Chapter 8
Backstepping Control with RBF

Abstract This chapter introduces backstepping controller design with RBF neural network approximation. Several controller design examples for mechanical systems are given, including backstepping controller for inverted pendulum, backstepping controller for single-link flexible joint robot, and adaptive backstepping controller for single-link flexible joint robot.

Keywords RBF neural network • Backstepping control • Single-link flexible joint robot

8.1 Introduction

Many physical systems do not satisfy the matching condition, which includes some uncertain flexible joint robots in particular.

The backstepping control approach has shown itself very effective in dealing with systems with multiple dynamics and with mismatched uncertainties, such as mechanical systems driven by electrical systems or multiple coupled mechanical systems [1, 2].

The idea of backstepping design is that some appropriate functions of state variables are selected recursively as pseudocontrol inputs for lower dimension subsystems of the overall system. Each backstepping stage results in a new pseudocontrol design, expressed in terms of the pseudocontrol designs from the preceding design stages. When the procedure terminates, a feedback design for the true control input results, which achieves the original design objective by virtue of a final Lyapunov function, formed by summing the Lyapunov functions associated with each individual design stage [3]. The backstepping design provides a systematic framework for the design of tracking and regulation strategies, suitable for a large class of state feedback linearizable nonlinear systems.
A major problem with the backstepping approach is that certain functions must be “linear in the unknown parameters” and some very tedious analysis is needed to determine “regression matrices.”

The possible solution of the problem is to use neural networks (NNs) to estimate certain nonlinear functions. A stable neural controller design using backstepping can be found in [4], where rigorous stability proofs are also provided.

A unified and general approach to backstepping control of nonlinear systems using neural networks is presented in [5]. By using the NNs in each stage of the backstepping procedure to estimate certain nonlinear functions, one can design control law using the backstepping approach, but the linear-in-the-parameters (LIP) assumption is not needed, and no regression matrices need to be found. The NNs weights are also tuned online, with no learning phase required. The boundedness of the tracking error and weight updates is guaranteed.

The very rapid developments described in adaptive and robust control techniques are accompanied by an increase in the use of neural networks for system identification-based control [6–9]. With the help of neural networks, the linear-in-the-parameters assumption of nonlinear function and the determination of regression matrices can be avoided.

A large number of backstepping design schemes are reported that combine the backstepping technique with adaptive neural network [10–13]. For example, an adaptive neural network control via backstepping design was presented for a class of minimum-phase nonlinear systems with known relative degree [4], and the combination of NN with backstepping has been proposed for multiple-input–multiple-output nonlinear systems in block-triangular form [14]. Based on implicit function theory, adaptive neural network control using backstepping was constructed for two special classes of non-affine pure-feedback systems [15].

In this chapter, we take single-rank inverted pendulum and single-link flexible joint robot as two typical examples to explain backstepping controller design with RBF.

### 8.2 Backstepping Control for Inverted Pendulum

The basic idea of backstepping design is that a complex nonlinear system is decomposed into the subsystems and the degree of each subsystem doesn’t exceed that of the whole system. Accordingly, the Lyapunov function and medial fictitious control are designed respectively, and the whole system is obtained through “backstepping.” Thus, the control rule is designed thoroughly. The backstepping method is called as the back-deduce method, and the desired dynamic indexes are satisfied.
8.2.1 System Description

Suppose the plant is a nonlinear system as follows:

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= f(x,t) + g(x,t)u
\end{align*}
\]  

(8.1)

where \(f(x,t)\) and \(g(x,t)\) are the nonlinear functions and \(g(x,t) \neq 0\).

Define \(e_1 = x_1 - x_{1d}\), where \(x_{1d}\) is the ideal position signal, the goal is \(e_1 \to 0\) and \(\dot{e}_1 \to 0\).

8.2.2 Controller Design

Basic backstepping control is designed as follows:

Step 1  
\[
\dot{e}_1 = \dot{x}_1 - \dot{x}_{1d} = x_2 - \dot{x}_{1d}
\]  

(8.2)

To realize \(e_1 \to 0\), design Lyapunov function as

\[
V_1 = \frac{1}{2} e_1^2
\]  

(8.3)

Then

\[
\dot{V}_1 = e_1 \dot{e}_1 = e_1 (x_2 - \dot{x}_{1d})
\]

To realize \(\dot{V}_1 < 0\), if we choose \(x_2 - \dot{x}_{1d} = -k_1 e_1\), \(k_1 > 0\), then

\[
\dot{V}_1 = -k_1 e_1^2.
\]

Step 2  
To realize \(x_2 - \dot{x}_{1d} = -k_1 e_1\), that is, \(x_2 = \dot{x}_{1d} - k_1 e_1\), we choose virtual control as

\[
x_{2d} = \dot{x}_{1d} - k_1 e_1
\]  

(8.4)

To realize \(x_2 \to x_{2d}\), we get a new error

\[
e_2 = x_2 - x_{2d}
\]  

(8.5)

Then, \(\dot{e}_2 = \dot{x}_2 - \dot{x}_{2d} = f(x,t) + g(x,t)u - \dot{x}_{2d}\)

To realize \(e_2 \to 0\) and \(e_1 \to 0\), design Lyapunov function as

\[
V_2 = V_1 + \frac{1}{2} e_2^2 = \frac{1}{2} (e_1^2 + e_2^2)
\]
Then
\[ \dot{V}_2 = e_1(x_2 - \dot{x}_1d) + e_2\dot{e}_2 \\
= e_1(x_{2d} + e_2 - \dot{x}_1d) + e_2\dot{e}_2 \\
= -k_1e_1^2 + e_1e_2 + e_2(f(x,t) + g(x,t)u - \dot{x}_{2d}) \]

To realize \( \dot{V}_2 < 0 \), we choose
\[ e_1 + f(x,t) + g(x,t)u - \dot{x}_{2d} = -k_2e_2, \quad k_2 > 0 \quad (8.6) \]

Then
\[ \dot{V}_2 = -k_1e_1^2 - k_2e_2^2 \]

From (8.6), consider \( \dot{x}_{2d} = \dot{x}_1d - k_1\dot{e}_1 \), and we can get the control law
\[ u = \frac{1}{g(x,t)}(-k_2e_2 - \dot{x}_{2d} - e_1 - f(x,t)) \quad (8.7) \]

In addition, if \( e_1 \to 0 \) and \( e_2 \to 0 \), then we can get \( \dot{e}_1 = x_2 - \dot{x}_1d = e_2 + \dot{x}_{2d} - \dot{x}_1d = e_2 - k_1e_1 \to 0 \).

To realize the control law (8.7), exact values of modeling information \( g(x), f(x) \) are needed, which are difficult in practical engineering. We can use RBF to approximate them.

### 8.2.3 Simulation Example

Consider one link inverted pendulum as follows:
\[ \dot{x}_1 = x_2 \\
\dot{x}_2 = f(x) + g(x)u \]

where \( f(x) = \frac{g \sin x_1 - ml_2^2 \cos x_1 \sin x_1/(m_1 + m)}{l(4/3 - m \cos^2 x_1 / (m_1 + m))} \), \( g(x) = \frac{\cos x_1 / (m_1 + m)}{l(4/3 - m \cos^2 x_1 / (m_1 + m))} \), and \( x_1 \) and \( x_2 \) are oscillation angle and oscillation rate, respectively. \( g = 9.8 \text{ m/s}^2 \), \( m_c = 1 \text{ kg} \) is the vehicle mass, \( m_c = 1 \text{ kg} \), \( m \) is the mass of pendulum bar, \( m = 0.1 \text{ kg} \), \( l \) is one half of pendulum length, \( l = 0.5 \text{ m} \), and \( u \) is the control input.

Consider the desired trajectory as \( x_2(t) = 0.1 \sin(\pi t) \), adopt the control law as (8.7), and select \( k_1 = 35 \) and \( k_2 = 15 \). The initial state of the inverted pendulum is \([\pi/60, 0]\). Simulation results are shown in Figs. 8.1 and 8.2.

The Simulink program of this example is chap8_1sim.mdl, and the Matlab programs of the example are given in the Appendix.
8.3 Backstepping Control Based on RBF for Inverted Pendulum

8.3.1 System Description

Suppose the plant is a nonlinear system as follows:

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= f(x, t) + g(x, t)u
\end{align*}
\]  

(8.8)

Fig. 8.1 Position and speed tracking

Fig. 8.2 Control input
where \( f(x, t) \) and \( g(x, t) \) are the nonlinear functions and \( g(x, t) \neq 0 \).

Define \( e_1 = x_1 - x_{1d} \), where \( x_{1d} \) is the ideal position signal, the goal is \( e_1 \to 0 \) and \( \dot{e}_1 \to 0 \).

### 8.3.2 Backstepping Controller Design

Basic backstepping control is designed as follows.

**Step 1**

\[
\dot{e}_1 = x_1 - \dot{x}_{1d} = x_2 - \dot{x}_{1d} \quad (8.9)
\]

To realize \( e_1 \to 0 \), design Lyapunov function as

\[
V_1 = \frac{1}{2}e_1^2 \quad (8.10)
\]

Then

\[
\dot{V}_1 = e_1 \dot{e}_1 = e_1(x_2 - \dot{x}_{1d})
\]

To realize \( \dot{V}_1 < 0 \), if we choose \( x_2 - \dot{x}_{1d} = -k_1 e_1, \) \( k_1 > 0 \), then we would have \( \dot{V}_1 = -k_1 e_1^2 \).

**Step 2**

To realize \( x_2 - \dot{x}_{1d} = -k_1 e_1 \), that is \( x_2 = \dot{x}_{1d} - k_1 e_1 \), we choose virtual control as

\[
x_{2d} = \dot{x}_{1d} - k_1 e_1 \quad (8.11)
\]

To realize \( x_2 \to x_{2d} \), we get a new error

\[
e_2 = x_2 - x_{2d} \quad (8.12)
\]

Then, \( \dot{e}_2 = \dot{x}_2 - \dot{x}_{2d} = f(x, t) + g(x, t)u - \dot{x}_{2d} \)

To realize \( e_2 \to 0 \) and \( e_1 \to 0 \), design Lyapunov function as

\[
V_2 = V_1 + \frac{1}{2}e_2^2 = \frac{1}{2}(e_1^2 + e_2^2)
\]

Then

\[
\dot{V}_2 = e_1(x_2 - \dot{x}_{1d}) + e_2 \dot{e}_2
= e_1(x_{2d} + e_2 - \dot{x}_{1d}) + e_2 \dot{e}_2
= -k_1 e_1^2 + e_1 e_2 + e_2(f + gu - \dot{x}_{2d})
= -k_1 e_1^2 + e_1 e_2 + e_2(f + \dot{g}u + (g - \ddot{g})u - \dot{x}_{2d})
\]

256 8 Backstepping Control with RBF
where $\hat{g}$ is estimation of $g$.

To realize $\dot{V}_2 < 0$, we design control law as

$$ u = \frac{-e_1 - \hat{f} + \dot{x}_{2d} - k_2 e_2}{\hat{g}}, \quad k_2 > 0 \quad (8.13) $$

where $\hat{f}$ is estimation of $f$.

Then, we can get

$$ \dot{V}_2 = -k_1 e_1^2 - k_2 e_2^2 + e_2 (f - \hat{f}) + e_2 (g - \hat{g}) u \quad (8.14) $$

### 8.3.3 Adaptive Law Design

The unknown functions of $\hat{f}$ and $\hat{g}$ can be approximated by neural network. Figure 8.3 shows the closed-loop neural-based adaptive control scheme.

We use two RBFs to approximate $f$ and $g$ respectively as follows:

$$ \begin{cases} f = W_1^T h_1 + \epsilon_1 \\ g = W_2^T h_2 + \epsilon_2 \end{cases} \quad (8.15) $$

where $W_i$ is the ideal neural network weight value, $h_i$ is the Gaussian function, $i = 1, 2$, $\epsilon_i$ is the approximation error, and $||e|| = \|[\epsilon_1 \quad \epsilon_2]^T|| < \epsilon_N$, $||W_i||_F \leq W_M$

Define

$$ \begin{cases} \hat{f} = \hat{W}_1^T h_1 \\ \hat{g} = \hat{W}_2^T h_2 \end{cases} \quad (8.16) $$

where $\hat{W}_i^T$ is the weight value estimation.
Define
\[ Z = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}, \quad ||Z||_F \leq Z_M, \quad \dot{Z} = \begin{bmatrix} \dot{W}_1 \\ \dot{W}_2 \end{bmatrix}, \quad \ddot{Z} = Z - \dot{Z} \]

Design Lyapunov function as
\[ V = \frac{1}{2} \xi^T \xi + \frac{1}{2} \text{tr} \left( \dot{Z}^T \Gamma^{-1} \dot{Z} \right) \tag{8.17} \]

where \[ V_2 = \frac{1}{2} \xi^T \xi, \quad \eta > 0, \quad \Gamma \] is a positive-definite matrix with proper dimension,
\[ \Gamma = \begin{bmatrix} \Gamma_1 \\ \Gamma_2 \end{bmatrix}, \quad \text{and} \quad \xi = [e_1 \ e_2]^T \]

Let adaptive law as
\[ \dot{Z} = \Gamma h \xi^T - n ||\xi|| \dot{Z} \tag{8.18} \]

where \[ h = [h_1 \ h_2]^T \] and \[ n \] is a positive number.

From (8.14), (8.15), and (8.16), we have
\[ \dot{V} = \xi^T \dot{\xi} + \text{tr} \left( \dot{Z}^T \Gamma^{-1} \dot{Z} \right) \]
\[ = -k_1e_1^2 - k_2e_2^2 + (\dot{W}_1^T h_1 + e_1) e_2 + (\dot{W}_2^T h_2 + e_2) e_3 + \text{tr} \left( \dot{Z}^T \Gamma^{-1} \dot{Z} \right) \]

where \[ \dot{W}_i^T = W_i^T - \dot{W}_i^T, \quad i = 1, 2 \]

Then
\[ \dot{V} = -\xi^T K_e \xi + \xi^T \epsilon + \xi^T Z \epsilon + \text{tr} \left( \dot{Z}^T \Gamma^{-1} \dot{Z} \right) + \bar{m} e_4 u \]
\[ = -\xi^T K_e \xi + \xi^T \epsilon + \text{tr} \left( \dot{Z}^T \Gamma^{-1} \dot{Z} + \ddot{Z} \dot{h} \xi^T \right) + \bar{m} e_4 u \]

where \[ K_e = [k_1 \ k_2]^T \] and \[ \epsilon = [e_1 \ e_2]^T \]

Since \[ \dot{Z} = -\dot{Z}, \] submitting the adaptive law (8.18), we have
\[ \dot{V} = -\xi^T K_e \xi + \xi^T \epsilon + n ||\xi|| \text{tr} \left( \dot{Z}^T (Z - \dot{Z}) \right) \tag{8.19} \]

According to Schwarz inequality, we have \( \text{tr} \left( \dot{Z}^T (Z - \dot{Z}) \right) \leq ||\dot{Z}||_F ||Z||_F - ||\dot{Z}||_F^2 \), since \( K_{\text{min}} ||\xi||^2 \leq \xi^T K \xi \), \( K_{\text{min}} \) is the minimum eigenvalue of \( K \), (8.19) becomes
\[
\dot{V} \leq - K_{\text{min}} ||\xi||^2 + \varepsilon_N ||\xi|| + n ||\xi|| \left( ||\tilde{Z}||_F ||Z||_F - ||\tilde{Z}||_F^2 \right) \\
\leq - ||\xi|| \left( K_{\text{min}} ||\xi|| - \varepsilon_N + n ||\tilde{Z}||_F \left( ||\tilde{Z}||_F - Z_M \right) \right)
\]

Since
\[
K_{\text{min}} ||\xi|| - \varepsilon_N + n \left( ||\tilde{Z}||_F^2 - ||\tilde{Z}||_F Z_M \right) \\
= K_{\text{min}} ||\xi|| - \varepsilon_N + n \left( \frac{1}{2} Z_M \right)^2 - n \frac{1}{4} Z_M^2
\]

this implies that \( \dot{V} < 0 \) as long as
\[
||\xi|| > \frac{\varepsilon_N + n \frac{1}{4} Z_M^2}{K_{\text{min}}} \quad \text{or} \quad ||\tilde{Z}||_F > \frac{1}{2} Z_M + \sqrt{\frac{Z_M^2}{4} + \frac{\varepsilon_N}{n}}
\]

(8.20)

From the above expression, we can see that the tracking performance is related to the value of \( \varepsilon_N, n \) and \( K_{\text{min}} \).

In addition, if \( e_1 \to 0 \) and \( e_2 \to 0 \), then we can get \( \dot{e}_1 = x_2 - x_{1d} = e_2 + x_{2d} \) \( -x_{1d} = e_2 - k_1 e_1 \to 0 \)

### 8.3.4 Simulation Example

Consider one link inverted pendulum as follows:
\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= g \sin x_1 - m l x_2 \cos x_1 \sin x_1 / (m_c + m) \\
&\quad + \cos x_1 / (m_c + m) u
\end{align*}
\]

where \( x_1 \) and \( x_2 \) are the oscillation angle and oscillation rate, respectively. \( g = 9.8 \text{ m/s}^2 \), \( m_c = 1 \text{ kg} \) is the vehicle mass, \( m_c = 1 \text{ kg} \), \( m \) is the mass of pendulum bar, \( m = 0.1 \text{ kg} \), \( l \) is one half of pendulum length, \( l = 0.5 \text{ m} \), and \( u \) is the control input.

Consider the desired trajectory as \( x_d(t) = 0.1 \sin t \), adopt the control law as (8.13) and adaptive law (8.18), and select \( \Gamma_1 = 500 \) and \( \Gamma_2 = 0.50, n = 0.10, \) and \( k_1 = k_2 = 35 \).

Two RBF neural networks are designed, for each Gaussian function, the parameters of \( c_i \) and \( b_i \) are designed as \([-0.5 -0.25 0.25 0.5] \) and 15. The initial weight value of each neural net in hidden layer is chosen as 0.10.

In the control law (8.13), to prevent from singularity, we should prevent the item \( \hat{\xi} \) value from changing frequently; thus, we should choose small \( \Gamma_2 \) in adaptive law (8.18).
The initial state of the inverted pendulum is $[\pi/60, 0]$. Simulation results are shown from Figs. 8.4, 8.5, and 8.6. The variation of $f(\cdot)$ and $g(\cdot)$ do not converge to $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$. This is due to the fact that the desired trajectory is not persistently exciting and the tracking error performance can be achieved by many possible values of $f(\cdot)$ and $g(\cdot)$, besides the true $f(\cdot)$ and $g(\cdot)$, which has been explained in Sect. 1.5, and this occurs quite often in real-world application.

The Simulink program of this example is chap8_2sim.mdl, and the Matlab programs of the example are given in the Appendix.

8.4 Backstepping Control for Single-Link Flexible Joint Robot

8.4.1 System Description

The dynamic equation of single-link flexible joint robot is

$$
\begin{align*}
I\ddot{q}_1 + MgL\sin q_1 + K(q_1 - q_2) &= 0 \\
J\ddot{q}_2 + K(q_2 - q_1) &= u
\end{align*}
$$

(8.21)

where $q_1 \in \mathbb{R}^n$ and $q_2 \in \mathbb{R}^n$ are the link angular and motor angular respectively, $K$ is the stiffness of the link, $u \in \mathbb{R}^n$ is control input, $I$ is motor inertia, $I$ is link inertia, $M$ is the link mass, and $L$ is the length from the joint to the center of the link.

Choose $x_1 = q_1$, $x_2 = \dot{q}_1$, $x_3 = q_2$, $x_4 = \dot{q}_2$, and then the system (8.21) can be written as
\[ \begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -\frac{1}{J} (MgL \sin x_1 + K (x_1 - x_3)) \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= \frac{1}{J} (u - K (x_3 - x_1))
\end{align*} \] (8.22)

Also, the simplified system equation can be written as
\[
\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= x_3 + g(x) \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= f(x) + mu
\end{aligned}
\]

(8.23)

where \( x = [x_1 \ x_2 \ x_3 \ x_4]^T \) is the state vector, \( g(x) = -x_3 - MgL \sin(x_1)/I - K(x_1 - x_3)/J \), \( f(x) = K(x_1 - x_3)/J \), and \( m = 1/J \).

Let \( e_1 = x_1 - x_{1d} \) and \( e_2 = x_1 - \dot{x}_{1d} \). The control goal is \( x_1 \) track \( x_{1d} \), \( \dot{x}_1 \) track \( \dot{x}_{1d} \), that is, \( e_1 \to 0 \) and \( e_2 \to 0 \). If we design Lyapunov function as \( V = \frac{1}{2} e_1^2 + \frac{1}{2} e_2^2 \), then we have \( \dot{V} = e_1 \dot{e}_1 + e_2 \dot{e}_2 \), and since the control input \( u \) does not appear in \( \dot{V} \) expression, the control input cannot be designed.

### 8.4.2 Backstepping Controller Design

Reference to the main idea of adaptive backstepping sliding controller design for single-link flexible joint robot proposed in [16], we discuss backstepping controller design for single-link flexible joint robot in several steps as follows:

**Step 1** Define \( e_1 = x_1 - x_{1d} \), and \( x_{1d} \) is the ideal position signal, and then

\[
\dot{e}_1 = \dot{x}_1 - \dot{x}_{1d} = x_2 - \dot{x}_{1d}
\]

To realize \( e_1 \to 0 \), define a Lyapunov function as

\[
V_1 = \frac{1}{2} e_1^2
\]

(8.24)

Then

\[
\dot{V}_1 = e_1 \dot{e}_1 = e_1(x_2 - \dot{x}_{1d})
\]

To realize \( \dot{V}_1 < 0 \), if we choose \( x_2 - \dot{x}_{1d} = -k_1 e_1, k_1 > 0 \), then we would have \( \dot{V}_1 = -k_1 e_1^2 \).

**Step 2** To realize \( x_2 - \dot{x}_{1d} = -k_1 e_1 \), that is, \( x_2 = \dot{x}_{1d} - k_1 e_1 \), we choose virtual control as

\[
x_{2d} = \dot{x}_{1d} - k_1 e_1
\]

(8.25)

To realize \( x_2 \to x_{2d} \), we get a new error

\[
e_2 = x_2 - x_{2d}
\]

Then \( \dot{e}_2 = \dot{x}_2 - \dot{x}_{2d} = x_3 + g(x) - \dot{x}_{2d} \), and
\[
V_1 = e_1(x_2 - \dot{x}_{1d}) = e_1(x_{2d} + e_2 - \dot{x}_{1d}) = e_1(\dot{x}_{1d} - k_1e_1 + e_2 - \dot{x}_{1d}) = -k_1e_1^2 + e_1e_2
\]

To realize \(e_2 \to 0\) and \(e_1 \to 0\), design Lyapunov function as
\[
V_2 = V_1 + \frac{1}{2}e_2^2 = \frac{1}{2}(e_1^2 + e_2^2) \tag{8.26}
\]

Then
\[
\dot{V}_2 = -k_1e_1^2 + e_1e_2 + e_2(x_3 + g(x) - \dot{x}_{2d})
\]

To realize \(\dot{V}_2 < 0\), if we choose
\[
e_1 + x_3 + g(x) - \dot{x}_{2d} = -k_2e_2, \quad k_2 > 0
\]

then we would have \(\dot{V}_2 = -k_1e_1^2 - k_2e_2^2\)

**Step 3** To realize \(e_1 + x_3 + g(x) - \dot{x}_{2d} = -k_2e_2\), that is, \(x_3 = \dot{x}_{2d} - g(x) - k_2e_2 - e_1\), we choose virtual control as
\[
x_{3d} = \dot{x}_{2d} - g(x) - k_2e_2 - e_1 \tag{8.27}
\]

To realize \(x_3 \to x_{3d}\), we get a new error
\[
e_3 = x_3 - x_{3d}
\]

Then \(\dot{e}_3 = \dot{x}_3 - \dot{x}_{3d} = x_4 - \dot{x}_{3d}\)

To realize \(e_3 \to 0\) and \(e_1 \to 0\) and \(e_2 \to 0\), design Lyapunov function as
\[
V_3 = V_2 + \frac{1}{2}e_3^2 = \frac{1}{2}(e_1^2 + e_2^2 + e_3^2) \tag{8.28}
\]

Then
\[
\dot{V}_3 = -k_1e_1^2 + e_1e_2 + e_2(x_3 + g(x) - \dot{x}_{2d}) + e_3\dot{e}_3
\]
\[
= -k_1e_1^2 + e_1e_2 + e_2(e_3 + x_{3d} + g(x) - \dot{x}_{2d}) + e_3\dot{e}_3
\]
\[
= -k_1e_1^2 - k_2e_2^2 + e_2e_3 + e_3(x_4 - \dot{x}_{3d})
\]
\[
= -k_1e_1^2 - k_2e_2^2 + e_3(e_2 + x_4 - \dot{x}_{3d})
\]

To realize \(\dot{V}_3 < 0\), we choose
\[
e_2 + x_4 - \dot{x}_{3d} = -k_3e_3, \quad k_3 > 0
\]
Then we would have

$$\dot{V}_3 = -k_1e_1^2 - k_2e_2^2 - k_3e_3^2$$

Step 4 To realize $e_2 + x_4 - \dot{x}_{3d} = -k_3e_3$, that is, $x_4 = \dot{x}_{3d} - k_3e_3 - e_2$, we choose virtual control as

$$x_{4d} = \dot{x}_{3d} - k_3e_3 - e_2 \quad (8.29)$$

To realize $x_4 \to x_{4d}$, we get a new error

$$e_4 = x_4 - x_{4d}$$

Then, $\dot{e}_4 = \dot{x}_4 - \dot{x}_{4d} = f(x) + mu - \dot{x}_{4d}$

To realize $e_4 \to 0$ and $e_1 \to 0$, $e_2 \to 0$, and $e_3 \to 0$, design Lyapunov function as

$$V_4 = V_3 + \frac{1}{2}e_4^2 = \frac{1}{2}(e_1^2 + e_2^2 + e_3^2 + e_4^2) \quad (8.30)$$

Then

$$\dot{V}_4 = -k_1e_1^2 - k_2e_2^2 + e_3(e_2 + x_4 - \dot{x}_{3d}) + e_4(f(x) + mu - \dot{x}_{4d})$$

Since

$$e_3(e_2 + x_4 - \dot{x}_{3d}) = e_3(e_2 + x_{4d} + e_4 - \dot{x}_{3d})$$

$$= e_3(e_2 + \dot{x}_{3d} - k_3e_3 - e_2 + e_4 - \dot{x}_{3d})$$

$$= e_3e_4 - k_3e_3^2$$

then

$$\dot{V}_4 = -k_1e_1^2 - k_2e_2^2 + e_3e_4 - k_3e_3^2 + e_4(f(x) + mu - \dot{x}_{4d})$$

$$= -k_1e_1^2 - k_2e_2^2 - k_3e_3^2 + e_4(e_3 + f(x) + mu - \dot{x}_{4d})$$

To realize $\dot{V}_4 < 0$, we choose

$$e_3 + f(x) + mu - \dot{x}_{4d} = -k_4e_4, \quad k_4 > 0 \quad (8.31)$$

Then

$$\dot{V}_4 = -k_1e_1^2 - k_2e_2^2 - k_3e_3^2 - k_4e_4^2$$
From (8.31), we can get the control law

$$u = \frac{1}{m}(-f(x) + \dot{x}_4 - k_4 e_4 - e_3) \tag{8.32}$$

To realize the control law (8.32), modeling information $g(x)$, $f(x)$, and $m$ are needed, which are difficult in practical engineering. We can use RBF to approximate them.

### 8.5 Adaptive Backstepping Control with RBF for Single-Link Flexible Joint Robot

For the system (8.23) in Sect. 8.4, we consider $g(x)$, $f(x)$, and $m$ are unknown, the lower bound of $m$ is known, $m \geq \hat{m}$, and $\hat{m} > 0$.

The unknown functions of $\hat{f}$, $\hat{g}$, and $\hat{d}$ can be approximated by neural network, and $\hat{m}$ can be estimated by adaptive law. Figure 8.7 shows the closed-loop neural-based adaptive control scheme.

#### 8.5.1 Backstepping Controller Design with Function Estimation

Reference to the main idea of adaptive neural network backstepping sliding controller design for single-link flexible joint robot proposed in [16], we discuss RBF-based backstepping controller design for single-link flexible joint robot in several steps as follows:

**Step 1** Define $e_1 = x_1 - x_{1d}$, and $x_{1d}$ is the ideal position signal, and then

$$\dot{e}_1 = \dot{x}_1 - \dot{x}_{1d} = x_2 - \dot{x}_{1d} \tag{8.33}$$

To realize $e_1 \to 0$, define a Lyapunov function as

$$V_1 = \frac{1}{2} e_1^2 \tag{8.34}$$

Then

$$\dot{V}_1 = e_1 \dot{e}_1 = e_1(x_2 - \dot{x}_{1d})$$

To realize $\dot{V}_1 < 0$, if we choose $x_2 - \dot{x}_{1d} = -k_1 e_1$, $k_1 > 0$, then we would have $\dot{V}_1 = -k_1 e_1^2$. 
Step 2  

To realize $x_2 - \dot{x}_{1d} = -k_1 e_1$, that is, $x_2 = \dot{x}_{1d} - k_1 e_1$, we choose virtual control as

$$x_{2d} = \dot{x}_{1d} - k_1 e_1$$  \hfill (8.35)

To realize $x_2 \rightarrow x_{2d}$, we get a new error

$$e_2 = x_2 - x_{2d}$$

Then $\dot{e}_2 = \dot{x}_2 - \dot{x}_{2d} = x_3 + g(x) - \dot{x}_{2d}$, and

$$\dot{V}_1 = e_1 (x_2 - \dot{x}_{1d}) = e_1 (x_{2d} + e_2 - \dot{x}_{1d}) = e_1 (\dot{x}_{1d} - k_1 e_1 + e_2 - \dot{x}_{1d})$$

$$= -k_1 e_1^2 + e_1 e_2$$

To realize $e_2 \rightarrow 0$ and $e_1 \rightarrow 0$, design Lyapunov function as

$$V_2 = V_1 + \frac{1}{2} e_2^2 = \frac{1}{2} (e_1^2 + e_2^2)$$  \hfill (8.36)

Then

$$\dot{V}_2 = -k_1 e_1^2 + e_1 e_2 + e_2 (x_3 + g(x) - \dot{x}_{2d})$$

To realize $\dot{V}_2 < 0$, if we choose

$$e_1 + x_3 + g(x) - \dot{x}_{2d} = -k_2 e_2, \quad k_2 > 0$$

then we would have $\dot{V}_2 = -k_1 e_1^2 - k_2 e_2^2$

Step 3  

To realize $e_1 + x_3 + g(x) - \dot{x}_{2d} = -k_2 e_2$, that is, $x_3 = \dot{x}_{2d} - g(x) - k_2 e_2 - e_1$, we choose virtual control as

$$x_{3d} = \dot{x}_{2d} - \hat{g}(x) - k_2 e_2 - e_1$$  \hfill (8.37)
To realize $x_3 \rightarrow x_{3d}$, we get a new error

$$e_3 = x_3 - x_{3d}$$

Then $\dot{e}_3 = \dot{x}_3 - \dot{x}_{3d} = x_4 - \dot{x}_{3d}$.

To realize $e_3 \rightarrow 0$ and $e_1 \rightarrow 0$ and $e_2 \rightarrow 0$, design Lyapunov function as

$$V_3 = V_2 + \frac{1}{2} e_3^2 = \frac{1}{2} (e_1^2 + e_2^2 + e_3^2) \quad (8.38)$$

Then

$$\dot{V}_3 = -k_1 e_1^2 + e_1 e_2 + e_2 (x_3 + g(x) - \dot{x}_{3d}) + e_3 \dot{e}_3$$

$$= -k_1 e_1^2 + e_1 e_2 + e_2 (e_3 + x_{3d} + g(x) - \dot{x}_{3d}) + e_3 \dot{e}_3$$

$$= -k_1 e_1^2 - k_2 e_2^2 + e_2 e_3 + e_3 (x_4 - \dot{x}_{3d}) + e_2 (g(x) - \dot{g}(x))$$

$$= -k_1 e_1^2 - k_2 e_2^2 + e_3 (e_2 + x_4 - \dot{x}_{3d}) + e_2 (g(x) - \dot{g}(x))$$

To realize $\dot{V}_3 < 0$, we choose

$$e_2 + x_4 - \dot{x}_{3d} = -k_3 e_3, \quad k_3 > 0$$

Then, we would have $\dot{V}_3 = -k_1 e_1^2 - k_2 e_2^2 - k_3 e_3^2 + e_2 (g(x) - \dot{g}(x))$.

**Step 4** Design control law

To realize $e_2 + x_4 - \dot{x}_{3d} = -k_3 e_3$, that is, $x_4 - \dot{x}_{3d} = -k_3 e_3 - e_2$, we choose virtual control as

$$x_{4d} = \hat{x}_{3d} - k_3 e_3 - e_2 \quad (8.39)$$

where $\dot{x}_{3d} = \dot{x}_{3d1} - d$, $\dot{x}_{3d} = \dot{x}_{3d1} - \dot{d}$, and

$$\dot{x}_{3d} = -\hat{g} + \ddot{x}_{2d} - k_2 \dot{e}_2 - \dot{e}_1$$

$$= -\hat{g} + \ddot{x}_{1d} - k_1 \dot{x}_1 - k_2 (x_3 + g - \dot{x}_{2d}) - (x_2 - \dot{x}_{1d})$$

$$= -\hat{g} + \ddot{x}_{1d} - k_1 (x_3 + g - \dot{x}_{1d}) - k_2 (x_3 + g - \dot{x}_{2d}) - x_2 + \dot{x}_{1d}$$

$$= \dot{x}_{3d1} - d$$

where $\dot{x}_{3d} = \dot{x}_{3d1} - \dot{x}_{3d2}$, $\dot{x}_{3d1} = \ddot{x}_{1d} - k_1 (x_3 - \dot{x}_{1d}) - k_2 (x_3 - \dot{x}_{2d}) + \dot{x}_{1d}$

and $\dot{x}_{3d2} = \hat{g} + k_1 g + k_2 g$, $\dot{x}_{3d1}$ is composed of known values, and $\dot{x}_{3d2}$ is the unknown part of $\dot{x}_{3d}$, and let $\dot{x}_{3d2} = d$.

To realize $x_4 \rightarrow x_{4d}$, we get a new error

$$e_4 = x_4 - x_{4d}$$
Then

\[ \dot{e}_4 = \dot{x}_4 - \dot{x}_{4d} = f + mu - \dot{x}_{4d1} - \dot{x}_{4d2} \]

where

\[ \dot{x}_{4d} = \dot{x}_{4d1} - \dot{d} - k_3\dot{e}_3 - \dot{e}_2 \]
\[ = \ddot{x}_{1d} - k_1(\dot{x}_3 - \dot{x}_{1d}) - k_2(\dot{x}_3 - \ddot{x}_{3d}) + \ddot{x}_{1d} - \dot{x}_2 - \dot{d} - k_3(x_4 - \dot{x}_{3d1} + \dot{x}_{3d2}) - (\dot{x}_2 - \dot{x}_{2d}) \]
\[ = \ddot{x}_{1d} - k_1(x_4 - \dot{x}_{1d}) - k_2(x_4 - \ddot{x}_{1d} + k_1(\dot{x}_2 - \dot{x}_{1d})) + \ddot{x}_{1d} - x_3 - g - \dot{d} \]
\[ - k_3(x_4 - \dot{x}_{3d1} + \dot{x}_{3d2}) - (x_3 + g - \dot{x}_{2d}) \]
\[ = \ddot{x}_{1d} - k_1(x_4 - \dot{x}_{1d}) - k_2(x_4 - \ddot{x}_{1d} + k_1(\dot{x}_3 - \dot{x}_{1d})) + \ddot{x}_{1d} - x_3 - x_3 - k_3(x_4 - \dot{x}_{3d1}) \]
\[ = \ddot{x}_{1d} - k_1k_2g - g - \dot{d} - k_3\dot{x}_{3d2} - g \]

where \( \dot{x}_{4d} \) is composed of known values \( \dot{x}_{4d1} \) and the unknown part of \( \dot{x}_{4d2} \)

\begin{align*}
\dot{x}_{4d1} &= \ddot{x}_{1d} - k_1(x_4 - \dot{x}_{1d}) - k_2(x_4 - \ddot{x}_{1d} + k_1(\dot{x}_3 - \dot{x}_{1d})) \\
+ &\ddot{x}_{1d} - x_3 - k_3(x_4 - \dot{x}_{3d1}) - (x_3 - \dot{x}_{2d}) \\
\dot{x}_{4d2} &= -k_1k_2g - g - \dot{d} - k_3\dot{x}_{3d2} - g
\end{align*} (8.40)

Define \( \bar{f} = f - \dot{x}_{4d2} \), and then

\[ \dot{e}_4 = \bar{f} - \dot{x}_{4d1} + mu = \bar{f} - \dot{x}_{4d1} + (m - \dot{m})u + \dot{m}u \]

where \( \dot{m} \) is estimation value of \( m \).

To realize \( e_4 \to 0 \) and \( e_1 \to 0 \), \( e_2 \to 0 \), and \( e_3 \to 0 \), design Lyapunov function as

\[ V_4 = V_3 + \frac{1}{2}e_4^2 = \frac{1}{2}(e_1^2 + e_2^2 + e_3^2 + e_4^2) \] (8.41)

Then

\[ \dot{V}_4 = -k_1e_1^2 - k_2e_2^2 + e_3(e_2 + x_4 - \dot{x}_{3d}) + e_2(g(x) - \ddot{g}(x)) \]
\[ + e_4(\bar{f} - \dot{x}_{4d1} + (m - \dot{m})u + \dot{m}u) \]

Since

\[ e_3(e_2 + x_4 - \dot{x}_{3d}) = e_3(e_2 + x_{4d} + e_4 - \dot{x}_{3d}) \]
\[ = e_3(e_2 + \dot{x}_{3d1} - \dot{d} - k_3e_3 - e_2 + e_4 - \dot{x}_{3d1} + d) \]
\[ = e_3e_4 - k_3e_3^2 + e_3(d - \dot{d}) \]
then

\[
\dot{V}_4 = - k_1 e_1^2 - k_2 e_2^2 + e_3 e_4 - k_3 e_3^2 + e_3 (d - \hat{d}) + e_2 (g - \hat{g}) + e_4 \left( \dot{f} - \dot{x}_{4d1} + (m - \hat{m}) u + \dot{m} u \right) \\
= - k_1 e_1^2 - k_2 e_2^2 - k_3 e_3^2 + e_2 (g - \hat{g}) + e_3 (d - \hat{d}) + e_4 (m - \hat{m}) u + e_4 (e_3 + \dot{f} - \dot{x}_{4d1} + \dot{m} u)
\]

To realize \( \dot{V}_4 < 0 \), we choose the control law as

\[
u = \frac{1}{m} \left( -\hat{f} + \dot{x}_{4d1} - k_4 e_4 - e_3 \right) \tag{8.42}
\]

then

\[
\dot{V}_4 = - k_1 e_1^2 - k_2 e_2^2 - k_3 e_3^2 - k_4 e_4^2 + (m - \hat{m}) u e_4 + (g - \hat{g}) e_2 + (d - \hat{d}) e_3 + (\hat{f} - \dot{f}) e_4 \tag{8.43}
\]

If \( \hat{m} = m, \hat{g} = g, \hat{d} = d, \) and \( \hat{f} = \dot{f} \), we can get \( \dot{V}_4 < 0 \)

### 8.5.2 Backstepping Controller Design with RBF Approximation

We use three kind of RBF neural network to approximate \( g, d \) and \( \dot{f} \) respectively as follows

\[
\begin{align*}
g &= W_i^T h_1 + \epsilon_i \\
d &= W_i^T h_2 + \epsilon_2 \\
\dot{f} &= W_i^T h_3 + \epsilon_3
\end{align*}
\]

where \( W_i \) is the ideal neural network weight value, \( h_i \) is the Gaussian function, \( i = 1, 2, 3, \) \( \epsilon_i \) is the approximation error and \( ||\epsilon|| = ||[\epsilon_1 \epsilon_2 \epsilon_3]^T|| < \epsilon_N, ||W||_F \leq W_M \)

Define

\[
\begin{align*}
\hat{g} &= \hat{W}_i^T h_1 \\
\hat{d} &= \hat{W}_i^T h_2 \\
\hat{f} &= \hat{W}_i^T h_3 
\end{align*} \tag{8.44}
\]

where \( \hat{W}_i^T \) is weight value estimation.
Define
\[
Z = \begin{bmatrix} 0 & W_1 & W_2 & W_3 \end{bmatrix}, \quad ||Z||_F \leq Z_M
\]
\[
\dot{Z} = \begin{bmatrix} 0 & \dot{W}_1 & \dot{W}_2 & \dot{W}_3 \end{bmatrix}, \quad \dot{Z} = Z - \dot{Z}
\]

Design Lyapunov function as
\[
V = \frac{1}{2} \xi^T \xi + \frac{1}{2} \text{tr}(Z^T \Gamma^{-1} \ddot{Z}) + \frac{1}{2} \eta m^2 \tag{8.45}
\]
where \(V_4 = \frac{1}{2} \xi^T \xi, \eta > 0, \Gamma\) is a positive-definite matrix with proper dimension,
\[
\Gamma = \begin{bmatrix} 0 & \Gamma_2 \\ \Gamma_3 & \Gamma_4 \end{bmatrix}, \quad \xi = [e_1 \ e_2 \ e_3 \ e_4]^T, \quad \text{and} \quad \dot{m} = m - \ddot{m}.
\]

Let adaptive law as
\[
\dot{Z} = \Gamma h \xi^T - n ||\xi|| \dot{Z} \tag{8.46}
\]
where \(h = [0 \ h_1 \ h_2 \ h_3]^T, \ n\) is a positive number, and \(\dot{m}(0) \geq m\)

From (8.43), (8.44), and (8.45), we have
\[
\dot{V} = \xi^T \dot{\xi} + \text{tr}(Z^T \Gamma^{-1} \ddot{Z}) + \eta \dot{m} \dot{m}
\]
\[
= -k_1 e_1^2 - k_2 e_2^2 - k_3 e_3^2 - k_4 e_4^2 + (\dot{\tilde{W}}_1^T h_1 + \epsilon_1)e_2 + (\dot{\tilde{W}}_2^T h_2 + \epsilon_2)e_3
\]
\[
+ (\dot{\tilde{W}}_3^T h_3 + \epsilon_3)e_4 + \text{tr}(\ddot{Z}^T \Gamma^{-1} \dot{Z}) + \dot{m} e_4 u + \eta \dot{m} \dot{m}
\]

where \(\dot{\tilde{W}}_i^T = W_i^T - \dot{W}_i^T, \quad i = 1, 2, 3\)

Then
\[
\dot{V} = -\xi^T K_e \xi + \xi^T \epsilon + \xi^T \tilde{Z} h + \text{tr}(\ddot{Z}^T \Gamma^{-1} \dot{Z}) + \dot{m} e_4 u + \eta \dot{m} \dot{m}
\]
\[
= -\xi^T K_e \xi + \xi^T \epsilon + \text{tr}(\dot{Z}^T \Gamma^{-1} \dot{Z} + \dot{Z}^T \Gamma^{-1} \dot{Z}) + \dot{m} e_4 u + \eta \dot{m} \dot{m}
\]

where \(K_e = [k_1 \ k_2 \ k_3 \ k_4]^T\), and \(\epsilon = [0 \ \epsilon_1 \ \epsilon_2 \ \epsilon_3]^T\).
Since \( \dot{\mathbf{Z}} = -\mathbf{Z} \), and \( \dot{m} = -\dot{\bar{m}} \), submitting the adaptive law (8.46), we have

\[
\dot{V} = -\xi^T K_\varepsilon \xi + \xi^T \mathbf{e} + n\|\xi\| \text{tr} \left( \mathbf{Z}^T (\mathbf{Z} - \bar{\mathbf{Z}}) \right) + \dot{\bar{m}} (e_4 u - \eta \dot{\bar{m}})
\]  
(8.47)

To guarantee \( \bar{m}(e_4 u - \eta \dot{\bar{m}}) \leq 0 \), at the same time to avoid singularity in (8.42) and guarantee \( \dot{m} \geq m \), we use an adaptive law for \( \dot{m} \) given in [16] as follows:

\[
\dot{m} = \begin{cases} 
\eta^{-1} e_4 u, & \text{if } e_4 u > 0 \\
\eta^{-1} e_4 u, & \text{if } e_4 u \leq 0 \text{ and } \dot{m} > m \\
\eta^{-1}, & \text{if } e_4 u \leq 0 \text{ and } \dot{m} \leq m
\end{cases}
\]  
(8.48)

where \( \dot{m}(0) \geq m \).

The adaptive law (8.48) can be analyzed as:

1. If \( e_4 u > 0 \), we get \( \bar{m}(e_4 u - \eta \dot{m}) = 0 \) and \( \dot{m} > 0 \); thus, \( \dot{m} > m \)
2. If \( e_4 u \leq 0 \) and \( \dot{m} > m \), we get \( \bar{m}(e_4 u - \eta \dot{m}) = 0 \)
3. If \( e_4 u \leq 0 \) and \( \dot{m} \leq m \), we have \( \dot{m} = m - \dot{m} \geq m - m > 0 \); thus, \( \bar{m}(e_4 u - \eta \dot{m}) = \bar{m} e_4 u - \dot{m} \leq 0 \), and \( \dot{m} \) will increase gradually, and then \( \dot{m} > m \) will be guaranteed with \( \dot{m} > 0 \)

According to Schwarz inequality, we have \( \text{tr} \left( \mathbf{Z}^T (\mathbf{Z} - \bar{\mathbf{Z}}) \right) \leq \|\mathbf{Z}\|_F^2 \|\mathbf{Z} - \bar{\mathbf{Z}}\|_F \); since \( K_{\text{min}} \|\xi\|^2 \leq \xi^T K \xi \), \( K_{\text{min}} \) is the minimum eigenvalue of \( K \), (8.47) becomes

\[
\dot{V} \leq -K_{\text{min}} \|\xi\|^2 + \varepsilon_N \|\xi\| + n \|\xi\| \left( \|\mathbf{Z}\|_F^2 - \|\mathbf{Z}\|_F^2 \right) + M
\]

\[
\leq - \|\xi\| \left( K_{\text{min}} \|\xi\| - \varepsilon_N + n \|\mathbf{Z}\|_F (\|\mathbf{Z}\|_F - Z_M) \right)
\]

Since

\[
K_{\text{min}} \|\xi\| - \varepsilon_N + n \left( \|\mathbf{Z}\|_F^2 - \|\mathbf{Z}\|_F Z_M \right)
\]

\[
= K_{\text{min}} \|\xi\| - \varepsilon_N + n \left( \|\mathbf{Z}\|_F^2 - \frac{1}{2} Z_M \right)^2 - \frac{n}{4} Z_M^2
\]

this implies that \( \dot{V} < 0 \) as long as

\[
\|\xi\| > \frac{\varepsilon_N + n Z_M^2}{K_{\text{min}}} \quad \text{or} \quad \|\mathbf{Z}\|_F > \frac{1}{2} Z_M + \sqrt{\frac{Z_M^2}{4} + \frac{\varepsilon_N}{n}}
\]  
(8.49)

From the above expression, we can see that the tracking performance is related to the value of \( \varepsilon_N \), \( n \), and \( K_{\text{min}} \).
8.5.3 Simulation Examples

8.5.3.1 First Example

For the system (8.23), we choose \( f(x) = 0 \) and \( g(x) = 0 \), and \( m = 3.0 \), and then (8.23) can be written as

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= x_3 \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= mu
\end{align*}
\]

We assume \( f(x) = 0 \) and \( g(x) = 0 \) are known and only \( m \) is unknown, and then Lyapunov function becomes \( V = \frac{1}{2} \xi^T \xi + \frac{1}{2} \eta \dot{m}^2 \); thus, only adaptive law (8.48) can be used.

The ideal position signal is \( x_{id} = \sin t \), and the initial value is \( x(0) = [0.5 \ 0 \ 0 \ 0] \). We use the control law (8.42) with adaptive law (8.48). The parameters are chosen as \( k_1 = k_2 = k_3 = k_4 = 35 \). In the adaptive law, we choose \( m = 1.0 \), \( \dot{m}(0) = 500 \), and \( \eta = 150 \). The results are shown from Figs. 8.8, 8.9, and 8.10.

The Simulink program of this example is chap8_3sim.mdl and the Matlab programs of the example are given in the Appendix.
8.5.3.2 Second Example

Consider a dynamic equation of single-link flexible joint robot as

\[
\begin{align*}
I\ddot{q}_1 + MgL \sin q_1 + K(q_1 - q_2) &= 0 \\
J\ddot{q}_2 + K(q_2 - q_1) &= u
\end{align*}
\]

where \( M = 0.2 \) kg, \( L = 0.02 \) m, \( I = 1.35 \times 10^{-3} \) kg \( \cdot \) m\(^2\); \( K = 7.47 \) Nm/rad, and \( J = 2.16 \times 10^{-1} \) kg \( \cdot \) m\(^2\).
The ideal position signal is $x_{1d} = \sin t$, and the initial value is $x(0) = [0, 0, 0, 0]$. Control law (8.42) is used with adaptive law (8.46) and (8.48). The parameters are chosen as $k_1 = k_2 = k_3 = k_4 = 3.5$, $\eta = 0.01$, $\Gamma_2 = \Gamma_3 = \Gamma_4 = 250$, and $\Gamma = \text{diag}\{0, \Gamma_2, \Gamma_3, \Gamma_4\}$. Since $J = 2.16 \times 10^{-1}\text{kg} \cdot \text{m}^2$, we choose $m = 1.0$.

We use three RBFs to approximate $g$, $d$, and $f$, respectively. The structure is used as 4-5-1, and the input vector of RBF is $z = [x_1, x_2, x_3, x_4]^T$. For each Gaussian function, the parameters of $c_i$ and $b_i$ are designed as $[-1, -0.5, 0, 0.5, 1]$ and $1.5$. The initial weight value is chosen as zero.

In the adaptive law (8.48), since there exists strong coupling between $\hat{m}$ and $u$, the choice of initial value of $\hat{m}$ is important. If $\hat{m}(0)$ is chosen as very small, $u$ become very big, and then $\dot{\hat{m}}$ will become very big, which will cause big chattering of $\hat{m}$, and control input will be singular. For the same reason, if $\hat{m}(0)$ is chosen as very big, $u$ will become very small, and then $\dot{\hat{m}}$ will become very small, which will cause small change of $\hat{m}$, and may cause $u$ failure. Therefore, in simulation, we should design initial value of $\hat{m}$ as big as possible. In this simulation, we set $\hat{m}(0) = 500$. In addition, to guarantee not big of $\dot{\hat{m}}$ value, we choose big $\eta$, that is, $\eta = 150$.

The results are shown from Figs. 8.11, 8.12, and 8.13. The estimation of $\hat{g}(x)$ and $\hat{m}$ do not converge to true value of $g(x)$ and $m$. This is due to the fact that the desired trajectory is not persistently exciting and the tracking error performance can be achieved by many possible values of $\hat{g}(x)$ and $\hat{m}$, besides the true $g(x)$ and $m$, which
has been explained in Sect. 1.5, and this occurs quite often in real-world application.

The Simulink program of this example is chap8_4sim.mdl, and the Matlab programs of the example are given in the Appendix.
Appendix

Programs for Sect. 8.2.3

Simulation programs:
  Simulink program: chap8_1sim.mdl

S function for control law: chap8_1ctrl.m

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

```matlab
function [sys,x0,str,ts] = mdlInitializeSizes
global M V x0 fai
sizes = simsizes;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
```
ts = [0 0];
function sys=mdlOutputs(t,x,u)
k1=35;
k2=15;
x1d=u(1);
dx1d=0.1*pi*cos(pi*t);
ddx1d=-0.1*pi^2*sin(pi*t);
x1=u(2);
x2=u(3);
g=9.8;mc=1.0;m=0.1;l=0.5;
S=l*(4/3-m*(cos(x1))^2/(mc+m));
fx=g*sin(x1)-m*l*x2^2*cos(x1)*sin(x1)/(mc+m);
fx=fx/S;
gx=cos(x1)/(mc+m);
gx=gx/S;
e1=x1-x1d;

de1=x2-dx1d;
x2d=dx1d-k1*e1;
dx2d=ddx1d-k1*de1;
e2=x2-x2d;

ut=(1/gx)*(-k2*e2+dx2d-e1-fx);

sys(1)=ut;

S function for plant:chap8_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;

sys(1) = x(2);
sys(2) = fx + gx*u;
function sys = mdlOutputs(t, x, u)
sys(1) = x(1);
sys(2) = x(2);

Plot program: chap8_1plot.m

close all;
figure(1);
subplot(211);
plot(t, y(:, 1), 'r', t, y(:, 3), 'k:', 'linewidth', 2);
xlabel('time(s)'); ylabel('Position tracking');
legend('Ideal position', 'Position tracking');
subplot(212);
plot(t, y(:, 2), 'r', t, y(:, 4), 'k:', 'linewidth', 2);
xlabel('time(s)'); ylabel('Speed tracking');
legend('Ideal speed', 'Speed tracking');
figure(2);
plot(t, ut(:, 1), 'k', 'linewidth', 2);
xlabel('time(s)'); ylabel('Control input');

Programs for Sect. 8.3.4

Simulink program: chap8_2sim.mdl
S function for control law: chap8_2ctrl.m

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

function [sys,x0,str,ts] = mdlInitializeSizes
global c b k1 k2 node
node=5;
sizes = simsizes;
sizes.NumContStates = 2*node;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 5;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0.1*ones(2*node,1)]; %m(0)>ml
str = [];
ts = [];
c=0.5*[-1 -0.5 0 0.5 1;
             -1 -0.5 0 0.5 1];
b=15;
k1=35;k2=35;
function sys=mdlDerivatives(t,x,u)
global c b k1 k2 node
x1d = u(1);
dx1d = 0.1*cos(t);
ddx1d = -0.1*sin(t);
x1 = u(2); x2 = u(3); % Plant states
z = [x1, x2]';
for j = 1:1:node
    h(j) = exp(-norm(z-c(:,j))^2/(2*b^2));
end
e1 = x1 - x1d;
del = x2 - dx1d;
x2d = dx1d - k1*e1;
e2 = x2 - x2d;
dx2d = ddx1d - k1*de1;
Kexi = [e1 e2]';
n = 0.1;
Gama1 = 500; Gama2 = 0.50;
Gama = [Gama1 0;
        0 Gama2];
%dZ = Gama*h*Kexi' - n*Gama*norm(Kexi)*Z;
w_fp = [x(1:node)]'; % fp weight
w_gp = [x(node+1:node*2)]'; % gp weight
for i = 1:1:node
    sys(i) = Gama(1,1)*h(i)*Kexi(1) - n*Gama(1,1)*norm(Kexi)
           *w_fp(i); % f estimation
    sys(i+node) = Gama(2,2)*h(i)*Kexi(2) - n*Gama(2,2)*norm
                 (Kexi)*w_gp(i); % g estimation
end
function sys = mdlOutputs(t, x, u)
global c b k1 k2 node
x1d = u(1);
dx1d = 0.1*cos(t);
ddx1d = -0.1*sin(t);
x1 = u(2); x2 = u(3);
z = [x1, x2]';
for j = 1:1:node
    h(j) = exp(-norm(z-c(:,j))^2/(2*b^2));
end
w_fp = [x(1:node)]'; % fp weight
w_gp = [x(node+1:node*2)]'; % gp weight
fp = w_fp*h';
gp = w_gp*h';
\[e_1 = x_1 - x_{1d};\]
\[de_1 = x_2 - dx_{1d};\]
\[x_{2d} = dx_{1d} - k_1 * e_1;\]
\[e_2 = x_2 - x_{2d};\]
\[dx_{2d} = ddx_{1d} - k_1 * de_1;\]
\[u_t = \frac{1}{(gp + 0.01)} * (-e_1 - f_p + dx_{2d} - k_2 * e_2);\]

\[
\begin{align*}
\text{sys}(1) &= u_t; \\
\text{sys}(2) &= f_p; \\
\text{sys}(3) &= gp;
\end{align*}
\]

S function for plant: chap8_2plant.m

function \[\text{sys}, x_0, \text{str}, ts\] = s_function(t, x, u,\]
\[\text{switch flag,}\]
\[\text{case 0,}\]
\[\quad [\text{sys}, x_0, \text{str}, ts] = \text{mdlInitializeSizes};\]
\[\text{case 1,}\]
\[\quad \text{sys} = \text{mdlDerivatives}(t, x, u);\]
\[\text{case 3,}\]
\[\quad \text{sys} = \text{mdlOutputs}(t, x, u);\]
\[\text{case \{2, 4, 9\}}\]
\[\quad \text{sys} = [];\]
\[\text{otherwise}\]
\[\quad \text{error}([''Unhandled flag = '', num2str(flag)]);\]
\[\text{end}\]

function \[\text{sys}, x_0, \text{str}, ts\] = mdlInitializeSizes

sizes = \text{simsizes};

sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

\[\text{sys} = \text{simsizes}(sizes);\]
\[x_0 = [\pi / 60 0];\]
\[\text{str} = [];\]
\[\text{ts} = [];\]

function sys = mdlDerivatives(t, x, u)

\[g = 9.8; mc = 1.0; m = 0.1; l = 0.5;\]

\[S = 1 * (4/3 - m*(\cos(x(1)))^2/(mc + m));\]

\[fx = g * \sin(x(1)) - m * l * x(2)^2 * \cos(x(1)) * \sin(x(1))/ (mc + m);\]

\[gx = \cos(x(1))/ (mc + m);\]

\[gx = gx / S;\]
sys(1) = x(2);
sys(2) = fx + gx u(1);
function sys = mdlOutputs(t, x, u)
g = 9.8; mc = 1.0; m = 0.1; l = 0.5;
S = 1 * (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);
gx = cos(x(1)) / (mc + m);
g = gx / S;

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

Plot program: chap8_2plot.m

close all;

figure(1);
subplot(211);
plot(t, y(:,1), 'r', t, y(:,3), 'k:', 'linewidth', 2);
xlabel('time(s)'); ylabel('Position tracking');
legend('Ideal position', 'Position tracking');
subplot(212);
plot(t, y(:,2), 'r', t, y(:,4), 'k:', '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');

figure(3);
subplot(211);
plot(t, p(:,3), 'r', t, p(:,6), 'k:', 'linewidth', 2);
xlabel('time(s)'); ylabel('fx and its estimated value');
legend('fx', 'estimated fx');
subplot(212);
plot(t, p(:,4), 'r', t, p(:,7), 'k:', 'linewidth', 2);
xlabel('time(s)'); ylabel('gx and its estimated value');
legend('gx', 'estimated gx');

Programs for Sect. 8.5.3.1

Programs:
Simulink program: chap8_3sim.mdl
S function for control law and adaptive law: chap8_3ctrl.m

```matlab
function [sys,x0,str,ts] = MIMO_Tong_s(t,x,u,flag)
switch flag,
    case 0,
        [sys,x0,str,ts]=mdlInitializeSizes;
    case 1,
        sys=mdlDerivatives(t,x,u);
    case 3,
        sys=mdlOutputs(t,x,u);
    case {2,4,9}
        sys=[];
    otherwise
        error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts] = mdlInitializeSizes
global k1 k2 k3 k4
sizes = simsizes;
sizes.NumContStates = 1;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 6;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [10]; %Let m(0)>>ml
str = [];
ts = [];
k1=35;k2=35;k3=35;k4=35;
function sys=mdlDerivatives(t,x,u)
global k1 k2 k3 k4
x1d=u(1);
dx1d=cos(t);
ddx1d=-sin(t);
```

Appendix 283
dddx1d = -cos(t);
ddddx1d = sin(t);
x1 = u(2); x2 = u(3); x3 = u(4); x4 = u(5); % Plant states
mp = x(1);
e1 = x1 - x1d;
x2d = dx1d - k1*e1;
e2 = x2 - x2d;
dx2d = ddx1d - k1*(x2 - dx1d);
x3d = dx2d - k2*e2 - e1;
dx3d1 = dddx1d - k1*(x3 - ddx1d) - k2*(x3 - dx2d) + dx1d - x2;
e3 = x3 - x3d;
x4d = dx3d1 - k3*e3 - e2;
e4 = x4 - x4d;
dx4d1 = ddddx1d - k1*(x4 - dddx1d) - k2*(x4 - ddx1d) + k1*(x3 - ddx1d) + ddx1d - x3 - k3*(x4 - dx3d1) - (x3 - dx2d);

ut = (1/mp) * (dx4d1 - k4*e4 - e3);
eta = 150;
ml = 1;
if (e4 * ut > 0)
    dm = (1/eta) * e4 * ut;
else
    if (mp > ml)
        dm = (1/eta) * e4 * ut;
    else
        dm = 1/eta;
    end
end
sys(1) = dm;
function sys=mdlOutputs(t,x,u)
global k1 k2 k3 k4
x1d = u(1);
dx1d = cos(t);
ddx1d = -sin(t);
dddx1d = -cos(t);
ddddx1d = sin(t);
x1 = u(2); x2 = u(3); x3 = u(4); x4 = u(5); % Plant states
mp = x(1);
e1 = x1 - x1d;
x2d = dx1d - k1*e1;
e2 = x2 - x2d;
dx2d = ddx1d - k1 * (x2 - dx1d);
x3d = dx2d - k2 * e2 - e1;
dx3d1 = ddx1d - (k1) * (x3 - ddx1d) - (k2) * (x3 - dx2d) + dx1d - x2;

e3 = x3 - x3d;
x4d = dx3d1 - k3 * e3 - e2;
e4 = x4 - x4d;

dx4d1 = ddddx1d - (k1) * (x4 - ddx1d) - (k2) * (x4 - ddx1d) + (k1) * (x3 - ddx1d) - (k3) * (x4 - dx3d1) - (x3 - dx2d) + ddx1d - x3;

ut = (1/mp) * (dx4d1 - k4 * e4 - e3);
sys(1) = ut;
sys(2) = mp;
S function for plant: chap8_3plant.m
function [sys, x0, str, ts] = MIMO_Tong_plant(t, x, u, flag)
switch flag,
case 0,
    [sys, x0, str, ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t, x, u);
case 3,
    sys = mdlOutputs(t, x, u);
case {2, 4, 9}
    sys = [];
otherwise
    error([‘Unhandled flag = ‘, num2str(flag)]);
end
function [sys, x0, str, ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 5;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0.5 0 0 0];
str = [];
ts = [];
function sys = mdlDerivatives(t, x, u)
m = 3.0;
ut = u(1);
sys(1) = x(2);
sys(2) = x(3);
sys(3) = x(4);
sys(4) = m * ut;
function sys=mdlOutputs(t,x,u)
    m=3.0;
    sys(1)=x(1);
    sys(2)=x(2);
    sys(3)=x(3);
    sys(4)=x(4);
    sys(5)=m;
    Plot program:chap8_3plot.m
    close all;
    figure(1);
    subplot(211);
    plot(t,y(:,1),'r',t,y(:,3),'k:','linewidth',2);
    xlabel('time(s)');ylabel('Position tracking');
    legend('Ideal position','Position tracking');
    subplot(212);
    plot(t,y(:,2),'r',t,y(:,4),'k:','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');
    figure(3);
    plot(t,m(:,5),'r',t,m(:,6),'k:','linewidth',2);
    xlabel('time(s)');ylabel('m and its estimated value');
    legend('m','estimated m');

Programs for Sect. 8.5.3.2

Simulink program: chap8_4sim.mdl
S function for control law and adaptive law: chap8_4ctrl.m

function [sys,x0,str,ts] = MIMO_Tong_s(t,x,u,flag)
switch flag,
    case 0,
        [sys,x0,str,ts]=mdlInitializeSizes;
    case 1,
        sys=mdlDerivatives(t,x,u);
    case 3,
        sys=mdlOutputs(t,x,u);
    case {2,4,9}
        sys=[];
    otherwise
        error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts] = mdlInitializeSizes
    global c b k1 k2 k3 k4 node
    node = 5;
    sizes = simsizes;
    sizes.NumContStates = 3*node+1;
    sizes.NumDiscStates = 0;
    sizes.NumOutputs = 3;
    sizes.NumInputs = 7;
    sizes.DirFeedthrough = 1;
    sizes.NumSampleTimes = 0;
    sys = simsizes(sizes);
    x0 = [zeros(3*node,1);500]; %m(0)>ml
    str = [];
    ts = [];
    c=[-1 -0.5 0 0.5 1; ...
       -1 -0.5 0 0.5 1; ...
       -1 -0.5 0 0.5 1];
    b=1.5;
    k1=0.35;k2=0.35;k3=0.35;k4=0.35;
function sys=mdlDerivatives(t,x,u)
global c b k1 k2 k3 k4 node
    x1d=u(1);
    dx1d=cos(t);
    ddx1d=-sin(t);
    dddx1d=-cos(t);
    dddddx1d=sin(t);
    x1=u(2);x2=u(3);x3=u(4);x4=u(5); %Plant states
    z=[x1,x2,x3,x4]’;
    for j=1:1:node
\[ h(j) = \exp\left(-\text{norm}(z-c(:,j))^2/(2*b^2)\right); \]

\[ \text{th_gp} = [\text{x}(1:node)']'; \quad \% \text{gp weight value} \]
\[ \text{th_dp} = [\text{x}(\text{node}+1:node*2)']'; \quad \% \text{dp weight value} \]
\[ \text{th_fp} = [\text{x}(\text{node}*2+1:node*3)']'; \quad \% \text{fp weight value} \]
\[ \text{gp} = \text{th_gp} * h'; \]
\[ \text{dp} = \text{th_dp} * h'; \]
\[ \text{fp} = \text{th_fp} * h'; \]
\[ \text{mp} = \text{x}(3*\text{node}+1); \]
\[ \text{e1} = \text{x1} - \text{x1d}; \]
\[ \text{x2d} = \text{dx1d} - \text{k1}*\text{e1}; \]
\[ \text{e2} = \text{x2} - \text{x2d}; \]
\[ \text{dx2d} = \text{ddx1d} - \text{k1}*(\text{x2} - \text{dx1d}); \]
\[ \text{x3d} = -\text{gp} + \text{dx2d} - \text{k2}^*\text{e2} - \text{e1}; \]
\[ \text{D3} = \text{dddx1d} - \text{k1}^*(\text{x3} - \text{ddx1d}) - \text{k2}^*(\text{x3} - \text{dx2d}) + \text{dx1d} - \text{x2}; \]
\[ \text{dx3d1} = \text{dddx1d} - \text{k1}^*(\text{x3} - \text{ddx1d}) - \text{k2}^*(\text{x3} - \text{dx2d}) + \text{dx1d} - \text{x2}; \]
\[ \text{e3} = \text{x3} - \text{x3d}; \]
\[ \text{x4d} = \text{dx3d1} - \text{dp} - \text{k3}^*\text{e3} - \text{e2}; \]
\[ \text{e4} = \text{x4} - \text{x4d}; \]
\[ \text{D4} = \text{dddddx1d} - \text{k1}^*(\text{x4} - \text{dddx1d}) - \text{k2}^*(\text{x4} - \text{dddx1d} + \text{k1}^*(\text{x3} - \text{ddx1d})) + \text{ddx1d} - \text{x3} - \text{k3}^*(\text{x4} - \text{dx3d1}) - (\text{x3} - \text{dx2d}); \]
\[ \text{ut} = (1/\text{mp})*(\text{-fp} + \text{dx4d1} - \text{k4}^*\text{e4} - \text{e3}); \]
\[ \text{Kexi} = [\text{e1} \text{e2} \text{e3} \text{e4}']'; \]
\[ \text{n} = 0.01; \]
\[ \text{Gama2} = 250; \text{Gama3} = 250; \text{Gama4} = 250; \]
\[ \text{Gama} = [0 \text{Gama2} \text{Gama3} \text{Gama4}]; \]
\[ \text{eta} = 150; \]
\[ \text{ml} = 1; \]
\[ \text{if (e4*ut > 0)} \]
\[ \quad \text{dm} = (1/\text{eta})*\text{e4}^*\text{ut}; \]
\[ \text{end} \]
\[ \text{if (e4*ut <= 0)} \]
\[ \quad \text{if (mp > ml)} \]
\[ \quad \quad \text{dm} = (1/\text{eta})*\text{e4}^*\text{ut}; \]
\[ \quad \text{else} \]
\[ \quad \quad \text{dm} = 1/\text{eta}; \]
\[ \text{end} \]
for i=1:1:node
    sys(i)=Gama(2,2)*h(i)*Kexi(2)-n*Gama(2,2)*norm(Kexi)*th_gp(i); %g estimation
    sys(i+node)=Gama(3,3)*h(i)*Kexi(3)-n*Gama(3,3)*norm(Kexi)*th_dp(i); %d estimation
    sys(i+node*2)=Gama(4,4)*h(i)*Kexi(4)-n*Gama(4,4)*norm(Kexi)*th_fp(i); %f estimation
end
sys(3*node+1)=dm;
function sysmdlOutputs(t,x,u)
global c b k1 k2 k3 k4 node
x1d=u(1);
dx1d=cos(t);
ddx1d=-sin(t);
dddx1d=-cos(t);
ddddx1d=sin(t);
x1=u(2);x2=u(3);x3=u(4);x4=u(5);
z=[x1,x2,x3,x4]’;
for j=1:1:node
    h(j)=exp(-norm(z-c(:,j))^2/(2*b^2));
end
th_gp=[x(1:node)]’; %gp weight
th_dp=[x(node+1:node*2)]’; %dp weight
th_fp=[x(node*2+1:node*3)]’; %fp weight
gp=th_gp*h’;
dp=th_dp*h’;
fp=th_fp*h’;
mp=x(node*3+1);
e1=x1-x1d;
x2d=dx1d-k1*e1;
e2=x2-x2d;
dx2d=ddx1d-k1*(x2-dx1d);
x3d=-gp+dx2d-k2*e2-e1;
%D3=dddx1d-(k1)*(x3-ddx1d)-(k2)*(x3-dx2d)+dx1d-x2;
dx3d1=dddx1d-(k1)*(x3-ddx1d)-(k2)*(x3-dx2d)+dx1d-x2;
e3=x3-x3d;
x4d=dx3d1-dp-k3*e3-e2;
e4=x4-x4d;
\[ d\dd x_1 = -(k_1)(x_4 - \dd x_1) - (k_2)(x_4 - \dd x_1 + k_1)(x_3 - \dd x_1) - (k_3)(x_4 - D_3) - (x_3 - dx_2) + \dd x_1 - x_3; \]
\[ dx_4 = -(k_1)(x_4 - \dd x_1) - (k_2)(x_4 - \dd x_1 + k_1)(x_3 - dx_2) + \dd x_1 - x_3; \]
\[ ut = (1/mp)*(-fp + dx_4 - k_4*e_4 - e_3); \]
\[ ut = u(1); \]
\[ sys(1) = ut; \]
\[ sys(2) = gp; \]
\[ sys(3) = mp; \]

S function for plant: chap8_4plant.m

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

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 6;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
s = simsizes(sizes);
x0 = [0 0 0 0];
str = [];
ts = [];
function sys = mdlDerivatives(t,x,u)
M = 0.2; L = 0.02; I = 1.35*0.001; K = 7.47; J = 2.16*0.1;
g = 9.8;
gx = -x(3) - M*g*L*sin(x(1))/I - (K/I)*(x(1) - x(3));
fx = (K/J)*(x(1) - x(3));
m = 1/J;
ut = u(1);
sys(1) = x(2);
sys(2) = x(3) + gx;
```
sys(3) = x(4);
sys(4) = fx + m*ut;
function sys = mdlOutputs(t, x, u)
M = 0.2; L = 0.02; I = 1.35*0.001; K = 7.47; J = 2.16*0.1;
g = 9.8;
gx = -x(3) - M*g*L*sin(x(1))/I - (K/I)*(x(1) - x(3));
m = 1/J;
sys(1) = x(1);
sys(2) = x(2);
sys(3) = x(3);
sys(4) = x(4);
sys(5) = gx;
sys(6) = m;

Plot program: chap8_4plot.m

close all;
figure(1);
subplot(211);
plot(t, y(:,1), 'r', t, y(:,3), 'k:', 'linewidth', 2);
xlabel('time(s)'); ylabel('Position tracking');
legend('Ideal position', 'Position tracking');
subplot(212);
plot(t, y(:,2), 'r', t, y(:,4), 'k:', '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');

figure(3);
subplot(211);
plot(t, p(:,5), 'r', t, p(:,8), 'k:', 'linewidth', 2);
xlabel('time(s)'); ylabel('gx and its estimated value');
legend('gx', 'estimated gx');
subplot(212);
plot(t, p(:,6), 'r', t, p(:,9), 'k:', 'linewidth', 2);
xlabel('time(s)'); ylabel('m and its estimated value');
legend('m', 'estimated m');

References