PENNSTATE

Diagonal Recurrent Neural Network-Based Control:

Convergence and Stability

Chao-Chee Ku and Kwang Y. Lee

Department of Electrical Engineering
The Pennsylvania State University
University Park, PA 16802

PENNSTATE

Outline of This Presentation

I. Introduction

II. DRNN-Based Control System

III. Convergence and Stability of the Closed-loop System

IV. Simulation Results

V. Conclusion

PENNSTATE

Artificial Neural Network Paradigms

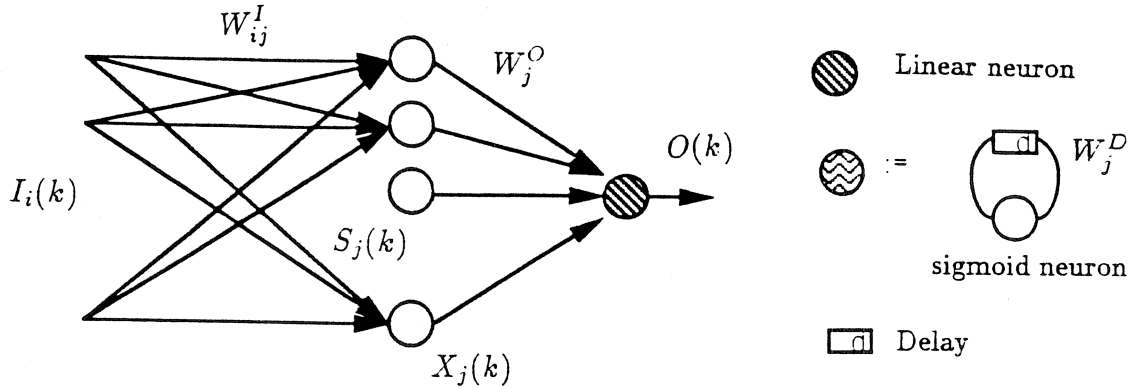
1. Feedforward Neural Network (FNN)
   . Static mapping
   . Can not represent a dynamic response w/o tapped delays

2. Fully Connected Recurrent Neural Network (FRNN)
   . Can naturally represent dynamic systems
   . Difficult to train and to converge in a short time

3. Diagonal Recurrent Neural Network (DRNN)
   . Fewer weights and shorter training time
   . Can be implemented easily for real-time control

PENNSTATE

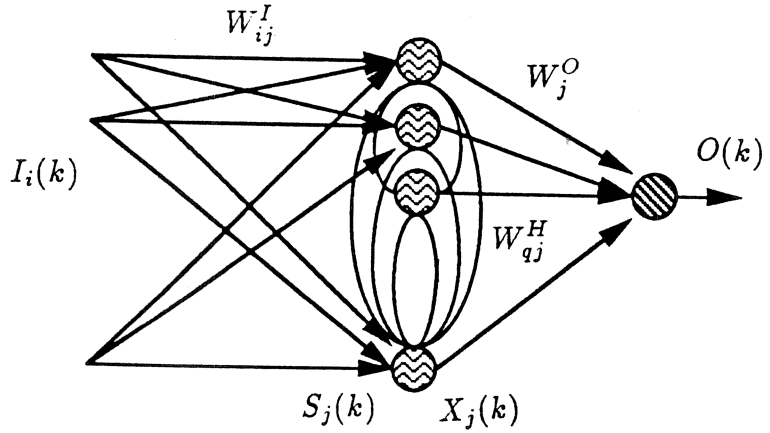
1. Feedforward Neural Network (FFNN)



* Static mapping
* Combine with tapped delays to perform dynamic mapping

PENNSTATE

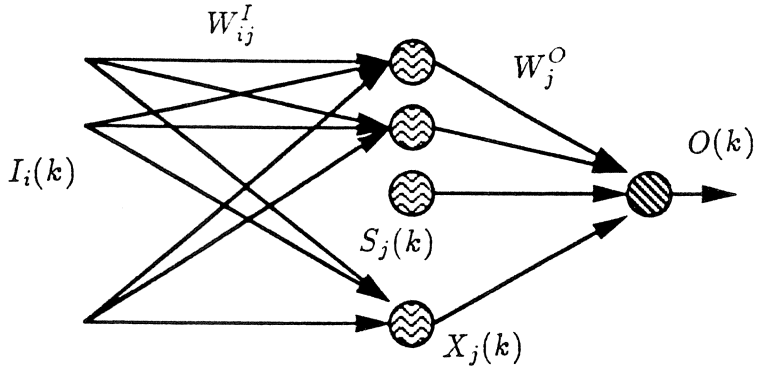
2. Fully Connected Recurrent Neural Network (FRNN)



* Dynamic mapping
* Convergence problem
* Stability problem

PENNSTATE

3. Diagonal Recurrent Neural Network (DRNN)



* Dynamic mapping
* Requires fewer weights
* Convergence is enhanced

A. Dynamic representation of DRNN

The mathematical model for the DRNN in Figure 1 is shown below:

$$O(k) = \sum_j W_j^O X_j(k), \quad X_j(k) = f(S_j(k)), \tag{1}$$

$$S_j(k) = W_j^D X_j(k-1) + \sum_i W_{ij}^I I_i(k), \tag{2}$$

where $I_i(k)$ is the i^{th} input to the DRNN, $S_j(k)$ is the sum of inputs to the j^{th} recurrent neuron, $X_j(k)$ is the output of the j^{th} recurrent neuron and $O(k)$ is the output of the DRNN. Here $f(\cdot)$ is the usual sigmoid function representing nonlinear threshold function, and W^I, W^D, and W^O are input, recurrent, and output weight vectors, respectively, in $\Re^{n_i}$, $\Re^{n_d}$, and $\Re^{n_o}$, which are Euclidean spaces with appropriate dimensions.

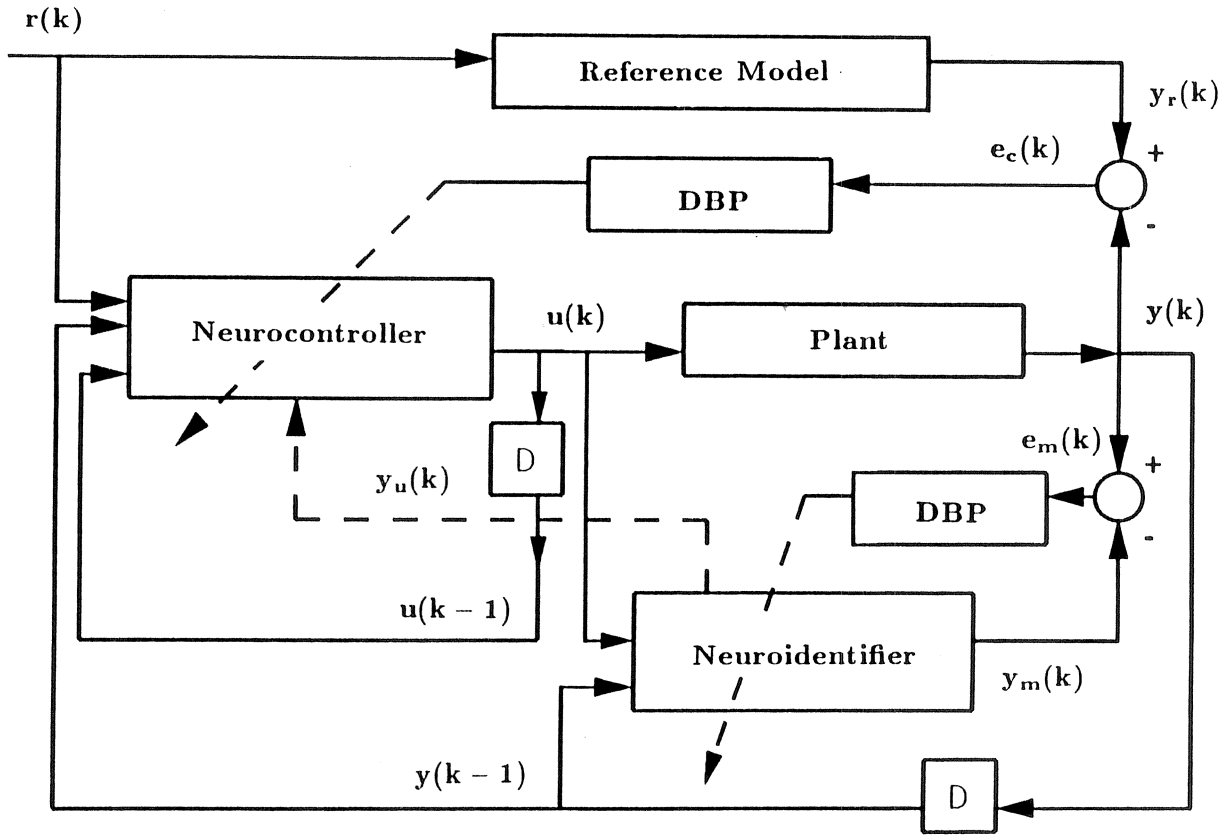


Figure 3.2 Block diagram of DRNN based control system.

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

$$E_c = \frac{1}{2}(y_r(k) - y(k))^2. \tag{3}$$

The error function (3) 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.,

$$E_m = \frac{1}{2}(y(k) - y_m(k))^2, \tag{4}$$

where $y_m(k) = O(k)$ in (1).

The gradient of error in (3) with respect to an arbitrary weight vector $W \in \Re^n$ is represented by

$$\frac{\partial E_c}{\partial W} = -e_c(k)\frac{\partial y(k)}{\partial W} = -e_c(k)y_u(k)\frac{\partial O(k)}{\partial W}, \tag{5}$$

where $e_c(k) = y_r(k) - y(k)$ is the error between the desired and output responses of the plant, and the factor $y_u(k) \equiv \frac{\partial y(k)}{\partial u(k)}$ 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 (4) simply becomes

$$\frac{\partial E_m}{\partial W} = -e_m(k)\frac{\partial y_m(k)}{\partial W} = -e_m(k)\frac{\partial O(k)}{\partial W}, \tag{6}$$

The output gradient $\frac{\partial O(k)}{\partial W}$ is common in (5) and (6), and needs to be computed for both DRNC and DRNI. The gradient with respect to output, recurrent, and input weights, respectively, are computed using the following equations :

$$\frac{\partial O(k)}{\partial W_j^O} = X_j(k) \tag{7a}$$

$$\frac{\partial O(k)}{\partial W_j^D} = W_j^O P_j(k) \tag{7b}$$

$$\frac{\partial O(k)}{\partial W_{ij}^I} = W_j^O Q_{ij}(k), \tag{7c}$$

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

$$P_j(k) = f'(S_j)\Big(X_j(k-1) + W_j^D P_j(k-1)\Big), \tag{8a}$$

$$Q_{ij}(k) = f'(S_j)\Big(I_i(k) + W_j^D Q_{ij}(k-1)\Big), \tag{8b}$$

B. Dynamic backpropagation for DRNI

From (6), the negative gradient of the error with respect to a weight vector in $\Re^n$ is

$$-\frac{\partial E_m}{\partial W} = e_m(k)\frac{\partial O(k)}{\partial W}, \tag{9}$$

where the output gradient is given by (7) and (8), and W represents W^O, W^D, or W^I in $\Re^{n_o}$, $\Re^{n_d}$, or $\Re^{n_i}$, respectively.

The weights can now be adjusted following any gradient method such as the steepest descent method, i.e., the update rule of the weights becomes

$$W(n+1) = W(n) + \eta(-\frac{\partial E_m}{\partial W}) + \alpha\Delta W(n), \tag{10}$$

where η is a learning rate, α is a momentum factor, and $\Delta W(n)$ represents the change in weight in the n^{th} iteration.

C. *Dynamic backpropagation for DRNC*

In the case of DRNC, from (5), the negative gradient of the error with respect to a weight vector in $\Re^n$ is

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

Since the plant is normally unknown, the sensitivity term $y_u(k)$ is unknown. This unknown value can be identified 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.

Therefore, the sensitivity was approximated in [2] and shown to be

$$y_u(k) \equiv \frac{\partial y(k)}{\partial u(k)} \approx \frac{\partial y_m(k)}{\partial u(k)} = \sum_j W_j^O f'(S_j(k)) W_{1j}^I, \tag{12}$$

where the variables and weights are those found in DRNI.

Adaptive Learning Rate for Training DRNN

A discrete-type Lyapunov function can be given by

$$V(k) = \frac{1}{2}e^2(k), \tag{13}$$

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

$$\Delta V(k) = V(k+1) - V(k) = \frac{1}{2}\Big[e^2(k+1) - e^2(k)\Big]. \tag{14}$$

The error difference due to the learning can be represented by

$$e(k+1) = e(k) + \Delta e(k) = e(k) + \left[\frac{\partial e(k)}{\partial W}\right]^T \Delta W, \tag{15}$$

where ΔW represents a change in an arbitrary weight vector in $\Re^n$.

A. Convergence of DRNI

From the update rule of (6) and (10) with $\alpha = 0$, and since $e_m(k) = y(k) - y_m(k)$ thus

$$\frac{\partial e_m(k)}{\partial W_I} = -\frac{\partial y_m(k)}{\partial W_I(k)},$$

$$\Delta W_I = -\eta_I e_m(k)\frac{\partial e_m(k)}{\partial W_I} = \eta_I e_m(k)\frac{\partial O(k)}{\partial W_I}. \tag{16}$$

Theorem 1: Let η_I, the learning rate for the weights of DRNI, satisfy $\eta_I = \eta_1/g_{I,max}^2$, with $0 < \eta_1 < 2$, and $g_{I,max}$ defined as $g_{I,max} := max_k\|g_I(k)\|$, where $g_I(k) = \frac{\partial O(k)}{\partial W_I}$, and $\|\cdot\|$ is the usual Euclidean norm in $\Re^n$. Then the convergence is guaranteed if η_I is chosen as

$$0 < \eta_I < \frac{2}{g_{I,max}^2}. \tag{17}$$

B. Convergence of DRNC

From the update rule of (10) and (11) with $\alpha = 0$, and following the same procedure as in Section A,

$$\Delta W_c = -\eta_c e_c(k) \frac{\partial e_c(k)}{\partial W_c} = \eta_c e_c(k) y_u(k) \frac{\partial O(k)}{\partial W_c}.$$

Theorem 2: Let η_c, the learning rate for the weights of DRNC, satisfy $\eta_c = \eta_2 / g_{c,max}^2 S_{max}^2$, with $0 < \eta_2 < 2$, $g_{c,max}$ defined as $g_{c,max} := max_k \|g_c(k)\|$, where $g_c(k) = \frac{\partial O(k)}{\partial W_c}$, and $S_{max} = \frac{h_I W_{I,max}^O W_{I,1max}^I}{2}$, where h_I is the number of neurons in the hidden layer. Then the convergence is guaranteed if η_c is chosen as

$$0 < \eta_c < \frac{2}{S_{max}^2 g_{c,max}^2}. \tag{18}$$

SIMULATION RESULTS

Example 1: A BIBO nonlinear plant

Reference Model:

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

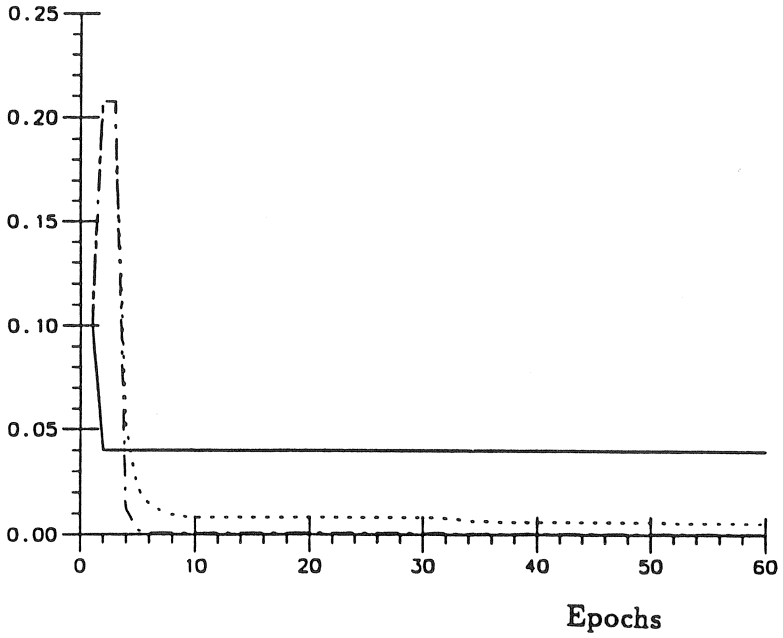
$$r(k) = sin\left(\frac{2\pi k}{25}\right) + sin\left(\frac{2\pi k}{10}\right)$$

Plant Model:

$$y(k+1) = \frac{y(k)}{1.0 + y^2(k)} + u^3(k)$$

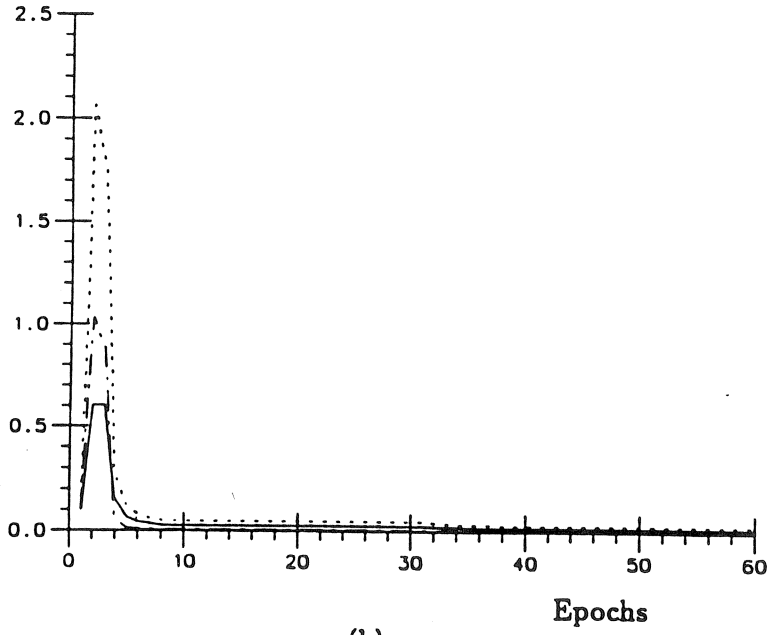
In this example, $Z_c = \{r(k), u(k-1), y(k-1)\}$ and $Z_I = \{u(k), y(k-1)\}$, thus $n_c = 3$ and $n_i = 2$. Also, $N_T = 14$ and $W_T = 67$. $\eta_c = 0.1$ and $\eta_I = 0.1$.

SIMULATION RESULTS



(a)

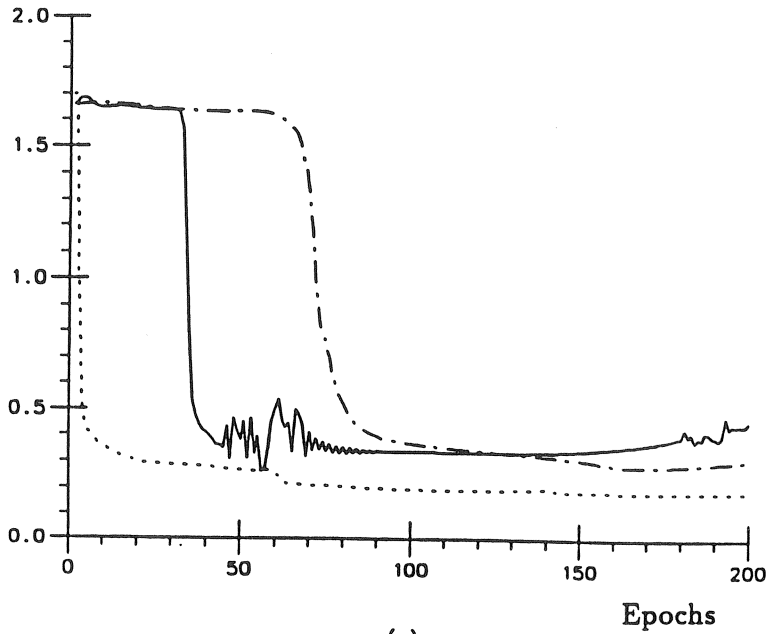
DRNI



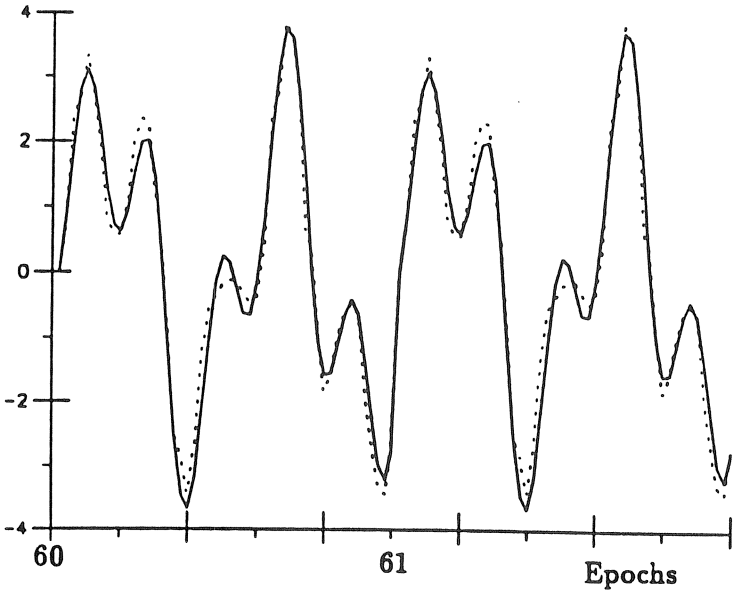
(b)

DRNC

SIMULATION RESULTS



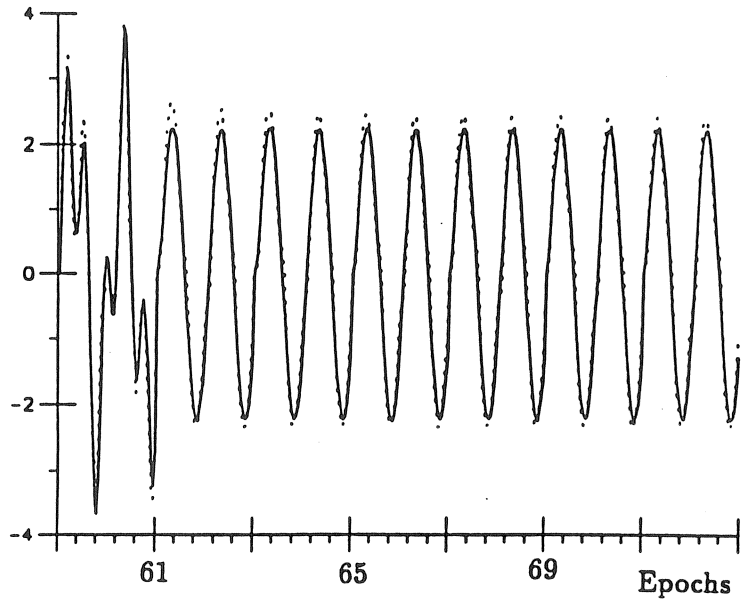
(c)



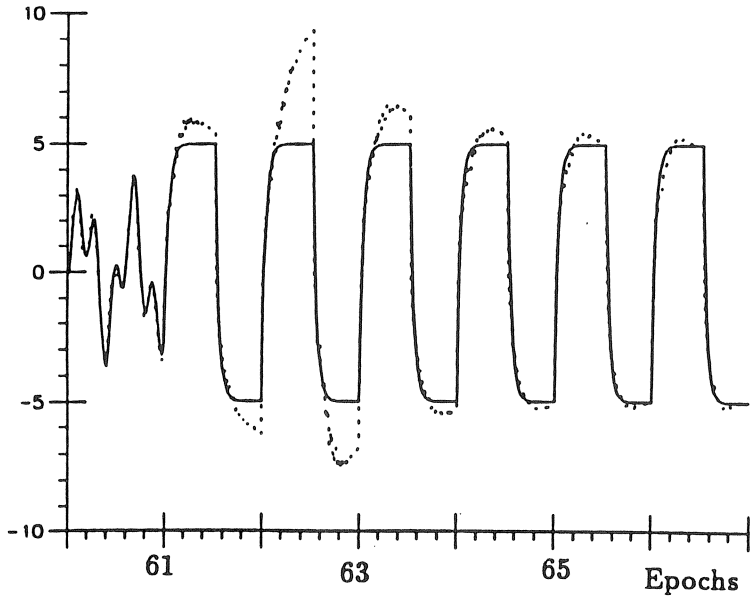
(d)

SIMULATION RESULTS

The on-line adapting ability of DRNN based control



(a)



(b)

Simulation Results : Example 2- MRC

Reference model:

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

Desired reference:

$$r(k) = 0.5sin(\frac{2\pi k}{50.0}) + 0.5sin(\frac{2\pi k}{100.0})$$

Plant :

$$y(k+1) = 0.2y^2(k) + 0.2y(k-1) + 0.4sin[0.5(y(k)+$$

$$y(k-1))]\cdot cos[0.5(y(k)+y(k-1))] + 1.2u(k)$$

Simulation Results: Example 2

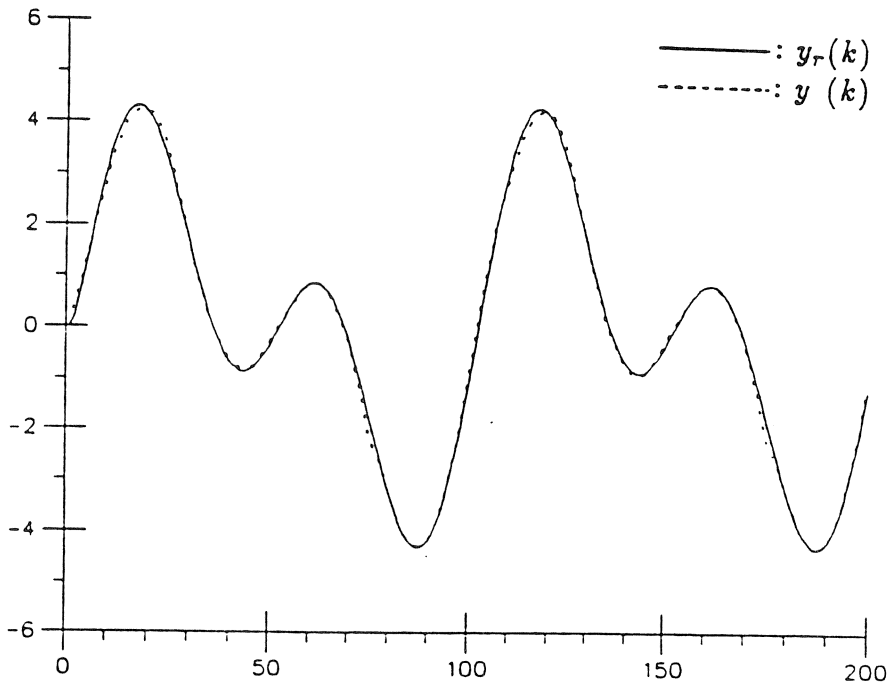


Fig. 4 (a) Outputs of reference model and plant

Simulation Results: Example 2

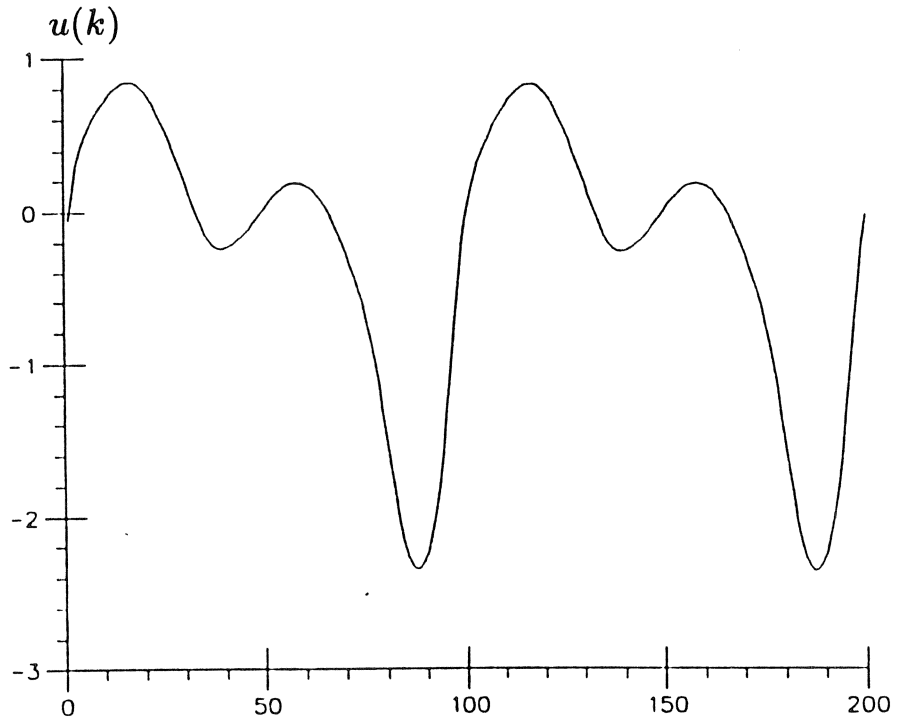


Fig. 4 (b) Control signal generated from DRNC

SIMULATION RESULTS

Example 3: Flight Control

Reference model:

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

Plant:

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

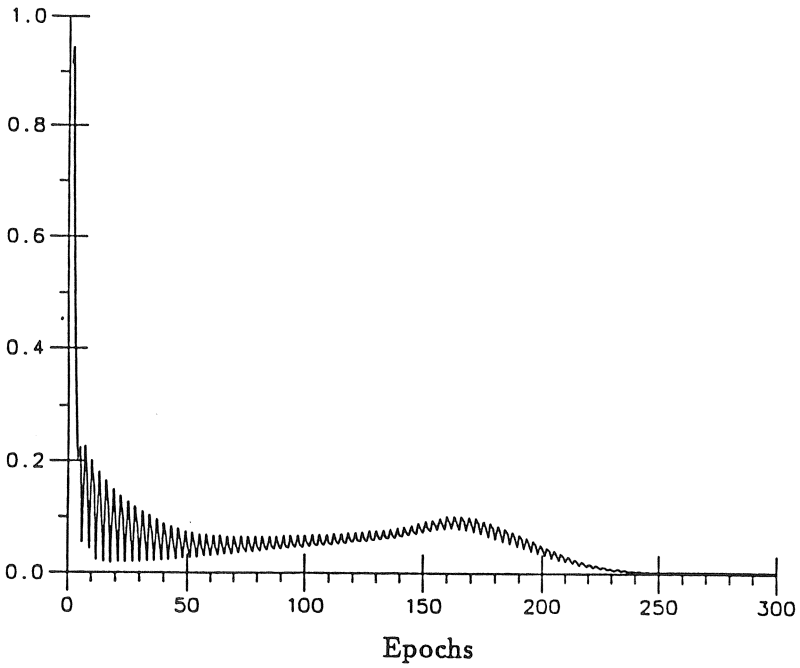
training sets $(x_1(0), x_2(0)) =$ $(0.0, 0.0)$, $(0.1, 0.3)$, and $(0.5, 0.75)$

testing set $(x_1(0), x_2(0)) = (0.8, 1.0)$

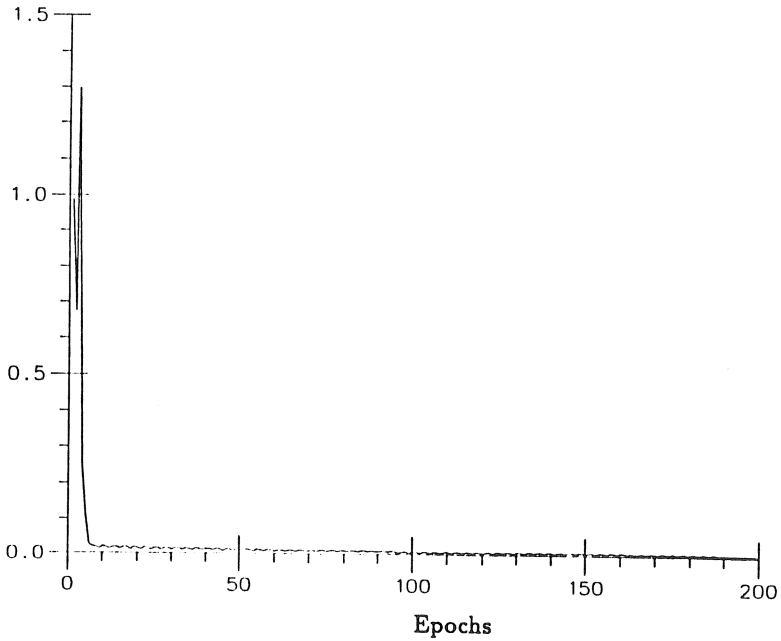
The step input is applied to the reference model, and $r(t) = 1$

$N_T = 16$ and $W_T = 84$. η_c and η_I , 20 , and both biases, b_c and b_I, 1.0.

Simulation Results



Fixed learning rates



Adaptive learning rates

Simulation Results: Example 1

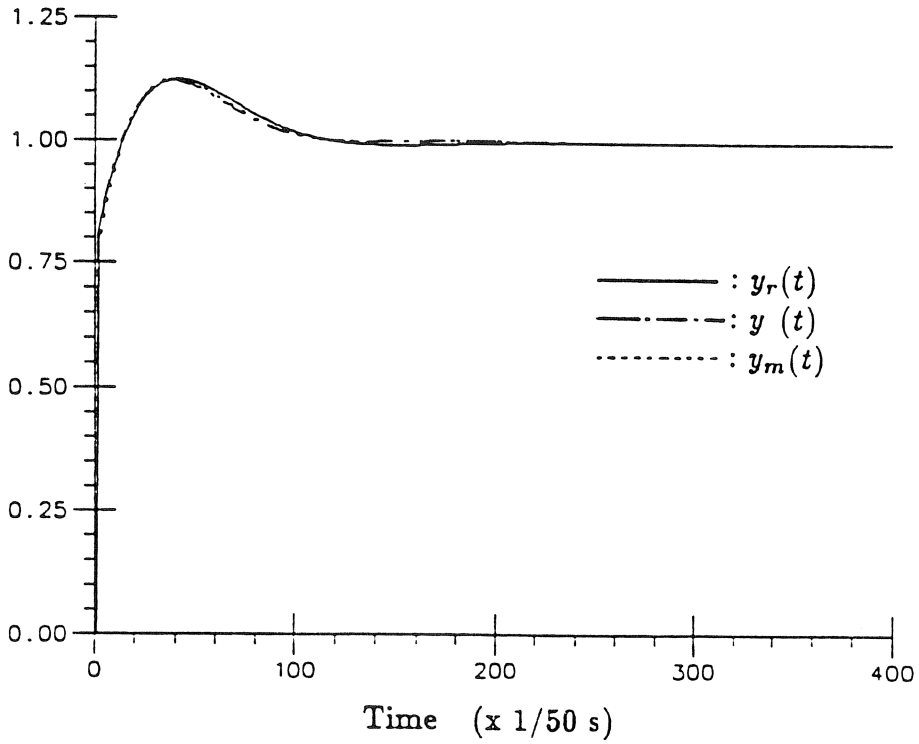


Fig. 3 (b) Outputs of reference model, plant, and DRNI

Example 2: A Controlled Van der Pol equation

The objective of this example is to investigate the ability of DRNN based control system in controlling a nonlinear plant with a Van der Pol dynamics.

Plant model:

$$\ddot{x}(t) - \mu(1 - x^2(t))\dot{x}(t) + x(t) = u(t)$$

or

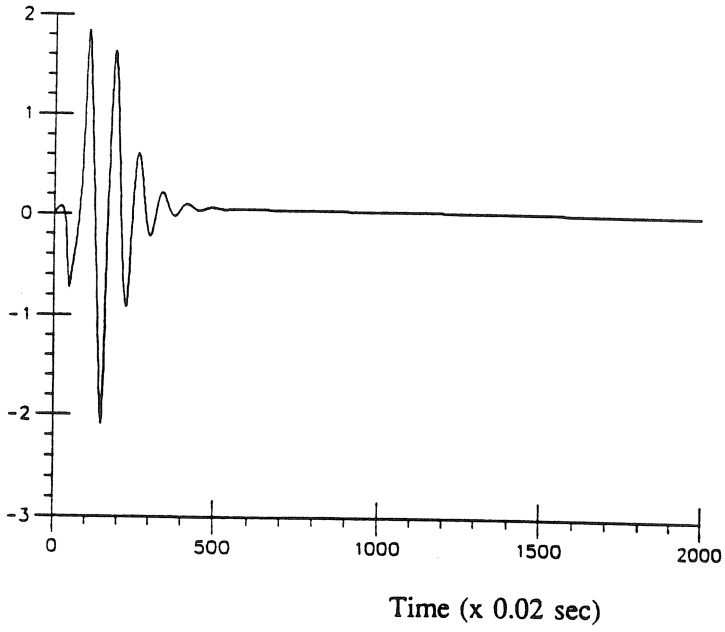
$$\dot{x}_1(t) = x_2(t)$$

$$\dot{x}_2(t) = -x_1(t) + \mu(1 - x_1^2(t))x_2(t) + u(t)$$

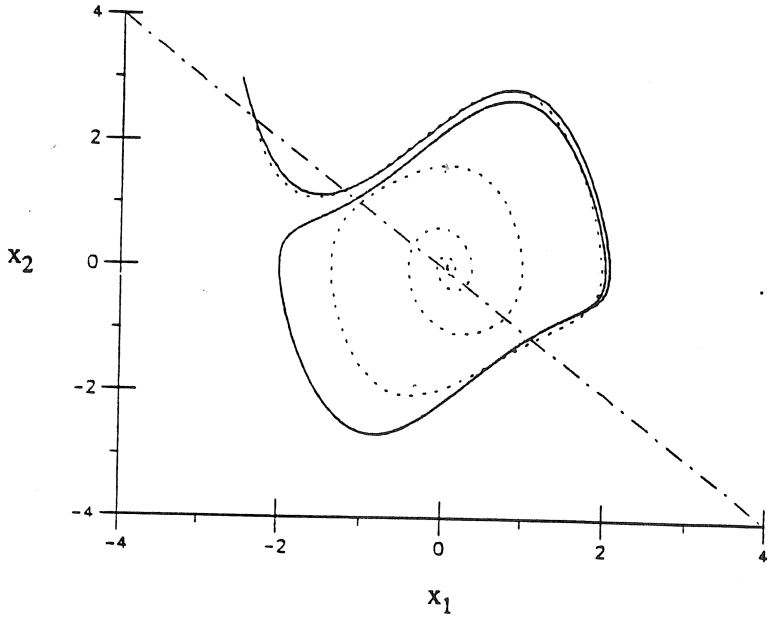
and

$$y(t) = x_1(t) + x_2(t).$$

PENNSTATE



The control signal generated from DRNC



The plots of (x_1, x_2); u(t)=0 (solid line), controlled tracking output (dotted line), and the line $x_1 + x_2 = 0$ (dotted-dashed line)

PENNSTATE

Conclusion

1. A new neural network paradigm of DRNN is developed as a minimal realization of the fully recurrent neural network.

2. The proposed paradigm has the desired features of simplicity and recurrence. This makes the convergence and stability possible.

3. Adaptive learning algorithm is developed for on-line approach.

4. Convergence theorems are developed which not only guarantees the error to converge to an arbitrary small value, but also guarantees the closed-loop stability of the BIBO stable system.