A Radial-Basis-Function-Network-Based Controller Derived from a Neural-Network-Based Controller and its Application to Controlling Mechanical Systems*

Kazuhiko TAKAHASHI** and Takayuki YAMADA***

In this paper we describe a method of designing a radial-basis-function-network-based(RBFN) controller. The RBFN controller is derived from a neural-network-based(NN) controller using differential operator focusing on the sigmoid function derivative of the NN controller. To overcome the Jacobian problem, the RBFN controller uses a learning algorithm and a neural identifier which uses an adaptive algorithm to estimate the plant Jacobian. A conventional feedback controller is incorporated into the RBFN controller to ensure both robustness and stability at the beginning of the learning process. Simulation results for mathematical plants demonstrate the applicability of the RBFN controller for controlling nonlinear systems and experimental results for 1-degree-of-freedom robots demonstrate its usefulness for controlling practical systems. Application to controlling an ODD positioner and a tunneling machine demonstrates the effectiveness of the RBFN controller for controlling mechanical systems.

Key Words: Radial Basis Function, Neural Networks, Control, Identification, Nonlinear Systems, Force Control, Positioning Control, Optical Disk Drive, Tunneling Machine

1. Introduction

Neural networks(NNs), which are modeled on biological neural networks, have excellent capabilities such as nonlinear mapping and learning. To apply the flexibility and learning ability of NNs to control systems, many types of neural-network-based controllers(neural controllers) have been studied⁽¹⁾⁻⁽¹⁰⁾. Features of neural controllers include (1) nonlinear control ability using nonlinear mapping of NNs, (2) generalization ability based on function knowledge of NNs, (3) optimization ability of NNs which can be used to tune system parameters. Although many studies on neural controllers for controlling nonlinear systems have been carried out, many problems remain. For example, it is known that many neurons

are required to achieve nonlinear mapping sufficiently using NNs. Consequently, there is a problem: it takes a long time to converge NN learning with a large number of neurons.

Several methods for using a radial basis function network(RBFN) have been proposed⁽¹¹⁾ in order to achieve fast convergence of a network's learning of complex nonlinear functions/mappings. Since the RBFN architecture is a linear connection weighted network of radial basis functions, such as Gaussian functions, fast linear algorithms such as the least squares algorithm can be used for the network learning. RBFNs are simple compared to NNs; therefore, RBFNs can be used in signal processing applications instead of two-layer NNs. Recently, there has been interest in using RBFNs for identification/control of nonlinear systems and several studies⁽¹²⁾⁻⁽¹⁶⁾ have been carried out. In these studies, the RBFN is almost always used to identify the nonlinear function/direct transfer function of an object plant. Then, this function is used to calculate the compensation input for the plant control input. However, there have been few studies on a direct controller based on the RBFN.

In our previous studies^{(9),(16)-(18)}, a direct neural

* Received 5th January, 1996.

** Integrated Information & Energy Systems Laboratories, Nippon Telegraph and Telephone Corporation, 3-9-11 Midori-cho, Musashino-shi, Tokyo 180, Japan

*** Access Network Systems Laboratories, Nippon Telegraph and Telephone Corporation, Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan

controller, in which the NN output is used as the plant control input (i.e. the plant inverse transfer function is determined using the NN after learning convergence), was investigated and its characteristics were discussed. Since the direct neural controller is the simplest type, it is suitable for investigating the basic characteristics of the neural controller and fully exploits the NN capability. In general, a sigmoid function is used as the neuron activation function of the neural controller. Study of the sigmoid function shows that the shape of the derivative of a sigmoid function is similar to that of a Gaussian function. Therefore, a network which uses the derivative of a sigmoid function as the activation function can be treated as a kind of RBFN⁽¹⁹⁾. In this study, by focusing on this characteristic of the sigmoid function, we derive a method of designing a direct RBFN controller by applying a differential operator to a direct neural controller. We also describe applications in which mechanical systems are controlled using the RBFN controller. Using our design method it should be possible to make an interface between a neural controller and an RBFN controller. Since the nonlinear mapping performance of an RBFN also depends on the chosen RBF centers, it is advantageous to simultaneously update the RBF centers and weights. In this study, however, the RBF centers are fixed in order to simplify the RBFN learning and only the weights of the RBFN are updated using a generalized δ -rule⁽²⁰⁾.

In this paper we describe an RBFN controller for controlling mechanical systems. In section 2, we describe a method of designing a direct RBFN controller derived from a direct neural controller using a differential operator. In section 3, the results of simulations carried out using a discrete-time single-input single-output plant to test the feasibility and characteristics of the RBFN controller are presented. In section 4, experimental results for controlling 1-degree-of-freedom robots using the RBFN controller are given and the usefulness of this controller is thus demonstrated. In section 5, examples of application to control of mechanical systems, such as an optical disk drive (ODD) and a tunneling machine, are described to demonstrate the effectiveness of the RBFN controller.

2. RBFN Controller Design

In this section, we describe a method of designing an RBFN controller, based on the sigmoid function derivative of a neural controller. Using a differential operator, the RBFN controller is derived from a learning-type direct neural controller, which learns at each iteration, as shown in Fig. 1(a). Therefore, the RBFN controller is also a learning-type controller.

$$y(k+1) = G \begin{bmatrix} y(k) & y(k-1) & \cdots & y(k-n) \\ u(k) & u(k-1) & \cdots & u(k-m) \end{bmatrix}, \quad (1)$$

where G is a function that expresses the plant characteristics, $y(k)$ is the plant output, $u(k)$ is the control input, k is the sampling number, and n and m are the plant orders. Here, we assume that (1) the orders n and m are known and (2) the inverse of the plant G^{-1} exists and is stable.

First, we design a direct neural controller using the plant equation, Eq. (1). The NN has two layers with no inner feedback loops and a sigmoid function is used to activate the input layer neurons⁽²¹⁾. To ensure both robustness and stability at the beginning of NN learning, a conventional feedback controller is incorporated into the neural controller. To simplify the design, the conventional controller uses proportional control (P-control). Figure 2 shows a direct neural controller incorporating a P-controller. As shown in Fig. 2, the plant control input $u(k)$ consists of the NN output $u_d(k)$ and the P-controller output $u_p(k)$. The P-controller output $u_p(k)$ is defined as

$$u_p(k) = K_p(p) e_c(k), \quad (2)$$

(a) learning-type scheme

(b) adaptive-type scheme

Fig. 1 Learning scheme of neural network

Fig. 2 Scheme of direct neural controller

where $K_p(p)$ is the feedback gain (P-gain), p is the iteration number, and $e_c(k)$ is the control error between the plant output $y(k)$ and the desired output $y_d(k)$, defined as

$$e_c(k) = y(k) - y_d(k). \quad (3)$$

The NN input vector $\mathbf{I}_d(k)$ of the direct neural controller is defined as⁽²¹⁾

$$\mathbf{I}_d(k) = [y(k) \ y(k-1) \ \cdots \ y(k-n+1) \ u(k-1) \ u(k-2) \ \cdots \ u(k-m) \ y_d(k+1) \ y_d(k)]^T. \quad (4)$$

The NN output $u_d(k)$ can be expressed as

$$u_d(k) = \mathbf{W}_d^T(p) \mathbf{f}\{\mathbf{I}_d(k)\} \quad (5)$$

where $\mathbf{W}_d^T(p)$ is the weight vector from the input layer to the output layer, and f is the sigmoid function given by

$$f(x) = (1.0 - e^{-x/X_g}) / (1.0 + e^{-x/X_g}), \quad (6)$$

where X_g is a parameter which changes the shape of the sigmoid function. In Eq. (5), the vector $\mathbf{f}\{\mathbf{I}_d(k)\}$ is defined as $\mathbf{f}\{\mathbf{I}_d(k)\} = [f\{y(k)\} \ f\{y(k-1)\} \ \cdots \ f\{u(k-1)\} \ f\{u(k-2)\} \ \cdots \ f\{y_d(k)\}]^T$.

Next, we show how the RBFN controller is derived from the neural controller. Assuming the existence of a weight vector $\mathbf{W}_{d0}$, which satisfies $e_c(k) = 0$, the control error $e_c(k)$ can be rewritten in terms of an arbitrary function g

$$e_c(k+1) = g(\mathbf{I}_d(k)) - u_d(k) \quad (7)$$

Assuming that the function g can be differentiated with respect to every component of the vector $\mathbf{I}_d(k)$, we apply the differential operator $\partial/\partial \mathbf{I}_d(k)$ to Eq. (7), and obtain

$$\begin{aligned} [\partial e_c(k+1)/\partial \mathbf{I}_d(k)]^T \mathbf{I}_d(k) &= [\partial \{g(\mathbf{I}_d(k)) - u_d(k)\}/\partial \mathbf{I}_d(k)]^T \mathbf{I}_d(k) \\ &= [\partial g(\mathbf{I}_d(k))/\partial \mathbf{I}_d(k)]^T \mathbf{I}_d(k) \\ &\quad - \sum_{i=1}^{m+n+2} W_{di}(p) I_{di}(k) f'\{I_{di}(k)\}, \end{aligned} \quad (8)$$

where $W_{di}(p)$ is the i th component of the weight vector $\mathbf{W}_d(p)$, f' is the derivative of the sigmoid function, and $I_{di}(k)$ is the i th component of the input vector $\mathbf{I}_d(k)$. If $\mathbf{W}_d(p) \approx \mathbf{W}_{d0}$, then the right-hand side of Eq. (8) approaches 0. Therefore, the partial derivative $[\partial e_c(k+1)/\partial \mathbf{I}_d(k)]^T \mathbf{I}_d(k)$ also approaches 0. As a result, from Eq. (8), the RBFN controller is obtained as

$$u_r(k) = \mathbf{W}_r^T(p) \mathbf{T}(k) \boldsymbol{\phi}\{\mathbf{I}_d(k)\}, \quad (9)$$

where $u_r(k)$ is the RBFN controller output, $\mathbf{W}_r^T(p)$ is the weight vector from the input layer to the output layer, $\mathbf{T}(k)$ is the diagonal matrix whose element in the i th row and the i th column is the i th component of the input vector $\mathbf{I}_d(k)$, and $\boldsymbol{\phi} (=f')$ is the RBF. In Eq. (9), the vector $\boldsymbol{\phi}\{\mathbf{I}_d(k)\}$ is defined as $\boldsymbol{\phi}\{\mathbf{I}_d(k)\} = [\phi\{y(k)\} \ \phi\{y(k-1)\} \ \cdots \ \phi\{u(k-1)\} \ \phi\{u(k-2)\} \ \cdots \ \phi\{y_d(k)\}]^T$. Consequently, if $\mathbf{W}_r(p) \approx \mathbf{W}_{d0}$, then the plant output $y(k)$ approaches the desired output $y_d(k)$ if the RBFN controller, Eq. (9), is used with a different integration constant. According to the

generalized δ -rule, the weight vector $\mathbf{W}_r^T(p)$ is updated during each iteration, using Eq. (11) to minimize the cost function $J_c(p)$ in Eq. (10).

$$J_c(p) = \sum_{k=1}^{\rho} e_c^2(k) / 2 \quad (10)$$

$$\mathbf{W}_r^T(p+1) = \mathbf{W}_r^T(p) - \eta_r \partial J_c(p) / \partial \mathbf{W}_r^T(p) \quad (11)$$

Here, η_r is the convergence factor, and ρ is the total sampling number within one iteration. The partial derivative of Eq. (11) is calculated as

$$\begin{aligned} \partial J_c(p) / \partial \mathbf{W}_r^T(p) &= \sum_{k=1}^{\rho} e_c(k) \{ \partial y(k) / \partial u(k-1) \} \{ \partial u(k-1) / \partial \mathbf{W}_r^T(p) \} \\ &= \sum_{k=1}^{\rho} e_c(k) \{ \partial y(k) / \partial u(k-1) \} \{ \partial u_r(k-1) / \partial \mathbf{W}_r^T(p) \}. \end{aligned} \quad (12)$$

In Eq. (12), the controller Jacobian, $\partial u_r(k-1) / \partial \mathbf{W}_r^T(p)$, can be calculated because the controller structure and parameters are known. However, the plant Jacobian, $\partial y(k) / \partial u(k-1)$, cannot be calculated when the plant structure and parameters are unknown⁽²²⁾. To estimate the plant Jacobian, we use an adaptive-type neural identifier^{(23),(24)}, which learns at each sampling time, as shown in Fig. 1(b), and models the direct transfer function of the plant. If the learning period is sufficiently large, the neural identifier's learning converges before the controller's learning begins. Therefore, the plant Jacobian can be calculated using the neural identifier and Eq. (11) can also be calculated.

In the following, we describe the design of a neural identifier. Figure 3 shows the neural identifier. The NN has two layers with no inner feedback loops. The NN output $y_m(k)$ can be expressed as

$$y_m(k) = \mathbf{W}_m^T(k) \mathbf{f}\{\mathbf{I}_m(k)\}, \quad (13)$$

where $\mathbf{W}_m^T(k)$ is the weight vector from the input layer to the output layer, f is the sigmoid function, and $\mathbf{I}_m(k)$ is the NN input vector, which is defined as⁽²¹⁾

$$\mathbf{I}_m^T(k) = [y(k-1) \ y(k-2) \ \cdots \ y(k-n-1) \ u(k-m-1) \ \cdots \ u(k-2) \ u(k-1)]. \quad (m+n \times 1) \quad (14)$$

Therefore, in Eq. (14), the vector $\mathbf{f}\{\mathbf{I}_m(k)\}$ is defined as $\mathbf{f}\{\mathbf{I}_m(k)\} = [f\{y(k-1)\} \ f\{y(k-2)\} \ \cdots \ f\{u(k-2)\} \ f\{u(k-1)\}]^T$. The NN learns during each sampling, using Eq. (16) to minimize the cost function $J_m(k)$ in Eq. (15).

$$J_m(k) = e_m^2(k) / 2, \quad (15)$$

$$\begin{aligned} \mathbf{W}_m^T(k+1) &= \mathbf{W}_m^T(k) - \eta_m \partial J_m(k) / \partial \mathbf{W}_m^T(k) \\ &= \mathbf{W}_m^T(k) - \eta_m e_m(k) \mathbf{f}\{\mathbf{I}_m(k)\}, \end{aligned} \quad (16)$$

Fig. 3 Scheme of neural identifier

Fig. 4 Block diagram of RBFN control system

where $e_m(k)$ is the model error, which is defined as

$$e_m(k) = y_m(k) - y(k), \quad (17)$$

and η_m is the convergence factor.

As a result, Eq. (12) is calculated as

$$\begin{aligned} \frac{\partial J_c(p)}{\partial \mathbf{W}_r^T(p)} &= \sum_{k=1}^p e_c(k) \left\{ \frac{\partial y_m(k)}{\partial u(k-1)} \right\} \left\{ \frac{\partial u_r(k-1)}{\partial \mathbf{W}_r^T(p)} \right\} \\ &= \sum_{k=1}^p e_c(k) W_{m(n+m)}(k) f'\{u(k-1)\} \mathbf{T}(k) \boldsymbol{\phi}\{\mathbf{I}_d(k)\}, \end{aligned} \quad (18)$$

where $W_{m(n+m)}(k)$ is the $(n+m)$ th component of the weight vector $\mathbf{W}_m^T(k)$.

The P-gain is also updated using the equation

$$\begin{aligned} K_p(p+1) &= K_p(p) - \eta_p \frac{\partial J_c(p)}{\partial K_p(p)} \\ &= K_p(p) - \eta_p \sum_{k=1}^p e_c(k) W_{m(n+m)}(k) f'\{u(k-1)\} \\ &\quad \times e_c(k-1), \end{aligned} \quad (19)$$

where η_p is the convergence factor.

Figure 4 shows a block diagram of a RBFN controller applied to a second-order plant ($n=2, m=1$). In this figure, T_i ($i=1, 2, \dots, 5$) are the diagonal components of the diagonal matrix $\mathbf{T}(k)$.

3. Simulation

In this section, we present the results of numerical simulations of the RBFN controller using mathematical plant to determine its feasibility and characteristics.

A simulation was carried out for the nonlinear plant described by

$$y(k+1) = f_s\{a_0 y(k) + a_1 y(k-1) + a_2 y(k-2) + c_{non} y^2(k-1) + b_0 u(k) + b_1 u(k-1)\}, \quad (20)$$

where a_i and b_i ($i=0, 1$) are the unknown parameters, a_2 and c_{non} are the coefficients of the parasitic term and the nonlinear term, respectively, and the function f_s is the sigmoid function, whose parameter X_g equals 0.5 in Eq. (6). In the simulation, the values of the parameters were set to $a_0=1.3$, $a_1=0.3$, $b_0=1.0$, $b_1=0.7$, $a_2=0.02$, and $c_{non}=-0.02$. The structure of the RBFN controller is the same as that shown in Fig. 4. The initial values of the weight vectors were randomly selected in the interval $[-0.05, 0.05]$, and the values

Fig. 5 Convergence of cost functions

of the controller parameters were set to $X_g=1.0$, $K_p(l)=-0.1$, $\eta_m=\eta_r=0.1$, and $\eta_p=0.01$. In the simulation, the purpose of the control is to make the plant output $y(k)$ track the desired output $y_d(k)$, which is a rectangular wave because it has many frequency components.

Figure 5 shows the learning process of the RBFN controller by illustrating the cost function $J_c(p)$ and the summation of the cost function $J_m(k)$ for one iteration. As shown in Fig. 5, both cost functions converge as learning progresses. At $p=1001$, the function f_s changes to

$$f_s(x) = 0.8 \sin(\pi x / 2.0). \quad (21)$$

When the change occurs, both cost functions increase and then converge after several iterations. Figure 6 shows the system responses at $p=1$, $p=1000$, and $p=2000$. As shown in the figure, the tracking of the desired output $y_d(k)$ was quick and precise after the RBFN controller learning converged ($p=1000$ and 2000). The P-controller output $u_p(k)$ is initially large ($p=1$). However, it approaches zero as the RBFN controller learning progresses. These simulation results show that the RBFN controller (1) can be used to control nonlinear systems and (2) can compensate for plant variations.

4. Experiment

In this section, we describe our experimental investigation of the RBFN controller and its feasibility in a practical plant.

4.1 Force control of a 1-degree-of-freedom robot

Figure 7 shows a block diagram of the experimental equipment. It consists of an axial robot, a force sensor, amplifiers, a microcomputer, and a contact object. The axial robot is a translational mechanism driven by a ball screw the rotation of which is controlled using a DC motor. The contact force at the tip of the robot arm is measured using the force sensor. The signal from the force sensor is filtered using a low-pass filter with a 10 Hz cut-off frequency.

Fig. 6 Simulation results for controlling nonlinear plant

Fig. 7 Block diagram of experimental setup for force control of 1-degree-of-freedom robot

The motor has a dead zone of approximately ± 0.12 A resulting from the harmonic-drive gear friction. The armature current of the motor is controlled directly by the microcomputer. Since this force control system can be approximated by a second-order plant⁽¹⁷⁾, the structure of the RBFN controller is the same as that shown in Fig. 4. The RBFN controller was implemented by the microcomputer. The sampling time was 40 ms. The initial values of the weight vectors were randomly selected in the interval $[-1, 1]$, and the values of the controller parameters were set to $X_g = 1.0$, $K_p(1) = -0.1$, $\eta_m = 0.01$, $\eta_r = 0.04$, and $\eta_p = 0.0001$. The contact object was made of silicone rubber. In the experiment, the aim of the control is to make the force sensor output $y(k)$ track the desired contact force $y_d(k)$. Figure 8(a) shows the learning process; the cost functions decrease as learning progresses. Figure 8(b) shows the system response at $p=100$; a small control error remains because of the dead zone

(a) relationship between iteration number and cost function

(b) response of system

Fig. 8 Experimental results for force control of 1-degree-of-freedom robot

in the motor current. However, the force sensor output $y(k)$ converges with the desired output $y_d(k)$.

4.2 Angular position control of 1-degree-of-freedom robot

Figure 9 shows a block diagram of the experimental setup. It consists of a DC motor, an optical encoder, an amplifier, a robot arm, and the microcomputer. One end of the arm is directly attached to the motor shaft, and the arm rotates in a horizontal plane. The rotation angle of the motor (joint angle) is measured using the optical encoder. The motor has a dead zone of approximately ± 0.36 A resulting from the solid friction at the joint. The armature current of the

Fig. 9 Block diagram of experimental setup for positioning control of 1-degree-of-freedom robot

(a) relationship between iteration number and cost function

(b) response of system

Fig. 10 Experimental results for force control of 1-degree-of-freedom robot

motor is controlled directly by the microcomputer with a sampling time of 40 ms. The aim of the control was to make the joint angle $y(k)$ track the desired output $y_d(k)$. Since this robot arm system is considered to be a second-order plant⁽⁹⁾, the structure of the RBFN controller and the controller parameters are the same as those in section 4.1. Figure 10(a) shows the learning process and Fig. 10(b) shows the system response at $p=100$. As shown in Fig. 10(a), both cost functions converge as learning progresses. As Fig. 10(b) shows, a small control error remains because of the dead zone in the motor current. However, the joint angle $y(k)$ tracks the desired output $y_d(k)$ after convergence of the RBFN controller learning.

These experimental results show that the RBFN controller is effective in an actual plant.

5. Application to Controlling Mechanical Systems

In this section, we describe example applications to demonstrate the effectiveness of the RBFN controller for controlling mechanical systems.

Fig. 11 Overview of optical disk drive

5.1 Positioning control of ODD positioner

Figure 11 shows a typical configuration of an ODD. It consists of two actuators, an optical head positioner, which coarsely moves the optical head incorporating the objective lens near the desired track, and a lens actuator, which moves the objective lens to the final track position. The lens actuator is used for jumps of only a few tracks, while the head positioner is used for track seeks exceeding 10 - 50 tracks. In this example, the object plant is the coarse head positioner and the aim is to control its position using the RBFN controller. We evaluated the ability of the RBFN controller by carrying out a simulation.

In ODDs, linear servomotors are widely used as optical head positioners. The motion equations for the positioner are⁽²⁵⁾

$$v(t) = dy(t)/dt, \quad (22)$$

$$Mdv(t)/dt + C_mv(t) = K_f i(t), \quad (23)$$

where M is the moving mass, C_m is the viscous frictional coefficient, K_f is the force constant, $y(t)$ is the position, $v(t)$ is the velocity, and $i(t)$ is the motor current.

Using Eqs.(22) and (23), the ODD positioner can be treated as a discrete-time second-order single-input (motor current) multi-output (position and velocity) plant. Therefore, the input vector $\mathbf{I}_d(k)$ of the RBFN controller is selected as $\mathbf{I}_d^T(k) = [y(k) \ y(k-1) \ v(k) \ v(k-1) \ u(k-1) \ y_d(k+1) \ y_d(k) \ v_d(k+1) \ v_d(k)]$, where $y_d(k)$ and $v_d(k)$ are the desired position and the desired velocity, respectively. The input vector $\mathbf{I}_m(k)$ of the NN identifier is chosen as $\mathbf{I}_m^T(k) = [y(k-1) \ y(k-2) \ v(k-1) \ v(k-2) \ u(k-2) \ u(k-1)]$ and the outputs are defined as $y_m(k)$ and $v_m(k)$. The cost functions in Eqs.(10) and (15) become

$$J_c(p) = \sum_{k=1}^p \{q_1 e_{c1}^2(k) + q_2 e_{c2}^2(k)\} / 2, \quad (24)$$

$$J_m(k) = \{e_{c1}^2(k) + e_{c2}^2(k)\} / 2, \quad (25)$$

where q_i ($i=1, 2$) are the weight parameters, and the

control errors are defined as $e_{c1}(k) = y(k) - y_d(k)$ and $e_{c2}(k) = v(k) - v_d(k)$. The P-controller in Eq. (2) can also be rewritten as

$$u_p(k) = K_{p1}(p)e_{c1}(k) + K_{p2}(p)e_{c2}(k), \quad (26)$$

where K_{pi} ($i=1, 2$) is the P-gain. In the simulation described below, the initial values of the weight vectors were randomly selected in the interval $[-0.01, 0.01]$, and the values of the controller parameters were set to $X_g=1.0$, $K_{p1}(1)=-0.0001$, $K_{p2}(1)=-0.01$, $\eta_m=\eta_r=1.0$, $\eta_p=0$, $q_1=1$, and $q_2=1000$. The sampling interval was 1 ms. A trapezoidal velocity curve and its integral are used as the desired outputs, $y_d(k)$ and $v_d(k)$.

Figure 12 shows the learning process and the system response after convergence of the RBFN controller learning. As seen in Fig. 12(b), quick, precise tracking of the desired outputs is achieved, since the plant is a linear system. Figure 13 shows the system response for various positioning distances after convergence of the RBFN controller learning, as shown in Fig. 12. Figure 13 shows that precise tracking of the desired output is also achieved for unlearned positioning distances. These results demonstrate the effectiveness of the RBFN controller for positioning

Fig. 12 Convergence of cost function and system response of ODD positioner after learning

control of an ODD positioner.

5.2 Directional control of tunneling machine

Figure 14 shows a schematic and a model of a small-diameter tunneling machine designed to construct telecommunication cable conduits that was developed by NTT. The operator corrects the tunneling machine's head angle λ , which can move in any direction with a maximum value of $2.2 \times 10^{-2} \pi$ rad, and controls the direction of the tunneling machine so that it moves along the planned path. The tunneling machine moves forward by pushing conduits one after another using an oil-pressure pushing device.

A simple model of the motion of the tunneling

Fig. 13 Response of system for various positioning distances after learning convergence

Fig. 14 Schematic and model of tunneling machine

machine is given by⁽²¹⁾

$$d\theta(x)/dx = \sigma(x)\xi\lambda/\tau, \quad (27)$$

$$dy(x)/dx = \theta(x), \quad (28)$$

where x and y denote the displacement of the tunneling machine in the X- and Y-directions, respectively, θ is the posture angle of the tunneling machine, τ is the stroke of the pushing device, ξ is the directional correction capability obtained by solving the nonlinear simultaneous equations with respect to the head angle λ , and $\sigma(x)$ is an arbitrary constant, which is considered to mainly reflect the variation in the ground condition. Figure 15 shows a comparison between the calculated and experimental paths of the tunneling machine. As shown in Fig. 15, the results are in good agreement, which indicates that the model approximates the movement of the tunneling machine.

Since the tunneling machine can be treated as a discrete dynamic system with a sampling interval equal to the stroke of the pushing device⁽²¹⁾, it can be treated as a discrete-time second-order single-input (head angle) multi-output (displacement and angle) plant. Therefore, the input vector $I_d(k)$ of the RBFN controller is chosen as $I_d^T(k) = [y(k) \ y(k-1) \ \theta(k) \ \theta(k-1) \ \lambda(k-1) \ y_d(k+1) \ y_d(k) \ \theta_d(k+1) \ \theta_d(k)]$, where $y_d(k)$ and $\theta_d(k)$ are the planned path and the planned angle, respectively. The input vector $I_m(k)$ of the NN identifier is chosen as $I_m^T(k) = [y(k-1) \ y(k-2) \ \theta(k-1) \ \theta(k-2) \ \lambda(k-2) \ \lambda(k-1)]$ and the outputs are defined as $y_m(k)$ and $\theta_m(k)$. The cost functions and the P-controller have the same form as Eqs. (24)–(26), except for the control error $e_{c2}(k) = \theta(k) - \theta_d(k)$. In the following simulation, the initial values of the weight vectors were randomly selected in the interval $[-0.05, 0.05]$, and the values of the controller parameters were set to $X_g = 1.0$, $K_{p1}(1) = -0.02$, $K_{p2}(1) = -0.1$, $\eta_m = 0.04$, $\eta_r = 1.0$, $\eta_p = 0$, $q_1 = 1$, and $q_2 = 1000$. The stroke of the pushing device was 1.25 m.

Fig. 15 Comparison of calculated and experimental results

Figure 16 shows the relationship between the iteration number p and the cost function $J_c(p)$ and illustrates the desired outputs (planned path and planned angle) for the RBFN controller. In the learning process, the signs of the desired outputs are reversed at every iteration and the parameter $\sigma(x)$ is set to 1. As shown in Fig. 16(a), the cost function $J_c(p)$ of each convergence factor η_c converges and the convergence speed increase as the value of the parameter η_c

Fig. 16 Convergence of cost functions

Fig. 17 Simulation results for controlling tunneling machine using RBFN controller after learning convergence

increases. After convergence of the RBFN controller learning, as shown in Fig. 16 at $p=5000$, the RBFN controller was tested for an unlearned planned path $y_d(k)$. Figure 17 shows the system response. Here, the parameter $\sigma(x)$ is randomly varied between 0 and 1, as shown at the top of Fig. 17. Nevertheless, quick, precise tracking of the desired outputs is achieved. These results indicate the effectiveness of the RBFN controller for directional control of the tunneling machine.

6. Conclusion

In this paper, we described a design method for an RBFN controller. By considering a sigmoid function derivative, the RBFN controller is derived by applying a differential operator to a direct neural controller. The designed controller is a learning-type direct RBFN controller, which uses an adaptive-type neural identifier to overcome the Jacobian problem in its learning process, and incorporates a conventional feedback controller to ensure both robustness and stability at the beginning of RBFN learning. Simulation results obtained for a mathematical discrete-time single-input single-output plant demonstrated the feasibility of the RBFN controller for controlling nonlinear systems. Experimental results for force control of an axial robot and angular position control of a robot arm demonstrated the effectiveness of the RBFN controller for controlling practical systems. Applications to the positioning control of an ODD positioner and the directional control of a tunneling machine were investigated. These demonstrated the effectiveness of the RBFN controller for controlling mechanical systems.

References

- (1) Kawato, M., Uno, Y., Isobe, M. and Suzuki, R., A Hierarchical Model for Voluntary Movement and its Application to Robotics, Proc. of 1987 IEEE Int. Conf. on Neural Networks, Vol. IV (1987), p. 573.
- (2) Tokita, M., Mitsuoka, T., Fukuda, T. and Kurihara, T., Force Control of Robot Manipulator by Neural Networks, J. of RSJ, (in Japanese), Vol. 7, No. 1 (1989), p. 47.
- (3) Narendra, K.S. and Parthasarathy, K., Identification and Control of Dynamic Systems Using Neural Networks, IEEE Trans. on Neural Networks, Vol. 1, No. 1 (1990), p. 4.
- (4) Chen, F.C. and Khalil, H.K., Adaptive Control of Nonlinear System using Neural Networks, Int. J. of Control, Vol. 55, No. 6 (1992), p. 1299.
- (5) Nagato, H. and Yoshida, K., Optimal Control of Nonlinear Systems by Neural Network, Trans. Jpn. Soc. Mech. Eng., (in Japanese), Vol. 58, No. 545, C (1992), p. 171.
- (6) Yabuta, T. and Yamada, T., Neural Network Controller Characteristics with Regard to Adaptive Control, IEEE Trans. on Systems, Man, and Cybernetics, Vol. 22, No. 1 (1992), p. 170.
- (7) Khalid, M., Omatu, S. and Yusof, R., MIMO Furnace Control with Neural Networks, IEEE Trans. on Control Systems Technology, Vol. 1, No. 4 (1993), p. 238.
- (8) Nordgren, R.E. and Meckel, P.H., An Analytical Comparison of a Neural Network and a Model Based Adaptive Controller, IEEE Trans. on Neural Networks, Vol. 4 (1993), p. 685.
- (9) Takahashi, K., Takayanagi, M., Yamada, I. and Torp, S., Design of Nonlinear Compensator for Positioning Control in Mass Storage System (A Neural-Network-Based Controller), JSME Int. J., Ser. C, Vol. 37, No. 3 (1994) p. 573.
- (10) Jin, L., Nikiforuk, P.N. and Gupta, M.M., Dynamic Recurrent Neural Networks for Control of Unknown Nonlinear Systems, ASME J. of Dyn. Syst., Meas., and Control, Vol. 116, No. 4 (1994), p. 567.
- (11) Poggio, T. and Edelman, S., A Network that Learns to Recognize Three-Dimensional Objects, Nature, Vol. 343, No. 18 (1990), p. 263.
- (12) Sanner, R.M. and Slotine, J.-J.E., Gaussian Networks for Direct Adaptive Control, IEEE Trans. on Neural Networks, Vol. 3, No. 6 (1992), p. 837.
- (13) Chen, S. and Billings, S.A., Neural Networks for Nonlinear Dynamic System Modeling and Identification, Int. J. of Control, Vol. 56, No. 2 (1992), p. 319.
- (14) Narendra, K. S. and Parthasarathy, K., Disturbance Rejection in Nonlinear Systems Using Neural Networks, IEEE Trans. on Neural Networks, Vol. 4, No. 1 (1993), p. 63.
- (15) Gorinevsky, D.M., On the Persistency of Excitation in Radial Basis Function Network Identification of Nonlinear Systems, IEEE Trans. on Neural Networks, Vol. 6, No. 5 (1995), p. 1237.
- (16) Yamada, T. and Yabuta, T., Nonlinear Neural Network Controller for Dynamic System, Proc. of IEEE IECON '90, (1990), p. 1244.
- (17) Yamada, T. and Yabuta, T., Some Remarks on Characteristics of Direct Neuro-Controller with Regard to Adaptive Control, Trans. of SICE, (in Japanese), Vol. 27, No. 7 (1991), p. 784.
- (18) Takahashi, K. and Yamada, I., Neural-Network-Based Learning Control of Flexible Mechanism With Application to a Single-Link Flexible Arm, ASME J. of Dyn. Syst., Meas., and Control, Vol. 116, No. 4 (1994), p. 792.
- (19) Yamada, T., Remarks on RBF Controller Induced from Neural Network Controller, Proc. of SICE, (in Japanese), (1995), p. 725.
- (20) Rumelhart, D.E., McClelland, J.L. and PDP Research Group, Parallel Distributed Processing, The MIT Press, (1988).
- (21) Takahashi, K. and Hosaka, H., A Neural-Network-Based Controller Used for Directional

- Control of a Tunneling Machine, to be published in JSME Int. J.
- (22) Jordan, M.I., Generic Constraints on Under-specified Target Trajectories, Proc. of IJCNN' 89, (1989), p. 217.
- (23) Takahashi, K., Neural-Network-Based Learning Control Applied to a Single-Link Flexible Arm, Proc. of 2nd Int. Conf. on MOVIC, (1994), p. 811.
- (24) Yamada, T., Remarks on Hybrid Neural Network Controller Using Different Convergence Speeds, Proc. of IEEE ICRA '95, Vol. 1, (1995), p. 562.
- (25) Kudo, K., Yamada, I. and Kato, K., Minimum Energy Positioning Control in Optical Disk Drives, Proc. of IEEE IECON '90, (1990), p. 46.
-