = -25x_2$, and $g(\boldsymbol{x}) = 133$.

We use ideal position signal as $y_d(t) = 0.1 \sin t$, and choose the initial states of the plant as $[\pi/60, 0]$. RBF network structure is chosen as 2-5-1. The choice of Gaussian function parameters value c_{ij} and b_j must be chosen according to the scope of practical input value, which have important role in the neural network control. If the parameter values are chosen inappropriately, Gaussian function will not be effectively mapped, and RBF network will be invalid.

According to the practical scope of x_1 and x_2, the parameters of $\boldsymbol{c}_i$ and b_i are designed as $[-2 \quad -1 \quad 0 \quad 1 \quad 2]$ and 0.20; the initial weight value is chosen as zero.

Adopting control law (4.6) and adaptive law (4.8), choosing $Q = \begin{bmatrix} 500 & 0 \\ 0 & 500 \end{bmatrix}$, $k_d = 50$, $k_p = 30$, and $\gamma = 1,200$.

The results are shown in Figs. 4.2 and 4.3. The Simulink program of this example is chap4_1sim.mdl; the Matlab programs of the example are given in the Appendix.

4.1.3.2 Second Simulation Example

Consider a single inverted pendulum system as in Fig. 4.4.

The dynamic equation is described as

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

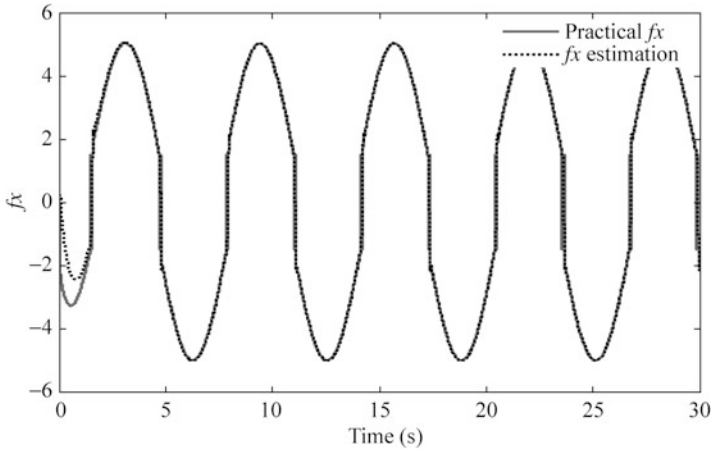


Fig. 4.3 $f(\boldsymbol{x})$ and $\hat{f}(\boldsymbol{x})$

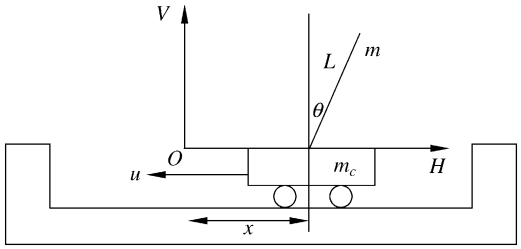


Fig. 4.4 Single inverted pendulum system

where $f(\boldsymbol{x}) = \frac{g \sin x_1 - mlx_2^2 \cos x_1 \sin x_1/(m_c+m)}{l(4/3 - m\cos^2 x_1/(m_c+m))}$; $g(\boldsymbol{x}) = \frac{\cos x_1/(m_c+m)}{l(4/3 - m\cos^2 x_1/(m_c+m))}$, x_1 and x_2 are angle and angle speed value, respectively; $g = 9.8\,\mathrm{m/s^2}$, $m_c = 1$ kg is mass of cart; $m = 0.1$ kg is mass of the pendulum; $l = 0.5$ m is the half length of the pendulum; and u is control input.

Consider ideal position signal as $y_\mathrm{d}(t) = 0.1 \sin t$, the initial states are chosen as $[\pi/60, 0]$. RBF network structure is chosen as 2-5-1.

According to the practical scope of x_1 and x_2, the parameters of $\boldsymbol{c}_\mathrm{i}$ and b_i are designed as $[-2 \quad -1 \quad 0 \quad 1 \quad 2]$ and 0.20; the initial weight value is chosen as zero.

Use control law (4.6) and adaptive law (4.8), and choose $Q = \begin{bmatrix} 500 & 0 \\ 0 & 500 \end{bmatrix}$, $k_\mathrm{d} = 50$, $k_\mathrm{p} = 30$, and $\gamma = 1,200$.

The results are shown in Figs. 4.5 and 4.6. The Simulink program of this example is chap4_2sim.mdl; the Matlab programs of the example are given in the Appendix.

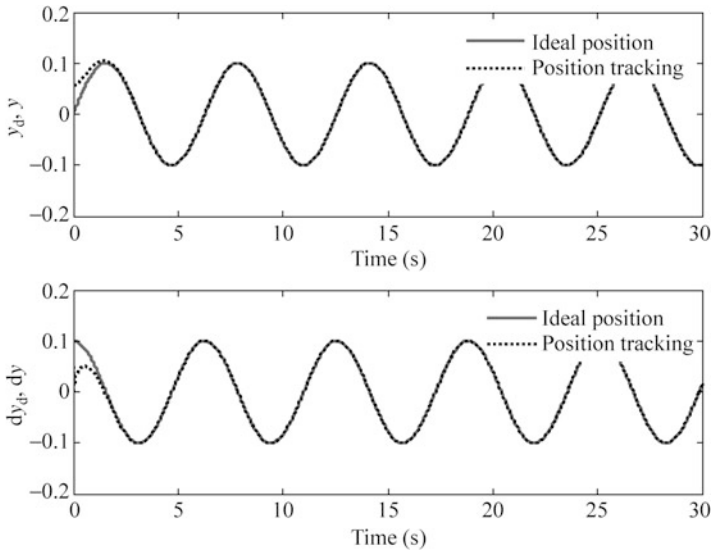


Fig. 4.5 Position and speed tracking

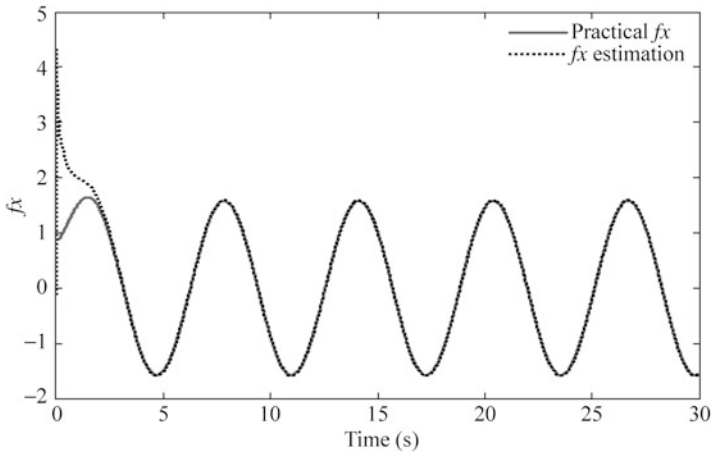


Fig. 4.6 $f(\boldsymbol{x})$ and $\hat{f}(\boldsymbol{x})$

4.2 Adaptive Control Based on Neural Approximation with Unknown Parameter

4.2.1 Problem Description

Consider a second-order nonlinear system

$$\ddot{x} = f(x, \dot{x}) + mu \tag{4.18}$$

where f is unknown nonlinear function, m are unknown, the lower bound of m is known, $m \geq \bar{m}$, and $\bar{m}>0$.

The Eq. (4.18) can also be written as

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(\boldsymbol{x}) + mu \\ y &= x_1. \end{aligned} \tag{4.19}$$

We assume the ideal position signal is y_d, and let

$$e = y_\mathrm{d} - y = y_\mathrm{d} - x_1, \quad \boldsymbol{E} = [e \quad \dot{e}]^\mathrm{T}.$$

We design the control law as

$$u^* = \frac{1}{m}\left[-f(\boldsymbol{x}) + \ddot{y}_\mathrm{d} + \boldsymbol{K}^\mathrm{T}\boldsymbol{E}\right] \tag{4.20}$$

Substitute (4.20) into (4.18), we can get the closed control system as

$$\ddot{e} + k_\mathrm{p} e + k_\mathrm{d}\dot{e} = 0 \tag{4.21}$$

We design $\boldsymbol{K} = [k_\mathrm{p} \quad k_\mathrm{d}]^\mathrm{T}$ so that all the roots of the polynomial $s^2 + k_\mathrm{d}s + k_\mathrm{p} = 0$ are in the left part of the complex plane. Then we have $t \to \infty$, $e(t) \to 0$, and $\dot{e}(t) \to 0$.

From (4.20), we know that if the function $f(\boldsymbol{x})$ and parameter m are unknown,the control law will not be realized.

4.2.2 Adaptive Controller Design

4.2.2.1 RBF Neural Network Design

In this section, just like Sect. 4.1, with reference to the indirect adaptive fuzzy controller tactics given in [1], now we use RBF to replace fuzzy system to design RBF indirect adaptive controller.

The algorithm of RBF to approximate $f(\boldsymbol{x})$ is described as

$$h_j = g\left(\left\|\boldsymbol{x} - \boldsymbol{c}_{ij}\right\|^2 \Big/ b_j^2\right)$$
$$f = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{h}(\mathbf{x}) + \varepsilon$$

where $\boldsymbol{x}$ is the input vector, i denotes input neural nets number in the input layer, j denotes hidden neural nets number in the hidden layer, $\boldsymbol{h} = [h_1, h_2, \ldots, h_n]^{\mathrm{T}}$ denotes the output of hidden layer, $\boldsymbol{W}$ is weight value, ε is approximation error, and $\varepsilon \leq \varepsilon_{\mathrm{N}}$.

We use RBF to approximate f, the input vector is chosen as $\boldsymbol{x} = \begin{bmatrix} e & \dot{e} \end{bmatrix}^{\mathrm{T}}$, and the output of RBF is

$$\hat{f}(\boldsymbol{x}) = \hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}). \tag{4.22}$$

4.2.2.2 Control Law and Adaptive Law Design

If we use RBF neural network to represent the unknown nonlinear function f, the control law becomes

$$u = \frac{1}{\hat{m}}\left[-\hat{f}(\boldsymbol{x}) + \ddot{y}_{\mathrm{d}} + \boldsymbol{K}^{\mathrm{T}}\boldsymbol{E}\right] \tag{4.23}$$

$$\hat{f}(\boldsymbol{x}) = \hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) \tag{4.24}$$

where $\boldsymbol{h}(\boldsymbol{x})$ is Gaussian function and $\hat{\boldsymbol{W}}$ is the estimated parameter for $\boldsymbol{W}$.

4.2.2.3 Stability Analysis

Submitting the control law (4.23) into (4.18), the closed-loop system is expressed as

$$\ddot{e} = -\boldsymbol{K}^{\mathrm{T}}\boldsymbol{E} + \left(\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\right) + (m - \hat{m})u. \tag{4.25}$$

Let

$$\boldsymbol{\Lambda} = \begin{bmatrix} 0 & 1 \\ -k_{\mathrm{p}} & -k_{\mathrm{d}} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{4.26}$$

Now, (4.25) can be rewritten as

$$\dot{\boldsymbol{E}} = \boldsymbol{\Lambda}\boldsymbol{E} + B\left[\left(\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\right) + (m - \hat{m})u\right]. \tag{4.27}$$

The optimal weight value is

$$W^* = \arg\min_{W\in\Omega}\left[\sup\left|\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\right|\right]. \tag{4.28}$$

Define the modeling error as

$$\omega = \hat{f}(\boldsymbol{x}|\boldsymbol{W}^*) - f(\boldsymbol{x}) \tag{4.29}$$

Then Eq. (4.27) becomes

$$\dot{\boldsymbol{E}} = \boldsymbol{\Lambda E} + B\left\{\left[\hat{f}(\boldsymbol{x}|) - \hat{f}(\boldsymbol{x}|\boldsymbol{W}^*)\right] + \omega + (m-\hat{m})u\right\}. \tag{4.30}$$

Submitting Eq. (4.24) into (4.13), we can get closed equation as

$$\dot{\boldsymbol{E}} = \boldsymbol{\Lambda E} + B\left[\left(\hat{\boldsymbol{W}} - \boldsymbol{W}^*\right)^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) + \omega + (m-\hat{m})u\right]. \tag{4.31}$$

Define a Lyapunov function as

$$V = \frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{E} + \frac{1}{2\gamma}\left(\hat{\boldsymbol{W}} - \boldsymbol{W}^*\right)^{\mathrm{T}}\left(\hat{\boldsymbol{W}} - \boldsymbol{W}^*\right) + \frac{1}{2}\eta\tilde{m}^2 \tag{4.32}$$

where γ is positive constant. $\hat{\boldsymbol{W}} - \boldsymbol{W}^*$ denotes the parameter estimation error, and the matrix $\boldsymbol{P}$ is symmetric and positive definite and satisfies the following Lyapunov equation

$$\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{P} + \boldsymbol{P}\Lambda = -\boldsymbol{Q}. \tag{4.33}$$

With $\boldsymbol{Q} \geq 0$, $\boldsymbol{\Lambda}$ is given by (4.26), η>0, and $\tilde{m} = m - \hat{m}$.

Choose $V_1 = \frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{E}$, $V_2 = \frac{1}{2\gamma}\left(\hat{\boldsymbol{W}} - \boldsymbol{W}^*\right)^{\mathrm{T}}\left(\hat{\boldsymbol{W}} - \boldsymbol{W}^*\right)$, and $V_3 = \frac{1}{2}\eta\tilde{m}^2$, let $\boldsymbol{M} = B\left[\left(\hat{\boldsymbol{W}} - \boldsymbol{W}^*\right)^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) + \omega + \tilde{m}u\right]$, and then Eq. (4.31) becomes

$$\dot{\boldsymbol{E}} = \boldsymbol{\Lambda E} + \boldsymbol{M}.$$

Then

$$\begin{aligned}\dot{V}_1 =& \frac{1}{2}\dot{\boldsymbol{E}}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{E} + \frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\dot{\boldsymbol{E}} = \frac{1}{2}\left(\boldsymbol{E}^{\mathrm{T}}\boldsymbol{\Lambda}^{\mathrm{T}} + \boldsymbol{M}^{\mathrm{T}}\right)\boldsymbol{P}\boldsymbol{E} + \frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}(\boldsymbol{\Lambda E} + M)\\ =& \frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\left(\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{P} + \boldsymbol{P}\Lambda\right)\boldsymbol{E} + \frac{1}{2}\boldsymbol{M}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{E} + \frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}M\\ =& -\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E} + \frac{1}{2}\left(\boldsymbol{M}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{E} + \boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}M\right) = -\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E} + \boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}M.\end{aligned}$$

Submitting $\boldsymbol{M}$ into above, noting that $\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}(\hat{\boldsymbol{W}}-\boldsymbol{W}^{*})^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x})=(\hat{\boldsymbol{W}}-\boldsymbol{W}^{*})^{\mathrm{T}}[\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\boldsymbol{h}(\boldsymbol{x})]$, we get

$$\begin{aligned}\dot{V}_1 &= -\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E}+\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\left(\hat{\boldsymbol{W}}-\boldsymbol{W}^{*}\right)^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x})+\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\omega+\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\tilde{m}u\\ &= -\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E}+\left(\hat{\boldsymbol{W}}-\boldsymbol{W}^{*}\right)^{\mathrm{T}}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\boldsymbol{h}(\boldsymbol{x})+\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\omega+\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\tilde{m}u\end{aligned}$$

$$\dot{V}_2=\frac{1}{\gamma}\left(\hat{\boldsymbol{W}}-\boldsymbol{W}^{*}\right)^{\mathrm{T}}\dot{\hat{\boldsymbol{W}}}$$

$$\dot{V}_3=-\eta\tilde{m}\dot{\hat{m}}.$$

Then the derivative V becomes

$$\begin{aligned}\dot{V} =&\dot{V}_1+\dot{V}_2+\dot{V}_3\\ =&-\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E}+\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\omega+\frac{1}{\gamma}\left(\hat{\boldsymbol{W}}-\boldsymbol{W}^{*}\right)^{\mathrm{T}}\left[\dot{\hat{\boldsymbol{W}}}+\gamma\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\boldsymbol{h}(\boldsymbol{x})\right]+\tilde{m}\left(\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u-\eta\dot{\hat{m}}\right)\end{aligned}$$

We choose the adaptive law as

$$\dot{\hat{\boldsymbol{W}}}=-\gamma\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}\boldsymbol{h}(\boldsymbol{x}). \tag{4.34}$$

To guarantee $\tilde{m}(\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u-\eta\dot{\hat{m}})\le 0$, at the same time to avoid singularity in (4.23) and guarantee $\hat{m}\ge\underline{m}$, we used the adaptive law tactics proposed in [2] as

$$\dot{\hat{m}}=\begin{cases}\frac{1}{\eta}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u, & \text{if}\quad \boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u>0\\ \frac{1}{\eta}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u, & \text{if}\quad \boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u\le 0 \text{ and } \hat{m}>\underline{m}\\ \frac{1}{\eta}, & \text{if}\quad \boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u\le 0 \text{ and } \hat{m}\le\underline{m}\end{cases} \tag{4.35}$$

where $\hat{m}(0)\ge\underline{m}$.

Reference to [2], the adaptive law (4.35) can also be analyzed as:

1. If $\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u>0$, we get $\tilde{m}(\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u-\eta\dot{\hat{m}})=0$ and $\dot{\hat{m}}>0$, thus, $\hat{m}>\underline{m}$;
2. If $\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u\le 0$ and $\hat{m}>\underline{m}$, we get $\tilde{m}(\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u-\eta\dot{\hat{m}})=0$;
3. If $\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u\le 0$ and $\hat{m}\le\underline{m}$, we have $\tilde{m}=m-\hat{m}\ge m-\underline{m}>0$, thus, $\tilde{m}(\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u-\eta\dot{\hat{m}})=\tilde{m}\mathbf{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}u-\tilde{m}\le 0$, $\hat{m}$ will increase gradually, and then $\hat{m}>\underline{m}$ will be guaranteed with $\dot{\hat{m}}>0$.

Submitting the adaptive law (4.34) and (4.35) into above, we have

$$\dot{V}=-\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E}+\boldsymbol{E}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\omega.$$

Since $-\frac{1}{2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{E} \leq 0$, consider the adaptive fuzzy control system convergence analysis in [1]; if we can make the approximation error ω very small by using RBF, we can get $\dot{V} \leq 0$.

4.2.3 Simulation Examples

Consider a simple plant as

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= f(\boldsymbol{x}) + mu\end{aligned}$$

where x_1 and x_2 are position and speed, respectively; u is control input; $f(\boldsymbol{x}) = -25x_2 - 10x_1$, and $m = 133$.

We use ideal position signal as $y_{\mathrm{d}}(t) = \sin t$, and choose the initial states of the plant as [0,50,0].

We use RBF to approximate $f(\boldsymbol{x})$, and design adaptive algorithm to estimate parameter m. The structure is used as 2-5-1; the input vector of RBF is $\boldsymbol{z} = [x_1 \quad x_2]^{\mathrm{T}}$. For each Gaussian function, the parameters of $\boldsymbol{c}_i$ and b_i are designed as $[-1 \quad -0.5 \quad 0 \quad 0.5 \quad 1]$ and 2.0. The initial weight value is chosen as zero.

In simulation, we use control law (4.23) and adaptive law (4.34) and (4.35); the parameters are chosen as $Q = \begin{bmatrix} 500 & 0 \\ 0 & 500 \end{bmatrix}$, $k_{\mathrm{p}} = 30$, $k_{\mathrm{d}} = 50$, $\gamma = 1,200$, $\eta = 0.0001$, $\underline{m} = 100$, and $\hat{m}(0) = 120$.

The results are shown from Figs. 4.7, 4.8, and 4.9. The estimation of $f(x)$ and m do not converge to true value of $f(x)$ and m. This is due to the fact that the desired trajectory is not persistently exciting, and this occurs quite often in real-world application, which has been explained in the Sect. 1.5 in Chap. 1.

The Simulink program of this example is chap4_3sim.mdl; the Matlab programs of the example are given in the Appendix.

4.3 A Direct Method for Robust Adaptive Control by RBF

4.3.1 System Description

Considering the proposed controller in paper [3] and book [4], we analyze design and simulation method of a direct method for robust adaptive control by RBF.

Consider the SISO second-order nonlinear system in the following form:

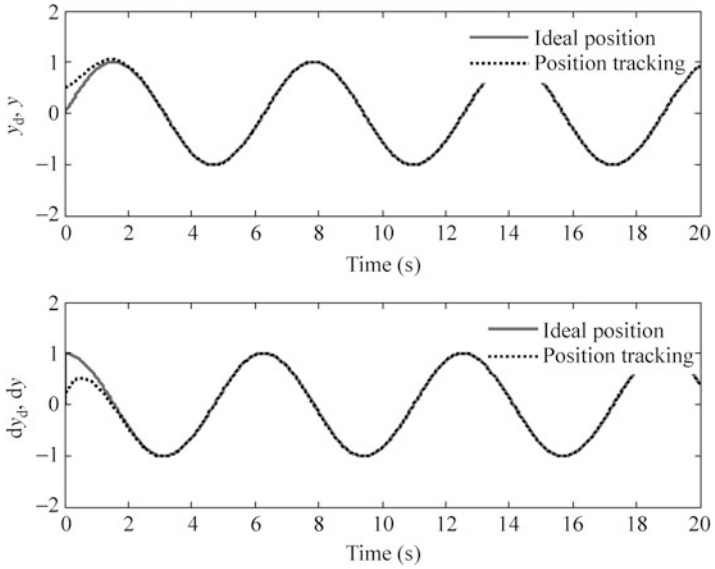


Fig. 4.7 Position and speed tracking

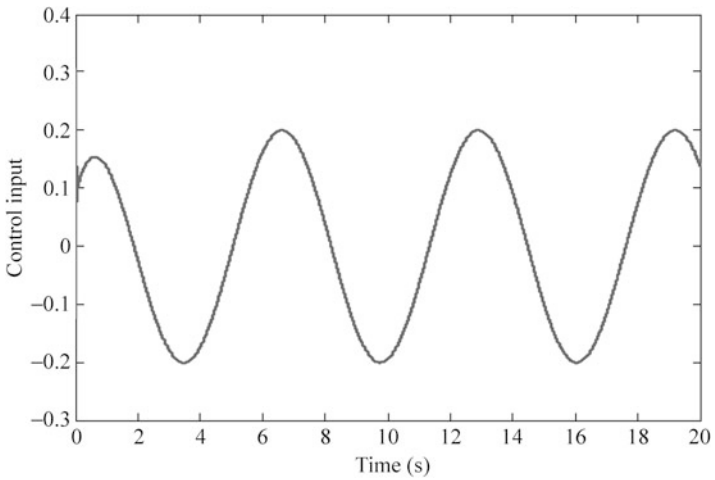


Fig. 4.8 Control input

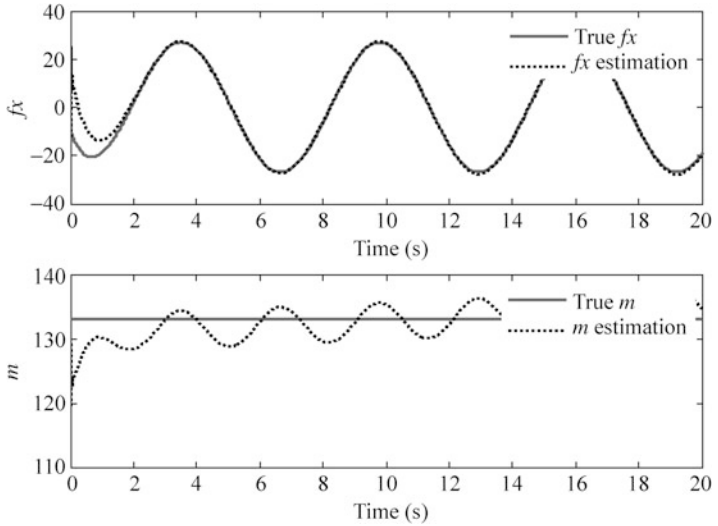


Fig. 4.9 Estimation of $f(x)$ and m

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= \alpha(x) + \beta(x)u + d(t)\\ y &= x_1\end{aligned} \tag{4.36}$$

where $\boldsymbol{x} = [x_1 \quad x_2]^{\mathrm{T}} \in \mathrm{R},\ u \in R,$ and $y \in R$ are state variables, system input, and output, respectively; $\alpha(x)$ and $\beta(x)$ are unknown smooth functions; and $d(t)$ denotes the external disturbance bound by a known constant $d_0{>}0,$ that is, $|d(t)| \le d_0$.

Assumption 4.1. The sign of $\beta(x)$ is known and $\beta(x) \neq 0,\ \forall x \in \Omega$.

Since the sign of $\beta(x)$ is known, without losing generality, assume that $\beta(x){>}0$.

Assumption 4.2. There exists a known smooth function $\bar{\beta}$ such that $|\beta(x)| \le \bar{\beta}$.

Define vector $\boldsymbol{x}_{\mathrm{d}},\ \boldsymbol{e}$, and an augmented s item as

$$\begin{aligned}&\boldsymbol{x}_{\mathrm{d}} = [y_{\mathrm{d}} \quad \dot{y}_{\mathrm{d}}]^{\mathrm{T}}\\ &\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_{\mathrm{d}} = [e \quad \dot{e}]^{\mathrm{T}}, \quad s = [\lambda \quad 1]\boldsymbol{e} = \lambda e + \dot{e}\end{aligned} \tag{4.37}$$

where λ is chosen as $\lambda > 0$ such that polynomial $s + \lambda$ is Hurwitz.

From (4.37), we have

$$\begin{aligned}\dot{s} &= \lambda\dot{e} + \ddot{e} = \lambda\dot{e} + \ddot{x}_1 - \ddot{y}_{\mathrm{d}}\\ &= \lambda\dot{e} + \alpha(x) + \beta(x)u + d(t) - \ddot{y}_{\mathrm{d}}\\ &= \alpha(x) + v + \beta(x)u + d(t)\end{aligned} \tag{4.38}$$

where $v = -\ddot{y}_{\mathrm{d}} + \lambda\dot{e}$.

4.3.2 Desired Feedback Control and Function Approximation

Lemma 4.1. Consider system (4.36) with Assumptions 4.1 and 4.2 satisfied and $d(t) = 0$. If the desired controller is chosen as

$$u^* = -\frac{1}{\beta(x)}(\alpha(x) + v) - \left(\frac{1}{\varepsilon\beta(x)} + \frac{1}{\varepsilon\beta^2(x)} - \frac{\dot{\beta}(x)}{2\beta^2(x)}\right)s \tag{4.39}$$

where $\varepsilon > 0$ is a design parameter, then $\lim_{t\to\infty} ||e(t)|| = 0$.

Proof. Substituting $u = u^*$ into (4.38) and noting $d(t) = 0$, we have

$$\begin{aligned}\dot{s} &= \alpha(x) + v + \beta(x)\left(-\frac{1}{\beta(x)}(\alpha(x) + v) - \left(\frac{1}{\varepsilon\beta(x)} + \frac{1}{\varepsilon\beta^2(x)} - \frac{\dot{\beta}(x)}{2\beta^2(x)}\right)s\right) \\ &= -\left(\frac{1}{\varepsilon} + \frac{1}{\varepsilon\beta(x)} - \frac{\dot{\beta}(x)}{2\beta(x)}\right)s = -\left(\frac{1}{\varepsilon} + \frac{1}{\varepsilon\beta(x)}\right)s + \frac{\dot{\beta}(x)}{2\beta(x)}s.\end{aligned}$$

Choosing a Lyapunov function as $V = \frac{1}{2\beta(x)}s^2$, then

$$\begin{aligned}\dot{V} &= \frac{1}{\beta(x)}s\dot{s} - \frac{\dot{\beta}(x)}{2\beta^2(x)}s^2 \\ &= \frac{1}{\beta(x)}s\left[-\left(\frac{1}{\varepsilon} + \frac{1}{\varepsilon\beta(x)}\right)s + \frac{\dot{\beta}(x)}{2\beta(x)}s\right] - \frac{\dot{\beta}(x)}{2\beta^2(x)}s^2 \\ &= -\left(\frac{1}{\varepsilon\beta(x)} + \frac{1}{\varepsilon\beta^2(x)}\right)s^2 \le 0\end{aligned} \tag{4.40}$$

Since $\beta(x) > 0$, then we have $\dot{V} \le 0$, which implies that $\lim_{t\to\infty} |s| = 0$; subsequently, we have $\lim_{t\to\infty} ||e(t)|| = 0$.

From (4.40), it is shown that the smaller the parameter ε is, the more negative $\dot{V}$ is. Hence, the convergence rate of the tracking error can be adjusted by tuning the design parameter ε.

From (4.39), the desired controller u^* can be written as a function of $\boldsymbol{x}$, s, and v as

$$z = \begin{bmatrix}\boldsymbol{x}^{\mathrm{T}} & s & \frac{s}{\varepsilon} & v\end{bmatrix}^{\mathrm{T}} \in \Omega_z \subset R^5 \tag{4.41}$$

where compact set Ω_z is defined as

$$\Omega_z = \left\{\left(\boldsymbol{x}^{\mathrm{T}} \quad s \quad \frac{s}{\varepsilon} \quad v\right) | x \in \Omega; |x_d \in \Omega_d; s = [\lambda \quad 1]\boldsymbol{e}; v = -\ddot{y}_{\mathrm{d}} + \lambda\dot{e}\right\} \tag{4.42}$$

When nonlinear functions $\alpha(x)$ and $\beta(x)$ are unknown, $u^*(z)$ is not available. In the following, RBF neural network is applied for approximating the unknown function $u^*(z)$.

It should be noted that though the elements s and $\frac{s}{\varepsilon}$ in the input vector z are dependent due to constant ε, they are in different scales when a very small ε is chosen. Thus, $\frac{s}{\varepsilon}$ is also fed into neural network as an input for improving the NN approximation accuracy.

There exist an ideal constant weight vector W^*, such that

$$u^*(z) = \boldsymbol{W}^{*\mathrm{T}}\boldsymbol{h}(z) + \mu_l, \quad \forall z \in \Omega_z \tag{4.43}$$

where $\boldsymbol{h}(z)$ is radial-basis function vector, μ_l is called the NN approximation error satisfying $|\mu_l| \leq \mu_0$, and

$$\boldsymbol{W}^* = \arg\min_{W \in R^l} \left\{ \sup_{z \in \Omega_z} \left| \boldsymbol{W}^{\mathrm{T}}\boldsymbol{h}(z) - u^*(z) \right| \right\}.$$

4.3.3 Controller Design and Performance Analysis

Figure 4.10 shows the closed-loop neural-based adaptive control scheme.

Let $\hat{\boldsymbol{W}}$ be an estimate of the ideal NN weight $\boldsymbol{W}^*$, the direct adaptive controller is designed as

$$u = \hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(z). \tag{4.44}$$

The adaptive law is designed as

$$\dot{\hat{\boldsymbol{W}}} = -\boldsymbol{\Gamma}\left(\boldsymbol{h}(z)s + \sigma\hat{\boldsymbol{W}}\right) \tag{4.45}$$

where $\boldsymbol{\Gamma} = \boldsymbol{\Gamma}^{\mathrm{T}} > 0$ is an adaptation gain matrix and $\sigma > 0$ is a constant.

Substituting controller (4.44) into (4.38), error equation (4.38) can be rewritten as

$$\dot{s} = \alpha(x) + v + \beta(x)\hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(z) + d(t). \tag{4.46}$$

Adding and subtracting on the right-hand side of (4.46) and noting (4.43), we have

$$\dot{s} = \alpha(x) + v + \beta(x)\left(\hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(z) - \boldsymbol{W}^{*\mathrm{T}}\boldsymbol{h}(z) - \mu_l\right) + \beta(x)u^*(z) + d(t). \tag{4.47}$$

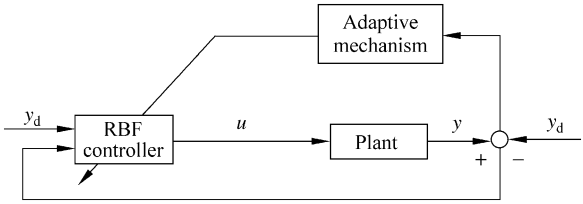


Fig. 4.10 Block diagram of direct RBF control scheme

From (4.39) we have

$$u^* = -\frac{1}{\beta(x)}(\alpha(x)+v) - \left(\frac{1}{\varepsilon\beta(x)} + \frac{1}{\varepsilon\beta^2(x)} - \frac{\dot{\beta}(x)}{2\beta^2(x)}\right)s.$$

Substituting above equation into (4.47) leads to

$$\dot{s} = \beta(x)\left(\tilde{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{z}) - \mu_l\right) - \left(\frac{1}{\varepsilon} + \frac{1}{\varepsilon\beta(x)} - \frac{\dot{\beta}(x)}{2\beta(x)}\right)s + d(t) \tag{4.48}$$

where $\tilde{W} = \hat{W} - W^*$.

The item $\beta(x)$ exists as a coefficient of $\tilde{W}^{\mathrm{T}}$ in (4.48), which can make $\beta(x)$ appear in $\dot{V}$ if we design $\frac{1}{2}s^2$ as a part of Lyapunov function V, and can cause $\beta(x)$ to be included in the adaptive law $\dot{\hat{W}}$.

To avoid $\beta(x)$ being included in the adaptive law $\dot{\hat{W}}$, $\frac{1}{2}\frac{s^2}{\beta(x)}$. should be used instead of $\frac{1}{2}s^2$, then the Lyapunov function is designed as

$$V = \frac{1}{2}\left(\frac{s^2}{\beta(x)} + \tilde{W}^{\mathrm{T}}\Gamma^{-1}\tilde{W}\right). \tag{4.49}$$

Differentiating (4.49) along (4.45) and (4.48) yields

$$\begin{aligned}
\dot{V} =& \frac{s\dot{s}}{\beta(x)} - \frac{\dot{\beta}(x)}{2\beta^2(x)}s^2 + \tilde{W}^{\mathrm{T}}\Gamma^{-1}\dot{\tilde{W}} \\
=& \frac{s}{\beta(x)}\left(\beta(x)\left(\tilde{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{z}) - \mu_l\right) - \left(\frac{1}{\varepsilon} + \frac{1}{\varepsilon\beta(x)} - \frac{\dot{\beta}(x)}{2\beta(x)}\right)s + d(t)\right) \\
& - \frac{\dot{\beta}(x)}{2\beta^2(x)}s^2 + \tilde{W}^{\mathrm{T}}\boldsymbol{\Gamma}^{-1}\left(-\boldsymbol{\Gamma}(\boldsymbol{h}(z)s + \sigma\hat{\boldsymbol{W}})\right) \\
=& -\left(\frac{1}{\varepsilon\beta(x)} + \frac{1}{\varepsilon\beta^2(x)}\right)s^2 + \frac{d(t)}{\beta(x)}s - \mu_l s - \sigma\tilde{W}^{\mathrm{T}}\hat{\boldsymbol{W}}
\end{aligned}$$

Use the facts that

$$\begin{aligned} 2\tilde{W}^{\mathrm{T}}\hat{W} &= \tilde{W}^{\mathrm{T}}(\tilde{W}+W^*)+(\hat{W}-W^*)^{\mathrm{T}}\hat{W} \\ &= \tilde{W}^{\mathrm{T}}\tilde{W}+(\hat{W}-W^*)^{\mathrm{T}}W^*+\hat{W}^{\mathrm{T}}\hat{W}-W^{*\mathrm{T}}\hat{W} \\ &= ||\tilde{W}||^2+||\hat{W}||^2-||W^*||^2 \geq ||\tilde{W}||^2-||W^*||^2 \end{aligned}$$

$$\frac{d(t)}{\beta(x)}s \leq \frac{s^2}{\varepsilon\beta^2(x)}+\frac{\varepsilon}{4}d(t)^2$$

$$|\mu_l s| \leq \frac{s^2}{2\varepsilon\beta(x)}+\frac{\varepsilon}{2}\mu_l^2\beta(x) \leq \frac{s^2}{2\varepsilon\beta(x)}+\frac{\varepsilon}{2}\mu_l^2\bar{\beta}$$

and note that $|\mu_l| \leq \mu_0,\ |d(t)| \leq d_0,$ the following inequality holds

$$\dot{V} \leq -\frac{s^2}{2\varepsilon\beta(x)}-\frac{\sigma}{2}||\tilde{W}||^2+\frac{\varepsilon}{2}\mu_0^2\bar{\beta}+\frac{\varepsilon}{4}d_0^2+\frac{\sigma}{2}||W^*||^2.$$

Considering $\tilde{W}^{\mathrm{T}}\Gamma^{-1}\tilde{W} \leq \bar{\gamma}||\tilde{W}||^2$ ($\bar{\gamma}$ is the largest eigenvalue of Γ^{-1}), we obtain

$$\dot{V} \leq -\frac{1}{\alpha_0}V+\frac{\varepsilon}{2}\mu_0^2\bar{\beta}+\frac{\varepsilon}{4}d_0^2+\frac{\sigma}{2}||W^*||^2$$

where $\alpha_0 = \max\{\varepsilon, \bar{\gamma}/\sigma\}$. Solving the above inequality using Lemma B.5 in [5], we have

$$\begin{aligned} V(t) &\leq \mathrm{e}^{-t/\alpha_0}V(0)+\left(\frac{\varepsilon}{2}\mu_0^2\bar{\beta}+\frac{\varepsilon}{4}d_0^2+\frac{\sigma}{2}||W^*||^2\right)\int_0^t \mathrm{e}^{-(t-\tau)/\alpha_0}\mathrm{d}\tau \\ &\leq \mathrm{e}^{-t/\alpha_0}V(0)+\alpha_0\left(\frac{\varepsilon}{2}\mu_0^2\bar{\beta}+\frac{\varepsilon}{4}d_0^2+\frac{\sigma}{2}||W^*||^2\right), \quad \forall t \geq 0. \end{aligned} \tag{4.50}$$

Since $V(0)$ is bounded, inequality (4.50) shows that s and $\hat{W}(t)$ are bounded. By (4.49) it follows that $V \geq \frac{1}{2}\frac{s^2}{\beta(x)}$, thus, $s \leq \sqrt{2\beta(x)V} \leq \sqrt{2\bar{\beta}V}$.

Combining with (4.50), noting $\sqrt{ab} \leq \sqrt{a}+\sqrt{b}(a>0, b>0)$, we obtain

$$|\mathrm{s}| \leq \mathrm{e}^{-t/2\alpha_0}\sqrt{2\bar{\beta}V(0)}+\sqrt{\alpha_0\bar{\beta}}\left(\varepsilon\mu_0^2\bar{\beta}+\frac{\varepsilon}{2}d_0^2+\sigma||\mathrm{s}||^2\right)^{1/2}, \quad \forall t \geq 0.$$

4.3.4 Simulation Example

4.3.4.1 First Example

The plant dynamics can be expressed in the form of system (4.36) as follows

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= -25x_2 + 133u + d(t)\\ y &= x_1\end{aligned}$$

where $\alpha(x) = -25x_2, \beta(x) = 133, d(t) = 100\sin t$, and $\boldsymbol{x} = [x_1 \quad x_2]^{\mathrm{T}} = [\theta \quad \dot{\theta}]^{\mathrm{T}}$.

The initial states are $\boldsymbol{x} = [0.5 \quad 0]^{\mathrm{T}}$, and the reference signal is $y_{\mathrm{d}} = \sin t$. The input vector of RBF is $\boldsymbol{z} = [x_1 \quad x_2 \quad s \quad s/\varepsilon \quad v]^{\mathrm{T}}$, the network structure 5-9-1 is used. The parameters of $\boldsymbol{c}_i$ and b_i must be chosen according to the scope of the input value. If the parameter values are chosen inappropriately, Gaussian function will not be effectively mapped, and RBF network will be invalid. In this example, according to the practical scope of x_1, x_2, s, s/ε, and v, according to (4.42), for each Gaussian function, the parameters of $\boldsymbol{c}_{\mathrm{i}}$ and b_i are designed as $[-2 \quad -1.5 \quad -1 \quad -0.5 \quad 0 \quad 0.5 \quad 1 \quad 1.5 \quad 2]$ and 5.0.

The adaptive controllers (4.44) and (4.45) are used; we set $\lambda = 5.0$, $\Gamma_{ii} = 500$ $(i = 9)$, $\varepsilon = 0.25$ and $\sigma = 0.005$, from $\beta(x)$ expression we set $\bar{\beta} = 150$; and the initial weight value is chosen as zero.

The results are shown from Figs. 4.11 and 4.12. The Simulink program of this example is chap4_4sim.mdl; the Matlab programs of the example are given in the Appendix.

4.3.4.2 Second Example

A pendulum plant dynamics can be expressed in the form of system (4.36) as follows [3]:

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= \alpha(x) + \beta(x)u + d(t)\\ y &= x_1\end{aligned}$$

where $\alpha(x) = \frac{0.5\sin x_1(1+0.5\cos x_1)x_2^2 - 10\sin x_1(1+\cos x_1)}{0.25(2+\cos x_1)^2}$, $\beta(x) = \frac{1}{0.25(2+\cos x_1)^2}$, $d(t) = d_1(t)\cos x_1$, $\boldsymbol{x} = [x_1 \quad x_2]^{\mathrm{T}} = [\theta \quad \dot{\theta}]^{\mathrm{T}}$, $d_1 = \cos(3t)$.

The initial states are $\boldsymbol{x} = [0 \quad 0]^{\mathrm{T}}$, and the reference signal is $y_{\mathrm{d}} = \frac{\pi}{6}\sin t$. The operation range of the system is chosen as $\Omega = \left\{(x_1, x_2) \middle| |x_1| \le \frac{\pi}{2}, |x_2| \le 4\pi\right\}$.

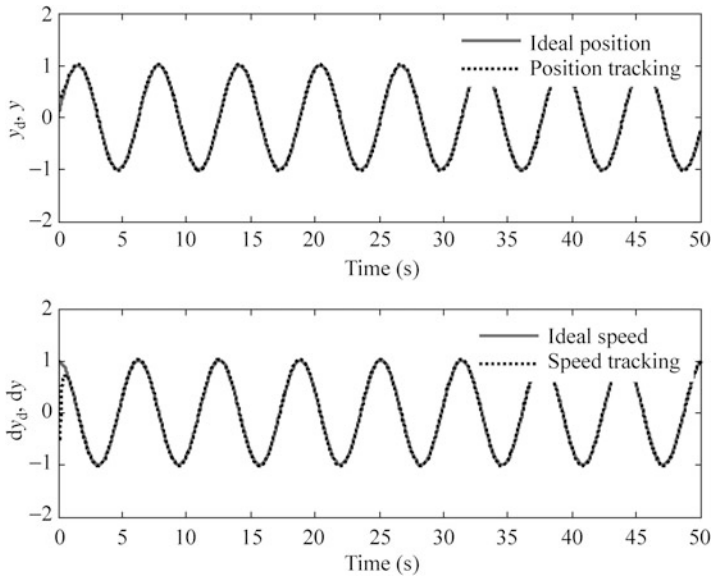


Fig. 4.11 Position and speed tracking

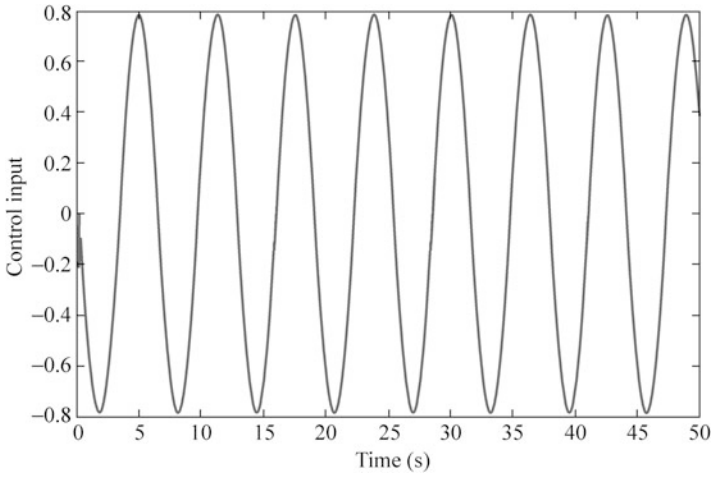


Fig. 4.12 Control input

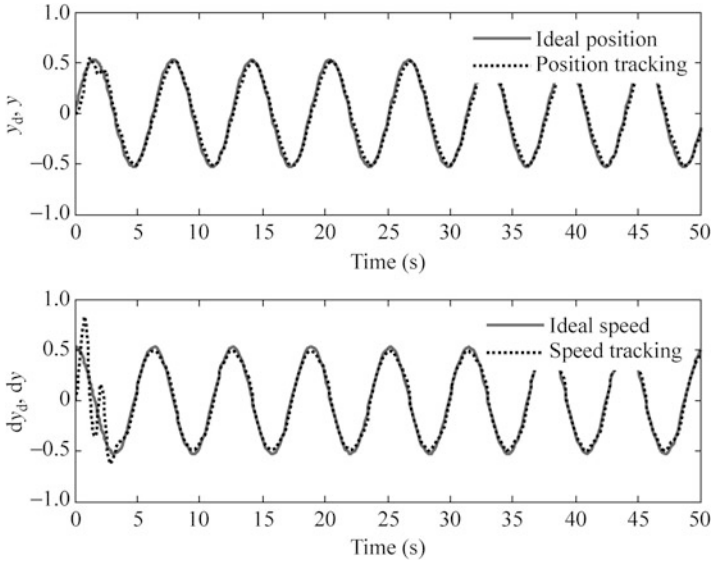


Fig. 4.13 Position and speed tracking

The input vector of RBF is $z = \begin{bmatrix} x_1 & x_2 & s & \frac{s}{\varepsilon} & v \end{bmatrix}^{\mathrm{T}}$, the network structure 5-13-1 is used. In this example, according to the practical scope of $x_1, x_2, s, s/\varepsilon$, and v, according to (4.42), for each Gaussian function, the parameters of $\boldsymbol{c}_{\mathrm{i}}$ and b_i are designed as $\begin{bmatrix} -6 & -5 & -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$ and 3.0.

The adaptive controllers (4.44) and (4.45) are used; we set $\lambda = 10$, $\Gamma_{ii} = 15$ $(i = 13)$, $\varepsilon = 0.25$ and $\sigma = 0.005$, and from $\beta(x)$ expression we set $\bar{\beta} = 1$. The initial weight value is chosen as zero.

The results are shown in Figs. 4.13 and 4.14. The Simulink program of this example is chap4_5sim.mdl; the Matlab programs of the example are given in the Appendix.

Appendix

Programs for Sect. 4.1.3.1

1. Simulink program: chap4_1sim.mdl

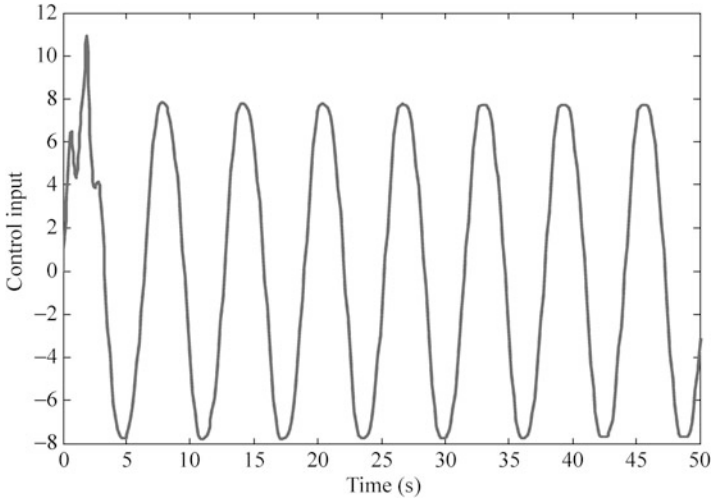
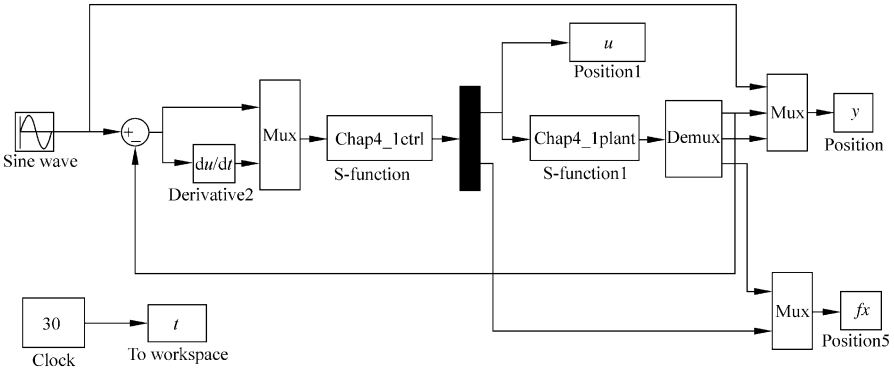


Fig. 4.14 Control input



2. S function for controller: chap4_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 c b
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(5,1)];
c= [-2 -1 0 1 2;
    -2 -1 0 1 2];
b=0.20;
str = [];
ts = [];
function sys=mdlDerivatives(t,x,u)
global c b
gama=1200;
yd=0.1*sin(t);
dyd=0.1*cos(t);
ddyd=-0.1*sin(t);

e=u(1);
de=u(2);
x1=yd-e;
x2=dyd-de;

kp=30;kd=50;
K=[kp kd]';

E=[e,de]';

Fai=[0 1;-kp -kd];
A=Fai';

Q=[500 0;0 500];
P=lyap(A,Q);

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

end
W=[x(1) x(2) x(3) x(4) x(5)]';

B=[0;1];
```

```
S=-gama*E'*P*B*h;

for i=1:1:5
    sys(i)=S(i);

end

function sys=mdlOutputs(t,x,u)
global c b
yd=0.1*sin(t);
dyd=0.1*cos(t);
ddyd=-0.1*sin(t);

e=u(1);
de=u(2);
x1=yd-e;
x2=dyd-de;

kp=30;kd=50;
K=[kp kd]';

E=[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-c(:,j))^2/(2*b^2));

end
fxp=W'*h;

gx=133;

ut=1/gx*(-fxp+ddyd+K'*E);

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

3. S function for plant: chap4_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)]);
```

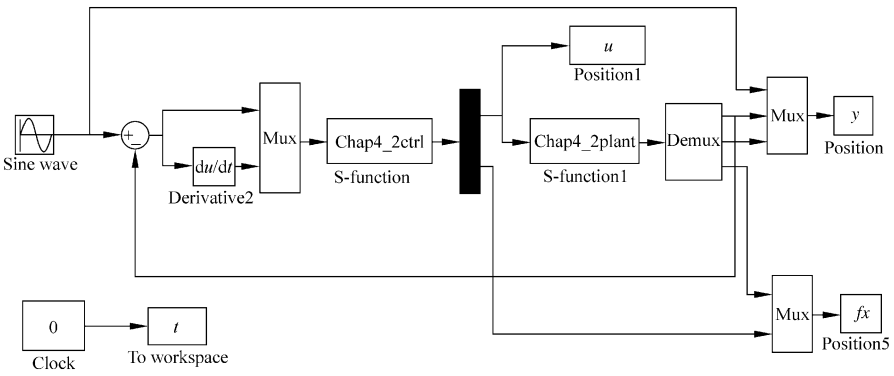
```
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[pi/60 0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
F=10*x(2)+1.5*sign(x(2));
fx=-25*x(2)-F;

sys(1)=x(2);
sys(2)=fx+133*u;
function sys=mdlOutputs(t,x,u)
F=10*x(2)+1.5*sign(x(2));
fx=-25*x(2)-F;
sys(1)=x(1);
sys(2)=x(2);
sys(3)=fx;
```

4. Plot program: chap4_1plot.m

```
close all;

figure(1);
subplot(211);
plot(t,y(:,1),'r',t,y(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('yd,y');
legend('ideal position','position tracking');
subplot(212);
plot(t,0.1*cos(t),'r',t,y(:,3),'k:','linewidth',2);
xlabel('time(s)');ylabel('dyd,dy');
legend('ideal speed','speed tracking');

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

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

Programs for Sect. 4.1.3.2

1. Simulink program: chap4_2sim.mdl;



2. S function for controller: chap4_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 c b
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(5,1)];
c= [-2 -1 0 1 2;
    -2 -1 0 1 2];
b=0.20;
str = [];
ts = [];
function sys=mdlDerivatives(t,x,u)
global c b
gama=1200;
yd=0.1*sin(t);
dyd=0.1*cos(t);
ddyd=-0.1*sin(t);

e=u(1);
de=u(2);
x1=yd-e;
x2=dyd-de;

kp=30;kd=50;
K=[kp kd]';
E=[e de]';

Fai=[0 1;-kp -kd];
A=Fai';

Q=[500 0;0 500];
P=lyap(A,Q);

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

end
W=[x(1) x(2) x(3) x(4) x(5)]';

B=[0;1];
S=-gama*E'*P*B*h;

for i=1:1:5
    sys(i)=S(i);

end

function sys=mdlOutputs(t,x,u)
global c b
yd=0.1*sin(t);
dyd=0.1*cos(t);
ddyd=-0.1*sin(t);
```

```
e=u(1);
de=u(2);
x1=yd-e;
x2=dyd-de;

kp=30;kd=50;
K=[kp kd]';
E=[e de]';

Fai=[0 1;-kp -kd];
A=Fai';

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-c(:,j))^2/(2*b^2));

end
fxp=W'*h;

%%%%%%%%%
g=9.8;mc=1.0;m=0.1;l=0.5;
S=l*(4/3-m*(cos(x(1)))^2/(mc+m));
gx=cos(x(1))/(mc+m);
gx=gx/S;
%%%%%%%%%%%%
ut=1/gx*(-fxp+ddyd+K'*E);

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

3. S function for plant: chap4_2plant.m;

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

```
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
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;
```

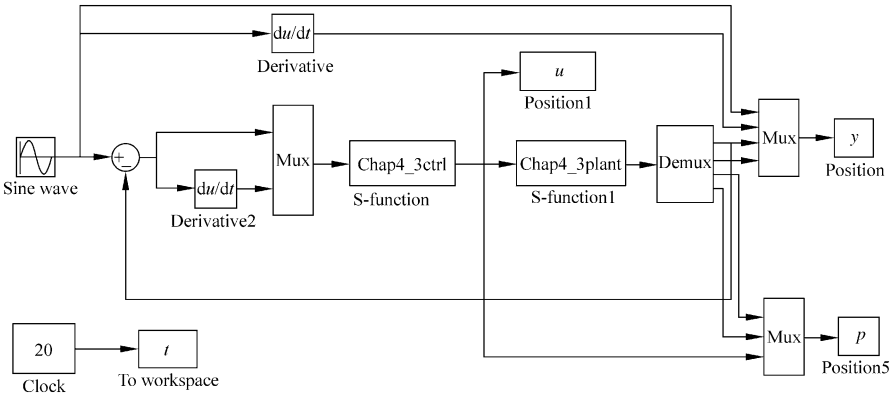
4. Plot program: chap4_2plot.m

```
close all;

figure(1);
subplot(211);
plot(t,y(:,1),'r',t,y(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('yd,y');
legend('ideal position','position tracking');
subplot(212);
plot(t,0.1*cos(t),'r',t,y(:,3),'k:','linewidth',2);
xlabel('time(s)');ylabel('dyd,dy');
legend('ideal speed','speed tracking');

figure(2);
```

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

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

Programs for Sect. 4.2

1. Simulink program: chap4_3sim.mdl;



2. S function for controller: chap4_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 node c b
```

```
node=5;
sizes = simsizes;
sizes.NumContStates = node+1;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [zeros(1,5),120];
c= [-1 -0.5 0 0.5 1;
    -1 -0.5 0 0.5 1];
b=2;
str = [];
ts = [];
function sys=mdlDerivatives(t,x,u)
global node c b
yd=sin(t);
dyd=cos(t);
ddyd=-sin(t);

e=u(1);
de=u(2);
x1=yd-e;
x2=dyd-de;

kp=30;
kd=50;
K=[kp kd]';
E=[e de]';

Fai=[0 1;-kp -kd];
A=Fai';
Q=[500 0;0 500];
P=lyap(A,Q);

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-c(:,j))^2/(2*b^2));
end
fxp=W'*h;

mp=x(node+1);

ut=1/mp*(-fxp+ddyd+K'*E);
```

```
B=[0;1];
gama=1200;
S=-gama*E'*P*B*h;
for i=1:1:node
     sys(i)=S(i);
end

eta=0.0001;
ml=100;
if (E'*P*B*ut>0)
     dm=(1/eta)*E'*P*B*ut;

end
if (E'*P*B*ut<=0)
     if (mp>ml)
     dm=(1/eta)*E'*P*B*ut;
     else
     dm=1/eta;
     end
end
sys(node+1)=dm;

function sys=mdlOutputs(t,x,u)
global node c b
yd=sin(t);
dyd=cos(t);
ddyd=-sin(t);

e=u(1);
de=u(2);
x1=yd-e;
x2=dyd-de;

kp=30;
kd=50;
K=[kp kd]';
E=[e de]';

W=[x(1) x(2) x(3) x(4) x(5)]';
xi=[e;de];
h=zeros(5,1);
for j=1:1:node
    h(j)=exp(-norm(xi-c(:,j))^2/(2*b^2));
end
fxp=W'*h;

mp=x(node+1);

ut=1/mp*(-fxp+ddyd+K'*E);
```

```
sys(1)=ut;
sys(2)=fxp;
sys(3)=mp;
```

3. S function for plant: chap4_3plant.m;

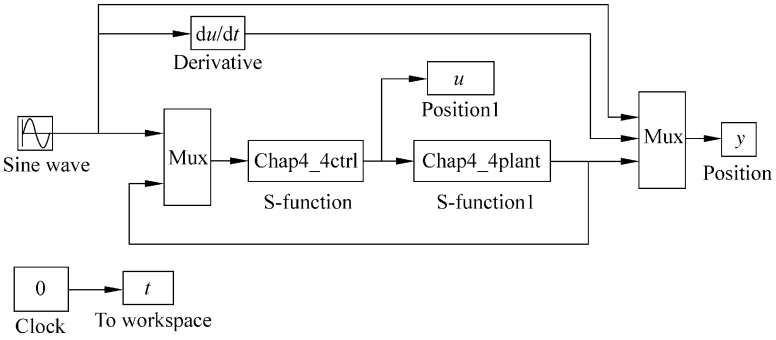
```
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
     [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
     sys=mdlDerivatives(t,x,u);
case 3,
     sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
     sys = [];
otherwise
     error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 3;
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=-25*x(2)-10*x(1);
m=133;

sys(1)=x(2);
sys(2)=fx+m*ut;
function sys=mdlOutputs(t,x,u)
fx=-25*x(2)-10*x(1);
m=133;
sys(1)=x(1);
sys(2)=x(2);
sys(3)=fx;
sys(4)=m;
```

4. Plot program: chap4_3plot.m

```
close all;

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

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

figure(3);
subplot(211);
plot(t,p(:,1),'r',t,p(:,4),'k:','linewidth',2);
xlabel('time(s)');ylabel('fx');
legend('True fx','fx estimation');
subplot(212);
plot(t,p(:,2),'r',t,p(:,5),'k:','linewidth',2);
xlabel('time(s)');ylabel('m');
legend('True m','m estimation');
```

Programs for Sect. 4.3.4.1

Main Simulink program: chap4_4sim.mdl



Control law program: chap4_4ctrl.m

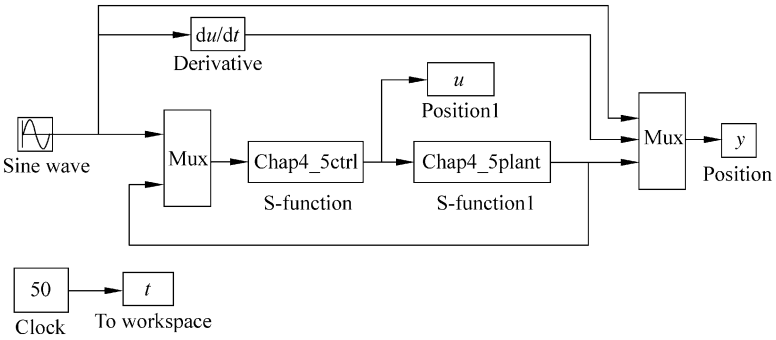
```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
     [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
     sys=mdlDerivatives(t,x,u);
case 3,
     sys=mdlOutputs(t,x,u);
case {2,4,9}
     sys=[];
otherwise
     error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
global node c b lambd epc
lambd=5;
epc=0.25;
node=9;
sizes = simsizes;
sizes.NumContStates = node;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = zeros(1,9);
c= [-2 -1.5 -1 -0.5 0 0.5 1 1.5 2;
    -2 -1.5 -1 -0.5 0 0.5 1 1.5 2;
    -2 -1.5 -1 -0.5 0 0.5 1 1.5 2;
    -2 -1.5 -1 -0.5 0 0.5 1 1.5 2;
    -2 -1.5 -1 -0.5 0 0.5 1 1.5 2];
b=5;
str = [];
ts = [];
function sys=mdlDerivatives(t,x,u)
global node c b lambd epc
yd=sin(t);
dyd=cos(t);
ddyd=-sin(t);
x1=u(2);
x2=u(3);
e=x1-yd;
de=x2-dyd;
```

```
s=lambd*e+de;
v=-ddyd+lambd*de;
xi=[x1;x2;s;s/epc;v];

h=zeros(9,1);
for j=1:1:9
    h(j)=exp(-norm(xi-c(:,j))^2/(2*b^2));
end

rou=0.005;
Gama=500*eye(node);
W=[x(1) x(2) x(3) x(4) x(5) x(6) x(7) x(8) x(9)]';
S=-Gama*(h*s+rou*W);

for i=1:1:node
    sys(i)=S(i);
end
function sys=mdlOutputs(t,x,u)
global node c b lambd epc
yd=sin(t);
dyd=cos(t);
ddyd=-sin(t);
x1=u(2);
x2=u(3);
e=x1-yd;
de=x2-dyd;
s=lambd*e+de;
v=-ddyd+lambd*de;

xi=[x1;x2;s;s/epc;v];

W=[x(1) x(2) x(3) x(4) x(5) x(6) x(7) x(8) x(9)]';
h=zeros(9,1);
for j=1:1:9
    h(j)=exp(-norm(xi-c(:,j))^2/(2*b^2));
end
betaU=150;
ut=1/betaU*W'*h;

sys(1)=ut;
Plant program: chap4_4plant.m
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
     [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
     sys=mdlDerivatives(t,x,u);
```

```
case 3,
     sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
     sys = [];
otherwise
     error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.5 0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
ut=u(1);
dt=100*sin(t);
sys(1)=x(2);
sys(2)=-25*x(2)+133*ut+dt;
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
Plot program: chap4_4plot.m
close all;

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

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

Programs for Sect. 4.3.4.2

Main Simulink program: chap4_5sim.mdl



Control law program: chap4_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 node c b lambd epc
lambd=5;
epc=0.25;
node=13;
sizes = simsizes;
sizes.NumContStates = node;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = zeros(1,13);
```

```
  c= 2*[-3 -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5 3;
      -3 -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5 3;
      -3 -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5 3;
      -3 -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5 3;
      -3 -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5 3];
  b=3;
  str = [];
  ts = [];
  function sys=mdlDerivatives(t,x,u)
  global node c b lambd epc
  yd=pi/6*sin(t);
  dyd=pi/6*cos(t);
  ddyd=-pi/6*sin(t);
  x1=u(2);
  x2=u(3);
  e=x1-yd;
  de=x2-dyd;

  s=lambd*e+de;
  v=-ddyd+lambd*de;
  xi=[x1;x2;s;s/epc;v];

  h=zeros(13,1);
  for j=1:1:13
       h(j)=exp(-norm(xi-c(:,j))^2/(2*b^2));
  end

  rou=0.005;
  Gama=15*eye(13);
  W=[x(1) x(2) x(3) x(4) x(5) x(6) x(7) x(8) x(9) x(10) x(11)
x(12) x(13)]';
  S=-Gama*(h*s+rou*W);

  for i=1:1:node
      sys(i)=S(i);
  end
  function sys=mdlOutputs(t,x,u)
  global node c b lambd epc
  yd=pi/6*sin(t);
  dyd=pi/6*cos(t);
  ddyd=-pi/6*sin(t);
  x1=u(2);
  x2=u(3);
  e=x1-yd;
  de=x2-dyd;

  s=lambd*e+de;
```

```
v=-ddyd+lambd*de;

xi=[x1;x2;s;s/epc;v];

W=[x(1) x(2) x(3) x(4) x(5) x(6) x(7) x(8) x(9) x(10) x(11)
x(12) x(13)]';
h=zeros(13,1);
for j=1:1:13
    h(j)=exp(-norm(xi-c(:,j))^2/(2*b^2));
end
ut=W'*h;

sys(1)=ut;
Plant program: chap4_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 = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0 0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
ut=u(1);
x1=x(1);
x2=x(2);
a1=0.5*sin(x1)*(1+cos(x1))*x2^2-10*sin(x1)*(1+cos
(x1));
a2=0.25*(2+cos(x1))^2;
```

```
alfax=a1/a2;

b=0.25*(2+cos(x1))^2;
betax=1/b;
d1=cos(3*t);
dt=0.1*d1*cos(x1);

sys(1)=x(2);
sys(2)=alfax+betax*ut+dt;
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
Plot program: chap4_5plot.m
close all;

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

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

References

1. Wang LX (1997) A course in fuzzy systems and control. Prentice-Hall, New York
2. Huang AC, Chen YC (2004) Adaptive sliding control for single-link flexible joint robot with mismatched uncertainties. IEEE Trans Control Syst Technol 12(5):770–775
3. Ge SS, Hang CC, Zhang T (1999) A direct method for robust adaptive nonlinear control with guaranteed transient performance. Syst Control Lett 37:275–284
4. Ge SS, Hang CC, Lee TH, Zhang T (2001) Stable adaptive neural network control. Kluwer, Boston
5. Krstic M, Kanellakopoulos I, Kokotovic P (1995) Nonlinear and adaptive control design. Wiley, New York