Diagonal Recurrent Neural Networks for Dynamic Systems Control

Chao-Chee Ku, *Student Member, IEEE*, and Kwang Y. Lee, *Senior Member, IEEE*

***Abstract*— A new neural paradigm called diagonal recurrent neural network (DRNN) is presented. The architecture of DRNN is a modified model of the fully connected recurrent neural network with one hidden layer, and the hidden layer is comprised of self-recurrent neurons. Two DRNN's are utilized in a control system, one as an identifier called diagonal recurrent neuroidentifier (DRNI) and the other as a controller called diagonal recurrent neurocontroller (DRNC). A controlled plant is identified by the DRNI, which then provides the sensitivity information of the plant to the DRNC. A generalized dynamic backpropagation algorithm (DBP) is developed and used to train both DRNC and DRNI. Due to the recurrence, the DRNN can capture the dynamic behavior of a system. To guarantee convergence and for faster learning, an approach that uses adaptive learning rates is developed by introducing a Lyapunov function. Convergence theorems for the adaptive backpropagation algorithms are developed for both DRNI and DRNC. The proposed DRNN paradigm is applied to numerical problems and the simulation results are included.**

I. Introduction

As Antsaklis [1] pointed out, the development in the control area has been fueled by three major needs: the need to deal with increasingly complex systems, the need to accomplish increasingly demanding design requirements, and the need to attain these requirements with less precise advanced knowledge of the plant and its environment. Increasingly complex dynamical systems with significant uncertainty have forced system designers to turn away from conventional control methods. However, the fundamental shortcomings of current adaptive control techniques [2], such as nonlinear control laws which are difficult to derive, geometrically increasing complexity with the number of unknown parameters, and the general unsuitability for real time applications have compelled researchers to look for solutions elsewhere.

The massive parallelism, natural fault tolerance and implicit programming of neural network computing architectures suggest that they may be good candidates for implementing real-time adaptive controllers for large-scale nonlinear dynamic systems [3]. Several neural network models and neural learning schemes were applied to system controller design during the last three decades, and many promising results are reported [4]–[13]. Most people used the feedforward neural network (FNN), combined with tapped delays, and the backpropagation training algorithm [14] to solve the dynamical problems; however, the feedforward network is a static mapping and without the aid of tapped delays it does not represent a dynamic system mapping. On the other hand, recurrent neural networks [15]–[19] have important capabilities not found in feedforward networks, such as attractor dynamics and the ability to store information for later use. Of particular interest is their ability to deal with time-varying input or output through their own natural temporal operation [20]. Thus the recurrent neural network (RNN) is a dynamic mapping and is better suited for dynamical systems than the feedforward network.

Almeida [21] pointed out that one should not expect a major increase in the performance of a perceptron in every situation, just by "throwing in" feedback. In most cases, the best network structure will probably turn out to have feedback only in a smaller group of units. Ljung [22] also mentioned that, for system identification, the identifier must be chosen to have a small number of parameters, i.e., fewer weights for our neural network model. This is because the more parameters we use, the higher is the random influence on the model. Since a recurrent neuron already has an internal feedback loop, it captures the dynamic response of a system without external feedback through tapped delays; thus the network model can be simplified. In most control applications, the real-time implementation is very important, and thus the neurocontroller also needs to be designed such that it converges with a relatively small number of training cycles.

With the objective of a simple recurrent network and a shorter training time for the neural network model, a diagonal recurrent neural network (DRNN), as shown in Fig. 1(b), is developed. It can be shown that the DRNN model is a dynamic mapping in a way the fully connected recurrent neural network (FRNN) shown in Fig. 1(c) is dynamic. Since there is no interlinks among neurons in the hidden layer, the DRNN has considerably fewer weights than the FRNN and the network is simplified considerably.

This paper is organized as follows. In Section II, a DRNN model is developed. Also, the comparison of DRNN, FNN, and FRNN in terms of their total number of weights, I/O mapping characteristics. Then a dynamic backpropagation training algorithm is developed to train a DRNN based control system. In Section III, the convergence of the DRNN based system is investigated, and an analytic method based on the

Manuscript received March 3, 1992; revised December 7, 1992. This work was supported in part by a Navy/ASEE Summer Faculty Fellowship, by the Environment Tectonics Corporation Neural Network Flight Control Project, by NSF Grant EID-9212132, by the Research and Curriculum Development for Power Plant Intelligent Distributed Control, and by NSF/EPRI Joint Project ECS-9216504, Experimental Development of Power Reactor Intelligent Control. However, any findings, conclusions, or recommendations expressed herein are those of authors and do not necessarily reflect the views of NAWC, ETC, NSF, or EPRI.

The authors are with the Department of Electrical and Computer Engineering, The Pennsylvania State University, University Park, PA 16802 USA.

IEEE Log Number 9207442.

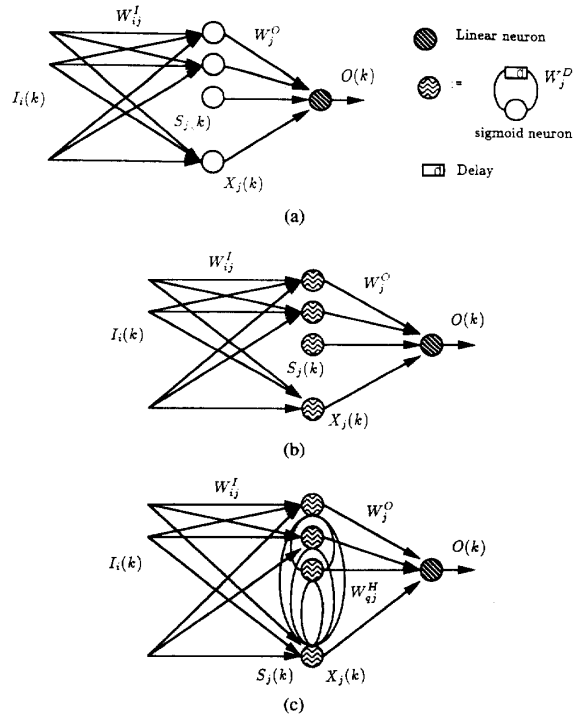


Fig. 1. Three different neural network architectures. (a) Feedforward neural network. (b) Diagonal recurrent neural network. (c) Fully connected recurrent neural network.

Lyapunov function is proposed to find the adaptive learning rates for the DRNN. Some simulations are conducted in Section IV. In the simulations, the bounded input bounded output (BIBO) nonlinear plant control, the on-line adapting ability of DRNN based control system, the non-BIBO nonlinear plant, the tolerance to disturbances, and the interpolation ability of the DRNN based control system are investigated.

II. Diagonal Recurrent Neural Networks

Consider Fig. 1, where for each discrete time k, $I_i(k)$ is the ith input, $S_j(k)$ is the sum of inputs to the jth recurrent neuron, $X_j(k)$ is the output of the jth recurrent neuron, and $O(k)$ is the output of the network. Depending on the network, W^I, W^O, W^D, or W^H represents input, output, diagonal, or hidden weight vectors, respectively. In this section the characteristics of DRNN, FNN, and FRNN shown in Fig. 1 are compared. Then the application of DRNN in control is developed, which includes developing the training algorithm for the DRNI and DRNC and finding the bounds of the learning rates such that the convergence of DRNN based control system is guaranteed.

A. The Comparison of DRNN, FNN, and FRNN

Definition 1: An ordered tuple $N^T = \{I^p, H^q, O^r\}$ represents a T-type neural network with p inputs (I^p), q sigmoid neurons in the hidden layer (H^q), and r linear neurons in the output layer (O^r); N^F, N^D, and N^R represent the feedforward, diagonal recurrent, and fully connected recurrent neural networks, respectively.

Definition 2: The variable G^T represents the total number of weights for a T-type neural network, where the symbol of type T is same as defined in Definition 1.

Lemma 1: The total number of weights (including q bias weights), for the N^F, N^D, and N^R neural networks are

$$G^F = (p+r+1)q \tag{1}$$

$$G^D = (p+r+2)q \tag{2}$$

$$G^R = (p+r+1)q + q^2 \tag{3}$$

where variables are defined in Definitions 1 and 2.

Remark 1: If $p = 4$, $q = 9$, and $r = 1$, then $G^F = 54$, $G^D = 63$, and $G^R = 135$. In this small neural network case, the number of weights of FRNN is about twice the number of weights of DRNN. The increase in the number of weights from DRNN to FRNN is $\Delta = q(q-1)$, which is 72 in the example.

Lemma 2: Define the I/O mapping, $M: I^m \rightarrow O^n$, and O^F, O^D, and O^R be the outputs of N^F, N^D, and N^R, respectively. If we assume $m = 3$, $n = 1$, and $I^3 = \{r(k), y(k), u(k)\}$. Then

$$O^F(k) = Q_N^F(r(k), y(k), u(k)) \tag{4}$$

$$O^D(k) = Q_N^D(r(l), y(l), u(l); l \le k) \tag{5}$$

$$O^R(k) = Q_N^R(r(l), y(l), u(l); l \le k) \tag{6}$$

where $Q_N(\cdot)$ is a nonlinear function, and l and k are nonnegative integers.

Proof: a) From Fig. 1(a), we obtain

$$O^F(k) = \sum_{j=1}^{n} W_j^O f_N(S_j(k)) = \sum_{j=1}^{n} W_j^O f_N\left(\sum_{i=1}^{m} W_{ij}^I I_i(k)\right)$$

where $f_N(\cdot)$ is a sigmoid function. Then for input vector I^3 assumed, it can be shown that the above equation can be written as

$$\begin{aligned} O^F(k) &= Q_N^F(I_i(k),\ i = 1, 2, \cdots, m) \\ &= Q_N^F(r(k), y(k), u(k)) \end{aligned}$$

where $Q_N^F(\cdot)$ is a nonlinear function, which represents a static mapping neural network.

b) From Fig. 1(b), we obtain

$$\begin{aligned} O^D(k) &= \sum_{j=1}^{n} W_j^O f_N\left[\sum_{i=1}^{m} W_{ij}^I I_i(k) + W_j^D X_j(k-1)\right] \\ &= \sum_{j=1}^{n} W_j^O f_N\left(\sum_{i=1}^{m} W_{ij}^I I_i(k) + W_j^D f_N \right. \\ &\quad \left. \cdot \left(\sum_{i=1}^{m} W_{ij}^I I_i(k-1) + W_j^D X_j(k-2)\right)\right) \\ &\ \ \vdots \\ &= \sum_{j=1}^{n} g_N(I_i(l);\ l \le k,\ i = 1, 2, \cdots, m) \\ &= Q_N^D(I_i(l);\ l \le k,\ i = 1, 2, \cdots, m) \\ &= Q_N^D(r(l), y(l), u(l);\ l \le k) \end{aligned}$$

where $Q_N^D(\cdot)$ is a nonlinear function. Since this includes all previous inputs, it represents a nonlinear dynamic mapping neural network.

c) From Fig. 1(c), we obtain

$$\begin{aligned} O^R(k) &= \sum_{j=1}^{n} W_j^O f_N\left(\sum_{i=1}^{m} W_{ij}^I I_i(k) + \sum_{q=1}^{n} W_{qj}^H X_q(k-1)\right) \\ &= \sum_{j=1}^{n} W_j^O f_N\left(\sum_{i=1}^{m} W_{ij}^I I_i(k) + \sum_{q=1}^{n} W_{qj}^H f_N \right. \\ &\quad \left. \cdot \left(\sum_{i=1}^{m} W_{ij}^I I_i(k-1) + \sum_{q'=1}^{n} W_{q'j}^H X_{q'}(k-2)\right)\right) \\ &\vdots \\ &= \sum_{j=1}^{n} W_j^O \tilde{g}_N(I_i(l);\ l \le k,\ i = 1, 2, \cdots, m) \\ &= Q_N^R(I_i(l);\ l \le k,\ i = 1, 2, \cdots, m) \\ &= Q_N^R(r(l), y(l), u(l);\ l \le k) \end{aligned}$$

where $Q_N^R(\cdot)$ is a nonlinear function, and it also represents a nonlinear dynamic mapping neural network. This completes the proof of Lemma 2. □

Remark 2: Lemma 2 shows that N^F is a static mapping, but both N^D and N^R are dynamic mappings. The difference between N^D and N^R is that N^R has more weights than N^D; thus N^R has more degrees of freedom to represent the output function. In the sense of memory capacity, N^R has more memory space than N^D, and thus both networks are very much different for associative memory purpose. However, N^D is much simpler in structure compared to N^R.

B. Diagonal Recurrent Neural Networks Based Control System

An approach for control and system identification using diagonal recurrent neural networks (DRNN) [23] is presented in this section. An unknown plant is identified by a system identifier, called the diagonal recurrent neuroidentifier (DRNI), which provides information about the plant to a controller, called the diagonal recurrent neurocontroller (DRNC). The neurocontroller is used to drive the unknown dynamic system such that the error between plant and desired output is minimized. A generalized algorithm, called the dynamic backpropagation (DBP), is developed to train both DRNC and DRNI. For simplicity, the plant is assumed to be single input/single output system.

Both DRNI and DRNC use the same DRNN architecture shown in Fig. 1(b), which has only one hidden layer with sigmoid type recurrent neurons. The block diagram of the DRNN based control system is shown in Fig. 2. The inputs to the DRNC are the reference input, the previous plant output, and the previous control signal, and the output of the DRNC is the control signal to the plant. By using the dynamic backpropagation (DBP) algorithm developed in this paper, the weights of the DRNC are adjusted such that the error between the output of the plant and the desired output from a reference model approaches a small value after some training cycles. When the DRNC is in training, the information on the plant is needed. Since the plant is normally unknown, the DRNI is used to estimate the plant sensitivity y_u for the DRNC.

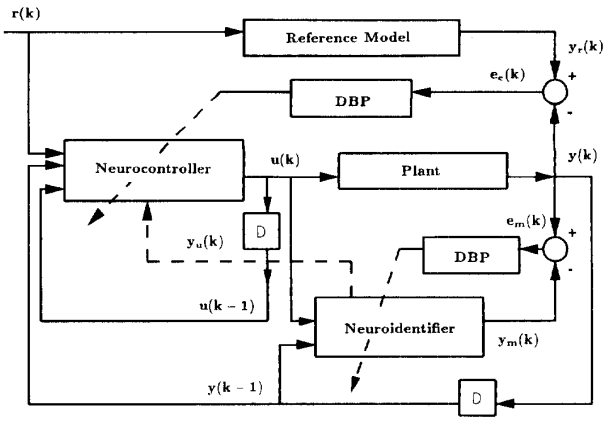


Fig. 2. Block diagram of DRNN based control system.

The current control signal generated from the DRNC and previous output of the plant are used as the inputs to the DRNI. The error between the output of the DRNI and plant is computed for each iteration, and is used to adjust the weights of the DRNI. By training the DRNI and DRNC alternately, the weights of the DRNC can be adjusted more effectively.

C. Dynamic Backpropagation Algorithm for Diagonal Recurrent Neural Networks

The mathematical model for the DRNN in Fig. 1(b) is shown below:

$$O(k) = \sum_j W_j^O X_j(k), \qquad X_j(k) = f(S_j(k)) \tag{7}$$

$$S_j(k) = W_j^D X_j(k-1) + \sum_i W_{ij}^I I_i(k) \tag{8}$$

where for each discrete time k, $I_i(k)$ is the ith input to the DRNN, $S_j(k)$ is the sum of inputs to the jth recurrent neuron, $X_j(k)$ is the output of the jth recurrent neuron, and $O(k)$ is the output of the DRNN. Here $f(\cdot)$ is the usual sigmoid function, and W^I, W^D, and W^O are input, recurrent, and output weight vectors, respectively, in $\mathcal{R}^{n_i}$, $\mathcal{R}^{n_d}$, and $\mathcal{R}^{n_o}$.

Let $y_r(k)$ and $y(k)$ be the desired and actual responses of the plant, then an error function for a training cycle for DRNC can be defined as

$$E_c = \frac{1}{2}(y_r(k) - y(k))^2. \tag{9}$$

In general, the plant response is a nonlinear mapping $G(\cdot)$ of input $u(k)$, i.e., $y(k) = G(u(i),\ i \le k)$. Here, the plant input $u(k)$ is the output of the DRNC, i.e., $u(k) = O(k)$ in (7). On the other hand, in the case of the DRNI, the plant input $u(k)$ is the input to the DRNI.

The error function (9) is also modified for the DRNI by replacing $y_r(k)$ and $y(k)$ with $y(k)$ and $y_m(k)$, respectively, where $y_m(k)$ is the output of the DRNI, i.e.,

$$E_m = \frac{1}{2}(y(k) - y_m(k))^2 \tag{10}$$

where $y_m(k) = O(k)$ of (7).

The gradient of error in (9) with respect to an arbitrary weight vector $W \in \mathcal{R}^n$ is represented by

$$\frac{\partial E_c}{\partial W} = -e_c(k)\frac{\partial y(k)}{\partial W} = -e_c(k)y_u(k)\frac{\partial u(k)}{\partial W} = -e_c(k)y_u(k)\frac{\partial O(k)}{\partial W} \tag{11}$$

where $e_c(k) = y_r(k) - y(k)$ is the error between the desired and output responses of the plant, and the factor $y_u(k) \equiv \partial y(k)/\partial u(k)$ represents the *sensitivity* of the plant with respect to its input.

Since the plant is normally unknown, the sensitivity needs to be estimated for the DRNC. However, in the case of the DRNI, the gradient of error in (10) simply becomes

$$\frac{\partial E_m}{\partial W} = -e_m(k)\frac{\partial y_m(k)}{\partial W} = -e_m(k)\frac{\partial O(k)}{\partial W} \tag{12}$$

where $e_m(k) = y(k) - y_m(k)$ is the error between the plant and the DRNI responses.

The output gradient $\partial O(k)/\partial W$ is common in (11) and (12) and needs to be computed for both DRNC and DRNI. Its computation is summarized in the following lemma:

Lemma 3: Given the DRNN shown in Fig. 1(b) and described by (7) and (8), the output gradients with respect to output, recurrent, and input weights, respectively, are given by

$$\frac{\partial O(k)}{\partial W_j^O} = X_j(k) \tag{13a}$$

$$\frac{\partial O(k)}{\partial W_j^D} = W_j^O P_j(k) \tag{13b}$$

$$\frac{\partial O(k)}{\partial W_{ij}^I} = W_j^O Q_{ij}(k) \tag{13c}$$

where $P_j(k) \equiv \partial X_j(k)/\partial W_j^D$ and $Q_{ij} \equiv \partial X_j(k)/\partial W_{ij}^I$ and satisfy

$$P_j(k) = f'(S_j)(X_j(k-1) + W_j^D P_j(k-1)), \quad P_j(0) = 0 \tag{14a}$$

$$Q_{ij}(k) = f'(S_j)(I_i(k) + W_j^D Q_{ij}(k-1)), \qquad Q_{ij}(0) = 0. \tag{14b}$$

Proof: From (7), the gradient with respect to the output weight is found as

$$\frac{\partial O(k)}{\partial W_j^O} = X_j(k).$$

Again, from (7), the gradient with respect to the recurrent weight is

$$\frac{\partial O(k)}{\partial W_j^D} = W_j^O \frac{\partial X_j(k)}{\partial W_j^D} \equiv W_j^O P_j(k).$$

From (7) and (8),

$$\frac{\partial X_j(k)}{\partial W_j^D} = \frac{\partial X_j(k)}{\partial S_j(k)}\frac{\partial S_j(k)}{\partial W_j^D} = f'(S_j(k))\frac{\partial S_j(k)}{\partial W_j^D}$$

and

$$\frac{\partial S_j(k)}{\partial W_j^D} = X_j(k-1) + W_j^D \frac{\partial X_j(k-1)}{\partial W_j^D}$$

which lead to (14a).

The procedure of deriving the gradient with respect to input weight is similar to the above derivation, and the corresponding equations, (13c) and (14b), follow. □

Remark 3: Equations (14a) and (14b) are nonlinear *dynamic recursive* equations for the state gradients $\partial X_j(k)/\partial W$, and can be solved recursively with given initial conditions. For the usual FNN, the recurrent weight W_j^D is zero and the equations become algebraic.

1) Dynamic backpropagation for DRNI: From (12), the negative gradient of the error with respect to a weight vector in $\mathcal{R}^n$ is

$$-\frac{\partial E_m}{\partial W} = e_m(k)\frac{\partial O(k)}{\partial W} \tag{15}$$

where the output gradient is given by (13) and (14), and W represents W^O, W^D, or W^I in $\mathcal{R}^{n_o}$, $\mathcal{R}^{n_d}$, or $\mathcal{R}^{n_i}$, respectively.

The weights can now be adjusted following a gradient method, i.e., the update rule of the weights becomes

$$W(n+1) = W(n) + \eta\left(-\frac{\partial E_m}{\partial W}\right) \tag{16}$$

where η is a learning rate. The equations (13)–(16) define the dynamic backpropagation algorithm (DBP) for DRNI.

2) Dynamic Backpropagation for DRNC: In the case of DRNC, from (11), the negative gradient of the error with respect to a weight vector in $\mathcal{R}^n$ is

$$-\frac{\partial E_c}{\partial W} = e_c(k)y_u(k)\frac{\partial O(k)}{\partial W}. \tag{17}$$

Since the plant is normally unknown, the sensitivity term $y_u(k)$ is unknown. This unknown value can be estimated by using the DRNI. When the DRNI is trained, the dynamic behavior of the DRNI is close to the unknown plant, i.e., $y(k) \approx y_m(k)$, where $y_m(k)$ is the output of the DRNI.

Once the training process is done, we assume the sensitivity can be approximated as

$$y_u(k) \equiv \frac{\partial y(k)}{\partial u(k)} \approx \frac{\partial y_m(k)}{\partial u(k)} \tag{18}$$

where $u(k)$ is an input to the DRNI.

Applying the chain rule to (18), and noting that $y_m(k) = O(k)$ of (7),

$$\frac{\partial y_m(k)}{\partial u(k)} = \frac{\partial O(k)}{\partial u(k)} = \sum_j \frac{\partial O(k)}{\partial X_j}\frac{\partial X_j(k)}{\partial u(k)} = \sum_j W_j^O \frac{\partial X_j(k)}{\partial u(k)}. \tag{19}$$

Also from (7),

$$\frac{\partial X_j(k)}{\partial u(k)} = f'(S_j(k))\frac{\partial S_j(k)}{\partial u(k)}. \tag{20}$$

Since inputs to the DRNI are $u(k)$ and $y(k-1)$ from Fig. 2, (8) becomes

$$S_j(k) = W_j^D X_j(k-1) + W_{1j}^I u(k) + W_{2j}^I y(k-1) + W_{3j}^I b_I \tag{21}$$

where b_I is the bias for DRNI.

Thus

$$\frac{\partial S_j(k)}{\partial u(k)} = W_{1j}^I. \tag{22}$$

From (19), (20), and (22),

$$y_u(k) \approx \frac{\partial y_m(k)}{\partial u(k)} = \sum_j W_j^O f'(S_j(k)) W_{1j}^I \tag{23}$$

where the variables and weights are those found in DRNI.

Using the negative gradients in (17), the weights for DRNC can now be adjusted using the update rule similar to (16). The equations (13), (14), (16), (17), and (23) define the dynamic backpropagation algorithm for DRNC.

III. Convergence and Stability

The update rule of (16) calls for a proper choice of the learning rate η. For a small value of η the convergence is guaranteed but the speed is very slow; on the other hand if η is too big, the algorithm becomes unstable. This section develops a guideline in selecting the learning rate properly, which leads to adaptive learning rate.

A discrete-type Lyapunov function can be given by

$$V(k) = \frac{1}{2} e^2(k) \tag{24}$$

where $e(k)$ represents the error in the learning process.

Thus, the change of the Lyapunov function due to the training process is obtained by

$$\Delta V(k) = V(k+1) - V(k) = \frac{1}{2}[e^2(k+1) - e^2(k)]. \tag{25}$$

The error difference due to the learning can be represented by [24]

$$e(k+1) = e(k) + \Delta e(k) = e(k) + \left[\frac{\partial e(k)}{\partial W}\right]^T \Delta W \tag{26}$$

where ΔW represents a change in an arbitrary weight vector in $\mathcal{R}^n$.

A. Convergence of DRNI

From the update rule of (12) and (16),

$$\Delta W_I = -\eta_I e_m(k) \frac{\partial e_m(k)}{\partial W_I} = \eta_I e_m(k) \frac{\partial O(k)}{\partial W_i} \tag{27}$$

where W_I and η_I, respectively, represent an arbitrary weight and the corresponding learning rate in DRNI, and $O(k)$ is the output of DRNI. Then we have the following general convergence theorem:

Theorem 1: Let η_I be the learning rate for the weights of DRNI and $g_{I,\max}$ be defined as $g_{I,\max} := \max_k \|g_I(k)\|$, where $g_I(k) = \partial O(k)/\partial W_I$, and $\|\cdot\|$ is the usual Euclidean norm in $\mathcal{R}^n$. Then the convergence is guaranteed if η_I is chosen as

$$0 < \eta_I < \frac{2}{g_{I,\max}^2}. \tag{28}$$

Proof: From (25)–(27), $\Delta V(k)$ can be represented as

$$\begin{aligned}\Delta V(k) &= \Delta e_m(k)\left[e_m(k) + \frac{1}{2}\Delta e_m(k)\right] \\ &= \left[\frac{\partial e_m(k)}{\partial W_I}\right]^T \eta_I e_m(k) \frac{\partial O(k)}{\partial W_I} \\ &\quad \cdot \left\{ e_m(k) + \frac{1}{2}\left[\frac{\partial e_m(k)}{\partial W_I}\right]^T \eta_I e_m(k) \frac{\partial O(k)}{\partial W_I} \right\}.\end{aligned} \tag{29}$$

Since for DRNI $\partial e_m(k)/\partial W_I = -\partial O(k)/\partial W_I$, we obtain

$$\begin{aligned}\Delta V(k) &= -\eta_I e_m^2(k) \left\| \frac{\partial O(k)}{\partial W_I} \right\|^2 + \frac{1}{2}\eta_I^2 e_m^2(k) \left\| \frac{\partial O(k)}{\partial W_I} \right\|^4 \\ &\equiv -\lambda_I e_m^2(k).\end{aligned} \tag{30}$$

Let $g_I(k) = \partial O(k)/\partial W_I$, $g_{I,\max} := \max_k \|g_I(k)\|$, and $\eta_1 = \eta_I g_{I,\max}^2$ [25]. Then

$$\begin{aligned}\lambda_I &= \frac{1}{2}\|g_I(k)\|^2 \eta_I (2 - \eta_I \|g_I(k)\|^2) \\ &= \frac{1}{2}\|g_I(k)\|^2 \eta_I \left(2 - \frac{\eta_1 \|g_I(k)\|^2}{g_{I,\max}^2}\right) \\ &\geq \frac{1}{2}\|g_I(k)\|^2 \eta_I (2 - \eta_1) > 0.\end{aligned} \tag{31}$$

From (31), we obtain $0 < \eta_1 < 2$, and (28) follows. □

Remark 4: The convergence is guaranteed as long as (31) is satisfied, i.e.,

$$\eta_I(2 - \eta_1) > 0$$

or

$$\eta_1(2 - \eta_1)/g_{I,\max}^2 > 0.$$

This implies that any η_1, $0 < \eta_1 < 2$, guarantees the convergence. However, the maximum learning rate which guarantees the most rapid or optimal convergence is corresponding to $\eta_1 = 1$, i.e.,

$$\eta_I^* = \frac{1}{g_{I,\max}^2}$$

which is the half of the upper limit in (28).

This shows an interesting result that any other learning rate larger than η_I^* does not guarantee the faster convergence.

The general convergence theorem can now be applied to find the specific convergence criterion for each type of weights:

Theorem 2: Let η_I^O, η_I^D, and η_I^I be the learning rates for the DRNI weights W_I^O, W_I^D, and W_I^I, respectively. Then the dynamic backpropagation algorithm converges if $0 < |W_{I,j}^D| < 1$, $j = 1, 2, \cdots, h_I$, and the learning rates are chosen as

$$0 < \eta_I^O < \frac{2}{h_I} \tag{32a}$$

$$0 < \eta_I^D < \frac{2}{h_I}\left[\frac{1}{W_{I,\max}^O}\right]^2 \tag{32b}$$

$$0 < \eta_I^I < \frac{2}{(n_I + h_I)}\left[\frac{1}{W_{I,\max}^O I_{I,\max}}\right]^2 \tag{32c}$$

where h_I is the number of recurrent neurons in the hidden layer, n_I is the number of inputs to the DRNI, $W_{I,\max}^O := \max_k \|W_I^O(k)\|$, $I_{I,\max} := \max_k \|I_I(k)\|$, and $\|\cdot\|$ is the sup-norm.

Proof: a) From (13a),

$$g_I(k) = \frac{\partial O(k)}{\partial W_I^O} = X^I(k)$$

where $X^I = [X_1^I, X_2^I, \cdots, X_{h_I}^I]^T$, and X_j^I is the output value of the jth neuron in the hidden layer, and h_I is the number of recurrent neurons in the DRNI hidden layer.

Since $0 < X_j^I < 1$, $j = 1, 2, \cdots, h_I$, by the definition of the usual Euclidean norm in $\mathcal{R}^{h_I}$, $\|g_I(k)\| < \sqrt{h_I}$ and $g_{I,\max}^2 = h_I$. Then from Theorem 1, (32a) follows.

b) and c) See the Appendix. □

B. Convergence of DRNC

From the update rule of (16) and (17),

$$\begin{aligned}\Delta W_c &= -\eta_c e_c(k)\frac{\partial e_c(k)}{\partial W_c} = \eta_c e_c(k) y_u(k)\frac{\partial u(k)}{\partial W_c} \\ &= \eta_c e_c(k) y_u(k)\frac{\partial O(k)}{\partial W_c}\end{aligned} \tag{33}$$

where $\partial y(k)/\partial u(k) = y_u(k)$ is the plant sensitivity, W_c and η_c, respectively, represent an arbitrary weight and the corresponding learning rate in DRNC, and $O(k)$ is the output of DRNC. Then we have the following general convergence theorem:

Theorem 3: Let η_c be the learning rate for the weights of DRNC and $g_{c,\max}$ be defined as $g_{c,\max} := \max_k \|g_c(k)\|$, where $g_c(k) = \partial O(k)/\partial W_c$, and $S_{\max} = h_I W_{I,\max}^O W_{I,\max}^I/2$. Then the convergence is guaranteed if η_c is chosen as

$$0 < \eta_c < \frac{2}{S_{\max}^2 g_{c,\max}^2}.$$

Proof: From (25), (26), (27) and (33), $\Delta V(k)$ can be represented as

$$\begin{aligned}\Delta V(k) &= \Delta e_c(k)\left[e_c(k) + \frac{1}{2}\Delta e_c(k)\right] \\ &= \left[\frac{\partial e_c(k)}{\partial W_c}\right]^T \eta_c e_c(k) y_u(k)\frac{\partial O(k)}{\partial W_c} \\ &\quad \cdot \left\{e_c(k) + \frac{1}{2}\left[\frac{\partial e_c(k)}{\partial W_c}\right]^T \eta_c e_c(k) \right. \\ &\quad \left. \cdot\, y_u(k)\frac{\partial O(k)}{\partial W_c}\right\}.\end{aligned}$$

Since in this case $\partial e_c(k)/\partial W_c = -y_u(k)\,\partial O(k)/\partial W_c$, we obtain

$$\begin{aligned}\Delta V(k) &= -\eta_c e_c^2(k) y_u^2(k)\left\|\frac{\partial O(k)}{\partial W_c}\right\|^2 \\ &\quad + \frac{1}{2}\eta_c^2 e_c^2(k) y_u^4(k)\left\|\frac{\partial O(k)}{\partial W_c}\right\|^4 \\ &\equiv -\lambda_c e_c^2(k).\end{aligned} \tag{34}$$

Comparing (34) with (30), it can be seen that both conditions are similar, except the sensitivity $y_u(k)$ needs to be incorporated in the DRNC. Therefore, it remains to find the limit on $y_u(k)$ or $y_u^2(k)$.

Since, from (23),

$$y_u(k) = \sum_j W_{I,j}^O f'(S_j) W_{I,1j}^I$$

where $0.0 < f'(S_j) < 0.5$. If $W_{I,1\max}^I$ is defined as $W_{I,1\max}^I := \max_k \|W_{I,1}^I(k)\|$, and $\|W_{I,1}^I(k)\| := \max_j |W_{I,1j}^I(k)|$, then

$$\begin{aligned}|y_u(k)| &\le h_I \|W_I^O(k)\|\,\|f'(k) W_{I,1}^I(k)\| \\ &\le \frac{h_I}{2} W_{I,\max}^O W_{I,1\max}^I \equiv S_{\max}\end{aligned}$$

where $S_{\max}$ is the limit on sensitivity. Thus following the proof of Theorem 1, we obtain

$$0 < \eta_c < \frac{2}{S_{\max}^2 g_{c,\max}^2} \tag{35}$$

where $g_{c,\max} := \max_k \|g_c(k)\|$, and $g_c(k) = \partial O(k)/\partial W_c$. □

Remark 5: As in the case of DRNI, the optimal convergence rate is

$$\eta_c^* = \frac{1}{S_{\max}^2 g_{c,\max}^2}$$

which is the half of the upper limit in (35). Again, any other learning rate larger than η_c^* does not guarantee the faster convergence.

Thus, in a way similar to DRNI, the specific convergence criteria can be found as following:

Theorem 4: Let η_c^O, η_c^D, and η_c^I be the learning rates for the DRNC weights W_c^O, W_c^D, and W_c^I, respectively. Then the dynamic backpropagation algorithm converges if $0 < |W_{c,j}^D| < 1, j = 1, 2, \cdots, h_c$, and the learning rates are chosen as

$$0 < \eta_c^O < \frac{2}{h_c S_{\max}^2} \tag{36a}$$

$$0 < \eta_c^D < \frac{2}{h_c S_{\max}^2}\left[\frac{1}{W_{c,\max}^O}\right]^2 \tag{36b}$$

$$0 < \eta_c^I < \frac{2}{(n_c + h_c) S_{\max}^2}\left[\frac{1}{W_{c,\max}^O I_{c,\max}}\right]^2 \tag{36c}$$

where h_c is the number of recurrent neurons in the hidden layer, n_c is the number of inputs to the DRNC, $W_{c,\max}^O := \max_k \|W_c^O(k)\|$, $I_{c,\max} := \max_k \|I_c(k)\|$, and $I_c(k) = \{b_c, u(k-1), y(k-1)\}$.

Proof: From Theorem 3, if we redefine $\hat{\eta}_c = \eta_c S_{\max}^2$, then we obtain

$$0 < \hat{\eta}_c < \frac{2}{g_{c,\max}^2} \tag{37}$$

which is the same form as (28) in Theorem 1 for DRNI.

Note that both DRNI and DRNC have the same DRNN architecture, and $g_{I,\max}$ and $g_{c,\max}$ are defined in the same way in terms of $\partial O(k)/\partial W$. Therefore Theorem 2 is valid for $\hat{\eta}_c$ for all three types of weights, i.e.,

$$0 < \hat{\eta}_c^O < \frac{2}{h_c}$$

$$0 < \hat{\eta}_c^D < \frac{2}{h_c}\left[\frac{1}{W_{c,\max}^O}\right]^2$$

$$0 < \hat{\eta}_c^I < \frac{2}{(h_c + n_c)}\left[\frac{1}{W_{c,\max}^O I_{c,\max}}\right]^2$$

where $\hat{\eta}_c^O = \eta_c^O S_{\max}^2$, $\hat{\eta}_c^D = \eta_c^D S_{\max}^2$, and $\hat{\eta}_c^I = \eta_c^I S_{\max}^2$. Thus (36) follows. □

IV. Simulation Results

The DRNC and DRNI are tested for five different examples. The numbers of inputs to DRNC and DRNI are denoted by n_c and n_I, respectively, and h_c and h_I denote the numbers of neurons in the hidden layer for DRNC and DRNI, respectively, and are chosen as $h_c = 2n_c + 1$, $h_I = 2n_I + 1$.

In the simulation study, P_c and P_I are the inputs to the DRNC and DRNI, respectively; N_T and W_T are the total number of neurons and weights in the system (both DRNC and DRNI), respectively; η_c and η_I are the initial learning rate for the DRNC and DRNI, respectively; and b_c, and b_I are the biases for the DRNC and DRNI, respectively. It can be verified that $N_T = h_c + h_I + 2$, and $W_T = (n_c + 3)h_c + (n_I + 3)h_I$.

Example 1: A BIBO Nonlinear Plant [26]: In this case the plant is described by the difference equation

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

and a reference model is described by the difference equation

$$y_r(k+1) = 0.6y_r(k) + r(k)$$

where $r(k) = \sin(2\pi k/25) + \sin(2\pi k/10)$. The objective is to determine an input $u(k)$ to the plant such that $\lim_{k\to\infty} |y(k) - y_r(k)| < \epsilon$, where ϵ is a suitably chosen constant. In this simulation study, $P_c = \{r(k), u(k-1), y(k-1)\}$ and $P_I = \{u(k), y(k-1)\}$, thus $n_c = 3$, $n_I = 2$, $N_T = 14$, and $W_T = 67$. In [26], however, a two-hidden layer feedforward neural network with $N_T = 31$ and $W_T = 281$ is used, i.e., more than two times for the number of neurons and four times for the number of weights compared to the DRNC and DRNI combined.

Adaptive learning rates were used starting from the initial rates of $\eta_c = 0.1$ and $\eta_I = 0.1$. Both learning rates are adjusted according to the criteria developed in Section III, and the adaptive learning rates of the DRNI and DRNC are shown in Fig. 3(a) and (b), respectively. As can be seen in Fig. 3(a) and (b), the learning rates adapt to reduce the tracking error. The learning rates for DRNI, η_I^O, η_I^D, and η_I^I, after 60 training epochs are adjusted to 0.04, 0.0012, and 0.00012, respectively, whereas the learning rates for DRNC, η_c^O, η_c^D, and η_c^I, after 60 training epochs to 0.0019, 0.0035, and 0.000354, respectively. The average errors for the cases of adaptive learning rates and the different values of fixed learning rates are shown in Fig. 3(c). As can be seen in the figure, the adaptive learning rate scheme is stable and converges much faster. The result after 60 training epochs (one epoch is equal to 50 time steps in this example) is shown in Fig. 3(d). In [26], however, the total of 100 000 training time steps were required, which is 30 times more than this result.

Example 2: The On-Line Adapting Ability of DRNN Based Control System: In this simulation study, all conditions are same as in Example 1, except the reference input, $r(k)$, is changed in order to observe the adapting ability of the DRNN based control systems.

After the system in Example 1 is trained 60 epochs, the reference input, $r(k)$, is changed to become $r(k) = \sin(2\pi k/25)$, and the result is shown in Fig. 4(a). This figure shows that the DRNN based control system can catch the new reference model quickly. When the new reference input $r(k)$ is changed to become a square wave with amplitude equal to 2, the simulation result is shown in Fig. 4(b). After about 6 epochs, the DRNN based control system catches the new reference model. These results show the on-line adapting ability of the proposed DRNN based model.

Example 3: The Recovering Ability from Disturbances: The same plant and reference models of Example 1 are used in this example. Both initial learning rates, η_c, and η_I, are chosen to be 0.1, and both biases, b_c and b_I, are equal to 1.0. After the DRNN's are trained already (i.e., after 60 epochs), a disturbance $v(k)$ is applied to the plant output during the 61th and 62th epochs.

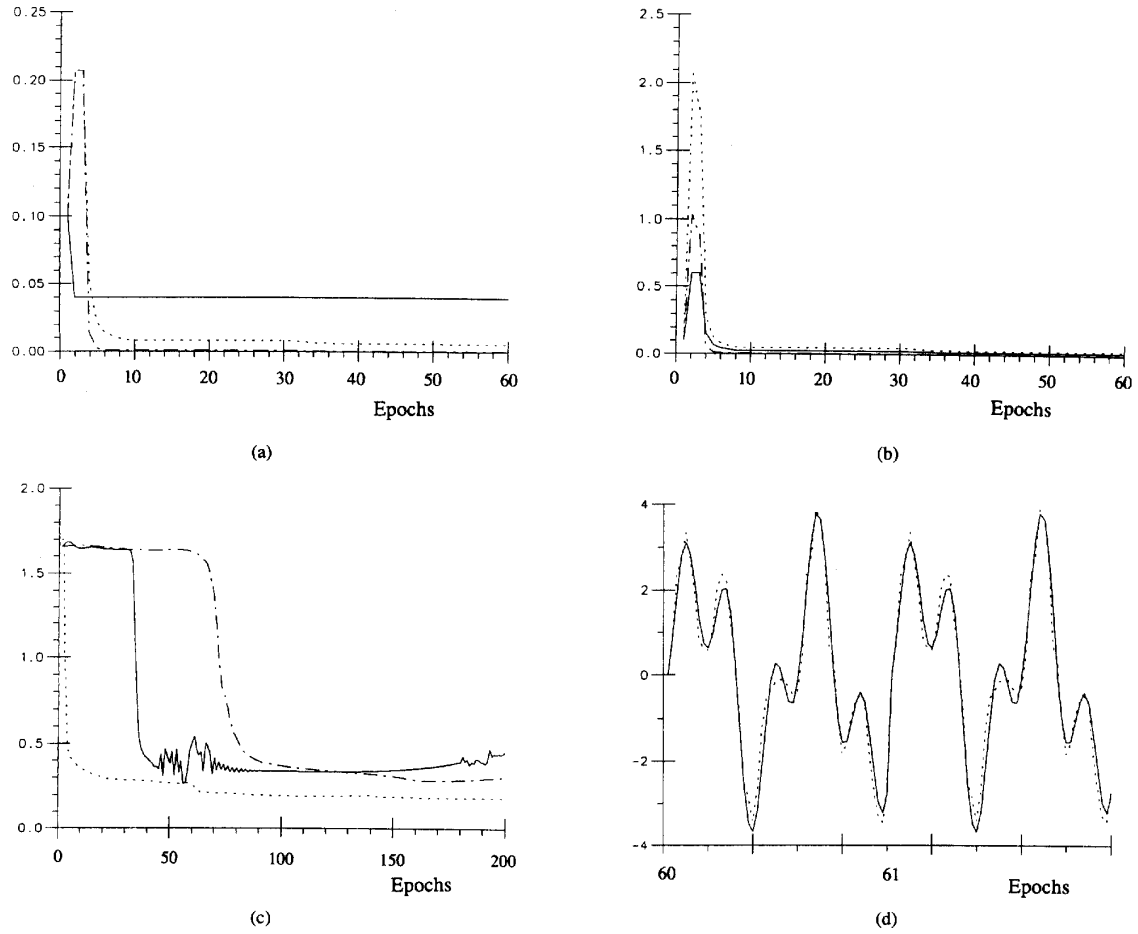


Fig. 3. Example 1: BIBO nonlinear plant control. (a) Adaptive learning rates of DRNI; η_I^O (solid line), η_I^D (dotted line), and η_I^I (dashed line). (b) Adaptive learning rates of DRNC; η_c^O (solid line), η_c^D (dotted line), and η_c^I (dashed line). (c) Average errors for the cases of the adaptive learning rates and the different values of fixed learning rates; adaptive learning rate (dotted line), fixed learning rate with $\eta_I = \eta_c = 0.1$ (solid line), and fixed learning rate with $\eta_I = 0.01$ and $\eta_c = 0.1$ (dashed line). (d) Outputs of reference model (solid line) and the plant (dotted line) after 60 training epochs.

In case 1, $v(k) = 1.0$ is applied to the plant output at the 61th and 62th epochs. In this case, the DRNN-based control system can recover from the disturbance quickly after about 4 epochs, as shown in Fig. 5(a). In case 2, $v(k) = 5.0$ is applied to the plant output at the 61th and 62th epochs. The DRNN based control system can recover from the disturbance after about 10 epochs, as shown in Fig. 5(b). When $v(k) = 10.0$ is applied, the DRNN based control system can still recover from the disturbance; however, it takes a longer time.

Example 4: A Non-BIBO Nonlinear Plant: The model reference control problem for a nonlinear plant with linear input is considered below.

Reference model:

$$y_r(k+1) = 0.6y_r(k) + r(k)$$

where $r(k) = \beta \sin(2\pi k/100)$.

Plant model:

$$y(k+1) = 0.2y^2(k) + 0.2y(k-1) + 0.4\sin[0.5(y(k)+y(k-1))] \cdot \cos[0.5(y(k)+y(k-1))] + 1.2u(k).$$

The plant is unstable in the sense that given a sequence of uniformly bounded controls $\{u(k)\}$, the plant output may diverge. The plant output diverges when the step input $u(k) = 0.83$, $\forall k \geq 0$, is applied to the plant [27]. Thus in order to guarantee the stability of the control system, the control signal, $u(k)$, if a uniformly step input is to be applied, must be less than 0.83. When a step input $u(k) < 0.83$ is applied to the plant, the maximum possible plant output becomes $y_{\max} \approx 2.26$. On the other hand, it can be seen that the maximum reference output that the reference model can generate is $y_{r,\max} \approx \frac{5}{2}\beta$, and this should not exceed the maximum allowed plant output $y_{\max}$. This implies that the reference signal needs to be restricted by $\beta \leq 0.9$.

In this simulation study, $P_c = \{r(k), u(k-1), y(k-1)\}$ and $P_I = \{u(k), y(k-1)\}$; thus $n_c = 3$, $n_I = 2$, $N_T = 14$, and $W_T = 67$. In [27], however, a two-hidden layer feedforward neural network with $N_T = 21$ and $W_T = 140$ is used. First, $\beta = 0.2$ is used. Both initial learning rates, η_c and η_I, are chosen to be 2.0, and both biases, b_c and b_I, are chosen to be 1.0. The tracking process during the 70th

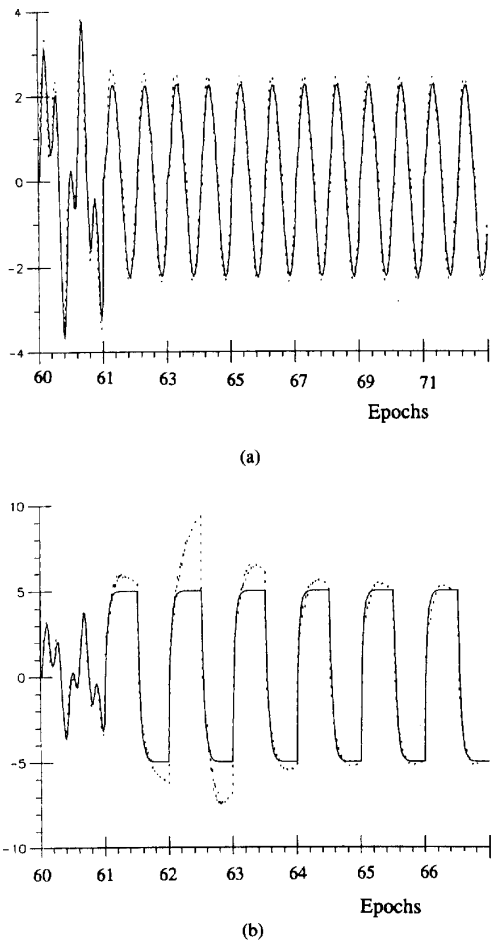


Fig. 4. Example 2: On-line adapting ability of the DRNN based control system. (a) Outputs of reference model (solid line) and the plant (dotted line). (b) Outputs of reference model (solid line) and the plant (dotted line).

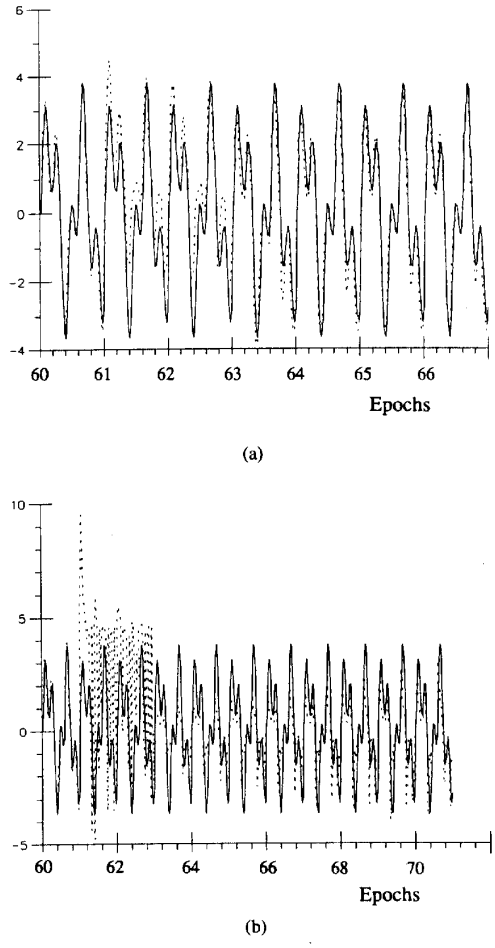


Fig. 5. Example 3: Recovering ability from disturbances. (a) Outputs of reference model (solid line) and the plant (dotted line) when the disturbance is 1.0. (b) Outputs of reference model (solid line) and the plant (dotted line) when the disturbance is 5.0.

to the 75th training epochs is shown in Fig. 6(a), and the simulation result during the 96th to the 100th epochs is shown in Fig. 6(b), which shows that the DRNC controls this plant very well. When $\beta = 1.0$ is used, since the control signal is not a uniform step input, this control system is still stable. However, since $y_{\max} < y_{r,\max}$, a sustained error exists.

One interesting observation is the control signal, $u(k)$. For $\beta = 0.2$, the control signal shown in Fig. 6(c), is much less than 0.83. This implies that the controller adapts itself in such a way that the plant will not be out of bound. However, as the value of β increases, the control signal $u(k)$ also increases. When $\beta > 1.0$ is used, a larger $y_{r,\max}$ is obtained. In order to track this increased reference output, the control signal $u(k)$ needs to be increased. However, when the control signal is getting larger, exceeding 0.83 for a sustained period, the plant becomes unstable. Another observation is on the plant sensitivity. When the output goes through a large transient, the sensitivity $y_u(k)$ increases, as shown in Fig. 6(d). Therefore, the weight changes become larger and the controller can capture the dynamic behavior of the plant quickly.

Example 5: The Interpolation Ability of the DRNN Based Control System: Flight control is a very complex task; thus a sophisticated control skill is needed. In this example, the interpolation ability of the DRNN based control system is demonstrated via a flight control application. During the training process, only few trim points are trained. After few training epochs, an untrained trim point is applied and tested in the DRNN based control system.

For initial study, a simplified second-order linear plant model is assumed for a typical flight dynamic subsystem. An objective of the flight controller is to maneuver the plant so that it will follow a reference response generated by a reference model. The assumed reference and plant models are given below:

Reference model:

$$H(s) = \frac{4.0}{s^2 + 2.82s + 4.0}.$$

Plant model:

$$H(s) = \frac{1.0}{s^2 + 2.0s + 1.0}.$$

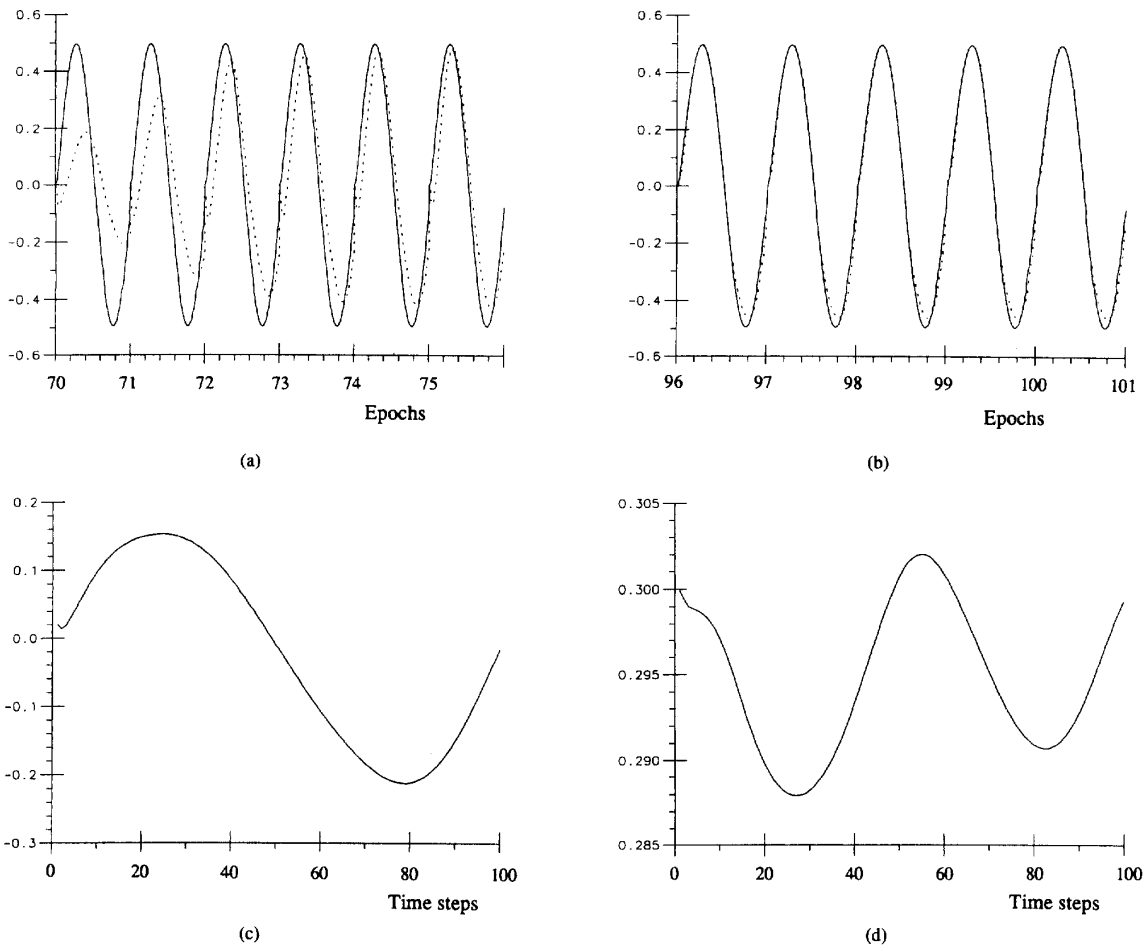


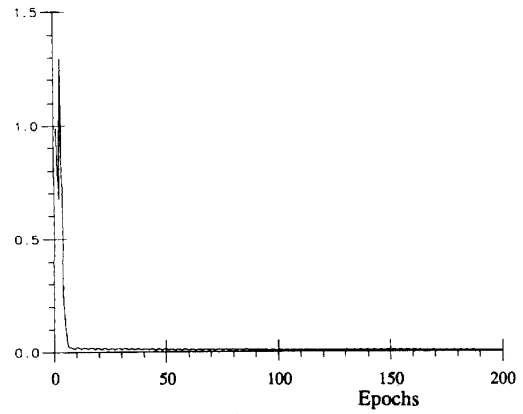
Fig. 6. Example 4: Non-BIBO nonlinear plant control. (a) Outputs of reference model (solid line) and the plant (dotted line) when $\beta = 0.2$. (b) Outputs of reference model (solid line) and the plant (dotted line), from 96 to 100 epochs. (c) Control signal generated from DRNC ($\beta = 0.2$)−. (d) Sensitivity generated from DRNI ($\beta = 0.2$).

For this second-order system [28], $x_1(t)$ and $x_2(t)$ are chosen as the state variables, $r(t)$ is the input to the reference model, and $u(t)$ and $y(t)$ are input and output of the plant, respectively. Here, the speed can be measured, i.e., $x_2(t)$ is available, $x_1(t) = y(t)$, and $\dot{x}_1(t) = x_2(t)$. The step input is applied to the reference model. Here, $P_c = \{r(t), y(t-1), x_2(t-1)\}$, and $P_I = \{u(t), y(t-1), x_2(t-1)\}$; thus $n_c = 3$ and $n_I = 3$. Also, $N_T = 16$ and $W_T = 84$. Here, three sets of initial conditions for $(x_1(0), x_2(0))$, (0.0, 0.0), (0.1, 0.3), and (0.5, 0.75), are selected as *training sets*. After the neural networks were trained, a *testing set* is applied to the system. In this simulation, $(x_1(0), x_2(0)) = (0.8, 1.0)$ is the testing set. Both initial learning rates, η_c and η_I, are equal to 2.0, and both biases, b_c and b_I, are equal to 1.0.

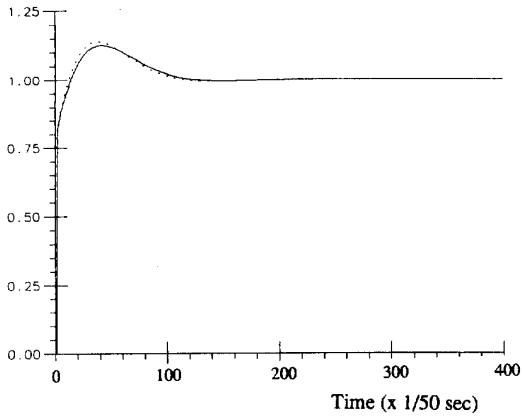
After 10 training epochs or cycles, the error converges to a small value. The error is shown in Fig. 7(a), and the outputs of the reference model (y_r) and plant (y) after 200 training epochs are shown in Fig. 7(b). It is observed that there is an oscillation in Fig. 7(a). This is because three different training sets are used sequentially. The neural networks must be adjusted such that the error will be minimum for all training sets. When a testing set, which is different but close to a training set is used, the result is very good as can be seen in Fig. 7(b). This implies that the neural network has the interpolation ability if an untrained set is close to a trained set.

V. Conclusion

This paper described the diagonal recurrent neural network (DRNN) based control architecture, which includes the diagonal recurrent neurocontroller (DRNC) and neuroidentifier (DRNI). The proposed model is compared with the feedforward neural network (FNN) and the fully connected recurrent neural network (FRNN) in terms of number of required weights and mapping characteristics. The DRNN model is shown to have dynamic mapping characteristic. Moreover, it requires fewer weights when compared with the FRNN model. The above features allow the DRNN model to be used for on-line applications.



(a)



(b)

Fig. 7. Example 5: Interpolation ability of the DRNN based control system. (a) Average error. (b) Outputs of reference model (y_r: solid line) and plant (y: dotted line).

In a neural network based control system, the plant is unknown in general; thus the sensitivity, which is required during training process, is normally not available. Many people simply ignore this sensitivity and use the direct control approach, whereas others use the simple method of sign changes in the plant response as the sensitivity. However, in this paper, a neuroidentifier is utilized to obtain the sensitivity, and along with the neurocontroller the weight adjustment becomes smoother than the case without the sensitivity information.

The convergence of a recurrent neural network is not easy to be guaranteed. Needless to say, when an unknown plant is combined with the neural network, it makes the convergence of neural network based system even more difficult. A large learning rate may make the system unstable while a small learning rate makes the training process too slow. An approach to find the bounds on learning rates based on the Lyapunov function is developed. The use of adaptive learning rates guarantees convergence, and the optimal learning rates are found.

The simulations of the proposed model are conducted for both BIBO and non-BIBO nonlinear plant control problems. The DRNN based control system is tested for its on-line adapting ability, recovering ability from disturbances, and the interpolation ability. The results show that the DRNN based control system is very promising for future real-time applications.

APPENDIX
PROOF OF THEOREM 3(B) AND (C)

Proof of Part (b): We denote $\partial X_j(k)/\partial W^D_{I,j} = P_{I,j}(k)$, then from (14a)

$$P_{I,j}(k) = f'(k)(X_j(k-1) + W^D_{I,j}P_{I,j}(k-1))$$

where $f'(k) \equiv f'(S_j(k))$. Thus the solution of the above equation can be written as

$$P_{I,j}(l) = \sum_{m=0}^{l-1}\prod_{n=0}^{l-m-1} f'(l-n)X_j(m)(W^D_{I,j})^{l-1-m} + \prod_{n=1}^{l} f'(n)P_{I,j}(0)(W^D_{I,j})^l.$$

Since $P_{I,j}(0) = 0$, thus we obtain

$$P_{I,j}(l) = \sum_{m=0}^{l-1}\prod_{n=0}^{l-m-1} f'(l-n)X_j(m)(W^D_{I,j})^{l-1-m}, \quad j = 1, 2, \cdots, h_I.$$

Since $0 < f'(S_j) < 0.5$, $0 < X_j(m) < 1$, and $0 < W^D_j < 1$ for $j = 1, 2, \cdots, h_I$, thus

$$|P_{I,j}(l)| \le \sum_{m=0}^{l-1}\prod_{n=0}^{l-m-1} |f'(l-n)|\,|X_j(m)|\,|W^D_{I,j}|^{l-m-1}$$

and

$$|P_{I,j}(l)| \le \sum_{m=0}^{l-1}(0.5)^{l-m}.$$

Let $r = l - m - 1$. Then

$$|P_{I,j}(l)| \le \sum_{r=0}^{l-1}(0.5)^{r+1} \le 0.5\sum_{r=0}^{\infty}(0.5)^r = 1, \quad j = 1, 2, \cdots, h_I.$$

Thus for the vector $P_I(l) = [P_{I,1}, P_{I,2}, \cdots, P_{I,h_I}]^T$,

$$\|P_I(l)\| \le \sqrt{h_I}$$

or

$$\left\|\frac{\partial X(k)}{\partial W^D_I}\right\| \le \sqrt{h_I}. \tag{A.1}$$

From (13b), we obtain

$$\left|\frac{\partial O(k)}{\partial W^D_{I,j}}\right| \le |W^O_{I,j}|\,|P_{I,j}| \le |W^O_{I,j}| \le \|W^O_I(k)\|$$

where $\|W^O_I(k)\| := \max_j |W^O_{I,j}(k)|$. Thus for the output gradient vector in $\mathcal{R}^{h_I}$,

$$\left\|\frac{\partial O(k)}{\partial W^D_I}\right\| \le \sqrt{h_I}\|W^O_I(k)\|. \tag{A.2}$$

Again we define $W^O_{I,\max} := \max_k \|W^O_I(k)\|$. Then we obtain

$$\left\| \frac{\partial O(k)}{\partial W^D_I} \right\| \le \sqrt{h_I} W^O_{I,\max}. \tag{A.3}$$

Hence, from Theorem 2 and (A.3), (32b) follows. □

Proof of Part (c): We denote $\partial X_j(k)/\partial W^I_{I,ij} = Q_{I,ij}(k)$. Then from (14b),

$$Q_{I,ij}(k) = f'(k)(I_{I,i}(k) + W^D_{I,j} Q_{I,ij}(k-1)).$$

Again,

$$Q_{I,ij}(l) = \sum_{m=0}^{l-1} \prod_{n=0}^{l-m-1} f'(l-n) I_{I,i}(m) (W^D_{I,j})^{l-1-m} + \prod_{n=0}^{l} f'(n) Q_{I,ij}(0) (W^D_{I,j})^l.$$

Since $Q_{I,ij}(0) = 0$, thus

$$Q_{I,ij}(l) = \sum_{m=0}^{l-1} \prod_{n=0}^{l-m-1} f'(l-n) I_{I,i}(m) (W^D_{I,j})^{l-1-m}$$

and in a way similar to the proof of part (b),

$$|Q_{I,ij}(l)| \le \sum_{m=0}^{l-1} (0.5)^{l-m} I_{I,i}(m).$$

Assume the plant is a BIBO stable system, and $I_{I,\max} = \max[b_I, u_{\max}, y_{\max}]$, where b_I is the bias, and $u_{\max}$ and $y_{\max}$ are maximum input and output, respectively. Thus we obtain

$$|Q_{I,ij}(l)| \le \sum_{m=0}^{l-1} (0.5)^{l-m} I_{I,\max} \le I_{I,\max}$$

and

$$\left\| \frac{\partial X(k)}{\partial W^I_I} \right\| \le \sqrt{n_I + h_I} I_{I,\max}. \tag{A.4}$$

Therefore from (13c), we obtain

$$\left\| \frac{\partial O(k)}{\partial W^I_I} \right\| \le \sqrt{n_I + h_I} I_{I,\max} \|W^O_I(k)\| \le \sqrt{n_I + h_I} W^O_{I,\max} I_{I,\max}. \tag{A.5}$$

Hence, from Theorem 2 and (A.5), (32c) follows. □

REFERENCES

[1] P. Antsaklis (Eds.), Special Issue on Neural Network in Control Systems, *IEEE Contr. Syst. Mag.*, vol. 10, no. 3, pp. 3–87, Apr. 1990.

[2] A. Guez and J. Selinsky, "A trainable neuromorphic controller," *J. Robotic Syst.*, vol. 5, pp. 364–388, Aug. 1988.

[3] A. Guez and J. Selinsky, "Neurocontroller design via supervised and unsupervised learning," *J. Intelligent and Robotic Syst.*, vol. 2, pp. 307–335, 1989.

[4] B. Widrow and F. W. Smith, "Pattern recognizing control systems," in *Computer and Information Sciences Symp. Proc.*, Washington, DC, 1964, pp. 288–317.

[5] D. Psaltis, A. Sideris, and A. A. Yamamura, "A multilayered neural network controller," *IEEE Contr. Syst. Mag.*, vol. 8, pp. 17–21, Apr. 1988.

[6] M. S. Lan, "Adaptive control of unknown dynamic systems via neural network approach," in *Proc. 1989 American Control Conf.*, Pittsburgh, June 1989, pp. 910–915.

[7] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamic systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, No. 1, pp. 4–27, Mar. 1990.

[8] R. Hoptroff, T. Hall, and R. Burge, "Experiments with a neural controller," in *Proc. IEEE Int. Joint Conf. Neural Networks*, Washington, DC, vol. II, 1990, pp. 735–740.

[9] H. Bleuler, D. Diez, G. Lauber, U. Meyer, and D. Zlatnik, "Nonlinear neural network control with application example," in *Proc. Int. Conf. Neural Networks*, Paris, 1990, pp. 201–204.

[10] G. J. Wang and D. K. Miu, "Unsupervised adaptive neural network control of complex mechanical systems," in *Proc. 1991 American Control Conf.*, Boston, 1991, pp. 28–29.

[11] R. Jacobs and M. Jordan, "A modular connectionist architecture for learning piecewise control strategies," in *Proc. 1991 American Control Conf.*, Boston, 1991, pp. 1597–1602.

[12] L. Rabelo and X. Avula, "Hierarchical neurocontroller architecture for intelligent robotic manipulation," *IEEE Int. Conf. Robotics and Automation*, Sacramento, CA, 1991, pp. 2656–2661.

[13] C. C. Ku, K. Y. Lee, and R. M. Edwards, "Neural network for adapting nuclear power plant control for wide-range operation," *Trans. American Nuclear Soc.*, vol. 63, pp. 114–115, June 1991.

[14] D. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in D. Rumelhart and J. McClelland (Eds.), *Parallel Distributed Processing*, vol. 1. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[15] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. Nat. Academy of Sciences*, vol. 79, 1982, pp. 2554–2558.

[16] A. Lapedes and R. Farber, "A self-optimizing nonsymmetrical neural network for content addressable memory and pattern recognition," *Physica*, vol. 22D, pp. 247–259, 1986.

[17] F. J. Pineda, "Generalization of backpropagation to recurrent networks," *Phys. Rev. Lett.*, vol. 59, no. 19, pp. 2229–2232, Nov. 1987.

[18] F. J. Pineda, "Recurrent backpropagation and the dynamical approach to adaptive neural computation," *Neural Computation*, vol. 1, pp. 161–172, 1989.

[19] B. A. Pearlmutter, "Learning state space trajectories in recurrent neural networks," *Neural Computation*, vol. 1, pp. 263–269, 1989.

[20] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270–280, 1989.

[21] L. B. Almeida, "Backpropagation in non-feedforward networks," in I. Aleksander (Ed.), *Neural Computing Architectures*. Cambridge, MA: MIT Press, 1989, pp. 75–91.

[22] L. Ljung, "Issue in system identification," *IEEE Contr. Syst. Mag.*, vol. 11, pp. 25–29, Jan. 1991.

[23] C. C. Ku and K. Y. Lee, "System identification and control using diagonal recurrent neural networks," in *Proc. 1992 American Control Conf.*, Chicago, June 24–26, 1992, pp. 545–549.

[24] T. Yabuta and T. Yamada, "Learning control using neural networks," in *Proc. IEEE Int. Conf. Robotics and Automation*, Sacramento, CA, Apr. 1991, pp. 740–745.

[25] M. M. Polycarpou and P. A. Ioannou, "Learning and convergence analysis of neural-type structured networks," *IEEE Trans. Neural Networks*, vol. 3, no. 1, pp. 39–50, Jan. 1992.

[26] K. S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 252–262, Mar. 1991.

[27] F. C. Chen, "Adaptive control of nonlinear systems using neural networks," Dept. of Electrical Engineering, Michigan State University, East Lansing, Ph.D dissertation, 1990.

[28] R. DiGirolamo and S. Donley, "Flight control law synthesis using neural network theory," Naval Air Development Center, Warminster, PA, Rept. NADC-91004-60, 1990.

Chao-Chee Ku (S'90) was born in Taichung, Taiwan, on April 27, 1959. He received the B.S. and M.S. degrees in electrical engineering, from Tatung Institute of Technology, Taipei, Taiwan, in 1981 and 1983, respectively, and the Ph.D. degree in electrical and computer engineering at the Pennsylvania State University, University Park, in 1992.

From 1983 to 1985 he was a Programmer at the Marine Corps Computer Center for two years military service. Between 1985 and 1987, he was a Project Leader at the PROTON Semiconductor Corp., Taiwan, involving microcontroller chip design. In 1987, he joined Telecommunication Laboratories, Taiwan, as a Research Assistant in the Switching Laboratory. Dr. Ku's areas of interests are intelligent control, VLSI design, digital signal processing, and neural networks.

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 1968, and the Ph.D. degree in system science from Michigan State University, East Lansing, in 1971.

He has been on the faculties of Michigan State University, Oregon State University, University of Houston, and the Pennsylvania State University, where he is Professor of Electrical Engineering. He is currently in charge of the Power Engineering Program and Power Systems Control Laboratory at Penn State. His interests are system theory, and artificial intelligent, and their applications to large scale system, and power systems.

Dr. Lee is a registered Professional Engineer.