Chapter 10
Discrete Neural Network Control

Abstract This chapter introduces two kinds of adaptive discrete neural network controllers for discrete nonlinear system, including a direct RBF controller and an indirect RBF controller. For the two control laws, the adaptive laws are designed based on the Lyapunov stability theory; the closed-loop system stability can be achieved.

Keywords Discrete RBF neural network • Discrete system • Adaptive control

10.1 Introduction

The discrete-time implementation of controllers is important. There are two methods for designing the digital controller. One method, called emulation, is to design a controller based on the continuous-time system, then discrete the controller. The other method is to design the discrete controller directly based on the discrete system. In this section, we consider the second approach to design the NN-based nonlinear controller.

Discrete-time adaptive control design is much more complicated than the continuous-time adaptive control design since the discrete-time Lyapunov derivatives tend to have pure and coupling quadratic terms in the states and/or NN weights.

There have been many papers to be published about adaptive neural control for discrete-time nonlinear systems [1–15], where a direct adaptive neural controller for second-order discrete-time nonlinear systems was proposed in [14], and a NN-based adaptive controller without off-line training is presented for a class of discrete-time multi-input multi-output (MIMO) nonlinear systems [15].

In this chapter, we introduce two typical examples for discrete neural network controller design: analysis and simulation.
10.2 Direct RBF Control for a Class of Discrete-Time Nonlinear System

10.2.1 System Description

A nonlinear model is
\[ y(k + 1) = f(y(k), u(k)). \]  
(10.1)

Assumption.
1. The unknown nonlinear function \( f(\cdot) \) is continuous and differentiable.
2. The neural number of hidden neurons is \( l \).
3. Assume that the partial derivative \( g_1 = \left| \frac{\partial f}{\partial u} \right| > 0 \), where both \( \epsilon \) and \( g_1 \) are positive constants.

Assume that \( y_m(k + 1) \) is the system’s desired output at time \( k + 1 \). In the ideal case, there is no disturbance; we can show that if the control input \( u^*(k) \) satisfying
\[ f(y(k), u^*(k)) - y_m(k + 1) = 0. \]  
(10.2)

Then the system’s output tracking error will converge to zero.

Define the tracking error as \( e(k) = y(k) - y_m(k) \), and then the tracking error dynamic equation is given as follows:
\[ e(k + 1) = f(y(k), u(k)) - y_m(k + 1). \]

10.2.2 Controller Design and Stability Analysis

For the system (10.1), a RBF neural network was designed to realize the controller directly [12, 16]. Figure 10.1 shows the closed-loop neural-based adaptive control scheme.

The ideal control input is \( u^* \), and
\[ u^*(k) = u^*(z) = w^*^T h(z) + \epsilon_u(z). \]  
(10.3)

On the compact set \( \Omega_z \), the ideal neural network weights \( w^* \) and the approximation error are bounded by
\[ ||w^*|| \leq w_m, \ |\epsilon_u(z)| \leq \epsilon_1 \]  
(10.4)
where $w_m$ and $\epsilon_1$ are positive constants.

Define $\hat{w}(k)$ as the actual neural network weight value, the control law is designed as by using RBF neural network directly

$$u(k) = \hat{w}^T(k)h(z)$$  \hspace{1cm} (10.5)

where $h(z)$ is Gaussian function output and $z(k)$ is input of RBF.

By noticing (10.3), we have

$$u(k) - u^*(k) = \hat{w}^T(k)h(z) - \left( w^*^T h(z) + \epsilon_u(z) \right)$$

$$= \hat{w}^T(k)h(z) - \epsilon_u(z)$$  \hspace{1cm} (10.6)

where $\hat{w}(k) = \hat{w}(k) - w^*$ is the weight approximation error.

The weight update law was designed as follows [12, 16]:

$$\dot{w}(k + 1) = \dot{w}(k) - \gamma(h(z)e(k + 1) + \sigma \hat{w}(k))$$  \hspace{1cm} (10.7)

where $\gamma > 0$ is a positive number and $\sigma > 0$.

Subtracting $w^*$ to both sides of Eq. (10.6), we have

$$\tilde{w}(k + 1) = \tilde{w}(k) - \gamma(h(z)e(k + 1) + \sigma \hat{w}(k))$$  \hspace{1cm} (10.8)

**Mean Value Theorem.** If a function $f(x)$ is continuous on the closed interval $[a, b]$ and differentiable on the open interval $(a, b)$, then there exists a point $c$ in $(a, b)$ such that

$$f(b) = f(a) + (b - a)f'(c)|_{c \in (a,b)}.$$  \hspace{1cm} (10.9)

Using mean value theorem, let $a = u^*(k)$, $b = u(k) = u^*(k) + \hat{w}^T(k)h(z) - \epsilon_u(z)$, $c = \zeta$, noticing Eqs. (10.2) and (10.6), then

$$f(y(k), u(k)) = f(y(k), u^*(k) + \hat{w}^T(k)h(z) - \epsilon_u(z))$$

$$= f(y(k), u^*(k)) + (\hat{w}^T(k)h(z) - \epsilon_u(z))f_u$$

$$= y_m(k + 1) + (\hat{w}^T(k)h(z) - \epsilon_u(z))f_u$$
where \( f_u = \frac{\partial f}{\partial u} \mid_{u=\zeta}, \ \zeta \in (u^*(k), u(k)) \).

Then we get

\[
e(k + 1) = f(y(k), u(k)) - y_m(k + 1) = (\tilde{w}^T(k)h(z) - \epsilon_u(z))f_u
\]

and

\[
\tilde{w}^T(k)h(z) = \frac{e(k + 1)}{f_u} + \epsilon_u(z). \tag{10.10}
\]

The stability analysis of the closed system is given by Zhang et al. [16] as follows.

Firstly the Lyapunov function is designed as

\[
J(k) = \frac{1}{g_1} e^2(k) + \frac{1}{\gamma} \tilde{w}(k)^T \tilde{w}(k). \tag{10.11}
\]

Then we have

\[
\Delta J(k) = J(k + 1) - J(k)
\]

\[
= \frac{1}{g_1} \left( e^2(k + 1) - e^2(k) \right) + \frac{1}{\gamma} \tilde{w}(k + 1)^T \tilde{w}(k + 1) - \frac{1}{\gamma} \tilde{w}(k)^T \tilde{w}(k)
\]

\[
= \frac{1}{g_1} \left( e^2(k + 1) - e^2(k) \right) + \frac{1}{\gamma} (\tilde{w}(k) - \gamma (h(z)e(k + 1) + \sigma \dot{w}(k)))^T
\]

\[
(\tilde{w}(k) - \gamma (h(z)e(k + 1) + \sigma \dot{w}(k))) - \frac{1}{\gamma} \tilde{w}(k)^T \tilde{w}(k)
\]

\[
= \frac{1}{g_1} \left( e^2(k + 1) - e^2(k) \right) - 2 \tilde{w}(k)^T h(z) e(k + 1) - 2 \sigma \tilde{w}(k)^T \dot{w}(k)
\]

\[
+ \gamma h^T(z) h(z) e^2(k + 1) + 2 \gamma \sigma \dot{w}(k)^T h(z) e(k + 1) + \gamma \sigma^2 \dot{w}(k)^T \dot{w}(k).
\]

Since

\[
|h_i(z)| \leq 1, \ ||h(z)|| \leq \sqrt{l} \leq l, \ h^T(z) h(z) = ||h(z)||^2 \leq l, \ i = 1, 2, \ldots, l
\]

\[
2 \sigma \tilde{w}(k)^T \dot{w}(k) = \sigma \tilde{w}(k)^T (\dot{w}(k) + w^*) + \sigma (\dot{w}(k) - w^*)^T \dot{w}(k)
\]

\[
= \sigma \left( ||\tilde{w}(k)||^2 + ||\tilde{w}(k)||^2 + \tilde{w}(k)^T w^* - w^* \dot{w}(k) \right) = \sigma \left( ||\tilde{w}(k)||^2 + ||\tilde{w}(k)||^2 - ||w^*||^2 \right)
\]

\[
\gamma h^T(z) h(z) e^2(k + 1) \leq \gamma e^2(k + 1)
\]
2γσ\hat{w}^T(k)h(z)e(k + 1) \leq γσl \left[ ||\hat{w}(k)||^2 + e^2(k + 1) \right]

γσ^2\hat{w}^T(k)\hat{w}(k) = γσ^2||\hat{w}(k)||^2,

we obtain

\[ \Delta J(k) \leq \frac{1}{g_1} \left( e^2(k + 1) - e^2(k) \right) - 2\left( \frac{e(k + 1)}{f_u} + ε_u(z) \right)e(k + 1) - \sigma \left( ||\hat{w}(k)||^2 + ||\hat{w}(k)||^2 - ||w^*||^2 \right) + \gamma l e^2(k + 1) + \gamma σ||\hat{w}(k)||^2 \]

\[ = \left( \frac{1}{g_1} - \frac{2}{f_u} + γ(1 + σ)l \right)e^2(k + 1) - \frac{1}{g_1} e^2(k) - 2ε_u(z)e(k + 1) - \sigma ||\hat{w}(k)||^2 + \sigma(-1 + γ l + γ σ)||\hat{w}(k)||^2. \]

Noticing Assumption (10.3), from 0 < ε < f_u ≤ g_1, we can derive that

\[ \frac{1}{g_1} - \frac{2}{f_u} \leq \frac{1}{g_1} - \frac{2}{g_1} = -\frac{1}{g_1} < 0 \]

\[ -2ε_u(z)e(k + 1) \leq k_0 e^2_k + \frac{1}{k_0} e^2(k + 1) \]

where k_0 is a positive number. Thus, we have

\[ \Delta J(k) \leq \left( -\frac{1}{g_1} + γ(1 + σ)l + \frac{1}{k_0} \right)e^2(k + 1) + \sigma(γ l + γ σ - 1)||\hat{w}(k)||^2 \]

\[ - \frac{1}{g_1} e^2(k) - σ||\hat{w}(k)||^2 + σω^2_m + k_0 e^2_k \]

\[ = - \left( \frac{1}{g_1} - (1 + σ)lγ - \frac{1}{k_0} \right)e^2(k + 1) + \sigma((l + σ)γ - 1)||\hat{w}(k)||^2 \]

\[ - \frac{1}{g_1} (e^2(k) - β) - σ||\hat{w}(k)||^2 \]

where β is a positive number as

\[ β = g_1(σω^2_m + k_0 e^2_k) \] (10.12)

Choosing the positive constants as k_0, λ and σ, these constants must be satisfied the following inequalities:

\[ \frac{1}{g_1} - \frac{1}{k_0} \geq 0, \quad \frac{1}{g_1} - (1 + σ)lγ - \frac{1}{k_0} \geq 0, \quad (l + σ)γ - 1 \leq 0, \]

that is,
Then we have $\Delta J(k) \leq 0$ once $e^2(k) \geq \beta$. This states that for all $k \geq 0$, $J(k)$ is bounded because

\[ J(k) = J(0) + \sum_{j=0}^{k} \Delta J(i) < \infty \]

Define compact set $\Omega_e = \{ e | e^2 \leq \beta \}$, then we can see that the tracking error $e(k)$ will converge to $\Omega_e$ if $e(k)$ is out of compact $\Omega_e$.

10.2.3 Simulation Examples

10.2.3.1 First Example: For Linear Plant

Consider a linear discrete-time system as

\[
\begin{align*}
    x_1(k+1) &= x_2(k) \\
    x_2(k+1) &= u(k) \\
    y(k) &= x_1(k).
\end{align*}
\]

Its model is described as the following form

\[ y(k+1) = f(y(k), u(k)). \]

In the program, we set $M = 1$ for linear plant. Since $g_1 \geq \frac{\partial f}{\partial u} = 1$, from simulation test, we can set $g_1 = 5$, and from (10.13), we can choose $k_0 = 10$.

From (10.14), we have $0 < (1 + \sigma)l_\gamma \leq \frac{1}{5} - \frac{1}{10} = \frac{1}{10} = 0.10$; from (10.15), we have $0 < (l + \sigma)\gamma \leq 1$; if we choose $l = 9$, we can get the conditions as: $0 < 9(1 + \sigma)\gamma \leq 0.10$ and $0 < (9 + \sigma)\gamma \leq 1$.

To satisfy (10.14) and (10.15), we can choose $\gamma = 0.01$, $\sigma = 0.001$.

For RBF neural network, the structure is 2-9-1, the input is chosen as $z(k) = [x_1(k) \ x_2(k)]^T$, the parameters of Gaussian function $c_i$ and $b_i$ are chosen as $[-2 -1.5 -1.0 -0.5 0 0.5 1.0 1.5 2]$ and 2.0, and the initial weight value is chosen as random value in the range $(0,1)$. The initial value of the plant is
The reference signal is $y_m(k) = \sin(\frac{\pi}{1000}k)$. The simulation results are shown from Figs. 10.2, 10.3, and 10.4.

The program of this example is chap10_1.m ($M = 1$), which is given in the Appendix.

10.2.3.2 Second Example: For Nonlinear Plant

Consider a nonlinear discrete-time system as

\[
\begin{align*}
x_1(k+1) &= x_2(k) \\
x_2(k+1) &= \frac{x_1(k)x_2(k)(x_1(k) + 2.5)}{1 + x_1^2(k) + x_2^2(k)} + u(k) + 0.1u^3(k).
\end{align*}
\]

The model is described as the following form

\[
y(k+1) = f(y(k), u(k)).
\]

In the program, we set $M = 2$ for nonlinear plant. Since $g_1 \geq \frac{\partial f}{\partial u} = 1 + 0.3u^2(k)$, from simulation test, we can set $g_1 = 5$, and from (10.13), we can choose $k_0 = 10$.

From (10.14), we have $0 < (1 + \sigma)\gamma \leq \frac{1}{5} - \frac{1}{10} = \frac{1}{10} = 0.10$, from (10.15), we have $0 < (I + \sigma)\gamma \leq 1$, if we choose $I = 9$, we can get the conditions as: $0 < 9(1 + \sigma)\gamma \leq 0.10$ and $0 < (9 + \sigma)\gamma \leq 1$.

To satisfy (10.14) and (10.15), we can choose $\gamma = 0.01$, $\sigma = 0.001$.  

![Fig. 10.2 Position tracking ($M = 1$)](image)
For RBF neural network, the structure is 2-9-1, the input is chosen as $z(k) = [x_1(k) \ x_2(k)]^T$, the parameters of Gaussian function $c_i$ and $b_i$ are chosen as $[-2 \ -1.5 \ -1.0 \ -0.5 \ 0 \ 0.5 \ 1.0 \ 1.5 \ 2]$ and 2.0, and the initial weight value is chosen as random value in the range (0,1). The initial value of the plant is $[0 \ 0]$. The reference signal is $y_m(k) = \sin(\frac{\pi}{1000}k)$. The simulation results are shown from Figs. 10.5, 10.6, and 10.7.

The program of this example is chap10_1.m ($M = 2$), which is given in the Appendix.
10.3 Adaptive RBF Control for a Class of Discrete-Time Nonlinear System

10.3.1 System Description

Consider a nonlinear discrete system as follows:

\[ y(k+1) = f(x(k)) + u(k) \]  \hspace{1cm} (10.16)
where \( x(k) = [y(k) y(k-1) \ldots y(k-n+1)]^T \) is the state vector, \( u(k) \) is the control input, and \( y(k) \) is the plant output. The nonlinear smooth function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is assumed unknown.

### 10.3.2 Traditional Controller Design

The tracking error \( e(k) \) is defined as \( e(k) = y(k) - y_d(k) \). If \( f(x(k)) \) were known, a feedback linearization-type control law can be designed as

\[
u(k) = y_d(k+1) - f(x(k)) - c_1 e(k).
\]

(10.17)

Submitting (10.17) into (10.16), we can get an asymptotical convergence error dynamic system as

\[
e(k+1) + c_1 e(k) = 0
\]

(10.18)

where \( |c_1| < 1 \).

### 10.3.3 Adaptive Neural Network Controller Design

If \( f(x(k)) \) is unknown, RBF neural network can be used to approximate \( f(x(k)) \). The network output is given as
\[ \hat{f}(x(k)) = \hat{w}(k)^T h(x(k)) \]  

where \( \hat{w}(k) \) denotes the network output weight vector and \( h(x(k)) \) denotes the vector of Gaussian basis functions.

Given any arbitrary nonzero approximation error bound \( \varepsilon_f \), there exist some optimal weight vector \( w^* \) such that

\[ f(x) = \hat{f}(x, w^*) - \Delta_f(x) \]  

where \( \Delta_f(x) \) denotes the optimal network approximation error, and \( |\Delta_f(x)| < \varepsilon_f \).

Then we can get the general network approximation error as

\[ \tilde{f}(x(k)) = \hat{f}(x(k)) - \hat{w}(k)^T h(x(k)) \]  

where \( \hat{w}(k) = \hat{w}(k) - w^* \).

The control law with RBF approximation was proposed in [17] as follows

\[ u(k) = y_d(k+1) - \hat{f}(x(k)) - c_1 e(k). \]  

Figure 10.8 shows the closed-loop neural-based adaptive control scheme. Substituting (10.22) into (10.16), yields

\[ e(k+1) = \tilde{f}(x(k)) - c_1 e(k). \]

Thus,

\[ e(k) + c_1 e(k-1) = \tilde{f}(x(k-1)). \]  

The term (10.23) can also be expressed as

\[ e(k) = \Gamma^{-1}(z^{-1}) \tilde{f}(x(k-1)) \]  

where \( \Gamma(z^{-1}) = 1 + c_1 z^{-1}, z^{-1} \) denotes the discrete-time delay operator.

Define a new augmented error as [17]

\[ e_1(k) = \beta (e(k) - \Gamma^{-1}(z^{-1}) v(k)) \]  

where \( \beta > 0 \).

Substituting (10.24) into (10.25) yields
which leads to the relation as

\[
e_1(k - 1) = \frac{\beta (\tilde{f}(x(k - 1)) - v(k)) - e_1(k)}{c_1}.
\]

The adaptive law as was proposed in [17] as

\[
\Delta \tilde{w}(k) = \begin{cases} 
\frac{\beta}{\gamma c_1} h(x(k - 1))e_1(k) & \text{if } |e_1(k)| > \varepsilon_f / G \\
0 & \text{if } |e_1(k)| \leq \varepsilon_f / G
\end{cases}
\]

where \(\Delta \tilde{w}(k) = \tilde{w}(k) - \tilde{w}(k - 1)\), \(\gamma\) and \(G\) are strictly positive constants.

### 10.3.4 Stability Analysis

The discrete-time Lyapunov function is designed by [17] as

\[
V(k) = e_1^2(k) + \gamma \tilde{w}^T(k)\tilde{w}(k)
\]

The first difference is

\[
\Delta V(k) = V(k) - V(k - 1) \\
= e_1^2(k) - e_1^2(k - 1) + \gamma (\tilde{w}^T(k) + \tilde{w}^T(k - 1)) (\tilde{w}(k) - \tilde{w}(k - 1)).
\]

The stability proof was given with the following three steps [17]. Firstly, using (10.26) for \(e_1(k - 1)\), it follows that
\[ \Delta V(k) = e_1^2(k) - \frac{e_1^2(k) + \beta^2(\tilde{f}(x(k-1)) - v(k))^2}{c_1^2} - 2\beta(\tilde{f}(x(k-1)) - v(k))e_1(k) \]

\[ + \gamma \left( (\hat{w}(k) - w^*)^T + (\hat{w}(k-1) - w^*)^T \right) (\hat{w}(k) - w^*) - (\hat{w}(k-1) - w^*) \]

\[ = -V_1 + \frac{2\beta(\tilde{f}(x(k-1)) - v(k))e_1(k)}{c_1^2} + \gamma (\Delta\hat{w}^T(k) + 2\hat{w}^T(k-1))\Delta\hat{w}(k) \]

where \( V_1 = \frac{e_1^2(k)(1 - c_1^2)}{c_1^2} + \frac{\beta^2(\tilde{f}(x(k-1)) - v(k))^2}{c_1^2} \geq 0. \)

Secondly, substituting for \( \tilde{f}(x(k-1)) \) via (10.21) yields

\[ \Delta V(k) = -V_1 + \frac{2\beta\left( -\hat{w}(k-1)^T h(x(k-1)) - \Delta_f(x(k-1)) - v(k) \right) e_1(k)}{c_1^2} \]

\[ + \gamma \Delta\hat{w}^T(k)\Delta\hat{w}(k) + 2\gamma \hat{w}^T(k-1)\Delta\hat{w}(k) \]

\[ = -V_1 + 2\hat{w}^T(k-1) \left( \gamma \Delta\hat{w}(k) - \frac{\beta}{c_1^2} h(x(k-1))e_1(k) \right) \]

\[ - \frac{2\beta}{c_1^2} (\Delta_f(x(k-1)) + v(k))e_1(k) + \gamma \Delta\hat{w}^T(k)\Delta\hat{w}(k). \]

Thirdly, substituting the adaptive law (10.27) into above, \( \Delta V(k) \) is

\[
\Delta V(k) = \begin{cases} 
-V_1 - \frac{2\beta}{c_1^2} (\Delta_f(x(k-1)) + v(k))e_1(k) + \\
\left( \frac{\beta}{\sqrt{\gamma} c_1} \right)^2 h^T(x(k-1))h(x(k-1))e_1^2(k), \text{ if } |e_1(k)| > \varepsilon_f/G \\
-V_1 - \frac{2\beta}{c_1^2} \left[ (\hat{w}^T(k-1)h(x(k-1))) + \right. \\
\left. (v(k) + \Delta_f(x(k-1))e_1(k)) \right], \text{ if } |e_1(k)| \leq \varepsilon_f/G 
\end{cases}
\]

The auxiliary signal \( v(k) \) must also be designed so that \( e_1(k) \to 0 \) could deduce \( e(k) \to 0 \). The auxiliary signal is designed as [17]

\[ v(k) = v_1(k) + v_2(k) \tag{10.30} \]

with

\[ v_1(k) = \frac{\beta}{2\sqrt{\gamma} c_1} h^T(x(k-1))h(x(k-1))e_1(k) \]

\[ v_2(k) = G e_1(k). \]

If \( |e_1(k)| > \varepsilon_f/G \), substituting for \( v(k) \) in (10.30) to (10.29), it follows that
\[
\Delta V(k) = -V_1 - \frac{2\beta}{c_1^2} \left( \Delta f(x(k-1)) + Ge_1(k) \right) e_1(k)
\leq -\frac{2\beta}{c_1^2} \left( \Delta f(x(k-1)) + Ge_1(k) \right) e_1(k).
\]

Since \(|\Delta f(x)|<\varepsilon_f\) and \(|e_1(k)|>\varepsilon_f/G\), then \(|e_1(k)|>\frac{|\Delta f(x(k-1))|}{G}\) and \(e_1^2(k)>-\frac{\Delta f(x(k-1))e_1(k)}{G}\), thus \((\Delta f(x(k-1)) + Ge_1(k))e_1(k)>0\), then \(\Delta V(k)<0\).

If \(|e_1(k)| \leq \varepsilon_f/G\), tracking performance can be satisfied and \(\Delta V(k)\) can be taken on any value.

In the simulation, we give three remarks as follows:

Remark 1. From (10.25), we have \(e_1(k) = \beta \left( e(k) - \frac{1}{1+c_1z^{-1}} v(k) \right)\), then \(e_1(k) \times (1+c_1z^{-1}) = \beta (e(k)(1+c_1z^{-1}) - v(k))\); therefore,

\[
e_1(k) = -c_1 e_1(k-1) + \beta (e(k) + c_1 e(k-1) - v(k))
\]  \hspace{1cm} (10.31)

Remark 2. If \(k \to \infty\), \(e_1(k) \to 0\), from (10.30), we have \(v(k) \to 0\), and then from (10.31), \(e(k) + c_1 e(k-1) \to 0\), considering \(|c_1|<1\), we get \(e(k) \to 0\).

Remark 3. Consider \(v(k)\) as a virtual variable, for (10.30), let \(v'(k) = \frac{\beta}{2c_1} h^T(x(k-1))h(x(k-1))\), then we get \(v(k) = (v'(k) + G)e_1(k)\); substituting \(v(k)\) into (10.31), we have \(e_1(k) = -c_1 e_1(k-1) + \beta (e(k) + c_1 e(k-1) - (v'(k) + G)e_1(k))\), then

\[
e_1(k) = -c_1 e_1(k-1) + \beta (e(k) + c_1 e(k-1)) \frac{1 + \beta (v'(k) + G)}{1 + \beta (v'(k) + G)}. \hspace{1cm} (10.32)
\]

### 10.3.5 Simulation Examples

#### 10.3.5.1 First Example: Linear Discrete-Time System

Consider a linear discrete-time system as

\[
y(k) = 0.5y(k-1) + u(k-1)
\]

where \(f(x(k-1)) = 0.5y(k-1)\).
We use RBF to approximate $f(x(k - 1))$. For RBF neural network, the structure is set as 1-9-1; from $f(x(k - 1))$ expression, only one input is chosen as $y(k - 1)$; the parameters of Gaussian function $c_i$ and $b_i$ are chosen as $[-1 \ -0.5 \ 0 \ 0.5 \ 1]$ and $15(i = 1, j = 1, 2, \ldots , 5)$; and the initial weight value is chosen as random value in the range $(0,1)$. The initial value of the plant is set as zero. The reference signal is $y_d(k) = \sin 2\pi t$. Using the control law (10.22) with adaptive law (10.27), $e_1(k)$ is calculated by (10.32); the parameters are chosen as $c_1 = -0.01, \beta = 0.001, \gamma = 0.001, \gamma = 0.001, G = 50000, \epsilon_f = 0.003$. The results are shown in Figs. 10.9, 10.10, and 10.11.
10.3.5.2 Second Example: Nonlinear Discrete-Time System

Consider a nonlinear discrete-time system as

\[ y(k) = \frac{0.5y(k-1)(1-y(k-1))}{1 + \exp(-0.25y(k-1))} + u(k-1) \]

where \( f(x(k-1)) = \frac{0.5y(k-1)(1-y(k-1))}{1 + \exp(-0.25y(k-1))} \).

Firstly, we assume \( f(x(k-1)) \) is known, use the control law (10.17), and set \( c_1 = -0.01 \), the results are shown in Figs. 10.12 and 10.13.

Then we use RBF to approximate \( f(x(k-1)) \). For RBF neural network, the structure is set as 1-9-1; from \( f(x(k-1)) \) expression, only one input \( y(k-1) \) is chosen; the parameters of Gaussian function \( c_i \) and \( b_j \) are chosen as \([-2 -1.5 -1.0 -0.5 0 0.5 1.0 1.5 2]\) and \( 15(i = 1, j = 1, 2, \ldots, 9) \), and the initial weight value is chosen as random value in the range (0,1). The initial value of the plant is set as zero. The reference signal is \( y_d(k) = \sin t \). Using the control law (10.22) with adaptive law (10.27), \( e_1(k) \) is calculated by (10.32); the parameters are chosen as \( c_1 = -0.01, \beta = 0.001, \gamma = 0.001, \gamma = 0.001, G = 50000, \varepsilon_f = 0.003 \). The results are shown in Figs. 10.14, 10.15, and 10.16.

The program of this example is chap10_3.m and chap10_4.m, which are given in the Appendix.
Consider a nonlinear discrete-time system as

\[ y(k) = f(x(k - 1)) + u(k - 1) \]

where \( f(x(k - 1)) = \frac{1.5y(k-1)y(k-2)}{1+y^2(k-1)+y^2(k-2)} + 0.35 \sin(y(k-1) + y(k-2)) \).

Then we use RBF to approximate \( f(x(k - 1)) \). For RBF neural network, the structure is 2-9-1; from \( f(x(k - 1)) \) expression, two inputs are chosen as \( y(k - 1) \) and \( y(k - 2) \), the parameters of Gaussian function \( c_{ij} \) and \( b_{j} \) are chosen as:

\[
\begin{bmatrix}
-2 & -1.5 & -1.0 & -0.5 & 0 & 0.5 & 1.0 & 1.5 & 2 \\
-2 & -1.5 & -1.0 & -0.5 & 0 & 0.5 & 1.0 & 1.5 & 2
\end{bmatrix}
\quad \text{and} \quad
15(i = 1, 2, \ldots, 9); \quad \text{and}
\]

10.3.5.3 Third Example: Nonlinear Discrete-Time System

Consider a nonlinear discrete-time system as

\[ y(k) = f(x(k - 1)) + u(k - 1) \]

where \( f(x(k - 1)) = \frac{1.5y(k-1)y(k-2)}{1+y^2(k-1)+y^2(k-2)} + 0.35 \sin(y(k-1) + y(k-2)) \).

Then we use RBF to approximate \( f(x(k - 1)) \). For RBF neural network, the structure is 2-9-1; from \( f(x(k - 1)) \) expression, two inputs are chosen as \( y(k - 1) \) and \( y(k - 2) \), the parameters of Gaussian function \( c_{ij} \) and \( b_{j} \) are chosen as:

\[
\begin{bmatrix}
-2 & -1.5 & -1.0 & -0.5 & 0 & 0.5 & 1.0 & 1.5 & 2 \\
-2 & -1.5 & -1.0 & -0.5 & 0 & 0.5 & 1.0 & 1.5 & 2
\end{bmatrix}
\quad \text{and} \quad
15(i = 1, 2, \ldots, 9); \quad \text{and}
\]
the initial weight value is chosen as random value in the range $(0,1)$. The initial value of the plant is set as zero. The reference signal is $y_d(k) = \sin t$. Using the control law (10.22) with adaptive law (10.27), $e_1(k)$ is calculated by (10.32); the parameters are chosen as $c_1 = -0.01, \beta = 0.001, \gamma = 0.001, \gamma = 0.001, G = 50000, \epsilon_f = 0.003$. The results are shown in Figs. 10.17, 10.18, and 10.19.

The program of this example is chap10_5.m, which is given in the Appendix.
Appendix

Programs for Sect. 10.2.3

Simulation program: chap10_1.m

```matlab
%Discrete neural controller
clear all;
close all;
L=9; %Hidden neural nets
c=[-2 -1.5 -1 -0.5 0 0.5 1 1.5 2;
-2 -1.5 -1 -0.5 0 0.5 1 1.5 2];
```

Fig. 10.16 $f(x(k - 1))$ and its estimation

Fig. 10.17 Position tracking
Fig. 10.18 Control input

Fig. 10.19 $f(x(k-1))$ and its estimation

```matlab
b=2;
w=rand(L,1);
w_1=w;
u_1=0;
x1_1=0; x1_2=0;
x2_1=0;
z=[0,0]';
Gama=0.01; rou=0.001;
L*(1+rou)*Gama <= 1/g1-1/k0
(L+rou)*Gama <= 1.0
for k=1:1:10000
time(k)=k;
```
ym(k) = sin(pi/1000*k);
y(k) = x1_1; %tol = 1
e(k) = y(k) - ym(k);

M = 2;
if M == 1 % Linear model
    x1(k) = x2_1;
    x2(k) = u_1;
elseif M == 2 % Nonlinear model
    x1(k) = x2_1;
    x2(k) = (x1(k) * x2_1 * (x1(k) + 2.5)) / (1 + x1(k)^2 + x2_1^2) + u_1 + 0.1*u_1^3;
end

z(1) = x1(k); z(2) = x2(k);
for j = 1:1:L
    h(j) = exp(-norm(z - c(:,j))^2 / (2*b^2));
end
w = w_1 - Gama*(h'*e(k) + rou*w_1);
w_1 = norm(w);

u(k) = w'*h;
%u(k) = 0.20*(ym(k) - x1(k)); % P control
x1_2 = x1_1;
x1_1 = x1(k);
x2_1 = x2(k);
w_1 = w;
u_1 = u(k);
end

figure(1);
plot(time, ym, 'r', time, x1, 'k:', 'linewidth', 2);
xlabel('k'); ylabel('ym, y');
legend('Ideal position signal', 'Position tracking');
figure(2);
plot(time, u, 'r', 'linewidth', 2);
xlabel('k'); ylabel('Control input');
figure(3);
plot(time, wn, 'r', 'linewidth', 2);
xlabel('k'); ylabel('Weight Norm');

**Programs for Sect. 10.3.5.1**

Simulation program: chap10_2.m
%Discrete RBF controller
clear all;
close all;
ts=0.001;
c1=-0.01;
beta=0.001;
epcf=0.003;
gama=0.001;
G=50000;
b=15;
c=[-1 -0.5 0 0.5 1];
w=rands(5,1);
w_1=w;
u_1=0;
y_1=0;
e_1=0;
f_1=0;
for k=1:1:2000
    time(k)=k*ts;
yd(k)=sin(2*pi*k*ts);
yd1(k)=sin(2*pi*(k+1)*ts);
%Nonlinear plant
fx(k)=0.5*y_1;
y(k)=fx_1+u_1;
e(k)=y(k)-yd(k);
x(1)=y_1;
for j=1:1:5
    h(j)=exp(-norm(x-c(:,j))^2/(2*b^2));
end
v1_bar(k)=beta/(2*gama*c1^2)*h*h';
e1(k)=(-c1*e_1+beta*(e(k)+c1*e_1))/(1+beta*(v1_bar(k)+G));
if abs(e1(k))>epcf/G
    w=w_1+beta/(gama*c1^2)*h'*e1(k);
elseif abs(e1(k))<=epcf/G
    w=w_1;
end
fnn(k)=w'*h';
u(k)=yd1(k)-fnn(k)-c1*e(k);
%u(k)=yd1(k)-fx(k)-c1*e(k); %With precise fx
Programs for Sect. 10.3.5.2

Simulation program with known $f(x(k-1))$: chap10_3.m

```matlab
%Discrete controller
clear all;
close all;
ts = 0.001;
c1 = -0.01;
u_1 = 0; y_1 = 0;
fx_1 = 0;
for k = 1:1:20000
    time(k) = k * ts;
yd(k) = sin(k * ts);
yd1 = sin((k+1) * ts);
end
%Nonlinear plant
fx(k) = 0.5 * y_1 * (1-y_1) / (1+exp(-0.25*y_1));
y(k) = fx_1 + u_1;
e(k) = y(k) - yd(k);
u(k) = yd1 - fx(k) - c1 * e(k);
y_1 = y(k);
```
Simulation program with unknown \( f(x(k - 1)) \): chap10_4.m

```matlab
% Discrete RBF controller
clear all;
close all;
ts = 0.001;
c1 = -0.01;
beta = 0.001;
epcf = 0.003;
gama = 0.001;
G = 50000;

b = 15;
c = [-2 -1.5 -1 -0.5 0 0.5 1 1.5 2];
w = rands(9, 1);
w_1 = w;

u_1 = 0;
y_1 = 0;
e1_1 = 0;
e_1 = 0;
fx_1 = 0;
for k = 1:1:10000
    time(k) = k * ts;
yd(k) = sin(k * ts);
yd1(k) = sin((k + 1) * ts);
% Nonlinear plant
fx(k) = 0.5 * y_1 * (1 - y_1) / (1 + exp(-0.25 * y_1));
y(k) = fx_1 + u_1;
e(k) = y(k) - yd(k);

x(1) = y_1;
for j = 1:1:9
    h(j) = exp(-norm(x - c(:, j))^2 / (2 * b^2));
end
```
\[ v_{1\text{\_bar}}(k) = \frac{\beta}{2gama*c_1^2}h*h' \]
\[ e_1(k) = \frac{-c_1*e_1+\beta*(e(k)+c_1*e_1)}{1+\beta*(v_{1\text{\_bar}}(k)+G)} \]

\[
\begin{align*}
\text{if abs(e}_1(k)\text{)} & > \text{epcf}/G \\
\qquad w & = \text{w}_1+\frac{\beta}{gama*c_1^2}h'*e_1(k) \\
\text{elseif abs(e}_1(k)\text{)} & < \text{epcf}/G \\
\qquad w & = \text{w}_1 \\
\end{align*}
\]

\[ f_{\text{nn}}(k) = w'*h' \]
\[ u(k) = yd_1(k) - f_{\text{nn}}(k) - c_1*e(k) \]
%\[ u(k) = yd_1(k) - f_{x}(k) - c_1*e(k) \] %With precise \( f_{x} \)
\[ f_{x_1} = f_{x}(k) \]
\[ y_1 = y(k) \]
\[ w_1 = w \]
\[ u_1 = u(k) \]
\[ e_1_1 = e_1(k) \]
\[ e_1 = e(k) \]

\text{end}

\text{figure(1);}
\text{plot(time,yd,'r',time,y,'k:','linewidth',2);}
\text{xlabel('time(s)');ylabel('yd,y');}
\text{legend('Ideal position signal','Position tracking');}
\text{figure(2);}
\text{plot(time,u,'r','linewidth',2);}
\text{xlabel('time(s)');ylabel('Control input');}
\text{figure(3);}
\text{plot(time,fx,'r',time,f_{nn},'k:','linewidth',2);}
\text{xlabel('time(s)');ylabel('fx and fx estimation');}

\textbf{Programs for Sect. 10.3.5.3}

Simulation program: chap10_5.m

%Discrete RBF controller
\text{clear all;}
\text{close all;}
\text{ts=0.001;}
\text{c1=-0.01;}
\text{beta=0.001;}
\text{epcf=0.003;}
\text{gama=0.001;}
\text{G=50000;}

b=15;
c=[-2 -1.5 -1 -0.5 0 0.5 1 1.5 2; -2 -1.5 -1 -0.5 0 0.5 1 1.5 2];
w=rands(9,1);
w_1=w;

u_1=0;y_1=0;y_2=0;
e_1_1=0;e_1=0;
x=[0 0]’;
fx_1=0;
for k=1:1:10000
time(k)=k*ts;
yd(k)=sin(k*ts);
yd1(k)=sin((k+1)*ts);
%Linear model
fx(k)=1.5*y_1*y_2/(1+y_1^2+y_2^2)+0.35*sin(y_1+y_2);
y(k)=fx_1+u_1;
e(k)=y(k)-yd(k);
x(1)=y_1;x(2)=y_2;
for j=1:1:9
    h(j)=exp(-norm(x-c(:,j))^2/(2*b^2));
end
v1_bar(k)=beta/(2*gama*c1^2)*h*h’;
e1(k)=(-c1*e1_1+beta*(e(k)+c1*e_1))/(1+beta*(v1_bar
    (k)+G));
if abs(e1(k))>epcf/G
    w=w_1+beta/(gama*c1^2)*h’*e1(k);
elseif abs(e1(k))<=epcf/G
    w=w_1;
end
fnn(k)=w’*h’;

u(k)=yd1(k)-fnn(k)-c1*e(k);
%u(k)=yd1(k)-fx(k)-c1*e(k); %With precise fx
%u(k)=0.10*(x1d(k)-x1(k)); %P control
fx_1=fx(k);
y_2=y_1;
y_1=y(k);
w_1=w;
u_1=u(k);
e1_1=e1(k);
e_1=e(k);
end
figure(1);
plot(time,yd,'r',time,y,'k:','linewidth',2);
xlabel('time(s)');ylabel('yd,y');
legend('Ideal position signal','Position tracking');
figure(2);
plot(time,u,'r','linewidth',2);
xlabel('time(s)');ylabel('Control input');
figure(3);
plot(time,fx,'r',time,fnn,'k:','linewidth',2);
xlabel('time(s)');ylabel('fx and fx estimation');

References