An Optimal Tracking Neuro-Controller for Nonlinear Dynamic Systems

Young-Moon Park, *Fellow, IEEE*, Myeon-Song Choi, *Member, IEEE*, and Kwang Y. Lee, *Senior Member, IEEE*

Abstract—**Multilayer neural networks are used to design an optimal tracking neuro-controller (OTNC) for discrete-time nonlinear dynamic systems with quadratic cost function. The OTNC is made of two controllers: feedforward neuro-controller (FFNC) and feedback neuro-controller (FBNC). The FFNC controls the steady-state output of the plant, while the FBNC controls the transient-state output of the plant. The FFNC is designed using a novel inverse mapping concept by using a neuro-identifier. A generalized backpropagation-through-time (GBTT) algorithm is developed to minimize the general quadratic cost function for the FBNC training. The proposed methodology is useful as an off-line control method where the plant is first identified and then a controller is designed for it. A case study for a typical plant with nonlinear dynamics shows good performance of the proposed OTNC.**

I. Introduction

TRADITIONAL controller design usually involves complex mathematical analysis and yet has many difficulties in controlling highly nonlinear plants. To overcome these difficulties, the number of new approaches using neural networks for control has increased significantly in recent years. The use of neural networks' learning ability helps controller design to be rather flexible, especially when plant dynamics are complex and highly nonlinear. This is a distinct advantage over traditional control methods.

Poggio and Girosi [1] stated that the problem of learning between input and output spaces is equivalent to that of synthesizing an associative memory that retrieves appropriate output when the input is present and generalizes when a new input is applied. It is equivalent to the problem of estimating an input–output transformation using given input–output pairs as training sets. It can also be included in the classical framework of approximation theory.

Nguyen and Widrow [2] showed the possibility of using neural networks in controlling a plant with high nonlinearities. They exploited the neural networks' self-learning ability in the Truck–Backer problem. Chen [3] proposed a self-tuning adaptive controller with neural networks, and Iiguni and Sakai [4] constructed a neural network controller combined with linear optimal controller to compensate for the uncertainties in model parameters. Recently, Ku and Lee [5] proposed an architecture of diagonal recurrent neural networks for identification and control and applied it to a nuclear reactor control problem [6]. There are a number of other cases in which neural networks' learning ability is applied for plant control [7]–[11].

The use of neural networks in control has been focused mostly on the model reference adaptive control (MRAC) problem [3], [5]–[11]. This paper introduces a new class of control problems, namely the optimal tracking problem, which minimizes a general quadratic cost function of tracking errors and control efforts. This results in a hybrid of *feedback* and *feedforward* neuro-controllers in parallel. The feedforward neuro-controller (FFNC) generates the steady-state control input to keep the plant output to a given reference value, and the feedback neuro-controller (FBNC) generates the transient control input to stabilize error dynamics along the optimal path while minimizing the cost function. A novel inverse mapping concept is developed to design the FFNC using a neuro-identifier.

The use of general quadratic cost function provides "optimal" performance with respect to tradeoffs between the tracking error and control effort. Since the cost function is defined over a finite time interval, a generalized backpropagation-through-time (GBTT) algorithm is developed to train the feedback controller by extending the Werbos' backpropagation-through-time (BTT) algorithm [10], which was originally developed for the cost function for the tracking error alone.

The control methodology in this paper is useful as an off-line control method where the plant is first identified and then a controller is designed for it. This assumes that the plant can be identified without making the plant unstable. Repeatability of the control experiment is also assumed in order to tune the parameters of the neural networks. This is often the case when we design a supplementary controller for an existing control system in a power plant.

The organization of the paper is as follows. Section II presents the formulation of the optimal tracking control problem and architecture for the optimal tracking neuro-controller (OTNC). Section III shows the development of neuro-controllers including their training algorithms. In Section IV the proposed OTNC is implemented in a typical nonlinear plant, and the conclusion is drawn in Section V.

Manuscript received August 4, 1993; revised February 20, 1994 and March 16, 1996. This work was supported in part by the Korea Science and Engineering Foundation (KOSEF) and the National Science Foundation (NSF) under Grants "U.S.–Korea Cooperative Research on Intelligent Distributed Control of Power Plants and Power Systems" (INT-9 223 030) and "Research and Curriculum Development for Power Plant Intelligent Distributed Control" (EID-921 232); and joint NSF and Electric Power Research Institute (EPRI) Grant "Experimental Development of Power Reactor Intelligent Control" (ECS-9 216 504/RP8030-4).

Y.-M. Park and M.-S. Choi are with the Department of Electrical Engineering, Seoul National University, Seoul 151-742 Korea.

K. Y. Lee is with the Department of Electrical Engineering, Pennsylvania State University, University Park, PA 16802 USA.

Publisher Item Identifier S 1045-9227(96)06597-6.

II. Problem Formulation

An optimal tracking control problem is first formulated in this section with an appropriate justification for the use of a general quadratic cost function. Motivation for the use of a feedforward controller is illustrated with the aid of linear optimal control theory. Finally, an architecture is given for the OTNC.

A. Optimal Tracking Problem

We consider a system in the form of the general nonlinear autoregressive moving average (NARMA) model

$$y_{(k+1)} = f(y_{(k)}, y_{(k-1)}, \cdots, y_{(k-n+1)}, u_{(k)} \\ u_{(k-1)}, \cdots, u_{(k-m+1)}) \quad (1)$$

where y and u, respectively, represent output and input variables, k represents time index, and n and m represent the respective output and input delay orders.

When the target output of a plant holds up for some time and varies from time to time, the control objectives can be defined as follows [12]:

1) Minimize the summation of the squares of regulating output error and the squares of input error in transient.
2) Reduce the steady-state error to zero.

The above control objectives can be achieved by minimizing the following well-known quadratic cost function:

$$J = \frac{1}{2}\sum_{k=1}^{N}(Q(y_{\text{ref}} - y_{(k+1)})^2 + R(u_{\text{ref}} - u_{(k)})^2) \quad (2)$$

where y_{ref} is a reference output, u_{ref} is the steady-state input corresponding to y_{ref}, and Q and R are positive weighting factors. This quadratic cost function or the performance index not only forces the plant output to follow the reference, but also forces the plant input to be close to the steady-state value in maintaining the plant output to its reference value.

It should be noted that the control input found by minimizing the performance index (2) is *subjective* and is "optimal" only with respect to a given set of values of the weighting factors Q and R. The choice of the weighting factors is done with engineering judgment and is often performed iteratively by observing the system responses in light of design specifications such as overshoot and rise time. Another aspect to be noted is that the quadratic performance index (2) is a very general optimization criterion. By properly choosing the weighting factors and/or the limit of summation, N, different control objectives can be achieved. For example, by setting $N = 1, R = 0$, and $Q = 1$, the performance index becomes the usual instantaneous error [12].

The instantaneous error alone as a performance index tends to exert large inputs and causes the plant to oscillate around the tracking reference. On the other hand, the quadratic performance index (2) limits the control energy to be expanded over some time interval while minimizing an accumulated error. For linear time-invariant dynamic systems, this performance index leads to a fixed feedback gain matrix as N goes to infinity.

Before solving the above control problem for a general nonlinear system, restriction of the model to a linear system will be helpful in gaining an insight for the need for both feedforward and feedback controls. A linear counter part to the NARMA model (1) is the following linear time-invariant system:

$$x_{(k+1)} = Ax_{(k)} + Bu_{(k)} \\ y_{(k)} = Cx_{(k)} \quad (3)$$

where x is an n-dimensional state vector, and A, B, and C are constant matrices of appropriate dimension. When the output $y_{(k)}$ has the set point y_{ref} in steady state, then the above state equation becomes

$$x_{\text{ref}} = Ax_{\text{ref}} + Bu_{\text{ref}} \\ y_{\text{ref}} = Cx_{\text{ref}} \quad (4)$$

where x_{ref} is the state vector corresponding to y_{ref} in steady state [13]. By subtracting (4) from (3) and shifting the vectors as

$$u'_{(k)} = u_{(k)} - u_{\text{ref}} \quad (5a)$$

$$x'_{(k)} = x_{(k)} - x_{\text{ref}} \quad (5b)$$

$$y'_{(k)} = y_{(k)} - y_{\text{ref}} \quad (5c)$$

the optimal tracking problem, (2) and (3), is converted to the optimal regulating problem with zero output in steady state

$$x'_{(k+1)} = Ax'_{(k)} + Bu'_{(k)} \\ y'_{(k)} = Cx'_{(k)} \quad (6)$$

with the quadratic cost function

$$J = \frac{1}{2}\sum_{k=1}^{N}(Q(y'_{(k+1)})^2 + R(u'_{(k+1)})^2). \quad (7)$$

The control law for the optimal regulator problem defined by (6) and (7) for $N = \infty$ is given as

$$u'_{(k)} = Fx'_{(k)} \quad (8)$$

where F is the optimal feedback gain matrix obtained by solving an algebraic matrix Riccati equation [12]. Thus from (5a), the optimal control for the original linear system, (3), is

$$u_{(k)} = Fx'_{(k)} + u_{\text{ref}}. \quad (9)$$

This shows an important observation that the control input consists of two parts, feedforward and feedback

$$u_{(k)} = u_{fb}(x'_{(k)}) + u_{ff}(y_{\text{ref}}) \quad (10)$$

where y_{fb} represents the feedback control, and u_{ff} represents the feedforward control corresponding to the steady-state output y_{ref}. When the problem is limited to the case where u_{ref} exists for any y_{ref}, it is reasonable to assume that the control input for a *nonlinear* system can also be separated into feedforward and feedback parts. The role of the feedforward control is to keep the plant output to the reference value in steady state, and that of the feedback control is to stabilize the tracking error dynamics during transient [14].

There have been a number of studies in solving the tracking control problem with feedforward and feedback controls

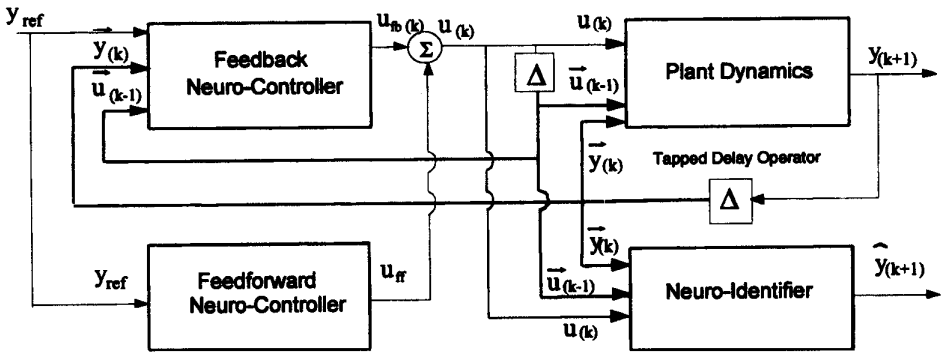


Fig. 1. Block diagram for the optimal tracking neuro-controller.

[15], [16]. Meystel and Uzzaman [15] used an algorithmic procedure, *approximate inversion*, to find the feedforward control input for a given reference trajectory; however, while guaranteeing admissible solutions, it leads to exhaustive binary search. Then, they used a simple constant-gain proportional control for feedback control input. Jordan and Rumelhart [16] used an inverse learning model to find the feedforward control when the reference trajectory of plant output was given. However, it did not provide an optimal controller for the optimal tracking problem defined with the general quadratic performance index (2).

While an explicit solution for the optimal tracking problem is available for linear systems, it is not possible for nonlinear systems in general. Iiguni and Sakai [4] tried to design a nonlinear regulator using neural networks by assuming the plant to be linear, but with uncertain parameters. For linear systems, minimization of the quadratic performance index is possible, and training of the neuro-controller can be simplified by using the Ricatti equation. However, this approximation is not possible for nonlinear systems, and hence a method of minimizing the quadratic performance index is developed by generalizing the BTT training algorithm.

B. Architecture for Optimal Tracking Neuro-Controller

Following the above observation, an OTNC is designed with two neuro-controllers in order to control a nonlinear plant that has a nonzero set point in steady state. An FFNC is constructed to generate feedforward control input corresponding to the set point, and trained by the well-known error backpropagation algorithm. An FBNC is constructed to generate feedback control input, and trained by a GBTT algorithm to minimize the quadratic performance index.

An independent neural network named neuro-identifier is used when the above two neuro-controllers are in training mode. This network is trained to emulate plant dynamics and to backpropagate an *equivalent error* or *generalized delta* [2] to the controllers under training. Fig. 1 shows an architecture for the optimal tracking neuro-controller for a nonlinear plant. In the figure, the *tapped delay operator* Δ is defined as a delay mapping from a sequence of scalar input, $\{x_{(i)}\}$ to a vector output with an appropriate dimension defined as $\vec{x}_{(i-1)} = (x_{(i-1)}, x_{(i-2)}, \cdots, x_{(i-p)})$, where $p = n$ for the output variable y, and $p = m - 1$ for the input variable u.

III. Design of Neuro-Controllers

The OTNC is made of three multilayer feedforward neural networks: neuro-identifier, feedforward neuro-controller, and feedback neuro-controller. A multilayer neural network represents a nonlinear function mapping multi-inputs to single output with weight parameters and nonlinear sigmoid functions.

A. Neuro-Identifier

The function of the neuro-identifier is to identify plant dynamics. It is then used to backpropagate the equivalent error to the neuro-controllers. Training the neuro-identifier can be regarded as an approximation process of a nonlinear function using input–output data sets [1].

A NARMA model (1) can be viewed as a nonlinear mapping from $(n + m)$-dimensional input space to a one-dimensional (1-D) output space

$$y_{(k+1)} = f(\vec{X}_{(k)}) \tag{11}$$

where

$$\vec{X}_{(k)} = (y_{(k)}, y_{(k-1)}, \cdots, y_{(k-n-1)}, u_{(k)}, u_{(k-1)}, \cdots, u_{(k-m+1)})$$

is regarded as input vector.

Therefore, the neuro-identifier for the plant can be represented as

$$\hat{y}_{(k+1)} = F(\vec{X}_{(k)}, \vec{W}) \tag{12}$$

where $\hat{y}_{(k+1)}$ is the output estimated and $\vec{W}$ is the weight parameter vector for the neuro-identifier.

Then, training of the neuro-identifier F is to adjust the weight parameters so that it emulates the nonlinear plant dynamics. Input–output training patterns are obtained from the operation history of the plant. The block diagram for training the neuro-identifier is given in Fig. 2.

The objective of training the neuro-identifier is to reduce the average error defined by

$$J = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} (y^i_{(k+1)} - \hat{y}^i_{(k+1)})^2 \tag{13}$$

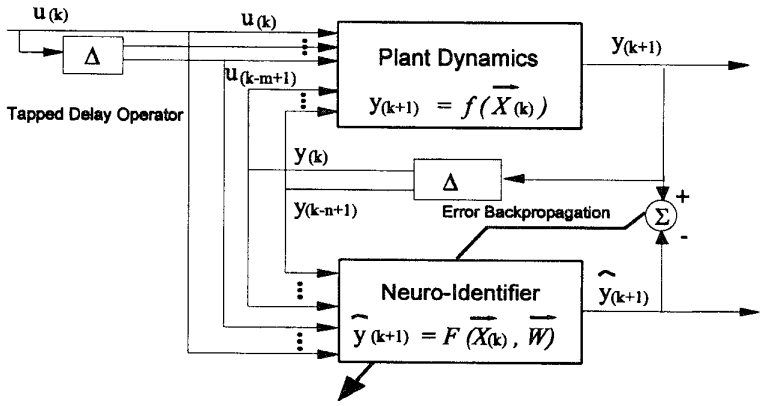


Fig. 2. Block diagram for training the neuro-identifier.

where N is the number of samples in a group of training set. The equivalent error on the output node of the network for the ith sampled data is defined as the negative of the gradient

$$\delta_o^i \triangleq -\frac{\partial J}{\partial o^i} = -\frac{\partial J}{\partial \hat{y}^i_{(k+1)}} = \frac{1}{N}(y^i_{(k+1)} - \hat{y}^i_{(k_1)}). \quad (14)$$

This error is then used backward to compute an equivalent error for a node in an arbitrary layer to update weight parameters in the backpropagation algorithm (BPA).

Through the learning process, the plant characteristics are stored in the weight parameters of the neuro-identifier. The training can be regarded as finished when the average error between the plant and the neuro-identifier outputs converges to a small value, and the neuro-identifier is presumed to have learned the plant characteristics approximately, i.e.,

$$y_{(k+1)} = f(\vec{X}_{(k)}) \approx \hat{y}_{(k+1)} = F(\vec{X}_{(k)}, \vec{W}). \quad (15)$$

B. Feedforward Neuro-Controller

In designing a controller for a plant to follow an arbitrary reference output, it is necessary to keep the steady-state tracking error to zero. For this purpose the FFNC is designed to generate a control input which will maintain the plant output to a given reference output in steady state. The FFNC is then required to learn the inverse dynamics of the plant in steady state. A novel approach is now proposed to develop the inverse mapping with the aid of the neuro-identifier.

Note that the steady-state control input can be obtained by setting $y_{(k)} \equiv y_{\text{ref}}$ and $u_{(k)} \equiv u_{\text{ref}}$ for all k in the NARMA model (1), i.e.,

$$y_{\text{ref}} = f(y_{\text{ref}}, y_{\text{ref}}, \cdots, y_{\text{ref}}, u_{\text{ref}}, u_{\text{ref}}, \cdots, u_{\text{ref}}) \quad (16)$$

or equivalently

$$u_{\text{ref}} = g(y_{\text{ref}}) \quad (17)$$

which is the inverse function of (16). The inverse is not unique in general and any one solution is sufficient for control purpose. However, noting that the control input is the control energy, the smallest solution of u_{ref} is preferred. The FFNC network G, as an inverse mapping of the plant in steady state, can be developed by using the neuro-identifier F as shown in Fig. 3, i.e.,

$$\hat{y}_{\text{ref}} = F(y_{\text{ref}}, y_{\text{ref}}, \cdots, y_{\text{ref}}, u_{ff}, u_{ff}, \cdots, u_{ff}, \vec{W}) \quad (18)$$

$$u_{ff} = G(y_{\text{ref}}, \vec{W}) \quad (19)$$

where u_{ff} is the feedforward control input and $\hat{y}_{\text{ref}}$ is the output of the neuro-identifier designed in (15). Training of the FFNC (19) is to adjust its weight parameters so that the output of the neuro-identifier $\hat{y}_{\text{ref}}$ approximates the given reference output y_{ref}, and when the training is finished u_{ff} approximates u_{ref}. Training of the FFNC can be understood as an approximation process of the inverse dynamics for the plant in steady state. To train the FFNC an equivalent error is defined, which then is propagated back through the neuro-identifier (15) that has been trained already.

The objective of training the FFNC is to reduce the average error defined by

$$J = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(y^i_{\text{ref}} - \hat{y}^i_{\text{ref}}(u^i_{ff}))^2 \quad (20)$$

where N is the number of samples in a group of training set.

To update the weight parameters in the FFNC the equivalent error is propagated backward through the neuro-identifier. The equivalent error on the output of the FFNC is defined as the negative sensitivity of the above performance index with respect to u_{ff}, which can be calculated from the equivalent error on the neuro-identifier input nodes

$$\delta^i_{u_{ff}} \triangleq -\frac{\partial J}{\partial u^i_{ff}} = -\frac{\partial J}{\partial \hat{y}^i_{\text{ref}}} \cdot \frac{\partial \hat{y}^i_{\text{ref}}}{\partial u^i_{ff}}. \quad (21)$$

Since u_{ff} is applied to the first m input nodes of the neuro-identifier, i.e., $(I_k)^i = u^i_{ff}, k = 1, 2, \cdots, m$, then

$$\delta^i_{u_{ff}} = \sum_{k=1}^{m} -\frac{\partial J}{\partial \hat{y}^i_{\text{ref}}} \cdot \frac{\partial \hat{y}^i_{\text{ref}}}{\partial (I_k)^i} = \sum_{k=1}^{m} \delta^i_{I_k} \quad (22)$$

where $\delta^i_{I_k}$ is the equivalent error of the u_{ff}-input node in the neuro-identifier, which is computed by the BPA. Since u_{ff}

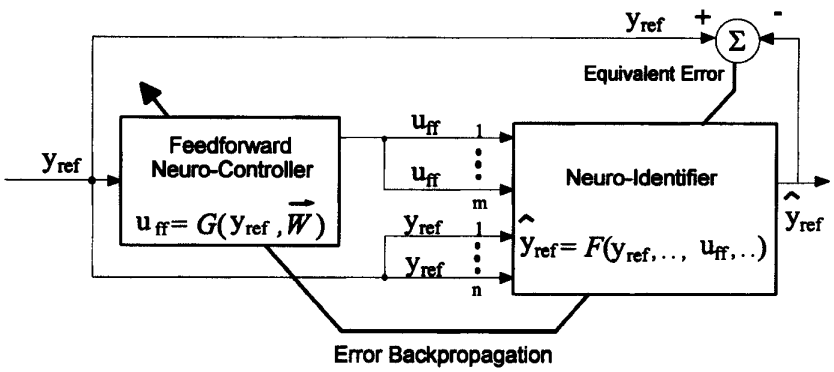


Fig. 3. Block diagram for training the feedforward controller.

is also the output of the FFNC, the equivalent error (22) can directly be used as the equivalent error for the network G in the BPA.

Training begins with small random values of weight parameters in the FFNC. This allows the feedforward control input to grow from a small random value, and converge to the smallest solution of u_{ref}, which is preferred over all other possible solutions.

At the end of the training, the weight parameters in the FFNC are adjusted so that the output of the neuro-identifier follows a given reference output. Training is finished when the average error between the neuro-identifier output and the given reference output converges to a small value, and FFNC has learned the steady-state inverse dynamics of the plant with the help of the neuro-identifier

$$\begin{aligned} y_{\text{ref}} &= f(y_{\text{ref}}, y_{\text{ref}}, \cdots, y_{\text{ref}}, u_{\text{ref}}, u_{\text{ref}}, \cdots, u_{\text{ref}},) \\ &\approx \hat{y}_{\text{ref}}, = F(y_{\text{ref}}, y_{\text{ref}}, \cdots, y_{\text{ref}}, u_{ff}, u_{ff}, \cdots, u_{ff}, \vec{W}) \\ u_{\text{ref}} &\approx u_{ff} = G(y_{\text{ref}}, \vec{W}). \end{aligned} \tag{23}$$

C. Feedback Neuro-Controller

The role of the FBNC is to stabilize tracking error dynamics when the plant output is following an arbitrarily given reference output. This objective can be achieved by minimizing the quadratic performance index (2) discussed previously.

Noting that $u_{\text{ref}} \approx u_{ff}$ and $u_{(k)} = u_{ff} + u_{fb(k)}$, the performance index (2) can be modified as

$$J = \sum_{k=1}^{N} J_k = \tfrac{1}{2} \sum_{k=1}^{N} (Q(y_{\text{ref}} - y_{(k+1)})^2 + R(u_{fb(k)})^2) \tag{24}$$

where $u_{fb(k)}$ is the feedback control input.

From the NARMA model (1), the feedback control input can be viewed as an inverse mapping

$$\begin{aligned} u_{fb(k)} = h(&y_{\text{ref}}, y_{(k)}, y_{(k-1)}, \cdots, y_{(k-n+1)}, u_{(k-1)} \\ &u_{(k-2)}, \cdots, u_{(k-m+1)}) \end{aligned} \tag{25}$$

where y_{ref} indicates the nonlinear dependency of the input function on the set-point. The corresponding FBNC can be represented as a nonlinear network H

$$\begin{aligned} u_{fb(k)} = H(&y_{\text{ref}}, y_{(k)}, y_{(k-1)}, \cdots, y_{(k-n+1)}, u_{(k-1)} \\ &u_{(k-2)}, \cdots, u_{(k-m+1)}, \vec{W}). \end{aligned} \tag{26}$$

Since the target value for the optimal feedback control $u_{fb(k)}$ is not available for training, traditional BPA method is not applicable here. Therefore, the FBNC will learn the control law by trial and error as it drives the neuro-identifier to generate the equivalent error for backpropagation.

The learning process by trial and error consists of two parts. First, from the given initial state and an arbitrarily given reference set-point, the combined FFNC and FBNC drive the neuro-identifier for N steps. Second, update the weight parameters in the FBNC using the equivalent error generated by the GBTT algorithm developed in the following section.

D. Generalized Backpropagation-Through-Time Algorithm

The GBTT is to generate an equivalent error from a general quadratic cost function (24), and it is an extension of the BTT algorithm of Werbos [10]. The original BTT was for the cost function with output error only. On the other hand, the GBTT is for the general quadratic cost function (24) which includes not only output errors, but also input variables.

The GBTT is based upon output and input sensitivities of the cost function defined by

$$\delta_y^k \triangleq -\frac{\partial J}{\partial y_{(k)}}, \qquad k = 1, 2, 3, \cdots, N+1 \tag{27}$$

$$\delta_u^k \triangleq -\frac{\partial J}{\partial u_{(k)}}, \qquad k = 0, 1, 2, \cdots, N. \tag{28}$$

Since, for a fixed feedforward control

$$\begin{aligned} \delta_{ufb}^k &= -\frac{\partial J}{\partial u_{fb(k)}} = -\frac{\partial J}{\partial u_{(k)}} \frac{\partial u_{(k)}}{\partial u_{fb(k)}} \\ &= -\frac{\partial J}{\partial u_{(k)}} \frac{\partial (u_{ff} + u_{fb(k)}}{\partial u_{fb(k)}} = -\frac{\partial J}{\partial u_{(k)}} = \delta_u^k \end{aligned} \tag{29}$$

the subscript fb will be dropped in the following development.

1) Output Sensitivity Equation (OSE): An output $y_{(k)}$ at an arbitrary time-step k influences both the plant dynamics (1) and the inverse dynamics (25). Since the plant dynamics (1) is defined with n delayed output variables, an arbitrary output $y_{(k)}$ will influence the plant dynamics for the next n steps, i.e., is a function of $y_{(k)}$ for $i = 1, 2, \cdots, n$. Similarly, since the inverse dynamics (25) also has n delayed output variables, an output $y_{(k)}$ will influence the input for the next n steps, i.e., is a function of $y_{(k)}$ for $i = 0, 1, 2, \cdots, n-1$.

Recall that the performance index (24) is defined on a finite interval, i.e.,

$$J = J(y_{(j+1)}, u_{(j)};\ j = 1, 2, \cdots, N). \tag{30}$$

Thus, the gradient of J with respect to an output $y_{(k)}$ for some k is

$$\begin{aligned}\frac{\partial J}{\partial y_{(k)}} &= \sum_{\substack{i=1\\ k+i\le N+1}}^{n} \frac{\partial J}{\partial y_{(k+i)}}\frac{\partial y_{(k+i)}}{\partial y_{(k)}} + \sum_{\substack{i=0\\ k+i\le N}}^{n-1} \frac{\partial J}{\partial u_{(k+i)}} \\ &\quad \cdot \frac{\partial u_{(k+i)}}{\partial y_{(k)}} - Q(y_{\text{ref}} - y_{(k)}) \\ &= \sum_{\substack{i=k+1\\ i\le N+1}}^{k+n} \frac{\partial J}{\partial y_{(i)}}\frac{\partial y_{(i)}}{\partial y_{(k)}} + \sum_{\substack{i=k\\ i\le N}}^{k+n-1} \frac{\partial J}{\partial u_{(i)}}\frac{\partial u_{(i)}}{\partial y_{(k)}} \\ &\quad - Q(y_{\text{ref}} - y_{(k)}).\end{aligned} \tag{31}$$

By using the definition of sensitivities, (27) and (28)

$$\delta_y^k = \sum_{\substack{i=k+1,\\ i\le N+1}}^{k+n} \delta_y^i \frac{\partial y_{(i)}}{\partial y_{(k)}} + \sum_{\substack{i=k,\\ i\le N}}^{k+n-1} \delta_u^i \frac{\partial u_{(i)}}{\partial y_{(k)}} + Q(y_{\text{ref}} - y_{(k)}). \tag{32}$$

Note that this OSE depends on the input sensitivities as well.

2) Input Sensitivity Equation (ISE): The ISE can be derived in a way similar to the OSE.

Since the plant dynamics (1) is defined with m delayed input variables, while the inverse dynamics (25) is with $m-1$ delayed input variables, $y_{(k+i)}$ is a function of $u_{(k)}$ for $i = 1, 2, \cdots, m$, and $u_{(k+i)}$ is a function of $u_{(k)}$ for $i = 1, 2, \cdots, m-1$.

Thus, the gradient of J with respect to an input $u_{fb(k)}$ for some k is

$$\begin{aligned}\frac{\partial J}{\partial u_{(k)}} &= \sum_{\substack{i=1\\ k+i\le N+1}}^{m} \frac{\partial J}{\partial y_{(k+i)}}\frac{\partial y_{(k+i)}}{\partial u_{(k)}} + \sum_{\substack{i=1\\ k+i\le N}}^{m-1} \frac{\partial J}{\partial u_{(k+i)}} \\ &\quad \cdot \frac{\partial u_{(k+i)}}{\partial u_{(k)}} + Ru_{fb(k)} \\ &= \sum_{\substack{i=k+1\\ i\le N+1}}^{k+m} \frac{\partial J}{\partial y_{(i)}}\frac{\partial y_{(i)}}{\partial u_{(k)}} + \sum_{\substack{i=k+1\\ i\le N}}^{k+m-1} \frac{\partial J}{\partial u_{(i)}}\frac{\partial u_{(i)}}{\partial u_{(k)}} \\ &\quad + Ru_{fb(k)}.\end{aligned} \tag{33}$$

By using the definition of sensitivities, (27) and (28)

$$\delta_u^k = \sum_{\substack{i=k+1,\\ i\le N+1}}^{k+m} \delta_y^i \frac{\partial y_{(i)}}{\partial u_{(k)}} + \sum_{\substack{i=k+1\\ i\le N}}^{k+m-1} \delta_u^i \frac{\partial u_{(i)}}{\partial u_{(k)}} - Ru_{fb(k)}. \tag{34}$$

This ISE also depends on the output sensitivities, and both are coupled to one another.

Since the plant dynamics (1) and the inverse dynamics (25) are not known, they are approximated by the corresponding networks, the neuro-identifier F and the feedback neuro-controller H, to yield

$$\delta_y^k = \sum_{\substack{i=k+1,\\ i\le N+1}}^{k+n} \delta_y^i \frac{\partial F_i}{\partial \hat{y}_{(k)}} + \sum_{\substack{i=k,\\ i\le N}}^{k+n-1} \delta_u^i \frac{\partial H_i}{\partial \hat{y}_{(k)}} + Q(y_{\text{ref}} - \hat{y}_{(k)}) \tag{35}$$

$$\delta_u^k = \sum_{\substack{i=k+1,\\ i\le N+1}}^{k+m} \delta_y^i \frac{\partial F_i}{\partial u_{(k)}} + \sum_{\substack{i=k+1,\\ i\le N}}^{k+m-1} \delta_u^i \frac{\partial H_i}{\partial u_{(k)}} - Ru_{fb(k)}. \tag{36}$$

It should be noted that the last terms in OSE and ISE are, respectively, the error terms for the output and input variables, and the terms under summation operations are the error (or delta) terms backpropagated through the networks F and H. For example, $\delta_y^i(\partial F_i/\partial \hat{y}_{(k)})$ [or $\delta_u^i(\partial F_i/\partial u_{(k)})$] is the error δ_y^i (or δ_u^i) backpropagated through the network F to the input node $\hat{y}_{(k)}$ (or $u_{(k)}$).

The objective of the GBTT is to compute the sensitivity δ_u^k, which will be used as the equivalent error for training of FBNC. This can be achieved by solving the OSE (35) and ISE (36) backward starting from $j = N+1$:

$j = N+1$:

$$\begin{aligned}\delta_u^{N+1} &= 0 \\ \delta_y^{N+1} &= Q(y_{\text{ref}} - \hat{y}_{(N+1)})\end{aligned}$$

$j = N$:

$$\begin{aligned}\delta_u^N &= \delta_y^{N+1}\frac{\partial F_{N+1}}{\partial u_{(N)}} - Ru_{fb(N)} \\ \delta_y^N &= \delta_y^{N+1}\frac{\partial F_{N+1}}{\partial \hat{y}_{(N)}} + \delta_y^N \frac{\partial H_N}{\partial \hat{y}_{(N)}} + Q(y_{\text{ref}} - \hat{y}_{(N)})\end{aligned}$$

$j = N-1$:

$$\begin{aligned}\delta_u^{N-1} &= \delta_y^{N+1}\frac{\partial F_{N+1}}{\partial u_{(N-1)}} + \delta_y^N \frac{\partial F_N}{\partial u_{(N-1)}} + \delta_u^N \frac{\partial H_N}{\partial u_{(N-1)}} \\ &\quad - Ru_{fb(N-1)} \\ \delta_y^{N-1} &= \delta_y^{N+1}\frac{\partial F_{N+1}}{\partial \hat{y}_{(N-1)}} + \delta_y^N \frac{\partial F_N}{\partial \hat{y}_{(N-1)}} + \delta_u^N \frac{\partial H_N}{\partial \hat{y}_{(N-1)}} \\ &\quad + \delta_u^{(N-1)}\frac{\partial H_{N-1}}{\partial \hat{y}_{(N-1)}} + Q(y_{\text{ref}} - \hat{y}_{(N-1)})\end{aligned}$$

$j = N-2$:

$$\begin{aligned}\delta_u^{(N-2)} &= \delta_y^{N+1}\frac{\partial F_{N+1}}{\partial u_{(N-2)}} + \delta_y^N \frac{\partial F_N}{\partial u_{(N-2)}} + \delta_y^{N-1}\frac{\partial F_{N-1}}{\partial u_{(N-2)}} \\ &\quad + \delta_u^N \frac{\partial H_N}{\partial u_{(N-2)}} + \delta_u^{N-1}\frac{\partial H_{N-1}}{\partial u_{(N-2)}} - Ru_{fb(N-2)} \\ \delta_y^{N-2} &= \delta_y^{N+1}\frac{\partial F_{N+1}}{\partial \hat{y}_{(N-2)}} + \delta_y^N \frac{\partial F_N}{\partial \hat{y}_{(N-2)}} + \delta_y^{N-1}\frac{\partial F_{N-1}}{\partial y_{(N-2)}} \\ &\quad + \delta_u^N \frac{\partial H_N}{\partial \hat{y}_{(N-2)}} + \delta_u^{N-1}\cdot\frac{\partial H_{N-1}}{\partial \hat{y}_{(N-2)}} + \delta_u^{N-2}\frac{\partial H_{N-2}}{\partial \hat{y}_{(N-2)}} \\ &\quad + Q(y_{\text{ref}} - \hat{y}_{(N-2)})\end{aligned}$$

$\cdots$

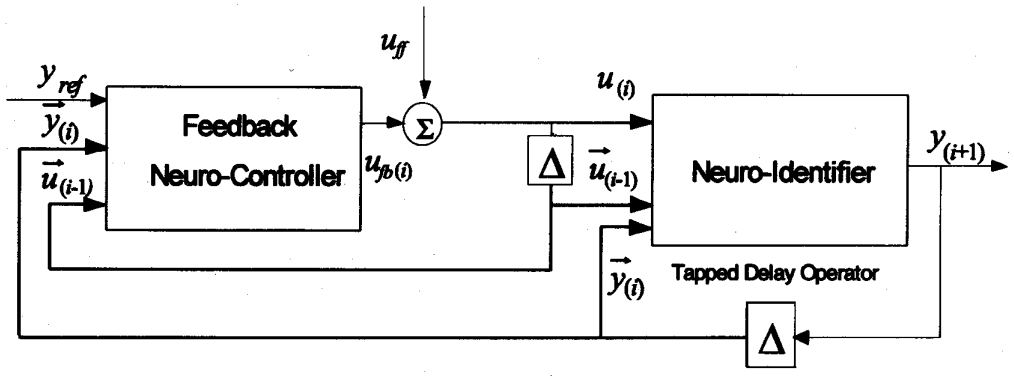


Fig. 4. Forward simulator for GBTT.

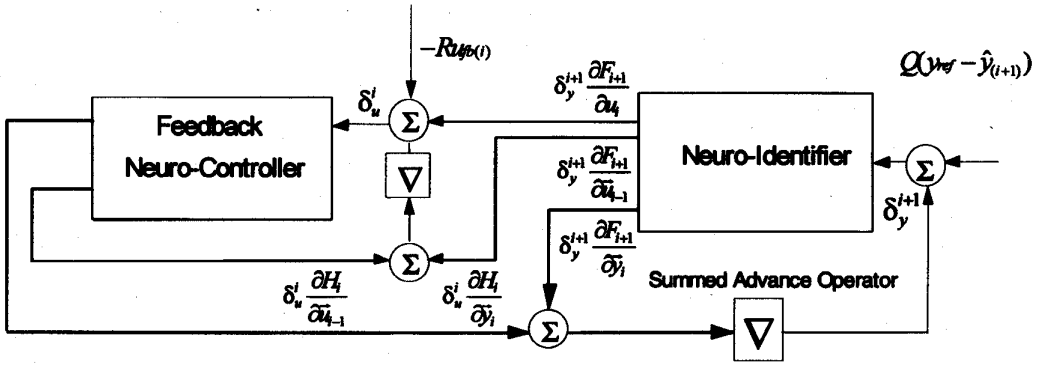


Fig. 5. Backward simulator for GBTT.

$j = k$:

$$\delta_u^k = \delta_y^{k+m}\frac{\partial F_{k+m}}{\partial u_{(k)}} + \cdots + \delta_y^{k+1}\frac{\partial F_{k+1}}{\partial u_{(k)}} + \delta_u^{k+m-1} \cdot \frac{\partial H_{k+m-1}}{\partial u_{(k)}} + \cdots + \delta_u^{k+1}\frac{\partial H^{k+1}}{\partial u_{(k)}} - Ru_{fb(k)}$$

$$\delta_y^k = \delta_y^{k+n}\frac{\partial F_{k+n}}{\partial \hat{y}_{(k)}} + \cdots + \delta_y^{k+1}\frac{\partial F_{k+1}}{\partial \hat{y}_{(k)}} + \delta_u^{k+n+1} \cdot \frac{\partial H_{k+n-1}}{\partial \hat{y}_{(k)}} + \cdots + \delta_u^{k+1}\frac{\partial H_{k+1}}{\partial \hat{y}_{(k)}} + \delta_u^k\frac{\partial H_k}{\partial \hat{y}_{(k)}} + Q(y_{\text{ref}} - \hat{y}_{(k)}).$$

3) Forward Simulator: Before solving the OSE and ISE, the error terms need to be generated. This can be done by driving the neuro-identifier with the controllers FFNC and FBNC for N steps forward. This process is illustrated by the *forward simulator* shown in Fig. 4, where the tapped delay operator is defined in Fig. 1.

Starting from initial conditions, $y_{(1)}, u_{(0)}$, the forward simulator generates sequences of inputs, $u_{fb(1)}, u_{fb(2)}, \cdots, u_{fb(N)}$, and outputs, $\hat{y}_{(2)}, \hat{y}_{(3)}, \cdots, \hat{y}_{(N+1)}$. These variables provide the error terms $(y_{\text{ref}} - \hat{y}_{(k)})$ and $u_{fb(k)}$ for any k.

4) Backward Simulator: The OSE and ISE are to be simulated backward in order to backpropagate the error terms. This can be performed by the *backward simulator* shown in Fig. 5, where *the summed advance operator* ∇ is defined as a "dual" of the tapped delay operator mapping from a sequence of vector input, $\{\vec{x}_{(i)}\}$, where $\vec{x}_{(i)} = (x_{(i,1)}, x_{(i,2)}, \cdots, x_{(i,n)})$, to a scalar output defined as $x_{(i)} = \Sigma_{k=1}^{n} x_{(i+k,k)}$.

The backward simulator can be shown to be the dual of the forward simulator, which is then constructed by using the *duality principle*, i.e., reversing the direction of arrows, interchanging the summers and nodes, and replacing the tapped delay operators with the summed advance operators.

Using the errors generated by the forward simulator, the backward simulator runs backward starting from $i = N$. It generates the sensitivities δ_y^{i+1} and δ_u^i. Noting that δ_u^i is in the output node of the FBNC, it is used as the equivalent error in computing the weight parameter adjustment in the FBNC, $\Delta\vec{W}^i$.

The process of the GBTT training algorithm is summarized as follows:

1) Set the weight parameters of the FBNC with small random numbers.
2) Set the reference output and initial state with random numbers in the operation region of the plant.
3) Run the forward simulator for N steps forward from $i = 1$.
4) Using the operation result in Step 3), run the backward simulator backward from $i = N$ to evaluate the equivalent error δ_u^i and the weight adjustment vector $\Delta\vec{W}^i$.
5) Update the weight parameters in the FBNC by using the average of the weight adjustment vectors found in Step 4).

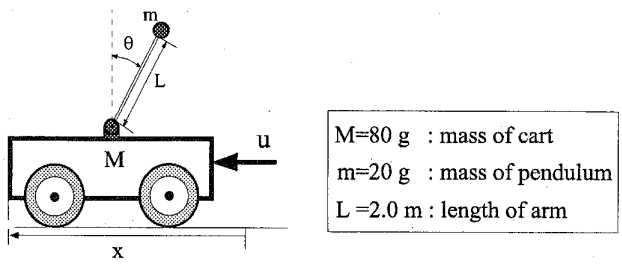


Fig. 6. Inverted pendulum.

6) Go to Step 2).

Training of the feedback controller is finished when the average decrease of the cost function converges to a small value for arbitrary reference outputs and initial conditions. Since the training algorithm is essentially a gradient descent method, the local minimum problem is a possibility. However, this problem can be avoided by starting with different initial weight parameters or with different numbers of nodes in the feedback controller [2], [18].

IV. Case Study

A prototypical control problem that has been widely used for neural network application is the pole balancing or the inverted pendulum problem. The problem involves balancing a pole hinged on a cart as shown in Fig. 6. This problem is of interest because it describes an inherently unstable system and is representative of a wide class of problems with severe nonlinearity in a broad operation region. Additional details of the pole balancing experiments including simulations can be found in [14] and [19]–[21]. In these works, the control problem has been limited to the case of balancing the pole at the vertical position.

The control objective of the pole-balancing problem in this paper is to extend the results of previous efforts by keeping the pole at an arbitrary angle, not necessarily at the vertical position, while minimizing a general quadratic cost function. Since a steady acceleration is required to maintain the pole at an angle, we assume that the cart can travel indefinitely in either direction. The OTNC is constructed and trained by the proposed method to meet the following control objectives:

1) Set the pendulum to an arbitrarily given reference angle.
2) Minimize the quadratic cost function while tracking the reference angle.

A. Training of the Neuro-Identifier

The nonlinear differential equation of the plant dynamics is as follows:

$$(M+m)\ddot{x} + mL\cos(\theta)\ddot{\theta} - mL\sin(\theta)\dot{\theta}^2 = u$$
$$m\ddot{x}\cos(\theta) + mL\ddot{\theta} = mg\sin(\theta) \quad (37)$$

where the variables and the parameters are defined in Fig. 6. The dynamics has severe nonlinearity when the angle deviates far from zero, in which case it is difficult to solve the control problem by any conventional method.

A paradigm for the neuro-identifier is chosen by trial and error. It consists of two hidden layers with 40 nodes each, an input layer with six input nodes and an output layer with one node. Three of the six input nodes are for output history, $y_{(k)}, y_{(k-1)}, y_{(k-2)}$; and two for input history, $u_{(k)}, u_{(k-1)}$; and one for bias input, 1.0. Training patterns of the neuro-identifier are generated from the mathematical model with random initial value and random input within the operation region of ± 1.1 [rad]. Discrete-time training patterns are obtained by applying the modified Euler method with time step-size of 0.13 [s] in simulation.

To avoid oscillation during training stage, weight parameters are corrected from the average of corrections calculated for every 10 patterns. After training the neuro-identifier for one hour in a SUN-SPARC2 workstation, it is tested with arbitrary initial conditions and sinusoidal inputs of different amplitude, which case is presented in Fig. 7. The neuro-identifier approximates the plant very closely and is sufficient for training the neuro-controllers.

B. Training of the Feedforward Controller

The FFNC has two hidden layers with 30 nodes each. The input layer has two nodes: one for reference output y_{ref} and one for the bias input, and the output layer has one node for the control u_{ff}. The reference output is given randomly to be within the operation region of ± 0.9 [rad] to train the FFNC, which is coupled with the neuro-identifier and the plant.

After training the FFNC for one hour, it is tested with a reference output varying within the operation region. Although the plant tracks the time-varying reference output, error remains small as shown in Fig. 8.

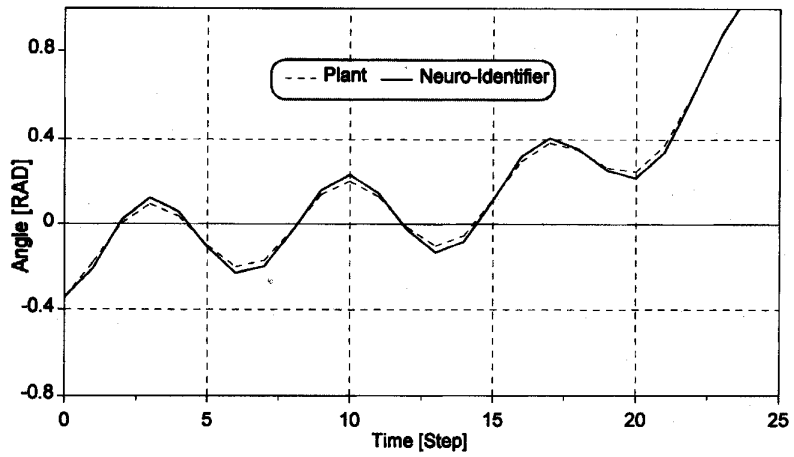
C. Training of the Feedback Neuro-Controller

The FBNC has two hidden layers with 30 nodes each. The input layer has five nodes: three for output history, $y_{(k)}, y_{(k-1)}, y_{(k-2)}$; one for previous input, $u_{(k-1)}$; and one for the bias input. The cost function for the N-step ahead optimal controller is set as

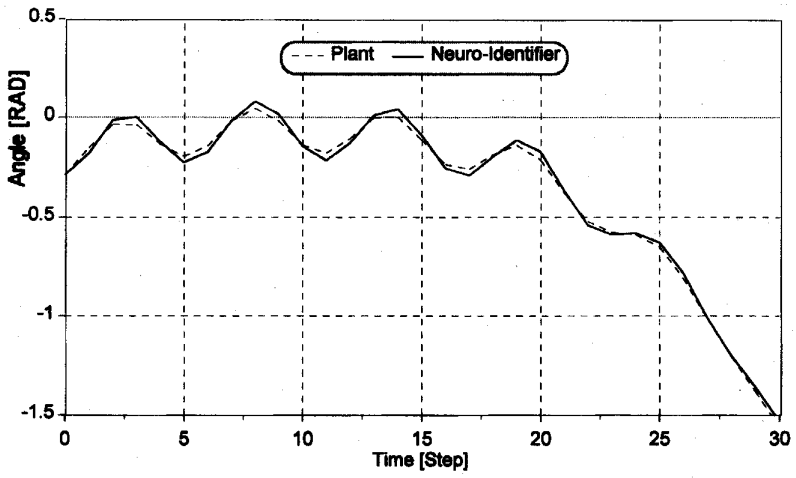
$$J = \frac{1}{2}\sum_{k=1}^{N}(1.0(y_{\text{ref}} - y_{(k+1)})^2 + 0.3(u_{fb(k)})^2). \quad (38)$$

The FBNC is trained once for an initial condition and a reference output, which are randomly selected while driving the plant for N steps. This training is repeated for other initial conditions and reference outputs. Each training is performed in two phases. First, the training is done with small N $(= 3)$ since the controller in the beginning has little knowledge of control. This also prevents the pendulum from falling down. Then, the step is increased gradually to $N = 15$. The second phase training is carried on with N fixed at 15.

After training the FBNC with the GBTT algorithm for two hours, it is tested with several nonzero set-points as presented in Fig. 9. It shows a larger overshoot for a larger set-point. A larger overshoot corresponds to an operating condition with severe nonlinearity.



(a)



(b)

Fig. 7. Training results of the neuro-identifier: (a) initial condition $\theta = -0.3$ and input $u = -0.9\cos(k)$ and (b) initial condition $\theta = -0.3$ and input $u = -0.8\cos(k)$.

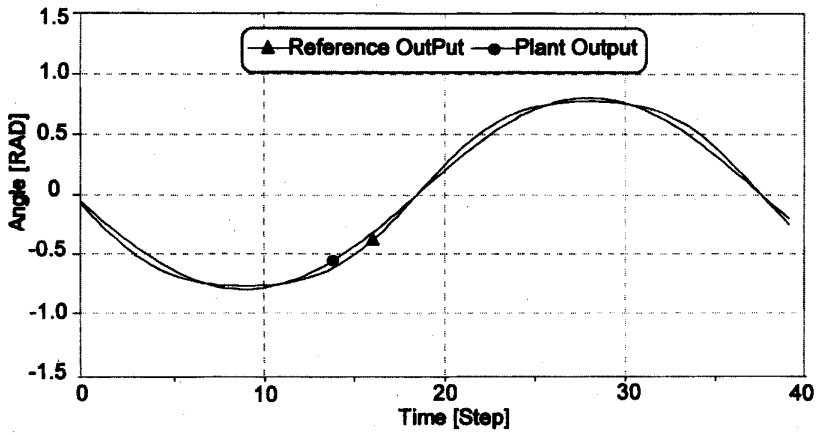


Fig. 8. Training result of the feedforward neuro-controller.

Fig. 10 shows a case for a set-point changing at each 40 time-steps. The trajectories of the plant output, reference output, and the control input are presented in the figure. It is seen that the OTNC for a nonlinear system behaves in a way similar to the usual optimal tracking controller for a linear quadratic problem [12]; the shapes of output

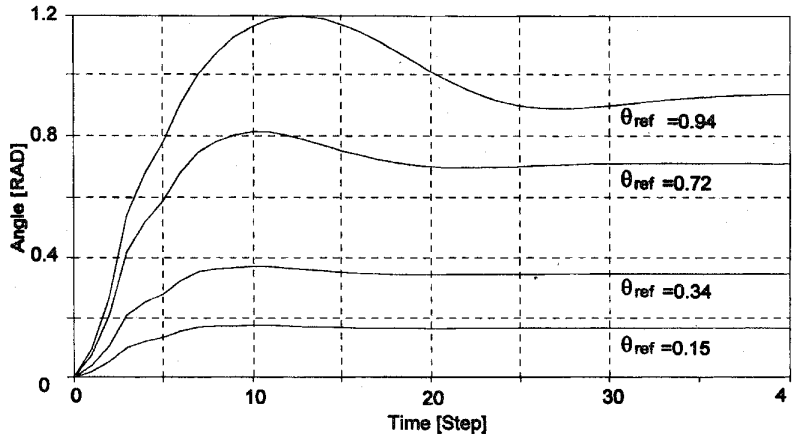


Fig. 9. Angle trajectories of the inverted pendulum for different reference set-points.

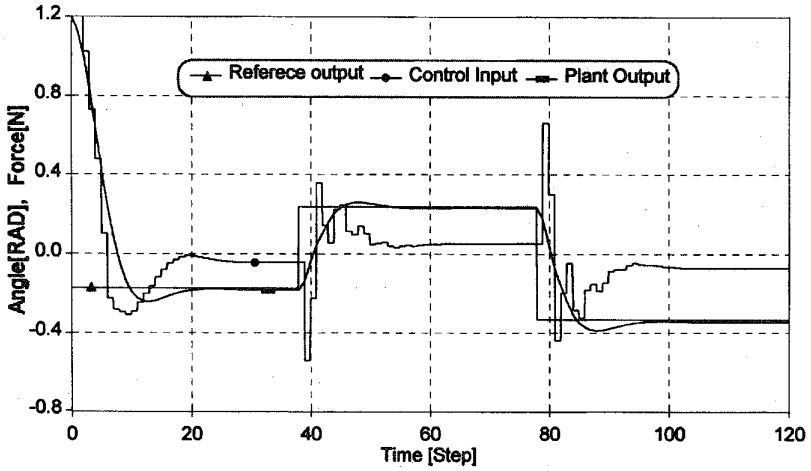


Fig. 10. Control result with the OTNC for changing reference set-point.

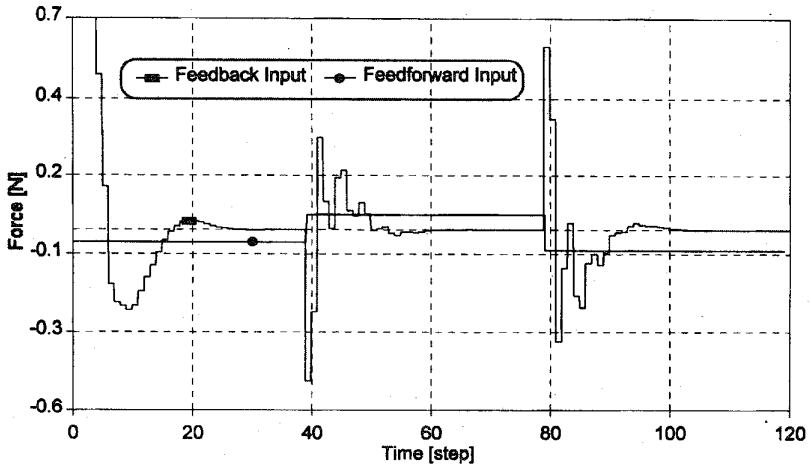


Fig. 11. Feedforward and the feedback control inputs for a changing reference set-point.

trajectories are typical fast responses with reasonable overshoots. Fig. 11 shows the corresponding feedforward and the feedback control inputs for the changing set-point. The FFNC generates the control input corresponding only to the reference output in steady state. On the other hand, the FBNC generates the control input corresponding to the regulating error between the reference and the plant outputs during transient.

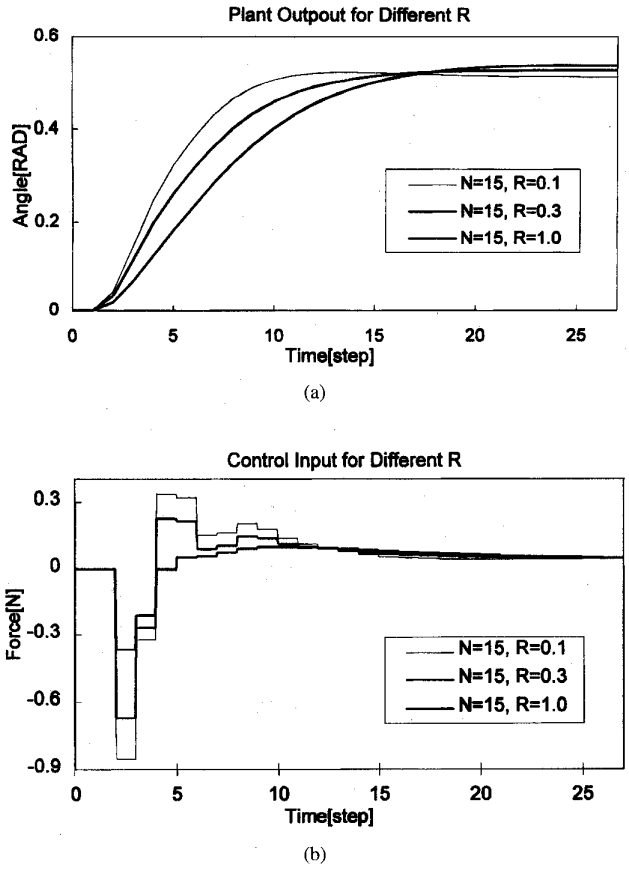


Fig. 12. Performance of the FBNC with different weight parameter R: (a) plant output and (b) control input.

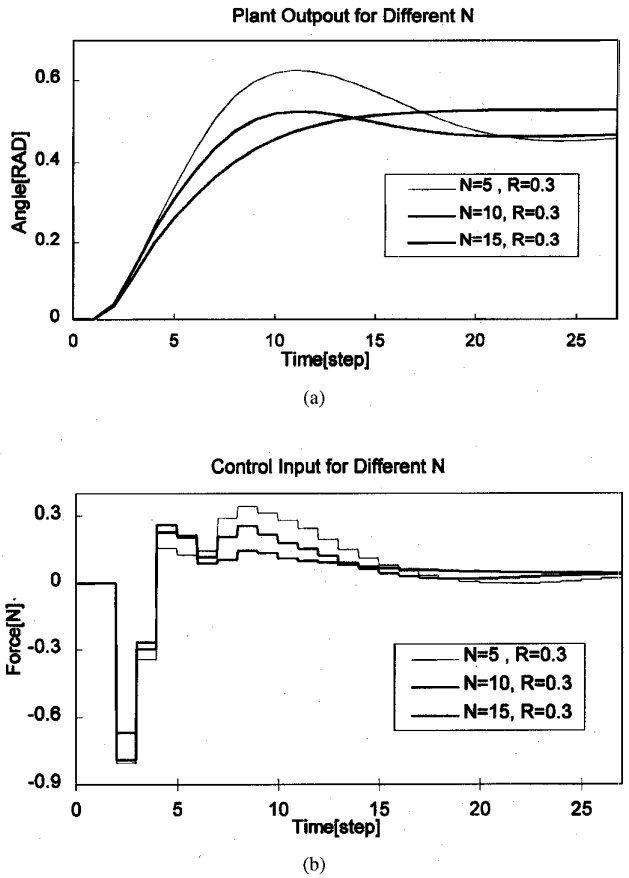


Fig. 13. Performance of the FBNC with different summation size N: (a) plant output and (b) control input.

D. Performance Evaluation

The general quadratic performance index (2) was used for the optimal tracking neuro-controlller. The performance index depends on the weighting factors Q and R, and the number of steps for summation, N. Fig. 12 shows the role of the weighting factor R when other parameters are fixed with $Q = 1.0$, and $N = 15$. As R increases the response slows down, becoming overdamped, and control effort becomes less with smaller magnitudes in force. This is due to the performance index giving more weight to the control effort as R increases.

The impact of N is also shown in Fig. 13. As N increases the response becomes less oscillating and more stable. This is because the performance index accumulates the error, both the input and output errors, over longer duration. This also reduces the control effort in the long run. When N becomes smaller than five, the plant becomes unstable. This demonstrates the importance of the general cost function, which is summed over some finite-time interval, compared to the usual instantaneous error function (corresponding to the case with $N = 1$).

V. Conclusions

For an optimal tracking control problem for nonlinear dynamic plants, a new architecture, the OTNC, is developed using feedback and feedforward controls.

First, the FFNC is introduced to solve the tracking problem with a nonzero set-point. A novel training method for the FFNC is developed by using the concept of an inverse mapping to generate the feedforward control input corresponding to the output set-point. Second, the FBNC is designed to solve an optimal regulator problem with general quadratic cost function. A GBTT training algorithm is developed to train the FBNC. The proposed OTNC scheme is demonstrated in a typical nonlinear plant, an inverted pendulum. The role of the general quadratic cost function is also demonstrated in the simulation. Simulation results show good performance over a wide range of nonlinear operation and the possibility of using the OTNC for the optimal tracking control of other nonlinear systems.

References

[1] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, pp. 1481–1497, Sept. 1990.
[2] D. Nguyen and B. Widrow, "The truck backer-upper: An example of self-learning in neural networks," *IEEE Contr. Syst. Mag.*, pp. 18–23, 1990.
[3] F. C. Chen, "Back-propagation neural networks for nonlinear self-tuning adaptive control," *IEEE Contr. Syst. Mag.*, pp. 44–48, 1990.
[4] Y. Iiguni and H. Sakai, "A nonlinear regulator design in the presence of system uncertainties using multilayered neural networks," *IEEE Trans. Neural Networks*, vol. 3, pp. 410–417, July 1991.
[5] C. C. Ku and K. Y. Lee, "Diagonal recurrent neural network for dynamic system control," *IEEE Trans. Neural Networks*, vol. 6, pp. 144–156, Jan. 1995.

[6] C. C. Ku, K. Y. Lee, and R. M. Edwards, "Improved nuclear reactor temperature control using diagonal neural networks," *IEEE Trans. Nucl. Sci.*, vol. 39, pp. 2298–2308, Dec. 1992.
[7] C. C. Ku and K. Y. Lee, "Diagonal recurrent neural network based control using adaptive learning rates," in *Proc. IEEE Conf. Decision and Control*, Dec. 1992, pp. 3485–3490.
[8] W. T. Miller, R. S. Sutton, and P. J. Werbos, *Neural Networks for Control.* Cambridge, MA: MIT Press, 1990, pp. 28–65.
[9] T. Yamada and T. Yabuta, "Neuro-controller using autotuning method for nonlinear functions," *IEEE Trans. Neural Networks*, vol. 3, pp. 595–601, July 1992.
[10] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, pp. 1550–1560, Oct. 1990.
[11] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamic systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–28, Mar. 1990.
[12] H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems.* New York: Wiley, 1972, pp. 201–319.
[13] U. diCaprio and P. P. Wang, "A study of the output regulator problem for linear system with input vector," in *Proc. 7th Annu. Allerton Conf. Circuits and System Theory*, 1969, pp. 186–188.
[14] J. E. Slotine, *Applied Nonlinear Control.* Englewood Cliffs, NJ: Prentice-Hall, 1991.
[15] A. Meystel and S. Uzzaman, "The planning of tracking control via search," in *Proc. Int Symp. Intelligent Control*, Chicago, 1993, pp. 554–559.
[16] M. I. Jordan and D. E. Rumelhart, "Forward models: Supervised learning with a distal teacher," *Cognitive Sci.*, vol. 16, pp. 307–354, 1992.
[17] T. Troudet, S. Garg, D. Mattern, and W. Merrill, "Toward practical control design using neural computation," in *Proc. Int. Joint Conf. Neural Networks*, Seattle, 1991, pp. II-675–681.
[18] R. Hecht-Nilsen, *Neuro Computing.* Reading, MA: Addison-Wesley, 1989, pp. 115–119.
[19] A. P. Wieland, "Evolving neural network controllers for unstable systems," in *Proc. Int. Joint Conf. Neural Networks*, Singapore, 1992, pp. 214–219.
[20] V. Williams and K. Matsuoka, "Learning to balance the inverted pendulum using neural networks," in *Proc. Int. Joint Conf. Neural Networks*, Singapore, 1992, pp. 214–219.
[21] B. Friedland, *Control System Design: An Introduction to State-Space Methods.* New York: McGraw-Hill, 1986.

Young-Moon Park (M'84–SM'91–F'96) was born in Masan, Korea on August 20, 1933. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from Seoul National University, Seoul, Korea, in 1956, 1959, and 1971, respectively.

Since 1959, he has been on the faculty of Seoul National University, where he is currently a Professor of Electrical Engineering. He was a Visiting Professor at Tokyo University, Japan, in 1981, the University of Houston, TX, in 1983, and the Pennsylvania State University, University Park, in 1987. His major research interests include power system operation and planning, and artificial intelligence applications to power systems.

Dr. Park is a past President of the Korea Institute of Electrical Engineers and is now serving as the President of the Electrical Engineering and Science Research Institute. He was the Chairman of the Organizing Committee and Co-chairman of the International Federation of the Automatic Control Symposium on Control of Power Plants and Power Systems held in Seoul in 1989.

Myeon-Song Choi (S'93–M'95) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Seoul National University, Seoul, Korea, in 1989, 1991, and 1996, respectively.

He is currently an Instructor of Electrical Engineering at Soong Sil University, Seoul, Korea. His research interests include robust control theory, artificial neural networks, and their applications to power system stabilizing control.

Kwang Y. Lee (S'70–M'72–SM'86) was born in Pusan, Korea on March 6, 1942. He received the B.S. degree in electrical engineering from Seoul National University, Seoul, Korea, in 1964, the M.S. degree in electrical engineering from North Dakota State University, Fargo, in 1967, and the Ph.D. degree in systems science from Michigan State University, East Lansing, in 1971.

He has been on the faculties of Michigan State, Oregon State, Corvallis, the University of Houston, TX, and the Pennsylvania State University, University Park, where he is currently a Professor of Electrical Engineering. He was a Visiting Professor at Seoul National University in 1993, where he conducted a U.S.–Korea Cooperative Research on Intelligent Distributed Control of Power Plants and Power Systems. His current research interests include control theory, artificial neural networks, fuzzy logic systems, and computational intelligence and their applications to power systems. He is currently the Director of the Power Systems Control Laboratory and Co-director of Intelligent Distributed Controls Research Laboratory at Pennsylvania State University.