'92 KACC 1992. 10. 19～21

# A Tracking Controller Using Multi-layered Neural Networks

Byeong Woo Bae and Gi Joon Jeon

Department of Electronics
Kyungpook National University
Taegu, Korea

Kyung Youn Kim

Department of Electronics
Cheju National University
Cheju, Korea

Abstract

This paper addresses the problem of designing a neural network based controller for a discrete-time nonlinear dynamical system. Using two multi-layered neural networks we first design an indirect controller the weights of which are updated by the informations obtained from system identification. The weight update is executed by parameter optimization method under Lagrangian formulation. For the nonlinear dynamical system, we define several cost functions and by computer simulations analyze the control performances of them and the effects of penalty-weighting values.

## 1. Introduction

Neural networks have been used for the design of control systems since they have potential to treat many problems that cannot be handled by conventional analytic approaches. A multi-layered neural network using back-propgation learning algorithm is the most prevalent neural network topology for control applications because they have the capability to learn system charateristics through nonlinear mapping[1].

Recently, the application of neural network to the control system design has tendencies to integrate linear control techniques and neural network learning methods [2-3], to utilize the input/output information of the system embedded in neural networks[4-5], and to autotune the parameters for training of the neural network, i.e., the shape of the activation function[6].

In real industial processes it is important to consider the control scheme with constrained control input. However, many studies[2,4-6] have ignored some restrictions on control efforts such as control energy, control rates and control bounds etc.. There have been some studies on solving the regulator and tracking problems: Iiguni et al.[3] have presented the optimal regulator realized by minimizing a quadratic cost function which includes a energy term. However, the analyses(or comparisons) on such performance indices as transient response, steady-state error and control energy have not been shown. Nguyen et al.[7] have presented that their self-learning control system with a neural emulator can be realized by minimizing a quadratic cost function involving the control energy as well as the final state error. Troudet et al.[8] have proposed a method of learning the neural controller by minimizing an objective function which is a weighted sum of tracking erorrs and control inputs.

In this paper we propose an effective method that designs an indirect controller using two multi-layered neural networks(MLNN), one for identification and the other for control, with backpropagation learning archi-tecture. The former is trained so that the MLNN has the chracteristics of a nonlinear dynamical system to be controlled, and the latter obatains the control input from Lagrangian formulation[9-10] with a given cost function and some constraints. Furthermore, before investigating the performace of the control system with the control method represented, we define several cost functions and compare the control performance not only by the types of cost functions but also in terms of the penalty-weighting values. Some computer simulations show the efficiency of the proposed control method.

## 2. Optimization Problem of Nonlinear Control Systems

### 2.1 Problem Formulation

Consider a discrete-time nonlinear dynamical system:

$$x(k+1) = \Phi(x(k), u(k)) \tag{1}$$
$$y(k) = \Psi(x(k)) \tag{2}$$

where $x(k) \in R^p$ and $y(k) \in R^n$ are the state vector and the mearsured output vector of the system, respectively, and $u(k) \in R^m$ is the control input vector. $\Phi$ and $\Psi$ are the nonlinear functions that describe the charateristics of the system.

When the model of a real process is given as linear equations, it is currently well established to find the optimal control input which minimizes a cost function similar to the following form:

$$J = \frac{1}{K}\sum_{k=1}^{K}\{(y_d(k+1)-y(k+1))^T Q (y_d(k+1)-y(k+1)) + u(k)^T R u(k)\} \tag{3}$$

where $y_d(k+1) \in R^n$ is the desired output vector of the system, and Q and R, the symmetric positive definite penalty-weighting matrices of the output vector and the input vector, are generally chosen with regard to the physical conditions.

On unknown nonlinear dynamical systems, however, it is difficult not only to obtain the model of the system but to derive the optimal control input satisfying a system objective. In this paper we use neural networks to obtain control input which minimizes a cost function subject to system equations, (1) and (2).

## 2.2 System Identification Scheme

In this section, since a system to be controlled is assumed to be unknown and nonlinear, a neural network based indentifier is developed for the solution of the problem stated in the last section. To find the dynamic charateristics or the parameters of the system, we use an MLNN with no inner feedback loop, which will be called NNI and is shown in Fig. 1. The inputs of the MLNN consist of the delayed system output vector and the control input vector. The training signals(signal vector) for updating the weights of the MLNN are errors(error vector) between the system output vector as desired values and the MLNN output vector as actual values.

The output vector of the NNI is described as follows:

$$\begin{aligned}\hat{y}(k+1) &\overset{\Delta}{=} z(k,T)\\ &= f(y(k),u(k)) \qquad (4)\end{aligned}$$

where $f(\cdot)$ and $z(k,T)$ denote the input/output mapping function and node output vector of the last layer of the NNI, respectively.

The node output vector of each layer is

$$\begin{aligned}z(t) &= f_t(s(k,t))\\ &= f_t(W(k,t),z(k,t-1)) \quad \forall\ t \in [1,\ T] \qquad (5)\end{aligned}$$

where $f_t(\cdot)$ denotes an activation function which is a sigmoid function type in the t-th hidden layer. And, $s(k,t)$ and $z(k,t)$ represent the activation value and node output vector of the t-th layer, respectively. $W(k,t)$ is the weight matrix between the (t-1)th and t-th layer in the time k. For simplicity, the (k,t) term of the above notations is denoted to (t). In case t=0, z(t) is defined to be the input vectors, y(k) and u(k).

When the error cost function for system identification is given as the following form

$$J_I = \tfrac{1}{2}\{(y(k+1)-\hat{y}(k+1))^T(y(k+1)-\hat{y}(k+1))\} \qquad (6)$$

the method how to minimize (6) can simply be summerized by the BP learning algorithm[1] with the steepest descent technique to update the weights of the MLNN.

$$W(k+1,t) = W(t) + \Delta W(t) \quad \forall\ t \in [1,\ T] \qquad (7)$$

where,

$$\Delta W(t) = \delta(t)z(t-1) + \alpha\ \Delta W(k-1,t) \quad \forall\ t \in [1,\ T] \qquad (8)$$

$$\delta(T) = \frac{\partial z^T(T)}{\partial W(T)}\ (y(k+1)-\hat{y}(k+1)) \qquad (9)$$

$$\delta(t) = \frac{\partial z^T(t)}{\partial W(t)}\ \delta(t+1)W(t+1) \quad \forall\ t \in [1,\ T-1] \qquad (10)$$

and $\eta$ is a step size and $\alpha$ is a momentum term.

## 2.3 Control Scheme

With the nueral indentifier developed in the last section, we find the control input which minimizes the cost function similar to (3). The cost function need to be chosen in such a way that several conditions be satisfied: the realization of a system objective, the uniformly bounded control input, the minimization of control energy, the lower control rates, etc.. To solve the above problem, we later define the cost function and the parameters of it.

In this section, for simplicity we consider the cost function without a control energy term as follows:

$$J_c = (y_d(k+1)-y(k+1))^T(y_d(k+1)-y(k+1)). \qquad (11)$$

Based on the Lagrangian formulism, the question to find the control input that minimizes the cost function is a parameter optimization problem by gradient method with equality constraints:

$$\begin{aligned}z_c(t) &= f_t(s_c(t))\\ &= f_t(V(t),z_c(t-1)) \qquad (12)\end{aligned}$$

Under the assumption that the system output vector is equivalent to the NNI output vector, the Lagrange function is defined as follows:

$$\begin{aligned}L(z_ct),V(k,t),\lambda(t)) &= (y_d(k+1)-\hat{y}(k+1))^T(y_d(k+1)-\hat{y}(k+1))\\ &+\sum_{t=1}^{T} \lambda^T(t)\ (z_c(t)-f_t(s_c(t))). \qquad (13)\end{aligned}$$

In the below, notations are summerized with illustration of an MLNN for controller(NNC). The NNC is the same structure as Fig. 1. The output vector and input vectors of the NNC is $u(k)$, $y_d(k+1)$ and $y(k)$, respectively. The weight matrix and output vector in the last section are represented by $V(t)$ and $z_c(t)$, respectively. Fig. 2 shows the overall structure of the NNC and the NNI.

In (12) and Fig. 2, $\lambda(t)$ denotes a Lagrange multiplier vector of the t-th layer and it may be recognized as the back-propagated gradient term. If we consider the optimization problem of the cost function (13), $\nabla L(z_c,V,\lambda)=0$ is a necessary condition which gives a local minimum of the cost function with respect to the stationary point. And the condition is split into three subconditions:

$$\frac{\partial L(z_c(t),V(t),\lambda(t))}{\partial \lambda(t)} = 0 \qquad (14a)$$

$$\frac{\partial L(z_c(t),V(t),\lambda(t))}{\partial z_c(t)} = 0 \qquad (14b)$$

$$\frac{\partial L(z_c(t),V(t),\lambda(t))}{\partial V(t)} = 0 \qquad (14c)$$

From (14a), the state vector $z_c(t)$ is given as the forward dynamic equation

$$z_c(t) = f_t(s_c(t)) \quad \forall\ t\in[1,\ T]. \tag{15}$$

From (14b), the costate vector $\lambda(t)$ is given as the backward dynamic equation

$$\lambda(t) = \frac{\partial z_c{}^T(t+1)}{\partial z_c(t)}\ \lambda(t+1) \quad \forall\ t\in[1,\ T-1] \tag{16}$$

with boundary condition which can be obtained as mapping the NNI output vector to the NNC output vector by the partial derivative and chain rule,

$$\lambda(T) = D(T)\ (y_d(k+1)-\hat{y}(k+1)) \tag{17}$$

where

$$D(T) = -2\prod_{t=0}^{T-1}\left(\frac{\partial z(t+1)}{\partial s(t+1)}\ \frac{\partial s(t+1)}{\partial z(t)}\right)^T. \tag{18}$$

From (14c), the gradient is used to compute the steepest descent for the weight update, i.e.,

$$\Delta V(t) = \eta\ \nabla_{V(t)} f(s(t))\ \lambda(t)\ z_c(t-1) \quad \forall\ t\in[1,\ T] \tag{19}$$

and

$$V(t+1) = V(t) + \Delta V(t) \tag{20}$$

where $\eta$ is step size. We repeat this procedure until the change in the cost function

$$\nabla J = \left[\frac{\partial L(z_c(t),V(t),\lambda(t))}{\partial V(t)}\right]^T \frac{\partial L(z_c(t),V(t),\lambda(t))}{\partial V(t)} \tag{21}$$

is smaller than some particular pre-chosen small number depending mainly on the accuracy of the control performance required.

Although the last condition (14c) is the problem of the weight parameter optimization, we can derive the optimal control based on cost function when this condition is satisfied. This can be shown to be equivalent to finding a minimum of Lagrangian function while satisfying the first two subconditions, (14a) and (14b).

The constrained optimization procedure described untill now is summerized as follows:

i ) identify the system to be controlled
ii ) solve the state and costate vector that satisfy the first two necessary conditions
iii ) compute the weight matrices using the state and costate vector

Before investigating the system performace of the control method represented, we define several cost functions corresponding to (3) or (11) and compare the control performances not only by the types of cost functions but also in terms of the selection of the penalty-weighting matrices.

Type A:

$$J_c = (y_d(k+1)-\hat{y}(k+1))^TQ(y_d(k+1)-\hat{y}(k+1)) \tag{22}$$

Type B:

$$J_c = (y_d(k+1)-\hat{y}(k+1))^TQ(y_d(k+1)-\hat{y}(k+1)) + u(k)^TRu(k) \tag{23}$$

Type C:

$$J_c = (y_d(k+1)-\hat{y}(k+1))^TQ(y_d(k+1)-\hat{y}(k+1)) + u(k)^TRu(k) + \Delta u(k)^TP\Delta u(k)\} \tag{24}$$

Type D:

$$J_c = (y_d(k+1)-\hat{y}(k+1))^TQ(y_d(k+1)-y(k+1)) + u(k)^TRu(k) + \Delta u(k)^TP\Delta u(k) \tag{25}$$

Type A has been widely used in the studies of a control system using a neural network. From the viewpoint of control efforts, however, the cost fuction need to be modified because controller may excessively supply control input for the system. Type B consists of penalty terms on both the control energy and the trajectory. In Type C, $\Delta u$ represents control input incremментants of which need to be kept in minimum in real control systems. When the dynamic characteristics of the system is not equal to that of the NNI, the control input derived from the NNI results in a poor system performance. To treat such a problem, the quasi-squared-error cost function of Type D can be employed.

The boundary conditions of the types of cost functions defined above are obtained as follows:

$$\text{Type B}: \lambda(T) = D(T)(y_d(k+1)-\hat{y}(k+1)) + Ru(k) \tag{26}$$

$$\text{Type C}: \lambda(T) = D(T)(y_d(k+1)-\hat{y}(k+1)) + Ru(k) + P\Delta u(k) \tag{27}$$

$$\text{Type D}: \lambda(T) = D(T)(y_d(k+1)-y(k+1)) + Ru(k) + P\Delta u(k) \tag{28}$$

The penalty-weighting matrices of the boundary conditions are determined by the physical characteristics and the objective of the system. This paper will analyze their influences by simulations only.

## 3. Simulations and Discussions

It is assumed that the system to be controlled is given as the SISO plant of the form

$$y(k+1) = \frac{u(k)y(k)}{1.\ + y^2(k)} + u^3(k)$$

where the output is a nonlinear function of the input and the past output. This plant is to be controlled so as to track a sine wave followed by a square wave as shown in Fig. 3(a). The bound of control input was contrained by the values between -1 and +1. MLNN's for indetification and control are two-layered MLNN with two inputs and one output. With the weights initialized at random, each MLNN was learned by on-line and concurrently run the plant so as to minimize each cost function defined in the last section.

The output responses during training on-line are shown in Fig. 3 and Fig. 4, and these figures show the results from the control input acquired using Type D cost function without penalty-weighting on the input and the increment of the input and Type A, respectively.

Fig. 5 and Fig. 6 show the effects of the penalty-weighting when Type B and C is used as a cost function. From the results, we could recognize that Fig. 6 shows a better transient response than Fig. 5.

## 4. Conclusions

Using two multi-layered neural networks we have designed an indirect controller the weights of which were updated by the informations obtained from system identification. The weight update was executed by parameter optimization method under Lagrangian formulation. For the nonlinear dynamical system, we have also defined several cost functions and analyzed the control performances of them and the effects of penalty-weighting values by computer simulations. It is noted from the simulation results that the output response with the quasi-squared-error cost function is much improved than the response with the conventional squared-error cost function.

## 5. References

[1] D.E. Rumelhart and et al., *Parrallel Distributed processing*, Cambridge, MA: MIT Press, 1, 1986.

[2] M. Kawato, Y. Uno, M. Isobe, and R. Suzuki, "Hierarchical Neural Network Model for Voluntary Movement with Application to Robotics,"*IEEE Contr. Syst. Mag.*, 8, 8-16, Apr. 1988.

[3] Y. Iiguni, H. Sakai and H. Tokumara, "A Nonlinear Regulator Design in the Presence of System Uncertainties Using Multi-layered Neural Networks," *IEEE Trans. Neural Networks*, 2, 410-417, 1991.

[4] M.A. Sartori and P.J. Antsaklis, "Implementations of Learning Control Systems Using Neural Networks," *IEEE Cont. Sys. Mag.*, 49-57, Apr. 1992.

[5] G.J. Jeon, B.W. Bae and J.O. Jang, "Supervised Self-Learning Controller Design Using Parallel Connected Neural Networks," *Proc. SICE*, 1273-1275, 1992.

[6] T. Yamada and T. Yabuta, "Neural Network Controller Using Autotunning Method for Nonlinear Functions," *IEEE Trans. Neural Networks*, 3, 595-601, 1992.

[7] D.H. Nguyer and B. Widrow, "Neural Networks for Self-Learning Control Systems," *IEEE Contr. Syst. Mag.*, 18-23, Apr. 1990.

[8] T. Troudet, S.Garg, D.Mattern and W.Merrill, "Towards Practical Control Design Using Neural Computaion," *IEEE Int. Conf. Neural Networks*, 675-681, Jun. 1991.

[9] A.E. Bryson and Y.C. Ho, *Applied Optimal Control*, Hemishpere Publishing Co., 1975.

[10] M.G. Singh and A. Titli, *Systems:Decomposition, Optimization and Control*, Pergamon Press, Chap. 7, 1978.

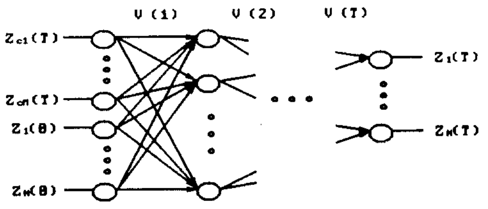


Fig. 1. The structure of the NNI

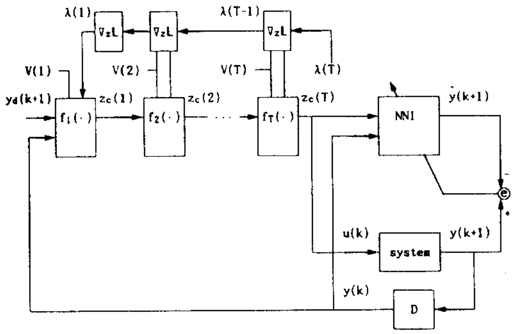
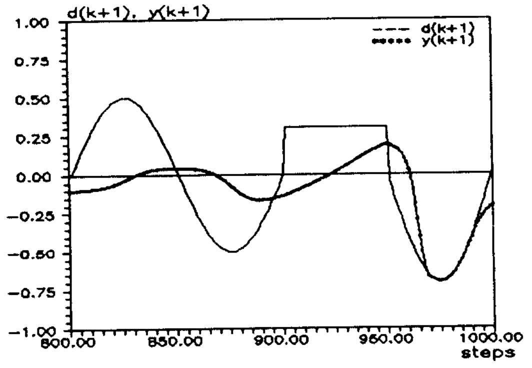
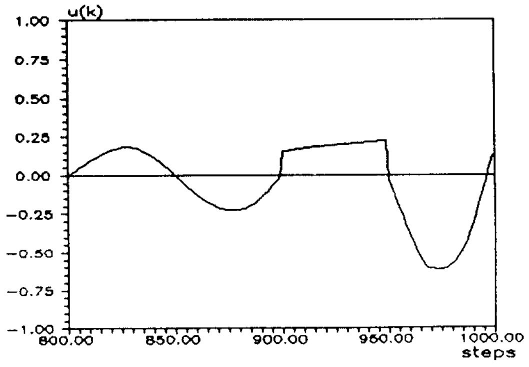


Fig. 2. The structure of the learning control system

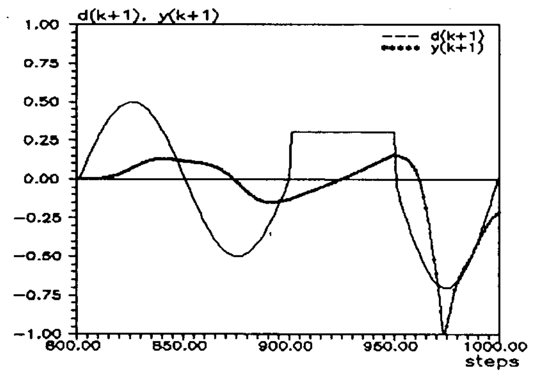


(a) Output response

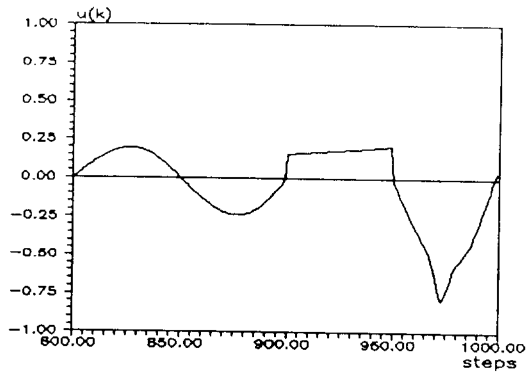


(b) Input response

Fig. 3. Input and output response of Type D without penalty-weighting on control



(a) Output response



(b) Input response

Fig. 4. Input and output response of Type C

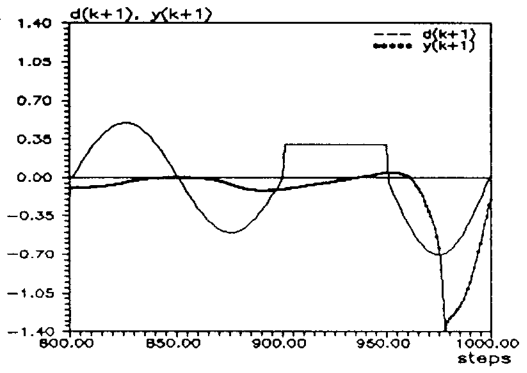


Fig. 5. Output response of Type B with R=0.01 and P=0.05

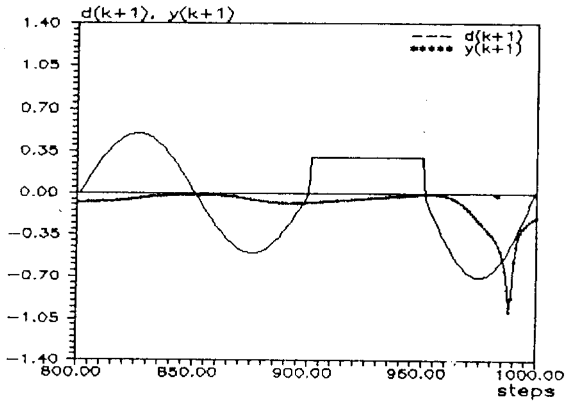


Fig. 6. Output response of Type C with R=0.01