Chapter 6
Adaptive RBF Control Based on Global Approximation

Abstract This chapter introduces three kinds of adaptive neural mode control laws for n-link manipulators based on RBF, including adaptive neural network control law, adaptive neural network control law with sliding mode robust term, and adaptive neural network control law with HJI. The closed-loop system stability can be achieved based on the Lyapunov stability.

Keywords RBF neural network • Adaptive control • Global approximation • Robotic manipulators

Recently, increasing attention has been paid to the use of neural networks in robot control. Previous use of neural networks to improve robot path following performance has concentrated on replacing either the entire control system or the feed-forward controller and/or prefilter with neural networks [1, 2]. Such research work was based on the desire to obtain the benefits of model-based control without a priori knowledge of system dynamics or without the computational burden of classical dynamic equations.

However, in many cases, the approximate dynamic model of the robot manipulators can be found a priori. Actually, the feasibility of model-based control has been demonstrated [3, 4]. Therefore, a priori knowledge of the robot dynamic models should be appropriately used rather than totally discarded. Recently, a direct adaptive control algorithm using neural networks was proposed in [5]. Their method is based on the BP feed-forward networks and the reinforcement learning.

In this chapter, we introduced three typical examples of neural network controller design, analysis, and simulation.

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_6,

6.1 Adaptive Control with RBF Neural Network Compensation for Robotic Manipulators

In this section, in reference to paper [6], a robot tracking control scheme is introduced. This scheme takes advantage of the computed torque methods and incorporates a compensating controller to achieve high tracking performance. The compensating controller is based on a radial basis function (RBF) neural network, which is trained on-line to identify the robot modeling error. A main feature of the proposed scheme is that the resulting closed-loop control system is guaranteed to be stable. Another important property of the proposed compensating controller is that the neural network-based correcting controller is an add-on device which can be added on many existing robot control systems.

6.1.1 Problem Description

Consider dynamic equation of $n-link$ manipulator as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{d} \tag{6.1}$$

where $\boldsymbol{M}(\boldsymbol{q})$ is an $n \times n$ inertia matrix, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ is an $n \times n$ matrix containing the centrifugal and Coriolis terms, $\boldsymbol{G}(\boldsymbol{q})$ is an $n \times 1$ vector containing gravitational forces and torques, $\boldsymbol{q}$ is generalized joint coordinates, τ is joint torques, and $\boldsymbol{d}$ denotes disturbances.

However, in practice, the perfect robot model could be difficult to obtain, and external disturbances are always present in practice. Usually, only a nominal model of the robot could be obtained. It is supposed that the nominal model of the robot is denoted by $\boldsymbol{M}_0(\boldsymbol{q}), \boldsymbol{C}_0(\boldsymbol{q},\dot{\boldsymbol{q}}), \boldsymbol{G}_0(\boldsymbol{q})$, and we assume $\Delta\boldsymbol{M} = \boldsymbol{M}_0 - \boldsymbol{M}$, $\Delta\boldsymbol{C} = \boldsymbol{C}_0 - \boldsymbol{C}$, $\Delta\boldsymbol{G} = \boldsymbol{G}_0 - \boldsymbol{G}$, then from (6.1), we have

$$(\boldsymbol{M}_0(\boldsymbol{q}) - \Delta\boldsymbol{M})\ddot{\boldsymbol{q}} + (\boldsymbol{C}_0(\boldsymbol{q},\dot{\boldsymbol{q}}) - \Delta\boldsymbol{C})\dot{\boldsymbol{q}} + \boldsymbol{G}_0(\boldsymbol{q}) - \Delta\boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{d}$$

Hence,

$$\boldsymbol{M}_0(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}_0(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}_0(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$$

where $\boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) = \Delta\boldsymbol{M}\ddot{\boldsymbol{q}} + \Delta\boldsymbol{C}\dot{\boldsymbol{q}} + \Delta\boldsymbol{G} + \boldsymbol{d}$.

Therefore, if the nominal model is used for the design of the computed torque controller, and if $\boldsymbol{f}(\cdot)$ is known, we can design control law as

$$\boldsymbol{\tau} = \boldsymbol{M}_0(\boldsymbol{q})(\ddot{\boldsymbol{q}}_\mathrm{d} - \boldsymbol{k}_\mathrm{v}\dot{\boldsymbol{e}} - \boldsymbol{k}_\mathrm{p}\boldsymbol{e}) + \boldsymbol{C}_0(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}_0(\boldsymbol{q}) - \boldsymbol{f}(\cdot) \tag{6.2}$$

where $\boldsymbol{k}_{\mathrm{p}} = \begin{bmatrix} \alpha^2 & 0 \\ 0 & \alpha^2 \end{bmatrix}$, $\boldsymbol{k}_{\mathrm{v}} = \begin{bmatrix} 2\alpha & 0 \\ 0 & 2\alpha \end{bmatrix}$, $\alpha > 0$.

Submitting the term (6.2) into (6.1), we can get the closed loop system as

$$\ddot{\boldsymbol{e}} + \boldsymbol{k}_{\mathrm{v}}\dot{\boldsymbol{e}} + \boldsymbol{k}_{\mathrm{p}}\boldsymbol{e} = 0 \tag{6.3}$$

where $\boldsymbol{q}_{\mathrm{d}}$ is ideal angle, $\boldsymbol{e} = \boldsymbol{q} - \boldsymbol{q}_{\mathrm{d}}$, $\dot{\boldsymbol{e}} = \dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{\mathrm{d}}$.

The goal is to design a stable robust controller based on nominal modeling information.

In practical engineering, the term $\boldsymbol{f}(\cdot)$ is always unknown. So we must estimate $\boldsymbol{f}(\cdot)$ and compensate it. In this section, we use RBF to approximate $\boldsymbol{f}(\cdot)$ and compensate it.

6.1.2 RBF Approximation

The algorithm of RBF is

$$h_i = g\left(\|\boldsymbol{x} - \boldsymbol{c}_i\|^2 / b_i^2\right), \quad i = 1, 2 \ldots, n \tag{6.4}$$

$$\boldsymbol{y} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) \tag{6.5}$$

where $\boldsymbol{x}$ is input vector, $\boldsymbol{y}$ is output, $\boldsymbol{h} = [h_1, h_2, \ldots, h_n]^{\mathrm{T}}$ is the output of Gaussian function, and $\boldsymbol{w}$ is neural network weight value.

Given a very small positive constant ε_0 and a continuous function $\boldsymbol{f}(\cdot)$, there exists a weight vector $\boldsymbol{w}^*$ such that the output $\hat{\boldsymbol{f}}(\cdot)$ of RBF satisfies

$$\max \left\| \boldsymbol{f}(\cdot) - \hat{\boldsymbol{f}}^*(\cdot) \right\| \leq \varepsilon_0 \tag{6.6}$$

where $\boldsymbol{w}^* = \arg \min_{\theta \in \beta(M_\theta)} \left\{ \sup_{x \in \phi(M_x)} \left\| \boldsymbol{f}(\cdot) - \hat{\boldsymbol{f}}(\cdot) \right\| \right\}$, $\boldsymbol{w}^*$ is $n \times n$ matrix, denote the optimal weight values for $\boldsymbol{f}(\cdot)$ approximation.

Define the approximation error as

$$\boldsymbol{\eta} = \boldsymbol{f}(\cdot) - \hat{\boldsymbol{f}}^*(\cdot) \tag{6.7}$$

Assume the modeling error $\boldsymbol{\eta}$ is bounded by a finite constant as

$$\boldsymbol{\eta}_0 = \sup \left\| \boldsymbol{f}(\cdot) - \hat{\boldsymbol{f}}^*(\cdot) \right\| \tag{6.8}$$

where $\hat{\boldsymbol{f}}^*(\cdot) = \boldsymbol{w}^{*\mathrm{T}}\boldsymbol{h}(\boldsymbol{x})$.

6.1.3 RBF Controller and Adaptive Law Design and Analysis

For the system (6.1), the following controller was proposed as [6]

$$\boldsymbol{\tau} = \boldsymbol{M}_0(\boldsymbol{q})(\ddot{\boldsymbol{q}}_\mathrm{d} - \boldsymbol{k}_\mathrm{v}\dot{\boldsymbol{e}} - \boldsymbol{k}_\mathrm{p}\boldsymbol{e}) + \boldsymbol{C}_0(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}_0(\boldsymbol{q}) - \hat{\boldsymbol{f}}(\cdot) \tag{6.9}$$

where $\hat{\boldsymbol{w}}$ is the estimation value of $\boldsymbol{w}^*$, $\|\boldsymbol{w}^*\|_\mathrm{F} \le \boldsymbol{w}_{\max}$, and $\hat{\boldsymbol{f}}(\cdot) = \hat{\boldsymbol{w}}^\mathrm{T}\boldsymbol{h}(\boldsymbol{x})$.

Submitting (6.9) into (6.1), we have

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{M}_0(\boldsymbol{q})(\ddot{\boldsymbol{q}}_\mathrm{d} - \boldsymbol{k}_\mathrm{v}\dot{\boldsymbol{e}} - \boldsymbol{k}_\mathrm{p}\boldsymbol{e}) + \boldsymbol{C}_0(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}_0(\boldsymbol{q}) - \hat{\boldsymbol{f}}(\cdot) + \boldsymbol{d}$$

Subtracting the right and left sides of above equation with $\boldsymbol{M}_0(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}_0(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}_0(\boldsymbol{q})$, we have

$$\begin{aligned}
&\Delta\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \Delta\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \Delta\boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{d} \\
&\quad = \boldsymbol{M}_0(\boldsymbol{q})\ddot{\boldsymbol{q}} - \boldsymbol{M}_0(\boldsymbol{q})(\ddot{\boldsymbol{q}}_\mathrm{d} - \boldsymbol{k}_\mathrm{v}\dot{\boldsymbol{e}} - \boldsymbol{k}_\mathrm{p}\boldsymbol{e}) + \hat{\boldsymbol{f}}(\cdot) \\
&\quad = \boldsymbol{M}_0(\boldsymbol{q})(\ddot{\boldsymbol{e}} + \boldsymbol{k}_\mathrm{v}\dot{\boldsymbol{e}} + \boldsymbol{k}_\mathrm{p}\boldsymbol{e}) + \hat{\boldsymbol{f}}(\cdot)
\end{aligned}$$

Thus,

$$\ddot{\boldsymbol{e}} + \boldsymbol{k}_\mathrm{v}\dot{\boldsymbol{e}} + \boldsymbol{k}_\mathrm{p}\boldsymbol{e} + \boldsymbol{M}_0{}^{-1}(\boldsymbol{q})\hat{\boldsymbol{f}}(\cdot) = \boldsymbol{M}_0{}^{-1}(\boldsymbol{q})(\Delta\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \Delta\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \Delta\boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{d})$$

Then, we can get

$$\ddot{\boldsymbol{e}} + \boldsymbol{k}_\mathrm{v}\dot{\boldsymbol{e}} + \boldsymbol{k}_\mathrm{p}\boldsymbol{e} = \boldsymbol{M}_0{}^{-1}(\boldsymbol{q})\big(\boldsymbol{f}(\cdot) - \hat{\boldsymbol{f}}(\cdot)\big)$$

Choose $\boldsymbol{x} = (\,\boldsymbol{e} \quad \dot{\boldsymbol{e}}\,)^\mathrm{T}$, then, above equation becomes

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\big\{\boldsymbol{f}(\cdot) - \hat{\boldsymbol{f}}(\cdot)\big\}$$

where $\boldsymbol{A} = \begin{pmatrix} 0 & \boldsymbol{I} \\ -\boldsymbol{k}_\mathrm{p} & -\boldsymbol{k}_\mathrm{v} \end{pmatrix}$, $\boldsymbol{B} = \begin{pmatrix} 0 \\ \boldsymbol{M}_0{}^{-1}(\boldsymbol{q}) \end{pmatrix}$.

Since

$$\boldsymbol{f}(\cdot) - \hat{\boldsymbol{f}}(\cdot) = \boldsymbol{f}(\cdot) - \hat{\boldsymbol{f}}^*(\cdot) + \hat{\boldsymbol{f}}^*(\cdot) - \hat{\boldsymbol{f}}(\cdot) = \boldsymbol{\eta} + \boldsymbol{w}^{*\mathrm{T}}\boldsymbol{h} - \hat{\boldsymbol{w}}^\mathrm{T}\boldsymbol{h} = \boldsymbol{\eta} - \tilde{\boldsymbol{w}}^\mathrm{T}\boldsymbol{h}$$

where $\tilde{\boldsymbol{w}} = \hat{\boldsymbol{w}} - \boldsymbol{w}^*$, $\boldsymbol{\eta} = \boldsymbol{f}(\cdot) - \hat{\boldsymbol{f}}^*(\cdot)$,

then,

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\big(\boldsymbol{\eta} - \tilde{\boldsymbol{w}}^\mathrm{T}\boldsymbol{h}\big)$$

In paper [6], the stability of the closed control system with the controller (6.9) was given by defining Lyapunov function as

$$V = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x} + \frac{1}{2\gamma}\|\tilde{\boldsymbol{w}}\|^2$$

where $\gamma > 0$.

The matrix $\boldsymbol{P}$ is symmetric and positive definite and satisfies the following Lyapunov equation

$$\boldsymbol{P}\boldsymbol{A} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{P} = -\boldsymbol{Q} \tag{6.10}$$

where $\boldsymbol{Q} \geq 0$.

Define

$$\|\boldsymbol{R}\|^2 = \sum_{i,j} \left|r_{ij}\right|^2 = \mathrm{tr}(\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}) = \mathrm{tr}(\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R})$$

where $\mathrm{tr}(\cdot)$ is the trace of matrix; then,

$$\|\tilde{\boldsymbol{w}}\|^2 = \mathrm{tr}\left(\tilde{\boldsymbol{w}}^{\mathrm{T}}\tilde{\boldsymbol{w}}\right)$$

The derivative of $\boldsymbol{V}$ is

$$\begin{aligned}
\dot{V} &= \frac{1}{2}\left[\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\dot{\boldsymbol{x}} + \dot{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}\right] + \frac{1}{\gamma}\mathrm{tr}\left(\dot{\tilde{\boldsymbol{w}}}^{\mathrm{T}}\tilde{\boldsymbol{w}}\right) \\
&= \frac{1}{2}\left[\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\left(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}(\boldsymbol{\eta} - \tilde{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{h})\right) + \left(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}} + (\boldsymbol{\eta} - \tilde{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{h})^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\right)\boldsymbol{P}\boldsymbol{x}\right] + \frac{1}{\gamma}\mathrm{tr}\left(\dot{\tilde{\boldsymbol{w}}}^{\mathrm{T}}\tilde{\boldsymbol{w}}\right) \\
&= \frac{1}{2}\left[\boldsymbol{x}^{\mathrm{T}}\left(\boldsymbol{P}\boldsymbol{A} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}\right)\boldsymbol{x} + \left(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\boldsymbol{\eta} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\tilde{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{h} + \boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x} - \boldsymbol{h}^{\mathrm{T}}\tilde{\boldsymbol{w}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}\right)\right] + \frac{1}{\gamma}\mathrm{tr}\left(\dot{\tilde{\boldsymbol{w}}}^{\mathrm{T}}\tilde{\boldsymbol{w}}\right) \\
&= -\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x} - \boldsymbol{h}^{\mathrm{T}}\tilde{\boldsymbol{w}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x} + \frac{1}{\gamma}\mathrm{tr}\left(\dot{\tilde{\boldsymbol{w}}}^{\mathrm{T}}\tilde{\boldsymbol{w}}\right)
\end{aligned}$$

where $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\tilde{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{h} = \boldsymbol{h}^{\mathrm{T}}\tilde{\boldsymbol{w}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}$, $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}\eta = \boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}$.

Since

$$\boldsymbol{h}^{\mathrm{T}}\tilde{\boldsymbol{w}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x} = \mathrm{tr}\left[\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}\boldsymbol{h}^{\mathrm{T}}\tilde{\boldsymbol{w}}\right]$$

then,

$$\dot{V} = -\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + \frac{1}{\gamma}\mathrm{tr}\left(-\gamma\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}\boldsymbol{h}^{\mathrm{T}}\tilde{\boldsymbol{w}} + \dot{\tilde{\boldsymbol{w}}}^{\mathrm{T}}\tilde{\boldsymbol{w}}\right) + \boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x} \tag{6.11}$$

In reference to the adaptive law proposed in [6], we design two adaptive laws as follows:

1. First adaptive law

Choose adaptive law as

$$\dot{\hat{\boldsymbol{w}}}^{\mathrm{T}} = \gamma \boldsymbol{B}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{x} \boldsymbol{h}^{\mathrm{T}}$$

Then

$$\dot{\hat{\boldsymbol{w}}} = \gamma \boldsymbol{h} \boldsymbol{x}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{B} \tag{6.12}$$

Since $\dot{\tilde{\boldsymbol{w}}} = \dot{\hat{\boldsymbol{w}}}$, then submitting (6.12) into (6.11), we have

$$\dot{V} = -\frac{1}{2} \boldsymbol{x}^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{\eta}^{\mathrm{T}} \boldsymbol{B}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{x}$$

From known, we have

$$\begin{aligned} &\left\| \boldsymbol{\eta}^{\mathrm{T}} \right\| \leq \| \boldsymbol{\eta}_0 \|, \ \| \boldsymbol{B} \| = \left\| \boldsymbol{M}_0{}^{-1}(\boldsymbol{q}) \right\| \\ &\dot{V} \leq -\frac{1}{2} \lambda_{\min}(\boldsymbol{Q}) \| \boldsymbol{x} \|^2 + \| \boldsymbol{\eta}_0 \| \ \left\| \boldsymbol{M}_0{}^{-1}(\boldsymbol{q}) \right\| \lambda_{\max}(\boldsymbol{P}) \| \boldsymbol{x} \| \\ &\quad = -\frac{1}{2} \| \boldsymbol{x} \| \left[\lambda_{\min}(\boldsymbol{Q}) \| \boldsymbol{x} \| - 2 \| \boldsymbol{\eta}_0 \| \ \left\| \boldsymbol{M}_0{}^{-1}(\boldsymbol{q}) \right\| \lambda_{\max}(\boldsymbol{P}) \right] \end{aligned}$$

where $\lambda_{\max}(\boldsymbol{P})$ and $\lambda_{\min}(\boldsymbol{Q})$ denote the maximum eigenvalue of matrix $\boldsymbol{P}$ and the minimum eigenvalue of matrix $\boldsymbol{Q}$ respectively.

To satisfy $\dot{V} \leq 0$, $\lambda_{\min}(\boldsymbol{Q}) \geq \frac{2 \| \boldsymbol{M}_0{}^{-1}(\boldsymbol{q}) \| \lambda_{\max}(\boldsymbol{P})}{\| \boldsymbol{x} \|} \| \boldsymbol{\eta}_0 \|$ should be satisfied, that is,

$$\| \boldsymbol{x} \| = \frac{2 \left\| \boldsymbol{M}_0{}^{-1}(\boldsymbol{q}) \right\| \lambda_{\max}(\boldsymbol{P})}{\lambda_{\min}(\boldsymbol{Q})} \| \boldsymbol{\eta}_0 \| \tag{6.13}$$

From (6.12), we can get a conclusion: the bigger value the eigenvalue of $\boldsymbol{Q}$ is, or the smaller value the eigenvalue of $\boldsymbol{P}$ is, or the smaller value of $\boldsymbol{\eta}_0$ is, the smaller of $\boldsymbol{x}$ radius convergence is.

The shortcoming of the first adaptive is the boundedness of $\tilde{\boldsymbol{w}} = \hat{\boldsymbol{w}} - \boldsymbol{w}^*$ can be guaranteed.

2. Second adaptive law

Choose adaptive law as

$$\dot{\hat{\boldsymbol{w}}}^{\mathrm{T}} = \gamma \boldsymbol{B}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{x} \boldsymbol{h}^{\mathrm{T}} + k_1 \gamma \| \boldsymbol{x} \| \hat{\boldsymbol{w}}^{\mathrm{T}}$$

then,

$$\dot{\hat{\boldsymbol{w}}} = \gamma \boldsymbol{h}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B} + k_1\gamma\|\boldsymbol{x}\|\hat{\boldsymbol{w}} \tag{6.14}$$

where $k_1 > 0$.

Submitting (6.14) into (6.11), we have

$$\begin{aligned}\dot{V} &= -\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + \frac{1}{\gamma}\mathrm{tr}\left(k_1\gamma\|\boldsymbol{x}\|\hat{\boldsymbol{w}}^{\mathrm{T}}\tilde{\boldsymbol{w}}\right) + \boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x} \\ &= -\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + k_1\|\boldsymbol{x}\|\mathrm{tr}\left(\hat{\boldsymbol{w}}^{\mathrm{T}}\tilde{\boldsymbol{w}}\right) + \boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}\end{aligned}$$

According to the characteristics of norm F, we have $\mathrm{tr}[\tilde{\boldsymbol{x}}^{\mathrm{T}}(\boldsymbol{x}-\tilde{\boldsymbol{x}})] \le ||\tilde{\boldsymbol{x}}||_{\mathrm{F}}||\boldsymbol{x}||_{\mathrm{F}} - ||\tilde{\boldsymbol{x}}||_{\mathrm{F}}^2$, then,

$$\mathrm{tr}\left[\hat{\boldsymbol{w}}^{\mathrm{T}}\tilde{\boldsymbol{w}}\right] = \mathrm{tr}\left[\tilde{\boldsymbol{w}}^{\mathrm{T}}\hat{\boldsymbol{w}}\right] = \mathrm{tr}\left[\tilde{\boldsymbol{w}}^{\mathrm{T}}(\boldsymbol{w}^* + \tilde{\boldsymbol{w}})\right] \le \|\tilde{\boldsymbol{w}}\|_{\mathrm{F}}\|\boldsymbol{w}^*\|_{\mathrm{F}} - \|\tilde{\boldsymbol{w}}\|_{\mathrm{F}}^2$$

Since

$$-k_1\|\tilde{\boldsymbol{w}}\|_{\mathrm{F}}w_{\max} + k_1\|\tilde{\boldsymbol{w}}\|_{\mathrm{F}}^2 = k_1\left(\|\tilde{\boldsymbol{w}}\|_{\mathrm{F}} - \frac{w_{\max}}{2}\right)^2 - \frac{k_1}{4}w_{\max}^2$$

then

$$\begin{aligned}\dot{V} &\le -\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + k_1\|\boldsymbol{x}\|\left(\|\tilde{\boldsymbol{w}}\|_{\mathrm{F}}\|\boldsymbol{w}^*\|_{\mathrm{F}} - \|\tilde{\boldsymbol{w}}\|_{\mathrm{F}}^2\right) + \boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x} \\ &\le -\frac{1}{2}\lambda_{\min}(\boldsymbol{Q})\|\boldsymbol{x}\|^2 + k_1\|\boldsymbol{x}\|\ \|\tilde{\boldsymbol{w}}\|_{\mathrm{F}}\|\boldsymbol{w}^*\|_{\mathrm{F}} - k_1\|\boldsymbol{x}\|\ \|\tilde{\boldsymbol{w}}\|_{\mathrm{F}}^2 + \|\boldsymbol{\eta}_0\|\lambda_{\max}(\boldsymbol{P})\|\boldsymbol{x}\| \\ &\le -\|\boldsymbol{x}\|\left(\frac{1}{2}\lambda_{\min}(\boldsymbol{Q})\|\boldsymbol{x}\| - k_1\|\tilde{\boldsymbol{w}}\|_{\mathrm{F}}w_{\max} + k_1\|\tilde{\boldsymbol{w}}\|_{\mathrm{F}}^2 - \|\boldsymbol{\eta}_0\|\lambda_{\max}(\boldsymbol{P})\right) \\ &= -\|\boldsymbol{x}\|\left(\frac{1}{2}\lambda_{\min}(\boldsymbol{Q})\|\boldsymbol{x}\| + k_1\left(\|\tilde{\boldsymbol{w}}\|_F - \frac{w_{\max}}{2}\right)^2 - \frac{k_1}{4}w_{\max}^2 - \|\boldsymbol{\eta}_0\|\lambda_{\max}(\boldsymbol{P})\right)\end{aligned}$$

To guarantee $\dot{V} \le 0$, the following conditions must be satisfied:

$$\frac{1}{2}\lambda_{\min}(\boldsymbol{Q})\|\boldsymbol{x}\| \ge \|\boldsymbol{\eta}_0\|\lambda_{\max}(\boldsymbol{P}) + \frac{k_1}{4}w_{\max}^2$$

or

$$\boldsymbol{k}_1\left(\|\tilde{\boldsymbol{w}}\|_F - \frac{w_{\max}}{2}\right)^2 \ge \|\boldsymbol{\eta}_0\|\lambda_{\max}(\boldsymbol{P}) + \frac{k_1}{4}w_{\max}^2$$

Then we get the boundedness of the closed-loop system as

$$\|\boldsymbol{x}\| \geq \frac{2}{\lambda_{\min}(\boldsymbol{Q})}\left(\|\boldsymbol{\eta}_0\|\lambda_{\max}(\boldsymbol{P}) + \frac{k_1}{4}w_{\max}^2\right) \tag{6.15}$$

or

$$\|\tilde{\boldsymbol{w}}\|_{\mathrm{F}} \geq \frac{w_{\max}}{2} + \sqrt{\frac{1}{k_1}\left(\|\boldsymbol{\eta}_0\|\lambda_{\max}(\boldsymbol{P}) + \frac{k_1}{4}w_{\max}^2\right)}$$

From (6.15), we can get a conclusion: the bigger value the eigenvalue of $\boldsymbol{Q}$ is, or the smaller value the eigenvalue of $\boldsymbol{P}$ is, or the smaller value of $\boldsymbol{\eta}_0$ is, or smaller value $w_{\max}$ is, the smaller radius of $\boldsymbol{x}$ convergence is.

The shortcoming of the controller is that nominal model of the robotic manipulators must be known.

6.1.4 Simulation Examples

6.1.4.1 First Example

Consider a simple servo system as

$$M\ddot{q} = \boldsymbol{\tau} + d(\dot{q})$$

where $M = 10$, $d(\dot{q})$ denotes friction force, and $d(\dot{q}) = -15\dot{q} - 30\mathrm{sgn}(\dot{q})$.

The desired trajectory is $q_{\mathrm{d}} = \sin t$, the initial value of the plant is $[0.6 \quad 0]^{\mathrm{T}}$. In simulation,we use control law (6.9) and first adaptive law (6.12). The parameters are chosen as $\boldsymbol{Q} = \begin{bmatrix} 50 & 0 \\ 0 & 50 \end{bmatrix}$, $\alpha = 3$, $\gamma = 200$, and $k_1 = 0.001$.

For RBF neural network, the parameters of Gaussian function $\boldsymbol{c}_{\mathrm{i}}$ and b_i are chosen as $[-2 \quad -1 \quad 0 \quad 1 \quad 2]$ and 5, and the initial weight value is chosen as zero.

In the simulation, $S1 = 1$ denotes first adaptive law, and $S1 = 2$ denotes second adaptive law. $S = 1$ denotes controller only based on nominal model, $S = 2$ denotes controller based on precise compensation, and $S = 3$ denotes controller based on RBF compensation.

In the simulation, we choose $S1 = 1$ and $S = 3$. To test the effect of hidden nets number on the approximation precision, we use 5 hidden nets and 19 hidden nets, respectively; the simulation results are shown from Figs. 6.1, 6.2, 6.3, and 6.4. From the results, it is shown that the more the hidden nets is chosen, the smaller approximation error can be gotten.

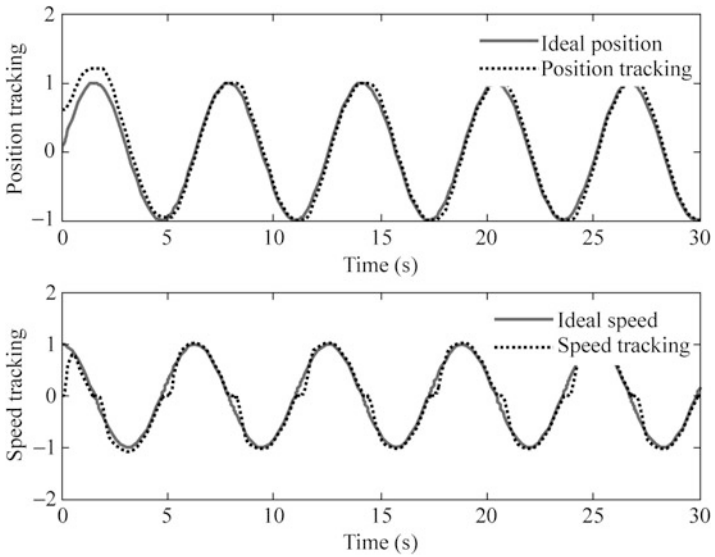


Fig. 6.1 Position and speed tracking with RBF compensation (five hidden nets, $S = 3$)

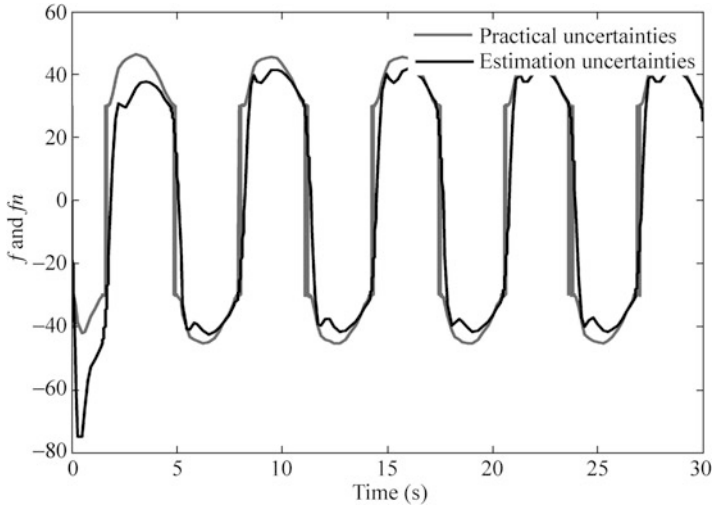


Fig. 6.2 $f(x)$ estimation with RBF (five hidden nets, $S = 3$)

Only choosing $S = 1$, the position and speed tracking without compensation is shown in Fig. 6.5.

The Simulink program of this example is chap6_1sim.mdl; the Matlab programs of the example are given in the Appendix.

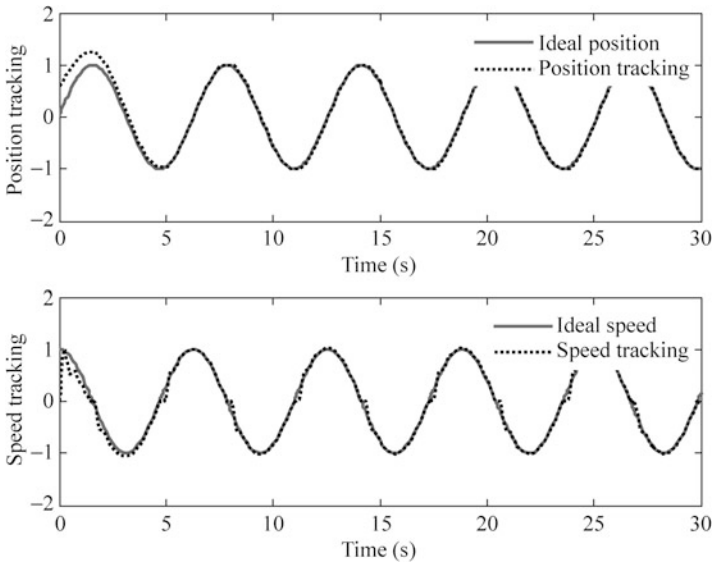


Fig. 6.3 Position and speed tracking with RBF compensation (19 hidden nets, $S = 3$)

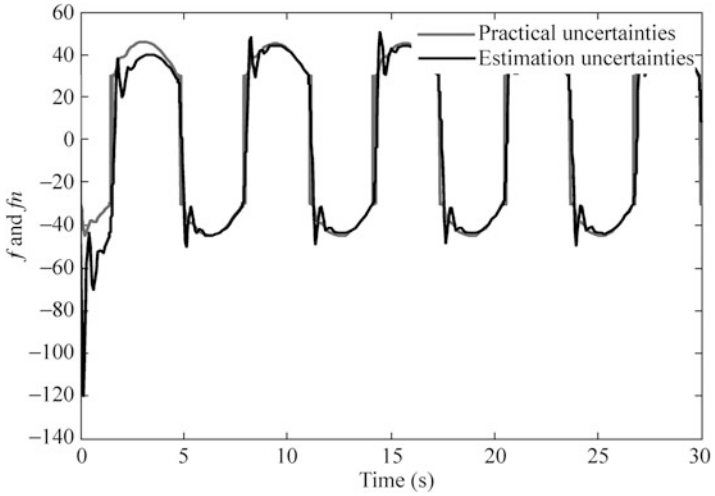


Fig. 6.4 $\boldsymbol{f}(\boldsymbol{x})$ estimation with RBF (19 hidden nets, $S = 3$)

6.1.4.2 Second Example

Consider a two-link manipulator dynamic equation as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{d}$$

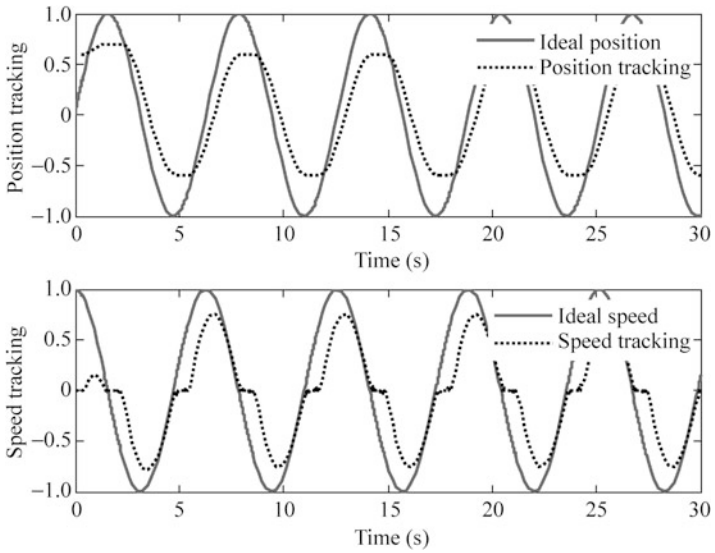


Fig. 6.5 Position and speed tracking without compensation ($S = 1$)

where

$$\boldsymbol{M}(\boldsymbol{q}) = \begin{bmatrix} v + q_{01} + 2\gamma\cos(q_2) & q_{01} + q_{02}\cos(q_2) \\ q_{01} + q_{02}\cos(q_2) & q_{01} \end{bmatrix}$$

$$\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \begin{bmatrix} -q_{02}\dot{q}_2\sin(q_2) & -q_{02}(\dot{q}_1 + \dot{q}_2)\sin(q_2) \\ q_{02}\dot{q}_1\sin(q_2) & 0 \end{bmatrix}$$

$$\boldsymbol{G}(\boldsymbol{q}) = \begin{bmatrix} 15g\cos q_1 + 8.75g\cos(q_1 + q_2) \\ 8.75g\cos(q_1 + q_2) \end{bmatrix}$$

and $v = 13.33,\ q_{01} = 8.98,\ q_{02} = 8.75,$ and $g = 9.8$.

The disturbance is $d = d_1 + d_2||\boldsymbol{e}|| + d_3||\dot{\boldsymbol{e}}||,\ d_1 = 2,\ d_2 = 3,$ and $d_3 = 6$. The desired joint trajectory is

$$\begin{cases} q_{1\mathrm{d}} = 1 + 0.2\sin(0.5\pi t) \\ q_{2\mathrm{d}} = 1 - 0.2\cos(0.5\pi t) \end{cases}$$

The initial value of the plant is $[q_1 \quad q_2 \quad q_3 \quad q_4]^{\mathrm{T}} = [0.6 \quad 0.3 \quad 0.5 \quad 0.5]^{\mathrm{T}}$, and assume $\Delta\boldsymbol{M} = 0.2\boldsymbol{M},\ \Delta\boldsymbol{C} = 0.2\boldsymbol{C},$ and $\Delta\boldsymbol{G} = 0.2\boldsymbol{G}$.

In simulation, we use control law (6.9) and first adaptive law (6.12), in the program, we set $S = 3,$ and $S_1 = 1$. The parameters are chosen as

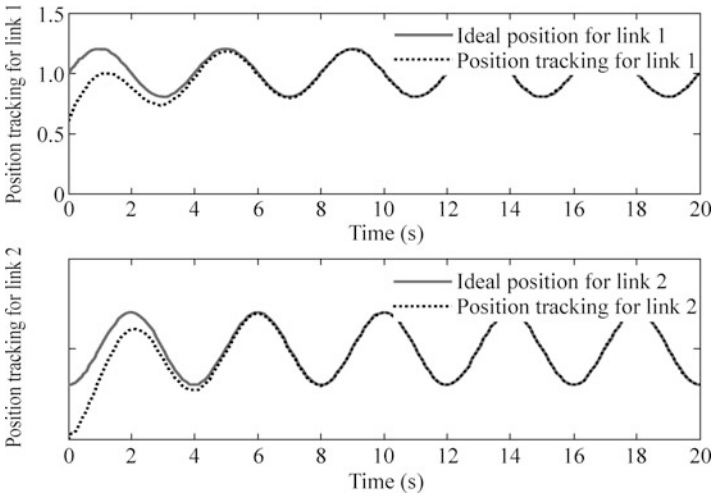


Fig. 6.6 Position tracking of link 1 and link 2

$$\boldsymbol{Q} = \begin{bmatrix} 50 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix},$$

$\alpha = 3,\ \gamma = 20,\ k_1 = 0.001.$

For RBF neural network, the parameters of Gaussian function $\boldsymbol{c}_i$ and b_i are chosen as $[-2 \quad -1 \quad 0 \quad 1 \quad 2]$ and 3.0, and the initial weight value is chosen as 0.10. In the simulation, $S_1 = 1$ denotes first adaptive law, and $S_1 = 2$ denotes second adaptive law. $S = 1$ denotes controller only based on nominal model, $S = 2$ denotes controller based on precise compensation, and $S = 3$ denotes controller based on RBF compensation. The simulation results are shown from Figs. 6.6, 6.7, 6.8, and 6.9.

The Simulink program of this example is chap6_2sim.mdl; the Matlab programs of the example are given in the Appendix.

6.2 RBF Neural Robot Controller Design with Sliding Mode Robust Term

6.2.1 Problem Description

Consider dynamic equation of n-link manipulator as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{\tau}_{\mathrm{d}} = \boldsymbol{\tau} \tag{6.16}$$

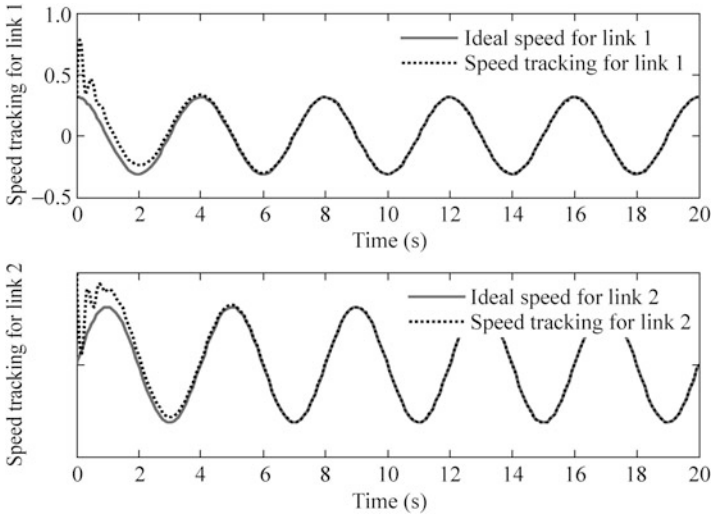


Fig. 6.7 Speed tracking of link 1 and link 2

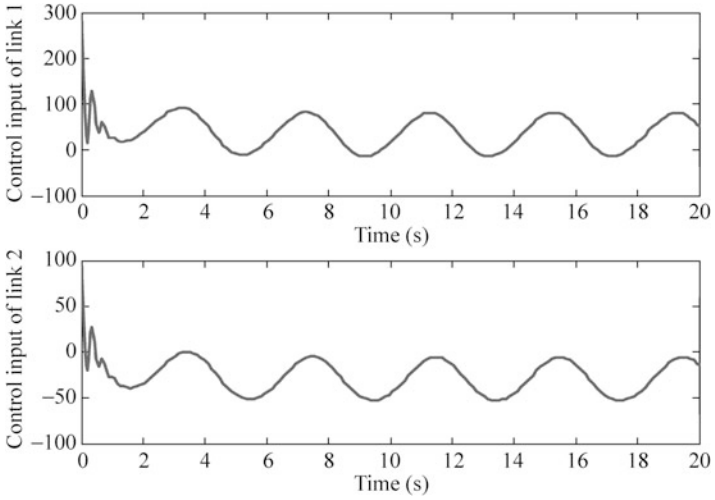


Fig. 6.8 Control input of link 1 and link 2

where $\boldsymbol{M}(\boldsymbol{q})$ is an $n \times n$ inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is an $n \times n$ matrix containing the centrifugal and Coriolis terms, $\boldsymbol{G}(\boldsymbol{q})$ is an $n \times 1$ vector containing gravitational forces and torques, q is generalized joint coordinates, τ is joint torques, and τ_{d} denotes disturbances.

The tracking error vector is designed as $\boldsymbol{e}(t) = \boldsymbol{q}_{\mathrm{d}}(t) - \boldsymbol{q}(t)$, and defines the sliding mode function as

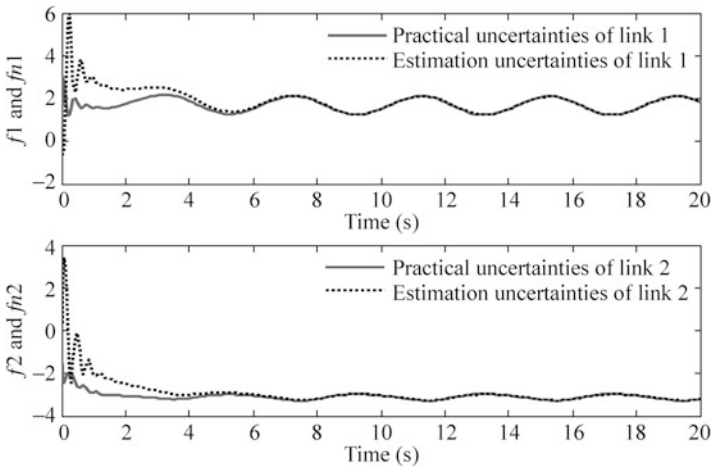


Fig. 6.9 $\boldsymbol{f}(\boldsymbol{x})$ estimation of link 1 and link 2

$$\boldsymbol{r} = \dot{\boldsymbol{e}} + \boldsymbol{\Lambda e} \tag{6.17}$$

where $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}^{\mathrm{T}} = [\lambda_1 \quad \lambda_2 \quad \cdots \quad \lambda_n]^{\mathrm{T}} > 0$ is an appropriately chosen coefficient vector such that $s^{n-1} + \lambda_{n-1}s^{n-2} + \cdots + \lambda_1$ is Hurwitz (i.e., $\boldsymbol{e} \to 0$ exponentially as $\boldsymbol{r} \to 0$).

The sliding mode tracking error $\boldsymbol{r}$ can be viewed as the real-valued utility function of the plant performance. When $\boldsymbol{r}$ is small, system performance is good. For the system (6.16), all modeling information was expressed as $\boldsymbol{f}(\boldsymbol{x})$ by using the sliding mode tracking error $\boldsymbol{r}$ [7].

The item (6.17) gives

$$\dot{\boldsymbol{q}} = -\boldsymbol{r} + \dot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\Lambda e} \tag{6.18}$$

and

$$\begin{aligned} \boldsymbol{M}\dot{\boldsymbol{r}} &= \boldsymbol{M}(\ddot{\boldsymbol{q}}_{\mathrm{d}} - \ddot{\boldsymbol{q}} + \boldsymbol{\Lambda}\dot{\boldsymbol{e}}) = \boldsymbol{M}(\ddot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\Lambda}\dot{\boldsymbol{e}}) - \boldsymbol{M}\ddot{\boldsymbol{q}} \\ &= \boldsymbol{M}(\ddot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\Lambda}\dot{\boldsymbol{e}}) + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{G} + \boldsymbol{F} + \boldsymbol{\tau}_{\mathrm{d}} - \boldsymbol{\tau} \\ &= \boldsymbol{M}(\ddot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\Lambda}\dot{\boldsymbol{e}}) - \boldsymbol{C}\boldsymbol{r} + \boldsymbol{C}(\dot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\Lambda e}) + \boldsymbol{G} + \boldsymbol{F} + \boldsymbol{\tau}_{\mathrm{d}} - \boldsymbol{\tau} \\ &= -\boldsymbol{C}\boldsymbol{r} - \boldsymbol{\tau} + \boldsymbol{f} + \boldsymbol{\tau}_{\mathrm{d}} \end{aligned} \tag{6.19}$$

where $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{M}\ddot{\boldsymbol{q}}_{\mathrm{r}} + \boldsymbol{C}\dot{\boldsymbol{q}}_{\mathrm{r}} + \boldsymbol{G} + \boldsymbol{F}, \quad \dot{\boldsymbol{q}}_{\mathrm{r}} = \dot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\Lambda e}$.

From $\boldsymbol{f}(\boldsymbol{x})$ expression, we can see that the term $\boldsymbol{f}(\boldsymbol{x})$ includes all the modeling information.

The goal is to design a stable robust controller without any modeling information. In this section, we use RBF to approximate $\boldsymbol{f}(\boldsymbol{x})$.

6.2.2 RBF Approximation

RBF algorithm is described as

$$h_j = \exp\frac{\left\|\boldsymbol{x} - \boldsymbol{c}_j\right\|^2}{b_j^2}, \quad j = 1, 2, \ldots, m$$
$$f(x) = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{h} + \varepsilon \tag{6.20}$$

where $\boldsymbol{x}$ is input of RBF, $\boldsymbol{W}$ is optimum weight value, $\boldsymbol{h} = [\,h_1 \quad h_2 \quad \cdots \quad h_m\,]^{\mathrm{T}}$, and ε is a very small value.

The output of RBF is used to approximate $\boldsymbol{f}(\boldsymbol{x})$,

$$\hat{\boldsymbol{f}}(\boldsymbol{x}) = \hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h} \tag{6.21}$$

where $\tilde{\boldsymbol{W}} = \boldsymbol{W} - \hat{\boldsymbol{W}}$, $||\boldsymbol{W}||_{\mathrm{F}} \leq W_{\max}$.

From (6.20) and (6.21), we have

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

From $\boldsymbol{f}(\boldsymbol{x})$ expression, the input of RBF should be chosen as $\boldsymbol{x} = [\,\boldsymbol{e}^{\mathrm{T}} \quad \dot{\boldsymbol{e}}^{\mathrm{T}} \quad \boldsymbol{q}_{\mathrm{d}}^{\mathrm{T}} \quad \dot{\boldsymbol{q}}_{\mathrm{d}}^{\mathrm{T}} \quad \ddot{\boldsymbol{q}}_{\mathrm{d}}^{\mathrm{T}}\,]$.

6.2.3 Control Law Design and Stability Analysis

For the system (6.16), the control law as proposed as [7],

$$\boldsymbol{\tau} = \hat{\boldsymbol{f}}(\boldsymbol{x}) + \boldsymbol{K}_{\mathrm{v}}\boldsymbol{r} - \boldsymbol{v} \tag{6.22}$$

with robust term $\boldsymbol{v} = -(\varepsilon_{\mathrm{N}} + b_{\mathrm{d}})\mathrm{sgn}(\boldsymbol{r})$, where $\hat{\boldsymbol{f}}(\boldsymbol{x})$ is estimation of $\boldsymbol{f}(\boldsymbol{x})$, and $\boldsymbol{v}$ is robustness term.

The corresponding RBF adaptive law is designed as

$$\dot{\hat{\boldsymbol{W}}} = \boldsymbol{\Gamma}\boldsymbol{h}\boldsymbol{r}^{\mathrm{T}} \tag{6.23}$$

where $\boldsymbol{\Gamma} = \boldsymbol{\Gamma}^{\mathrm{T}} > 0$.

Inserting (6.19) yields

$$\begin{aligned}\boldsymbol{M}\dot{\boldsymbol{r}} &= -\boldsymbol{C}\boldsymbol{r} - (\hat{\boldsymbol{f}}(\boldsymbol{x}) + \boldsymbol{K}_{\mathrm{v}}\boldsymbol{r} - \boldsymbol{v}) + \boldsymbol{f} + \boldsymbol{\tau}_{\mathrm{d}} \\ &= -(\boldsymbol{K}_{\mathrm{v}} + \boldsymbol{C})\boldsymbol{r} + \tilde{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h} + (\boldsymbol{\varepsilon} + \boldsymbol{\tau}_{\mathrm{d}}) + \boldsymbol{v} \\ &= -(\boldsymbol{K}_{\mathrm{v}} + \boldsymbol{C})\boldsymbol{r} + \boldsymbol{\varsigma}_1\end{aligned} \tag{6.24}$$

where $\boldsymbol{\varsigma}_1 = \tilde{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{\varphi} + (\boldsymbol{\varepsilon} + \boldsymbol{\tau}_{\mathrm{d}}) + \boldsymbol{v}$.

The stability proof of the closed system was given with two steps as follows [7]. Firstly, define Lyapunov function as

$$L = \frac{1}{2}\boldsymbol{r}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{r} + \frac{1}{2}\mathrm{tr}\left(\tilde{\boldsymbol{W}}^{\mathrm{T}}\Gamma^{-1}\tilde{\boldsymbol{W}}\right)$$

Thus,

$$\dot{L} = \boldsymbol{r}^{\mathrm{T}}\boldsymbol{M}\dot{\boldsymbol{r}} + \frac{1}{2}\boldsymbol{r}^{\mathrm{T}}\dot{\boldsymbol{M}}\boldsymbol{r} + \mathrm{tr}\left(\tilde{\boldsymbol{W}}^{\mathrm{T}}\Gamma^{-1}\dot{\tilde{\boldsymbol{W}}}\right)$$

Secondly, inserting (6.24) into above yields

$$\dot{L} = -\boldsymbol{r}^{\mathrm{T}}\boldsymbol{K}_{\mathrm{v}}\boldsymbol{r} + \frac{1}{2}\boldsymbol{r}^{\mathrm{T}}(\dot{\boldsymbol{M}} - 2\boldsymbol{C})\boldsymbol{r} + \mathrm{tr}\tilde{\boldsymbol{W}}^{\mathrm{T}}\left(\Gamma^{-1}\dot{\tilde{\boldsymbol{W}}} + \boldsymbol{h}\boldsymbol{r}^{\mathrm{T}}\right) + \boldsymbol{r}^{\mathrm{T}}(\boldsymbol{\varepsilon} + \boldsymbol{\tau}_{\mathrm{d}} + \boldsymbol{v})$$

Since:

1. According to the skew-symmetric characteristics of manipulator dynamic equation, $\boldsymbol{r}^{\mathrm{T}}(\dot{\boldsymbol{M}} - 2\boldsymbol{C})\boldsymbol{r} = 0$;
2. $\boldsymbol{r}^{\mathrm{T}}\tilde{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h} = \mathrm{tr}(\tilde{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}\boldsymbol{r}^{\mathrm{T}})$;
3. $\dot{\tilde{\boldsymbol{W}}} = -\dot{\hat{\boldsymbol{W}}} = -\boldsymbol{\Gamma}\boldsymbol{h}\boldsymbol{r}^{\mathrm{T}}$.

then,

$$\dot{L} = -\boldsymbol{r}^{\mathrm{T}}\boldsymbol{K}_{\mathrm{v}}\boldsymbol{r} + \boldsymbol{r}^{\mathrm{T}}(\boldsymbol{\varepsilon} + \boldsymbol{\tau}_{\mathrm{d}} + \boldsymbol{v})$$

Consider

$$\begin{aligned}\boldsymbol{r}^{\mathrm{T}}(\boldsymbol{\varepsilon} + \boldsymbol{\tau}_{\mathrm{d}} + \boldsymbol{v}) &= \boldsymbol{r}^{\mathrm{T}}(\boldsymbol{\varepsilon} + \boldsymbol{\tau}_{\mathrm{d}}) + \boldsymbol{r}^{\mathrm{T}}(-(\varepsilon_{\mathrm{N}} + b_{\mathrm{d}})\mathrm{sgn}(\boldsymbol{r})) \\ &= \boldsymbol{r}^{\mathrm{T}}(\boldsymbol{\varepsilon} + \boldsymbol{\tau}_{\mathrm{d}}) - \|\boldsymbol{r}\|(\boldsymbol{\varepsilon}_{\mathrm{N}} + \boldsymbol{b}_{\mathrm{d}}) \leq 0\end{aligned}$$

There results finally

$$\dot{L} \leq -\boldsymbol{r}^{\mathrm{T}}\boldsymbol{K}_{\mathrm{v}}\boldsymbol{r} \leq 0$$

6.2.4 *Simulation Examples*

6.2.4.1 First Example

Consider a simple servo system as

$$M\ddot{q} + F(\dot{q}) = \tau$$

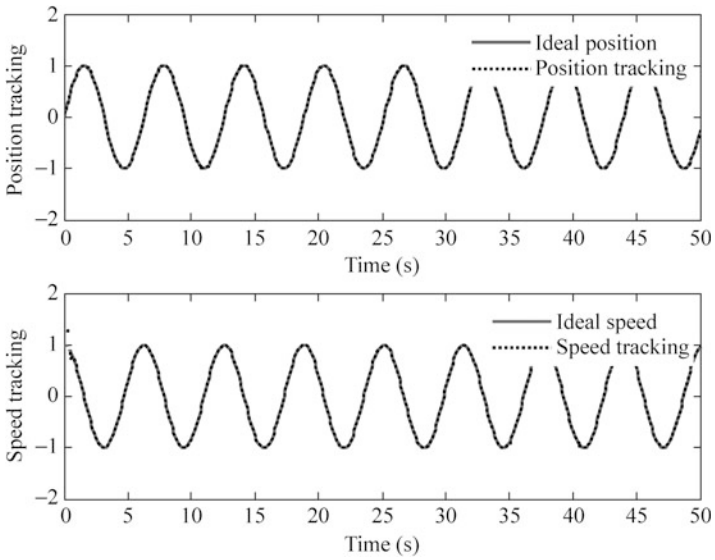


Fig. 6.10 Position and speed tracking with RBF compensation

where $M = 10$, $F(\dot{q})$ denotes friction force, and $F(\dot{q}) = 15\dot{q} + 30\text{sgn}(\dot{q})$.

From (6.16), (6.17), (6.18), and (6.19), we have $f(x) = M\ddot{q}_r + F = M(\ddot{q}_d + \Lambda\dot{e}) + F$.

For RBF neural network, the structure is 2-7-1, the input is chosen as $z = [e \quad \dot{e}]$, the parameters of Gaussian function c_i and b_i are chosen as $[-1.5 \quad -1.0 \quad -0.5 \quad 0 \quad 0.5 \quad 1.0 \quad 1.5]$ and 10, and the initial weight value is chosen as zero. The desired trajectory is $q_d = \sin t$, and the initial value of the plant is $[0.10 \quad 0]^T$.

In simulation, we use control law (6.22) and adaptive law (6.23), the parameters are chosen as $\Lambda = 15$, $\Gamma = 100$, and $K_v = 110$. The simulation results are shown from Figs. 6.10 and 6.11.

The Simulink program of this example is chap6_3sim.mdl; the Matlab programs of the example are given in the Appendix.

6.2.4.2 Second Example

Consider a plant as

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

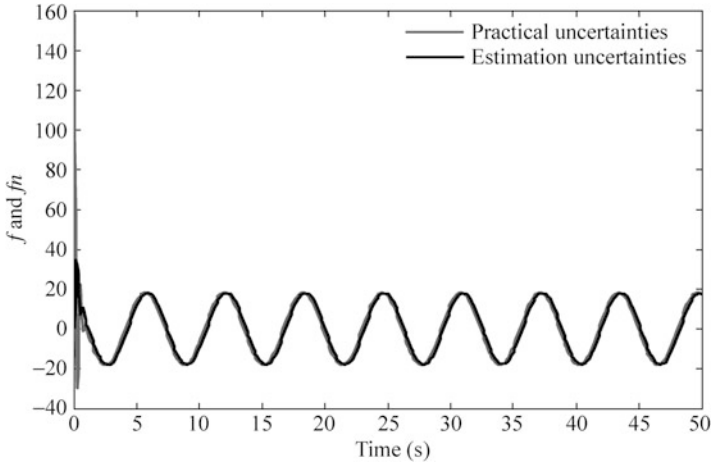


Fig. 6.11 $f(x)$ estimation with RBF

where

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

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

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

$\boldsymbol{F}(\dot{\boldsymbol{q}}) = 0.02\,\mathrm{sgn}(\dot{\boldsymbol{q}})$, $\boldsymbol{\tau}_\mathrm{d} = [0.2\sin(t) \quad 0.2\sin(t)]^\mathrm{T}$, $\boldsymbol{p} = [p_1, p_2, p_3, p_4, p_5] = [2.9, 0.76, \ 0.87, 3.04, 0.87]$.

For RBF neural network, the structure is 2-7-1, the input is chosen as $\boldsymbol{z} = [\boldsymbol{e} \quad \dot{\boldsymbol{e}}]$, the parameters of Gaussian function $\boldsymbol{c}_\mathrm{i}$ and b_i are chosen as $[-1.5 \quad -1.0 \quad -0.5 \quad 0 \quad 0.5 \quad 1.0 \quad 1.5]$ and 10, and the initial weight value is chosen as zero. The desired trajectory is $q_{1\mathrm{d}} = 0.1\sin t$, $q_{2\mathrm{d}} = 0.1\sin t$. The initial value of the plant is $[0.09 \quad 0 \quad -0.09 \quad 0]$.

Using control law (6.22) and adaptive law (6.23), $\boldsymbol{K}_\mathrm{v} = \mathrm{diag}\{10, 10\}$, $\boldsymbol{\Gamma} = \mathrm{diag}\{15, 15\}$, and $\boldsymbol{\Lambda} = \mathrm{diag}\{5, 5\}$. The simulation results are shown from Figs. 6.12, 6.13, 6.14, and 6.15.

The Simulink program of this example is chap6_4sim.mdl; the Matlab programs of the example are given in the Appendix.

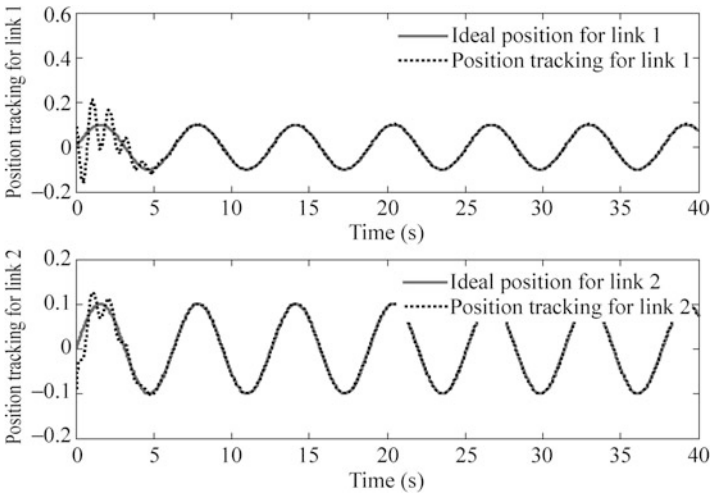


Fig. 6.12 Position tracking

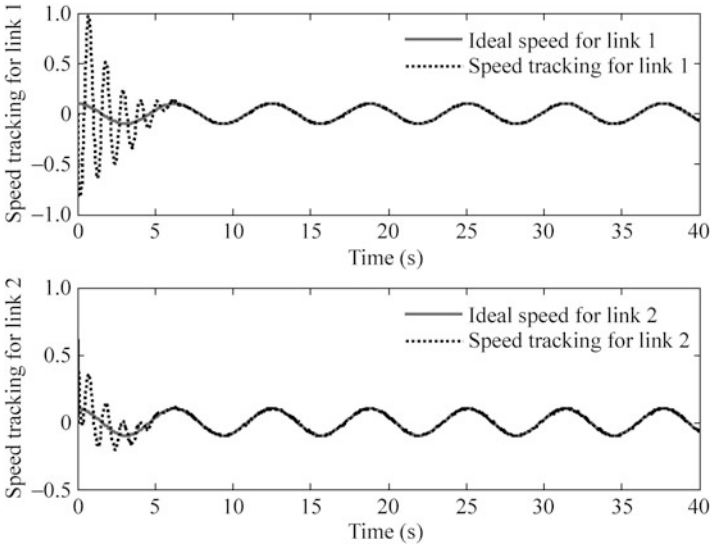


Fig. 6.13 Speed tracking

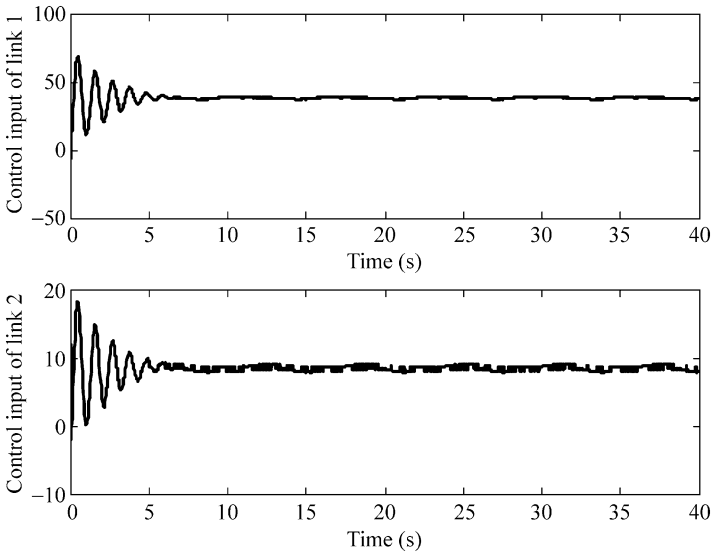


Fig. 6.14 Control input of link 1 and 2

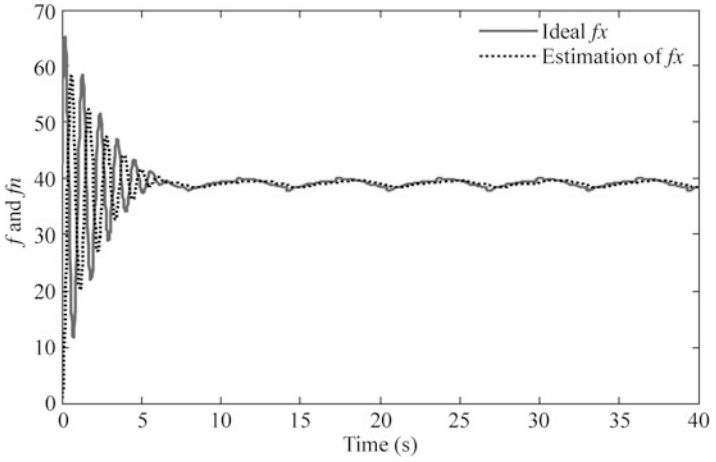


Fig. 6.15 $||\boldsymbol{f}(\boldsymbol{x})||$ and $||\hat{\boldsymbol{f}}(\boldsymbol{x})||$

6.3 Robust Control Based on RBF Neural Network with HJI

6.3.1 Foundation

Consider a system as

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{d} \\ \boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}) \end{cases} \tag{6.25}$$

where $\boldsymbol{d}$ is disturbance and $\boldsymbol{z}$ is critic signal for the system.

Definition. For signal $\boldsymbol{d}(t)$, its L_2 is $\|\boldsymbol{d}(t)\|_2 = \left\{\int_0^\infty \boldsymbol{d}^{\mathrm{T}}(t)\boldsymbol{d}(t)\mathrm{d}t\right\}^{\frac{1}{2}}$, which denotes the energy of $\boldsymbol{d}(t)$.

To express suppression ability, the following performance index is defined as

$$J = \sup_{\|d\|\neq 0} \frac{\|\boldsymbol{z}\|_2}{\|\boldsymbol{d}\|_2} \tag{6.26}$$

The term J is called L_2 gain of the system, which denotes the robust performance index of the system. The smaller J is, the better robust performance is.

With theorem 2 given in [8] and the system (6.25), HJI (Hamilton–Jacobi inequality) can be described as follows: for a positive number γ, if there exists positive definite quasi-differentiable function $L(\boldsymbol{x}) \geq 0$ and

$$\dot{L} \leq \frac{1}{2}\left\{\gamma^2\|\boldsymbol{d}\|^2 - \|\boldsymbol{z}\|^2\right\} \quad (\forall \boldsymbol{d}) \tag{6.27}$$

then, $J \leq \gamma$.

6.3.2 Controller Design and Analysis

HJI inequation was applied to design robust neural network controller for robots in [9]. In reference to theorem 3 in [9], a neural network adaptive controller is designed with HJI for $n-$ link manipulators as follows.

Consider dynamic equation of $n-$ link manipulators as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{V}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + \Delta(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{d} = \boldsymbol{T} \tag{6.28}$$

where $\boldsymbol{M}(\boldsymbol{q})$ is an $n \times n$ inertia matrix, $\boldsymbol{V}(\boldsymbol{q},\dot{\boldsymbol{q}})$ is a $n \times n$ matrix containing the centrifugal and Coriolis terms, $\boldsymbol{G}(\boldsymbol{q})$ is an $n \times 1$ vector containing gravitational

forces and torques, $\boldsymbol{q}$ is the angle vector, τ is joint torques, $\Delta(\boldsymbol{q}, \dot{\boldsymbol{q}})$ denotes uncertainties, and $\boldsymbol{d}$ denotes outer disturbances.

Consider desired angle signal $\boldsymbol{q}_{\mathrm{d}}$, the tracking error vector is $\boldsymbol{e} = \boldsymbol{q} - \boldsymbol{q}_{\mathrm{d}}$, and design a feed-forward control law as

$$\boldsymbol{T} = \boldsymbol{u} + \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{V}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{G}(\boldsymbol{q}) \tag{6.29}$$

where $\boldsymbol{u}$ is a feedback control law and to be designed in the following.

Submitting (6.29) into (6.28), we get closed system as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{e}} + \boldsymbol{V}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{e}} + \Delta(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{d} = \boldsymbol{u} \tag{6.30}$$

Letting $\Delta \boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \Delta(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{d}$, we have

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{e}} + \boldsymbol{V}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{e}} + \Delta \boldsymbol{f} = \boldsymbol{u} \tag{6.31}$$

Use RBF to approximate $\Delta \boldsymbol{f}$, then,

$$\Delta \boldsymbol{f} = \boldsymbol{W}_{\mathrm{f}}^{*}\boldsymbol{\sigma}_{\mathrm{f}} + \boldsymbol{\varepsilon}_{\mathrm{f}} \tag{6.32}$$

where $\boldsymbol{\varepsilon}_{\mathrm{f}}$ denotes the approximation error, $\boldsymbol{\sigma}_{\mathrm{f}}$ denotes the Gaussian function of RBF, and $\boldsymbol{W}_{\mathrm{f}}$ is weight value.

From (6.31) and (6.32), we have

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{e}} + \boldsymbol{V}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{e}} + \boldsymbol{W}_{\mathrm{f}}^{*}\boldsymbol{\sigma}_{\mathrm{f}} + \boldsymbol{\varepsilon}_{\mathrm{f}} = \boldsymbol{u}$$

Define two variables as

$$\begin{cases} \boldsymbol{x}_1 = \boldsymbol{e} \\ \boldsymbol{x}_2 = \dot{\boldsymbol{e}} + \alpha \boldsymbol{e} \end{cases} \tag{6.33}$$

where $\alpha > 0$.

Then,

$$\begin{cases} \dot{\boldsymbol{x}}_1 = \boldsymbol{x}_2 - \alpha \boldsymbol{x}_1 \\ \boldsymbol{M}\dot{\boldsymbol{x}}_2 = -\boldsymbol{V}\boldsymbol{x}_2 + \boldsymbol{\omega} - \boldsymbol{W}_{\mathrm{f}}^{*}\boldsymbol{\sigma}_{\mathrm{f}} - \boldsymbol{\varepsilon}_{\mathrm{f}} + \boldsymbol{u} \end{cases} \tag{6.34}$$

where $\boldsymbol{\omega} = \boldsymbol{M}a\dot{\boldsymbol{e}} + \boldsymbol{V}\alpha \boldsymbol{e}$.

To utilize the HJI inequation, the Eq. (6.34) should be written as (6.25),

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{d} \\ \boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}) \end{cases} \tag{6.35}$$

where $\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{x}_2 - \alpha \boldsymbol{x}_1 \\ \frac{1}{M}(-\boldsymbol{V}\boldsymbol{x}_2 + \boldsymbol{\omega} - \boldsymbol{W}_{\mathrm{f}}^{*}\boldsymbol{\sigma}_{\mathrm{f}} + \boldsymbol{u}) \end{bmatrix}$, $\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} 0 \\ -\frac{1}{M} \end{bmatrix}$, and $d = \varepsilon_{\mathrm{f}}$.

Since $d = \varepsilon_{\mathrm{f}}$, we can consider the approximation error $\boldsymbol{\varepsilon}_{\mathrm{f}}$ as the disturbance d, and define $z_{\mathrm{R}} = \dot{\boldsymbol{e}} + \alpha \boldsymbol{e}$, then, we can write the L_2 gain as $J_{\mathrm{R}} = \sup\limits_{||\boldsymbol{\varepsilon}_{\mathrm{f}}||\neq 0} \frac{||z||_2}{||\boldsymbol{\varepsilon}_{\mathrm{f}}||_2}$.

For the system (6.34), we design adaptive law as

$$\dot{\hat{\boldsymbol{W}}}_{\mathrm{f}} = -\eta \boldsymbol{x}_2 \boldsymbol{\sigma}_{\mathrm{f}}^{\mathrm{T}} \tag{6.36}$$

Design feedback control law as

$$\boldsymbol{u} = -\boldsymbol{\omega} - \frac{1}{2\gamma^2}\boldsymbol{x}_2 + \hat{\boldsymbol{W}}_{\mathrm{f}}\boldsymbol{\sigma}_{\mathrm{f}} - \frac{1}{2}\boldsymbol{x}_2 \tag{6.37}$$

where $\hat{\boldsymbol{W}}_{\mathrm{f}}$ and $\boldsymbol{\sigma}_{\mathrm{f}}$ are weight value and Gaussian function of RBF.

Then for the closed system (6.30), $J \leq \gamma$,

For the closed control system, in reference to the analysis in paper [9], the stability can be analyzed as follows:

Firstly, define the Lyapunov function as

$$L = \frac{1}{2}\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{M}\boldsymbol{x}_2 + \frac{1}{2\eta}\mathrm{tr}\left(\tilde{\boldsymbol{W}}_{\mathrm{f}}{}^{\mathrm{T}}\tilde{\boldsymbol{W}}_{\mathrm{f}}\right)$$

where $\tilde{\boldsymbol{W}}_{\mathrm{f}} = \hat{\boldsymbol{W}}_{\mathrm{f}} - \boldsymbol{W}_{\mathrm{f}}^{*}$.

Then, using (6.34) and (6.37), and considering the skew-symmetric characteristics of manipulator dynamic equation, we have

$$\begin{aligned}
\dot{L} &= \boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{M}\dot{\boldsymbol{x}}_2 + \frac{1}{2}\boldsymbol{x}_2^{\mathrm{T}}\dot{\boldsymbol{M}}\boldsymbol{x}_2 + \frac{1}{\eta}\mathrm{tr}\left(\dot{\tilde{\boldsymbol{W}}}_{\mathrm{f}}{}^{\mathrm{T}}\tilde{\boldsymbol{W}}_{\mathrm{f}}\right) \\
&= \boldsymbol{x}_2^{\mathrm{T}}\left(-\boldsymbol{V}\boldsymbol{x}_2 + \boldsymbol{\omega} - \boldsymbol{W}_{\mathrm{f}}^{*}\boldsymbol{\sigma}_{\mathrm{f}} - \boldsymbol{\varepsilon}_{\mathrm{f}} + \boldsymbol{u}\right) + \frac{1}{2}\boldsymbol{x}_2^{\mathrm{T}}\dot{\boldsymbol{M}}\boldsymbol{x}_2 + \frac{1}{\eta}\mathrm{tr}\left(\dot{\tilde{\boldsymbol{W}}}_{\mathrm{f}}{}^{\mathrm{T}}\tilde{\boldsymbol{W}}_{\mathrm{f}}\right) \\
&= \boldsymbol{x}_2^{\mathrm{T}}\left(-\boldsymbol{V}\boldsymbol{x}_2 - \boldsymbol{W}_{\mathrm{f}}^{*}\boldsymbol{\sigma}_{\mathrm{f}} - \boldsymbol{\varepsilon}_{\mathrm{f}} - \frac{1}{2\gamma^2}\boldsymbol{x}_2 + \hat{\boldsymbol{W}}_{\mathrm{f}}\boldsymbol{\sigma}_{\mathrm{f}} - \frac{1}{2}\boldsymbol{x}_2\right) + \frac{1}{2}\boldsymbol{x}_2^{\mathrm{T}}\dot{\boldsymbol{M}}\boldsymbol{x}_2 + \frac{1}{\eta}\mathrm{tr}\left(\dot{\tilde{\boldsymbol{W}}}_{\mathrm{f}}{}^{\mathrm{T}}\tilde{\boldsymbol{W}}_{\mathrm{f}}\right) \\
&= \boldsymbol{x}_2^{\mathrm{T}}\left(-\boldsymbol{\varepsilon}_{\mathrm{f}} - \frac{1}{2\gamma^2}\boldsymbol{x}_2 + \tilde{\boldsymbol{W}}_{\mathrm{f}}\boldsymbol{\sigma}_{\mathrm{f}} - \frac{1}{2}\boldsymbol{x}_2\right) + \frac{1}{2}\boldsymbol{x}_2^{\mathrm{T}}(\dot{\boldsymbol{M}} - 2\boldsymbol{V})\boldsymbol{x}_2 + \frac{1}{\eta}\mathrm{tr}\left(\dot{\tilde{\boldsymbol{W}}}_{\mathrm{f}}{}^{\mathrm{T}}\tilde{\boldsymbol{W}}_{\mathrm{f}}\right) \\
&= -\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{\varepsilon}_{\mathrm{f}} - \frac{1}{2\gamma^2}\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{x}_2 + \boldsymbol{x}_2^{\mathrm{T}}\tilde{\boldsymbol{W}}_{\mathrm{f}}\boldsymbol{\sigma}_{\mathrm{f}} - \frac{1}{2}\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{x}_2 + \frac{1}{\eta}\mathrm{tr}\left(\dot{\tilde{\boldsymbol{W}}}_{\mathrm{f}}{}^{\mathrm{T}}\tilde{\boldsymbol{W}}_{\mathrm{f}}\right)
\end{aligned}$$

Define

$$H = \dot{L} - \frac{1}{2}\gamma^2\|\boldsymbol{\varepsilon}_{\mathrm{f}}\|^2 + \frac{1}{2}\|z_{\mathrm{R}}\|^2 \tag{6.38}$$

Then,

$$H = -\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{\varepsilon}_{\mathrm{f}} - \frac{1}{2\gamma^2}\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{x}_2 + \boldsymbol{x}_2^{\mathrm{T}}\tilde{\boldsymbol{W}}_{\mathrm{f}}\boldsymbol{\sigma}_{\mathrm{f}} - \frac{1}{2}\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{x}_2 + \frac{1}{\eta}\mathrm{tr}\left(\dot{\tilde{\boldsymbol{W}}}_{\mathrm{f}}^{\mathrm{T}}\tilde{\boldsymbol{W}}_{\mathrm{f}}\right) - \frac{1}{2}\gamma^2\|\boldsymbol{\varepsilon}_{\mathrm{f}}\|^2 + \frac{1}{2}\|z_{\mathrm{R}}\|^2$$

Consider

1. $-\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{\varepsilon}_{\mathrm{f}} - \frac{1}{2\gamma^2}\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{x}_2 - \frac{1}{2}\gamma^2\|\boldsymbol{\varepsilon}_{\mathrm{f}}\|^2 = -\frac{1}{2}\left\|\frac{1}{\gamma}\boldsymbol{x}_2 + \gamma\boldsymbol{\varepsilon}_{\mathrm{f}}\right\|^2 \leq 0;$
2. $\boldsymbol{x}_2^{\mathrm{T}}\tilde{\boldsymbol{W}}_{\mathrm{f}}\boldsymbol{\sigma}_{\mathrm{f}} + \frac{1}{\eta}\mathrm{tr}\left(\dot{\tilde{\boldsymbol{W}}}_{\mathrm{f}}^{\mathrm{T}}\tilde{\boldsymbol{W}}_{\mathrm{f}}\right) = 0;$
3. $-\frac{1}{2}\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{x}_2 + \frac{1}{2}\|z_{\mathrm{R}}\|^2 = 0.$

Therefore we have $H \leq 0$, according to H definition in (6.38), we have

$$\dot{L} \leq \frac{1}{2}\gamma^2||\boldsymbol{\varepsilon}_{\mathrm{f}}||^2 - \frac{1}{2}||z_{\mathrm{R}}||^2$$

Thus from theorem 1, we can get $J \leq \gamma$.

6.3.3 Simulation Examples

6.3.3.1 First Example

Consider a servo system as

$$M\ddot{q} = T - d$$

where $M = 1.0$ is the inertia moment.

Suppose the ideal position signal as $q_{\mathrm{d}} = \sin t$, the disturbance signal is $d = 150\,\mathrm{sgn}\,\dot{q} + 10\dot{q}$, and the initial states of the plant are zero. Choose $\eta = 1,000$, $\alpha = 200$, and $\gamma = 0.10$, and the parameters of Gaussian function $\boldsymbol{c}_{\mathrm{i}}$ and b_i are chosen as $[-1 \quad -0.5 \quad 0 \quad 0.5 \quad 1]$ and 50. Use adaptive law (6.36) and control laws (6.29) and (6.37); the results are shown from Figs. 6.16 and 6.17.

The Simulink program of this example is chap6_5sim.mdl; the Matlab programs of the example are given in the Appendix.

6.3.3.2 Second Example

Consider two link manipulators dynamic equation as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{V}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{D} = \boldsymbol{T}$$

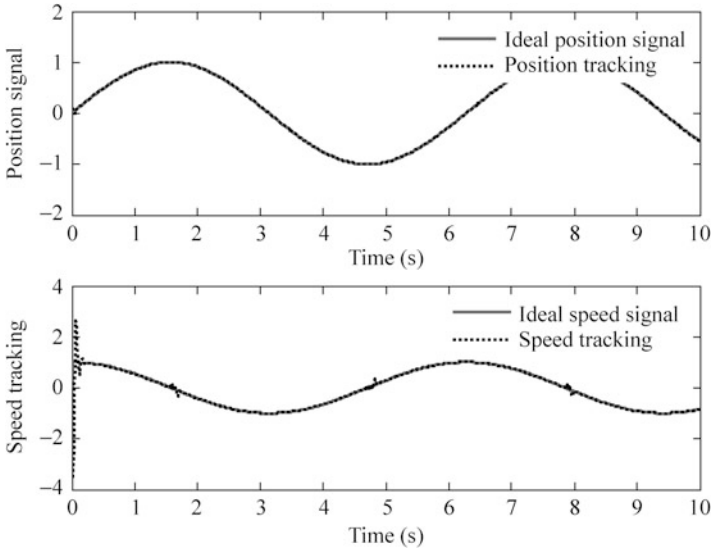


Fig. 6.16 Position and speed tracking with RBF compensation ($S = 2$)

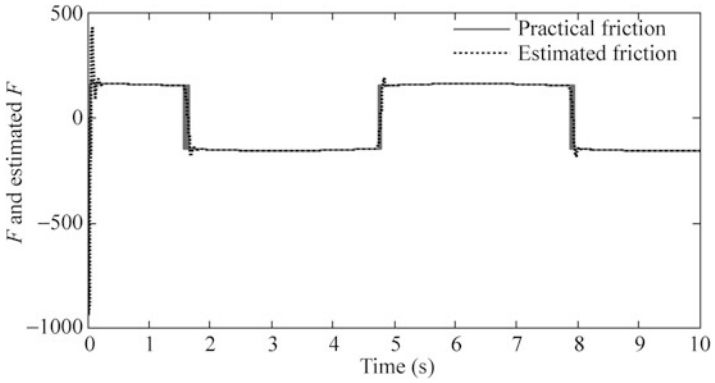


Fig. 6.17 Disturbance and its estimation ($S = 2$)

where $\boldsymbol{D} = \Delta(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{d}$, $M_{11} = (m_1 + m_2)r_1^2 + m_2 r_2^2 + 2m_2 r_1 r_2 \cos\, q_2$, $M_{12} = M_{21} = m_2 r_2^2 + m_2 r_1 r_2 \cos\, q_2$, $M_{22} = m_2 r_2^2$, $\boldsymbol{V} = \begin{bmatrix} -V_{12}\dot{q}_2 & -V_{12}(\dot{q}_1 + \dot{q}_2) \\ V_{12} q_1 & 0 \end{bmatrix}$, $V_{12} = m_2 r_1 \sin\, q_2$, $G_1 = (m_1 + m_2) r_1 \cos\, q_2 + m_2 r_2 \cos\,(q_1 + q_2)$, $G_2 = m_2 r_2 \cos\,(q_1 + q_2)$, $\boldsymbol{D} = \begin{bmatrix} 30\, \mathrm{sgn}\, q_2 \\ 30\ \mathrm{sgn}\, q_4 \end{bmatrix}$, $r_1 = 1,\ r_2 = 0.8,\ m_1 = 1,\ m_2 = 1.5$.

Suppose the ideal position signals as $q_{1\mathrm{d}} = \sin\, t$, and $q_{2\mathrm{d}} = \sin\, t$, and the initial states of the plant is zero. Choose 4-7-1 RBF structure, and set $\eta = 1{,}500$, $\alpha = 20$, $\gamma = 0.05$, the parameters of Gaussian function $\boldsymbol{c}_{\mathrm{i}}$ and b_i are chosen as

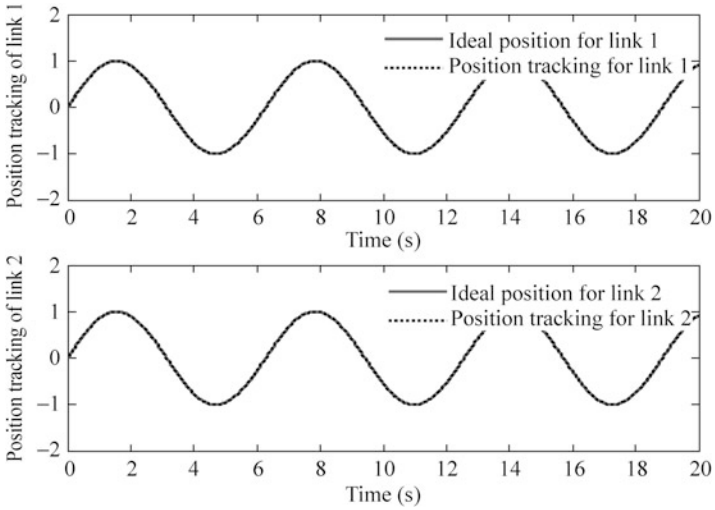


Fig. 6.18 Position tracking of with RBF compensation ($S = 2$)

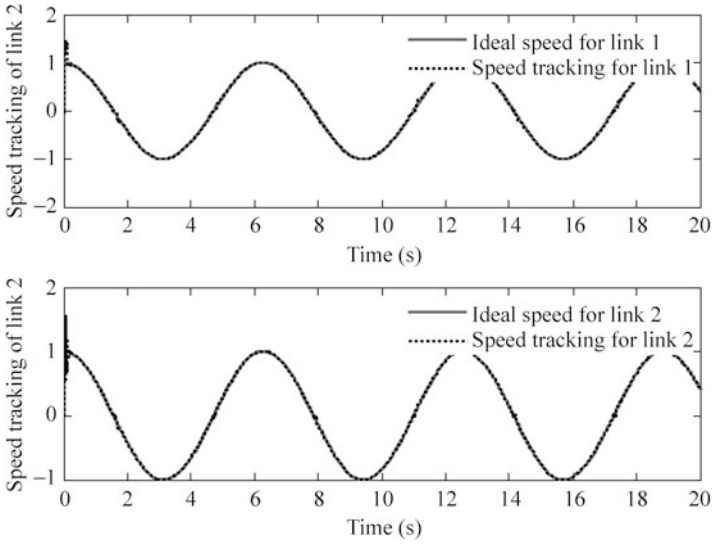


Fig. 6.19 Speed tracking of with RBF compensation ($S = 2$)

$[-1.5 \quad -1.0 \quad -0.5 \quad 0 \quad 0.5 \quad 1.0 \quad 1.5]$ and 10. Use adaptive law (6.36)) and control laws (6.29) and (6.37); the results are shown from Figs. 6.18, 6.19, and 6.20.

The Simulink program of this example is chap6_6sim.mdl; the Matlab programs of the example are given in the Appendix.

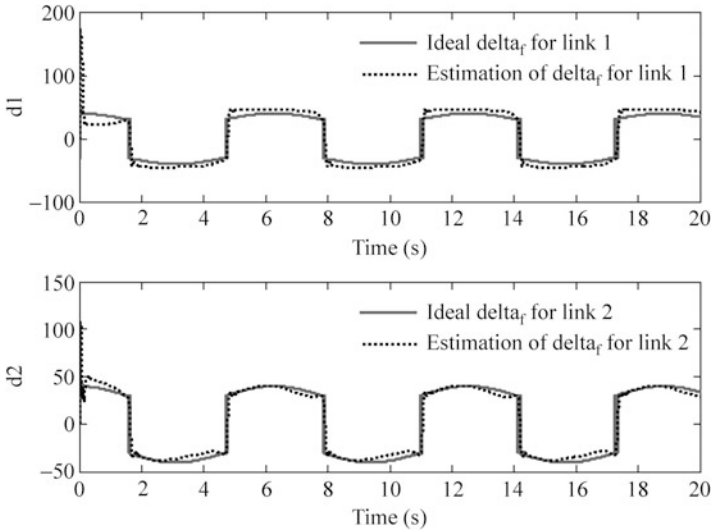
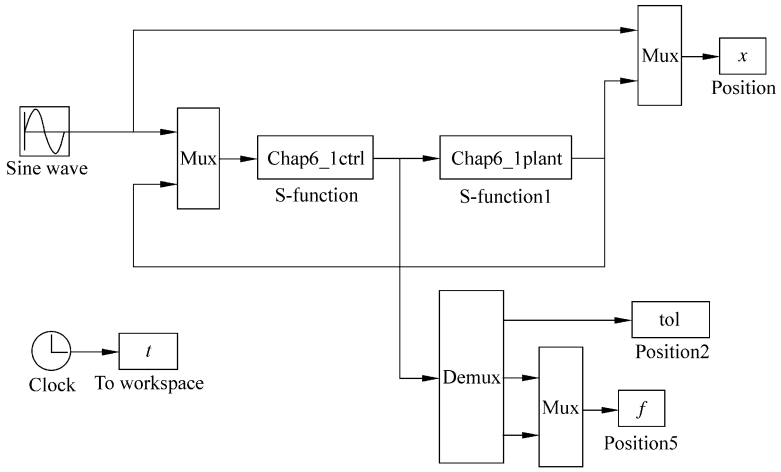


Fig. 6.20 Disturbance and its estimation ($S = 2$)

Appendix

Programs for Sect. 6.1.4.1

Simulink main program: chap6_1sim.mdl



S function for control law and adaptive law: chap6_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 kv kp
sizes = simsizes;
sizes.NumContStates = 5;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = 0.1*ones(1,5);
str = [];
ts = [0 0];
c=0.5*[-2 -1 0 1 2;
       -2 -1 0 1 2];
b=1.5*ones(5,1);
alfa=3;
kp=alfa^2;
kv=2*alfa;
function sys=mdlDerivatives(t,x,u)
global c b kv kp
qd=u(1);
dqd=cos(t);
ddqd=-sin(t);

q=u(2);
```

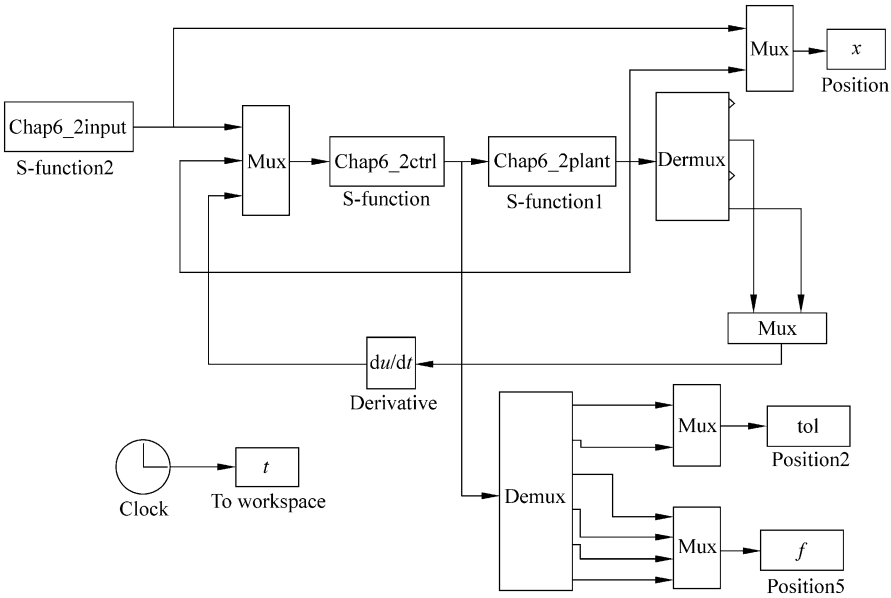
```
dq=u(3);

e=q-qd;
de=dq-dqd;

A=[0 1;-kp -kv];
D=10;
B=[0 1/D]';

Q=50*eye(2);
P=lyap(A',Q);
eig(P);

th=[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(j)*b(j)));
end
gama=1200;

S1=1;
if S1==1          % First adaptive Law
    S=gama*h*xi'*P*B;
elseif S1==2      % Secod adaptive Law with UUB
    k1=0.001;
    S=gama*h*xi'*P*B+k1*gama*norm(xi)*th;
end
S=S';
for i=1:1:5
    sys(i)=S(i);
end

function sys=mdlOutputs(t,x,u)
global c b kv kp
qd=u(1);
dqd=cos(t);
ddqd=-sin(t);

q=u(2);
dq=u(3);

e=q-qd;
de=dq-dqd;
```

```
M=10;

tol1=M*(ddqd-kv*de-kp*e);

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

d=-15*dq-30*sign(dq);
f=d;

S=3;
if S==1          %Nominal model based controller
    fn=0;
    tol=tol1;
elseif S==2       %Modified computed torque controller
    fn=0;
    tol2=-f;
    tol=tol1+tol2;
elseif S==3       %RBF compensated controller
    th=[x(1) x(2) x(3) x(4) x(5)]';
    fn=th'*h;
    tol2=-fn;
    tol=tol1+1*tol2;
end

sys(1)=tol;
sys(2)=f;
sys(3)=fn;
S function for plant:chap6_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 = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.6;0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
M=10;
d=-15*x(2)-30*sign(x(2));

tol=u(1);

sys(1)=x(2);
sys(2)=1/M*(tol+d);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
Plot program:chap6_1plot.m
close all;

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

figure(2);
plot(t,f(:,1),'r',t,f(:,2),'b','linewidth',2);
xlabel('time(s)');ylabel('f and fn');
legend('Practical          uncertainties','Estimation
uncertainties');
```

Programs for Sect. 6.1.4.2

Simulink main program: chap6_2sim.mdl



Tracking command program: chap6_2input.m

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

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

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 0;
```

```
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];

function sys=mdlOutputs(t,x,u)
qd1=1+0.2*sin(0.5*pi*t);
qd2=1-0.2*cos(0.5*pi*t);
dqd1=0.2*0.5*pi*cos(0.5*pi*t);
dqd2=0.2*0.5*pi*sin(0.5*pi*t);

sys(1)=qd1;
sys(2)=qd2;
sys(3)=dqd1;
sys(4)=dqd2;
S function for control law and adaptive law:chap6_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 kv kp
sizes = simsizes;
sizes.NumContStates = 10;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 6;
sizes.NumInputs = 10;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = 0.1*ones(1,10);
str = [];
ts = [0 0];
```

```
c= [-2 -1 0 1 2;
    -2 -1 0 1 2;
    -2 -1 0 1 2;
    -2 -1 0 1 2];
b=3.0;
alfa=3;
kp=[alfa^2 0;
    0 alfa^2];
kv=[2*alfa 0;
    0 2*alfa];
function sys=mdlDerivatives(t,x,u)
global c b kv kp

A=[zeros(2) eye(2);
   -kp -kv];
B=[0 0;0 0;1 0;0 1];

Q=[50 0  0  0;
   0  50 0  0;
   0  0  50 0;
   0  0  0  50];
P=lyap(A',Q);
eig(P);

qd1=u(1);
qd2=u(2);
d_qd1=u(3);
d_qd2=u(4);

q1=u(5);dq1=u(6);q2=u(7);dq2=u(8);
e1=q1-qd1;
e2=q2-qd2;
de1=dq1-d_qd1;
de2=dq2-d_qd2;

w=[x(1) x(2) x(3) x(4) x(5);x(6) x(7) x(8) x(9) x(10)]';
xi=[e1;e2;de1;de2];
h=zeros(5,1);
for j=1:1:5
    h(j)=exp(-norm(xi-c(:,j))^2/(2*b^2));
end
gama=20;

S1=1;
if S1==1          % Adaptive Law
    dw=gama*h*xi'*P*B;
elseif S1==2      % Adaptive Law with UUB
    k1=0.001;
    dw=gama*h*xi'*P*B+k1*gama*norm(x)*w;
end
```

```
dw=dw';
for i=1:1:5
     sys(i)=dw(1,i);
     sys(i+5)=dw(2,i);
end

function sys=mdlOutputs(t,x,u)
global c b kv kp
qd1=u(1);
qd2=u(2);
d_qd1=u(3);
d_qd2=u(4);

dd_qd1=-0.2*(0.5*pi)^2*sin(0.5*pi*t);
dd_qd2=0.2*(0.5*pi)^2*cos(0.5*pi*t);
dd_qd=[dd_qd1;dd_qd2];

q1=u(5);dq1=u(6);q2=u(7);dq2=u(8);

ddq1=u(9);ddq2=u(10);
ddq=[ddq1;ddq2];

e1=q1-qd1;
e2=q2-qd2;
de1=dq1-d_qd1;
de2=dq2-d_qd2;
e=[e1;e2];
de=[de1;de2];

v=13.33;
q01=8.98;
q02=8.75;
g=9.8;

M0=[v+q01+2*q02*cos(q2) q01+q02*cos(q2);
    q01+q02*cos(q2) q01];
C0=[-q02*dq2*sin(q2) -q02*(dq1+dq2)*sin(q2);
    q02*dq1*sin(q2) 0];
G0=[15*g*cos(q1)+8.75*g*cos(q1+q2);
    8.75*g*cos(q1+q2)];

dq=[dq1;dq2];
tol1=M0*(dd_qd-kv*de-kp*e)+C0*dq+G0;

d_M=0.2*M0;
d_C=0.2*C0;
d_G=0.2*G0;
d1=2;d2=3;d3=6;
```

```
d=[d1+d2*norm([e1,e2])+d3*norm([de1,de2])];
%d=[20*sin(2*t);20*sin(2*t)];
f=inv(M0)*(d_M*ddq+d_C*dq+d_G+d);

xi=[e1;e2;de1;de2];
h=zeros(5,1);
for j=1:1:5
    h(j)=exp(-norm(xi-c(:,j))^2/(2*b^2));
end

S=3;
if S==1          %Nominal model based controller
    tol=tol1;
elseif S==2        %Modified computed torque controller
    tol2=-M0*f;
 tol=tol1+tol2;
elseif S==3         %RBF compensated controller
     w=[x(1) x(2) x(3) x(4) x(5);x(6) x(7) x(8) x(9)
        x(10)]';
     fn=w'*h;
     tol2=-M0*fn;
     tol=tol1+1*tol2;
end

sys(1)=tol(1);
sys(2)=tol(2);
sys(3)=f(1);
sys(4)=fn(1);
sys(5)=f(2);
sys(6)=fn(2);
S function for plant:chap6_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;
```

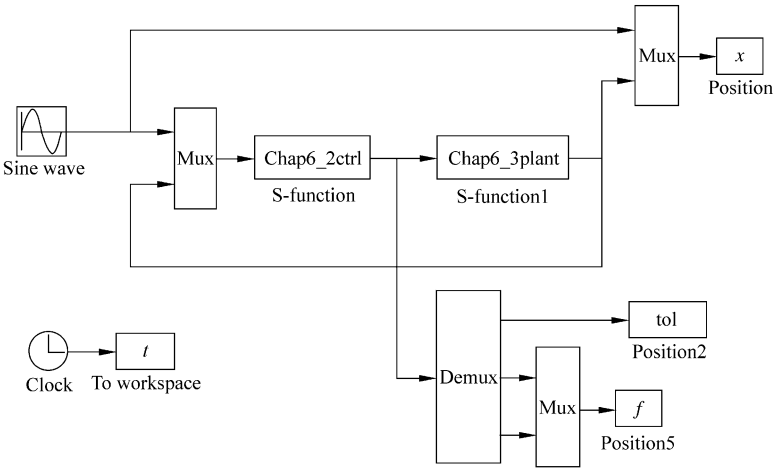
```
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 6;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.6;0.3;0.5;0.5];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
persistent ddx1 ddx2
if t==0
    ddx1=0;
    ddx2=0;
end
qd1=1+0.2*sin(0.5*pi*t);
dqd1=0.2*0.5*pi*cos(0.5*pi*t);
qd2=1-0.2*cos(0.5*pi*t);
dqd2=0.2*0.5*pi*sin(0.5*pi*t);

e1=x(1)-qd1;
e2=x(3)-qd2;
de1=x(2)-dqd1;
de2=x(4)-dqd2;

v=13.33;
q1=8.98;
q2=8.75;
g=9.8;

M0=[v+q1+2*q2*cos(x(3)) q1+q2*cos(x(3));
    q1+q2*cos(x(3)) q1];
C0=[-q2*x(4)*sin(x(3)) -q2*(x(2)+x(4))*sin(x(3));
    q2*x(2)*sin(x(3)) 0];
G0=[15*g*cos(x(1))+8.75*g*cos(x(1)+x(3));
    8.75*g*cos(x(1)+x(3))];
d_M=0.2*M0;
d_C=0.2*C0;
d_G=0.2*G0;

d1=2;d2=3;d3=6;
d=[d1+d2*norm([e1,e2])+d3*norm([de1,de2])];
%d=20*sin(2*t);
tol(1)=u(1);
tol(2)=u(2);
dq=[x(2);x(4)];
```

```
ddq=[ddx1;ddx2];
f=inv(M0)*(d_M*ddq+d_C*dq+d_G+d);

ddx=inv(M0)*(tol'-C0*dq-G0)+1*f;

sys(1)=x(2);
sys(2)=ddx(1);
sys(3)=x(4);
sys(4)=ddx(2);
ddx1=ddx(1);
ddx2=ddx(2);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);
sys(4)=x(4);
Plot program:chap6_2plot.m
close all;

figure(1);
subplot(211);
plot(t,x(:,1),'r',t,x(:,5),'k:','linewidth',2);
xlabel('time(s)');ylabel('Position tracking for link 1');
legend('ideal position for link 1','position tracking for
link 1');
subplot(212);
plot(t,x(:,2),'r',t,x(:,7),'k:','linewidth',2);
xlabel('time(s)');ylabel('Position tracking for link 2');
legend('ideal position for link 2','position tracking for
link 2');

figure(2);
subplot(211);
plot(t,x(:,3),'r',t,x(:,6),'k:','linewidth',2);
xlabel('time(s)');ylabel('Speed tracking for link 1');
legend('ideal speed for link 1','speed tracking for
link 1');
subplot(212);
plot(t,x(:,4),'r',t,x(:,8),'k:','linewidth',2);
xlabel('time(s)');ylabel('Speed tracking for link 2');
legend('ideal speed for link 2','speed tracking for
link 2');

figure(3);
subplot(211);
plot(t,tol(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('Control input of link 1');
subplot(212);
```

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

figure(4);
subplot(211);
plot(t,f(:,1),'r',t,f(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('f1 and fn1');
legend('Practical uncertainties of link 1','Estimation
uncertainties of link 1');
subplot(212);
plot(t,f(:,3),'r',t,f(:,4),'k:','linewidth',2);
xlabel('time(s)');ylabel('f2 and fn2');
legend('Practical uncertainties of link 2','Estimation
uncertainties of link 2');
```

Programs for Sect. 6.2.4.1

Simulink main program: chap6_3sim.mdl



S function for control law and adaptive law: chap6_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);
```

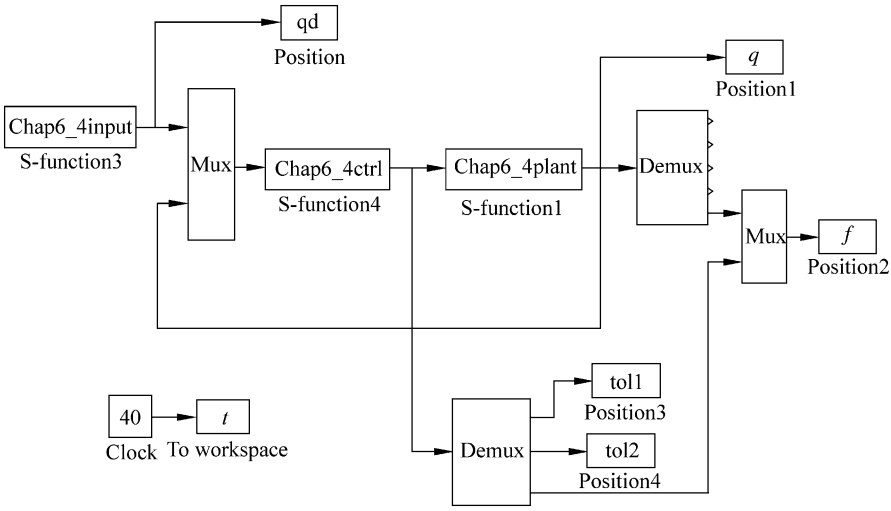
```
case 3,
     sys=mdlOutputs(t,x,u);
case {2,4,9}
     sys=[];
otherwise
     error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes
global node c b Fai
node=7;
c=[-1.5 -1 -0.5 0 0.5 1 1.5;
   -1.5 -1 -0.5 0 0.5 1 1.5];
b=10;
Fai=15;

sizes = simsizes;
sizes.NumContStates = node;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = zeros(1,node);
str = [];
ts = [];
function sys=mdlDerivatives(t,x,u)
global node c b Fai
qd=u(1);
dqd=cos(t);
ddqd=-sin(t);

q=u(2);
dq=u(3);

e=qd-q;
de=dqd-dq;
r=de+Fai*e;

z=[e;de];
for j=1:1:node
     h(j)=exp(-norm(z-c(:,j))^2/(b*b));
end

Gama=100;
for i=1:1:node
sys(i)=Gama*h(i)*r;
end
```

```
function sys=mdlOutputs(t,x,u)
global node c b Fai
qd=u(1);
dqd=cos(t);
ddqd=-sin(t);

q=u(2);
dq=u(3);

e=qd-q;
de=dqd-dq;
r=de+Fai*e;

dqr=dqd+Fai*e;
ddqr=ddqd+Fai*de;

z=[e;de];
w=[x(1:node)]';

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

fn=w*h';
Kv=110;

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

F=15*dq+0.3*sign(dq);
M=10;
f=M*ddqr+F;

fn_norm=norm(fn);
sys(1)=tol;
sys(2)=f;
sys(3)=fn;
S function for plant:chap6_3plant.m
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
    sys = [];
```

```
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.1;0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
M=10;
F=15*x(2)+0.30*sign(x(2));

tol=u(1);

sys(1)=x(2);
sys(2)=1/M*(tol-F);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
Plot program:chap6_3plot.m
close all;

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

figure(2);
plot(t,f(:,1),'r',t,f(:,2),'b','linewidth',2);
xlabel('time(s)');ylabel('f and fn');
legend('Practical          uncertainties','Estimation
uncertainties');
```

Programs for Sect. 6.2.4.2

Simulink main program: chap6_4sim.mdl



Tracking command program: chap6_4input.m

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

```
x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)
qd1=0.1*sin(t);
d_qd1=0.1*cos(t);
dd_qd1=-0.1*sin(t);
qd2=0.1*sin(t);
d_qd2=0.1*cos(t);
dd_qd2=-0.1*sin(t);

sys(1)=qd1;
sys(2)=d_qd1;
sys(3)=dd_qd1;
sys(4)=qd2;
sys(5)=d_qd2;
sys(6)=dd_qd2;
S function for control law and adaptive law:chap6_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 Fai
node=7;
c=[-1.5 -1 -0.5 0 0.5 1 1.5;
   -1.5 -1 -0.5 0 0.5 1 1.5];
b=10;
Fai=5*eye(2);

sizes = simsizes;
sizes.NumContStates = 2*node;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 11;
sizes.DirFeedthrough = 1;
```

```
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = zeros(1,2*node);
str = [];
ts = [];
function sys=mdlDerivatives(t,x,u)
global node c b Fai
qd1=u(1);
d_qd1=u(2);
dd_qd1=u(3);
qd2=u(4);
d_qd2=u(5);
dd_qd2=u(6);

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

q=[q1;q2];

e1=qd1-q1;
e2=qd2-q2;
de1=d_qd1-d_q1;
de2=d_qd2-d_q2;
e=[e1;e2];
de=[de1;de2];
r=de+Fai*e;
qd=[qd1;qd2];
dqd=[d_qd1;d_qd2];
dqr=dqd+Fai*e;
ddqd=[dd_qd1;dd_qd2];
ddqr=ddqd+Fai*de;

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

F=15*eye(node);
for i=1:1:node
    sys(i)=15*h1(i)*r(1);
    sys(i+node)=15*h2(i)*r(2);
end
function sys=mdlOutputs(t,x,u)
```

```
global node c b Fai
qd1=u(1);
d_qd1=u(2);
dd_qd1=u(3);
qd2=u(4);
d_qd2=u(5);
dd_qd2=u(6);

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

q=[q1;q2];

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

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

W_f1=[x(1:node)]';
W_f2=[x(node+1:node*2)]';

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

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

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

fn_norm=norm(fn);
sys(1)=tol(1);
```

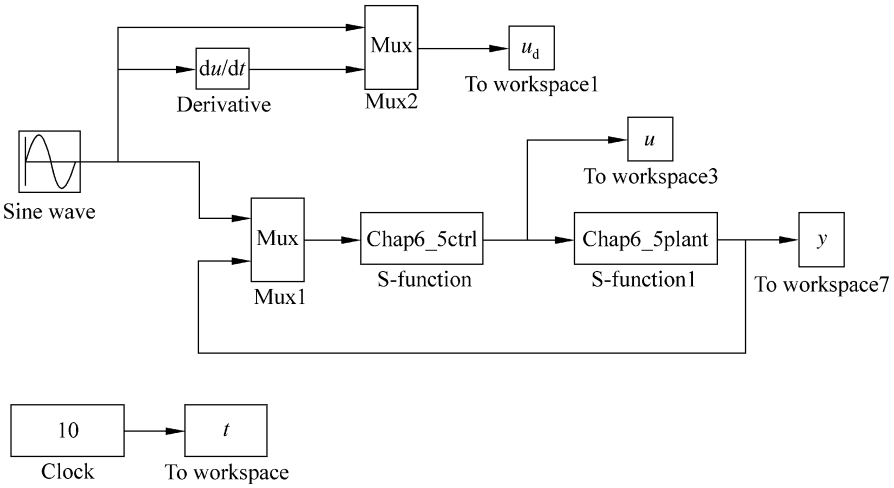
```
sys(2)=tol(2);
sys(3)=fn_norm;
S function for plant:chap6_4plant.m
function [sys,x0,str,ts]=s_function(t,x,u,flag)

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
global p g
sizes = simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 5;
sizes.NumInputs =3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.09 0 -0.09 0];
str=[];
ts=[];
p=[2.9 0.76 0.87 3.04 0.87];
g=9.8;
function sys=mdlDerivatives(t,x,u)
global p g

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

tol=u(1:2);

S=inv(D)*(tol-C*dq-G-F-told);
```

```
sys(1)=x(2);
sys(2)=S(1);
sys(3)=x(4);
sys(4)=S(2);
function sys=mdlOutputs(t,x,u)
global p g
D=[p(1)+p(2)+2*p(3)*cos(x(3)) p(2)+p(3)*cos(x(3));
   p(2)+p(3)*cos(x(3)) p(2)];
C=[-p(3)*x(4)*sin(x(3)) -p(3)*(x(2)+x(4))*sin(x(3));
   p(3)*x(2)*sin(x(3)) 0];
G=[p(4)*g*cos(x(1))+p(5)*g*cos(x(1)+x(3));
   p(5)*g*cos(x(1)+x(3))];
dq=[x(2);x(4)];
F=0.2*sign(dq);
told=[0.1*sin(t);0.1*sin(t)];

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

q1=x(1);
d_q1=dq(1);
q2=x(3);
d_q2=dq(2);
q=[q1;q2];
e1=qd1-q1;
e2=qd2-q2;
de1=d_qd1-d_q1;
de2=d_qd2-d_q2;
e=[e1;e2];
de=[de1;de2];
Fai=5*eye(2);
dqd=[d_qd1;d_qd2];
dqr=dqd+Fai*e;
ddqd=[dd_qd1;dd_qd2];
ddqr=ddqd+Fai*de;
```

```
f=D*ddqr+C*dqr+G+F;
f_norm=norm(f);

sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);
sys(4)=x(4);
sys(5)=f_norm;
Plot program:chap6_4plot.m
close all;

figure(1);
subplot(211);
plot(t,qd(:,1),'r',t,q(:,1),'k:','linewidth',2);
xlabel('time(s)');ylabel('Position tracking for link 1');
legend('ideal position for link 1','position tracking for
link 1');
subplot(212);
plot(t,qd(:,4),'r',t,q(:,3),'k:','linewidth',2);
xlabel('time(s)');ylabel('Position tracking for link 2');
legend('ideal position for link 2','position tracking for
link 2');

figure(2);
subplot(211);
plot(t,qd(:,2),'r',t,q(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('Speed tracking for link 1');
legend('ideal speed for link 1','speed tracking for
link 1');
subplot(212);
plot(t,qd(:,5),'r',t,q(:,4),'k:','linewidth',2);
xlabel('time(s)');ylabel('Speed tracking for link 2');
legend('ideal speed for link 2','speed tracking for
link 2');

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

figure(4);
plot(t,f(:,1),'r',t,f(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('f and fn');
legend('ideal fx','estimation of fx');
```

Programs for Sect. 6.3.3.1

Simulink main program: chap6_5sim.mdl



S function for control law and adaptive law: chap6_5ctrl.m

```
function [sys,x0,str,ts] = Robust_RBF(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 alfa
c=[-1 -0.5 0 0.5 1;
 -1 -0.5 0 0.5 1];
b=50*ones(5,1);
alfa=200;

sizes = simsizes;
sizes.NumContStates = 5;
sizes.NumDiscStates = 0;
```

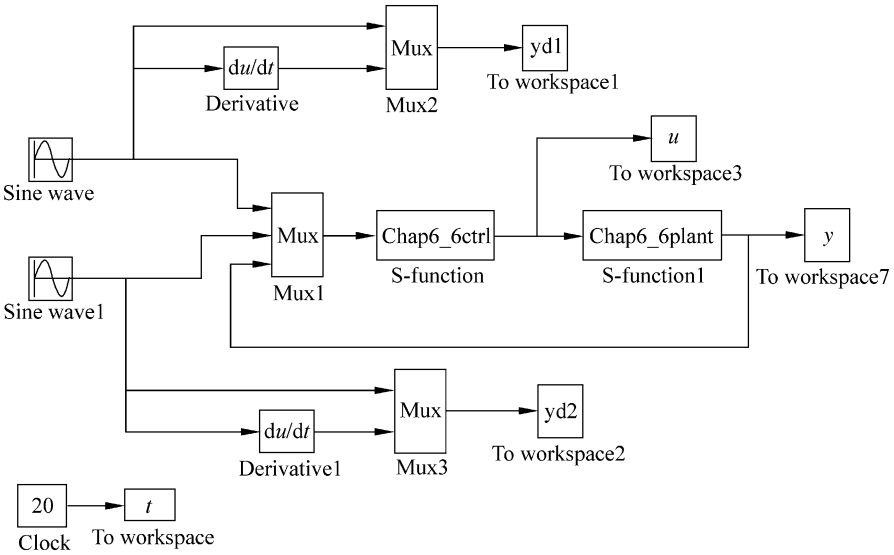
```
sizes.NumOutputs = 2;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [zeros(5,1)];
str = [];
ts = [];
function sys=mdlDerivatives(t,x,u)
global c b alfa
qd=u(1);
dqd=cos(t);
ddqd=-sin(t);

q=u(2);dq=u(3);

e=q-qd;
de=dq-dqd;
x2=de+alfa*e;

xi=[e;de];
h=zeros(5,1);
for j=1:1:5
    h(j)=exp(-norm(xi-c(:,j))^2/(2*b(j)*b(j)));
end
% Adaptive Law
xite=1000;
S=-xite*x2*h';
for i=1:1:5
    sys(i)=S(i);
end
function sys=mdlOutputs(t,x,u)
global c b alfa

qd=u(1);
dqd=cos(t);
ddqd=-sin(t);

q=u(2);dq=u(3);

e=q-qd;
de=dq-dqd;
M=1.0;
w=M*alfa*de;
Gama=0.10;

xi=[e;de];
h=zeros(5,1);
```

```
for j=1:1:5
    h(j)=exp(-norm(xi-c(:,j))^2/(2*b(j)*b(j)));
end
Wf=[x(1) x(2) x(3) x(4) x(5)]';

x2=de+alfa*e;

S=2;
if S==1 %Without RBF compensation
    ut=-w-0.5*1/Gama^2*x2-0.5*x2;
elseif S==2 %With RBF compensation
    ut=-w+Wf'*h-0.5*1/Gama^2*x2-0.5*x2;
end

T=ut+M*ddqd;

NN=Wf'*h;

sys(1)=T;
sys(2)=NN;
S function for plant:chap6_5plant.m
function [sys,x0,str,ts]=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 = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.10 0];
str=[];
```

```
ts=[];
function sys=mdlDerivatives(t,x,u)
J=1.0;
d=150*sign(x(2))+10*x(2);
T=u(1);

sys(1)=x(2);
sys(2)=1/J*(T-d);
function sys=mdlOutputs(t,x,u)
d=150*sign(x(2))+10*x(2);
sys(1)=x(1);
sys(2)=x(2);
sys(3)=d;
Plot program:chap6_5plot.m
close all;

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

figure(2);
subplot(211);
plot(t,yd(:,1)-y(:,1),'k','linewidth',2);
xlabel('time(s)');ylabel('position signal error');
legend('position tracking error');
subplot(212);
plot(t,yd(:,2)-y(:,2),'k','linewidth',2);
xlabel('time(s)');ylabel('Speed tracking error');
legend('speed tracking error');

figure(3);
plot(t,y(:,3),'r',t,u(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('F and Estimated F');
legend('Practical Friction','Estimated Friction');
```

Programs for Sect. 6.3.3.2

Simulink main program: chap6_6sim.mdl



S function for control law and adaptive law: chap6_6ctrl.m

```
function [sys,x0,str,ts] = Robust_RBF(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 alfa
c=0.5*[-3 -2 -1 0 1 2 3;
       -3 -2 -1 0 1 2 3;
       -3 -2 -1 0 1 2 3;
       -3 -2 -1 0 1 2 3];
b=10*ones(7,1);
alfa=20;
```

```
sizes = simsizes;
sizes.NumContStates = 14;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 8;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [zeros(14,1)];
str = [];
ts = [];
function sys=mdlDerivatives(t,x,u)
global c b alfa

qd1=u(1);qd2=u(2);
dqd1=cos(t);dqd2=cos(t);
dqd=[dqd1 dqd2]';
ddqd1=-sin(t);ddqd2=-sin(t);
ddqd=[ddqd1 ddqd2]';

q1=u(3);dq1=u(4);
q2=u(5);dq2=u(6);

e1=q1-qd1;
e2=q2-qd2;
e=[e1 e2]';
de1=dq1-dqd1;
de2=dq2-dqd2;
de=[de1 de2]';
x2=de+alfa*e;

xi=[e1;e2;de1;de2];
%xi=[q1;dq1;q2;dq2];
h=zeros(7,1);
for j=1:1:7
    h(j)=exp(-norm(xi-c(:,j))^2/(2*b(j)*b(j)));
end

% Adaptive Law
xite=1500;
S=-xite*x2*h';
for i=1:1:7
    sys(i)=S(1,i);
    sys(i+7)=S(2,i);
end
function sys=mdlOutputs(t,x,u)
global c b alfa
```

```
qd1=u(1);qd2=u(2);
dqd1=cos(t);dqd2=cos(t);
dqd=[dqd1 dqd2]';
ddqd1=-sin(t);ddqd2=-sin(t);
ddqd=[ddqd1 ddqd2]';

q1=u(3);dq1=u(4);
q2=u(5);dq2=u(6);

e1=q1-qd1;
e2=q2-qd2;
e=[e1 e2]';
de1=dq1-dqd1;
de2=dq2-dqd2;
de=[de1 de2]';

r1=1;r2=0.8;
m1=1;m2=1.5;

M11=(m1+m2)*r1^2+m2*r2^2+2*m2*r1*r2*cos(x(3));
M22=m2*r2^2;
M21=m2*r2^2+m2*r1*r2*cos(x(3));
M12=M21;
M=[M11 M12;M21 M22];

V12=m2*r1*sin(x(3));
V=[-V12*x(4) -V12*(x(2)+x(4));V12*x(1) 0];
g1=(m1+m2)*r1*cos(x(3))+m2*r2*cos(x(1)+x(3));
g2=m2*r2*cos(x(1)+x(3));
G=[g1;g2];

w=M*alfa*de+V*alfa*e;
Gama=0.050;

xi=[e1;e2;de1;de2];
%xi=[q1;dq1;q2;dq2];
h=zeros(7,1);
for j=1:1:5
    h(j)=exp(-norm(xi-c(:,j))^2/(2*b(j)*b(j)));
end
Wf=[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) x(14)]';

S=2;
if S==1 %Without RBF compensation
    ut=-e-w-0.5*1/Gama^2*(de+alfa*e);
elseif S==2 %Without RBF compensation
    x2=de+alfa*e;
    ut=-w+Wf'*h-0.5*1/Gama^2*x2-0.5*x2;
```

```
end
T=ut+M*ddqd+V*dqd+G;

NN=Wf'*h;
sys(1)=T(1);
sys(2)=T(2);
sys(3)=NN(1);
sys(4)=NN(2);
S function for plant:chap6_6plant.m
function [sys,x0,str,ts]=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 = 4;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0 0 0 0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
r1=1;r2=0.8;
m1=1;m2=1.5;

M11=(m1+m2)*r1^2+m2*r2^2+2*m2*r1*r2*cos(x(3));
M22=m2*r2^2;
M21=m2*r2^2+m2*r1*r2*cos(x(3));
M12=M21;
M=[M11 M12;M21 M22];

V12=m2*r1*sin(x(3));
V=[-V12*x(4) -V12*(x(2)+x(4));V12*x(1) 0];
```

```
g1=(m1+m2)*r1*cos(x(3))+m2*r2*cos(x(1)+x(3));
g2=m2*r2*cos(x(1)+x(3));
G=[g1;g2];

D=[10*x(2)+30*sign(x(2)) 10*x(4)+30*sign(x(4))]';

T=[u(1) u(2)]';

S=inv(M)*(T-V*[x(2);x(4)]-G-D);

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

D=[10*x(2)+30*sign(x(2)) 10*x(4)+30*sign(x(4))]';
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);
sys(4)=x(4);
sys(5)=D(1);
sys(6)=D(2);
Plot program:chap6_6plot.m
close all;

figure(1);
subplot(211);
plot(t,yd1(:,1),'r',t,y(:,1),'k:','linewidth',2);
xlabel('time(s)');ylabel('Position tracking of link 1');
legend('ideal position for link 1','position tracking for
link 1');
subplot(212);
plot(t,yd2(:,1),'r',t,y(:,3),'k:','linewidth',2);
xlabel('time(s)');ylabel('Position tracking of link 2');
legend('ideal position for link 2','position tracking for
link 2');

figure(2);
subplot(211);
plot(t,yd1(:,2),'r',t,y(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('Speed tracking of link 2');
legend('ideal speed for link 1','speed tracking for link
1');
subplot(212);
plot(t,yd2(:,2),'r',t,y(:,4),'k:','linewidth',2);
xlabel('time(s)');ylabel('Speed tracking of link 2');
```

```
legend('ideal speed for link 2','speed tracking for
link 2');

figure(3);
subplot(211);
plot(t,y(:,5),'r',t,u(:,3),'k:','linewidth',2);
xlabel('time(s)');ylabel('d1');
legend('ideal delta_f for link 1','estimation of delta_f
for link1');
subplot(212);
plot(t,y(:,6),'r',t,u(:,4),'k:','linewidth',2);
xlabel('time(s)');ylabel('d2');
legend('ideal delta_f for link 2','estimation of delta_f
for link2');
```

References

1. Miller TW, Hewes RP, Galnz FH, Kraft LG (1990) Real time dynamic control of an industrial manipulator using a neural network based learning controller. IEEE Trans Robot Autom 6 (1):1–9
2. Kuperstein M, Wang J (1990) Neural controller for adaptive movements with unforeseen payloads. IEEE Trans Neural Netw 1(1):137–142
3. Khosla PK, Kanade T (1989) Real-time implementation and evaluation of computed-torque scheme. IEEE Trans Robot Autom 5(2):245–253
4. Leahy MB Jr (1990) Model-based control of industrial manipulators: an experimental analysis. J Robot Syst 7(5):741–758
5. Zomaya AY, Nabhan TM (1993) Centralized and decentralized neuro-adaptive robot controllers. Neural Netw 6(2):223–244
6. Feng G (1995) A compensating scheme for robot tracking based on neural networks. Robot Auton Syst 15(3):199–206
7. Lewis FL, Liu K, Yesildirek A (1995) Neural net robot controller with guaranteed tracking performance. IEEE Trans Neural Netw 6(3):703–715
8. Schaft AJV (1992) L_2 gain analysis of nonlinear systems and nonlinear state feedback H_∞ control. IEEE Trans Autom Control 37(6):770–784
9. Wang Y, Sun W, Xiang Y, Miao S (2009) Neural network-based robust tracking control for robots. Int J Intel Autom Soft Comput 15(2):211–222