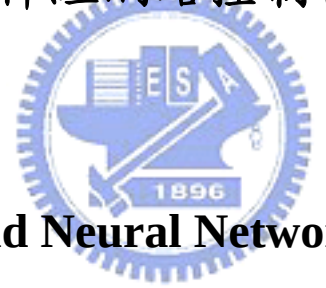
國 立 交 通 大 學

電 機 與 控 制 工 程 學 系

碩 士 論 文

霍普菲爾類神經網路控制器設計及其應用



Design of Hopfield Neural Network Controller with Its applications

研究生：甘能捷

指導教授：王啟旭　教授

中華民國九十五年十月

霍普菲爾類神經網路控制器設計及其應用

Design of Hopfield Neural Network Controller with Its applications

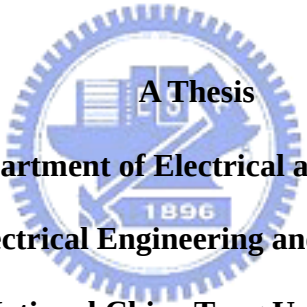
研究生：甘能捷　　Student: Neng-Chieh Kan

指導教授：王啟旭　教授　　Advisor: Chi-Hsu Wang

國 立 交 通 大 學

電機與控制工程學系

碩士論文



A Thesis

Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master

in

Electrical and Control Engineering

October 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年十月

霍普菲爾類神經網路控制器設計及其應用

研究生：甘能捷　　指導教授：王啟旭 教授

國立交通大學電機與控制工程研究所

摘要

本篇論文是將霍普菲爾類神經網路當作一個控制器應用在控制領域上。先將霍普菲爾神經網路做訓練，使其能夠產生較佳的控制訊號，訓練完之後再當一個即時的控制器使用。霍普菲爾神經網路是一個有迴授的具有保持穩定特性的類神經網路。我們利用兩種方法來做網路的訓練法則，其中一個是倒傳遞訓練演算法，而另一個動態最佳學習則可以加速我們的學習過程。要使用倒傳遞訓練演算法來訓練霍普菲爾類神經網路當作一個即時控制器，最小能量的條件扮演了一個重要的角色。最後我們用倒單擺系統和飛機控制系統當作我們的受控體得到了良好的結果，並討論如何實現霍普菲爾類神經網路。

Design of Hopfield Neural Network Controller with Its applications

Student: Neng-Chieh Kan **Advisor: Chi-Hsu Wang**

Department of Electrical and Control Engineering

National Chiao Tung University

ABSTRACT

This thesis explores the design of Hopfield neural network (NN) as a controller for control system. The training algorithms for Hopfield NN are first developed to generate proper control signal and then the Hopfield NN is used as a real-time controller after training. The Hopfield NN is a recurrent neural network which has the potential of maintaining stability. We use two approaches for training algorithm, one is normal back-propagation training, the other is dynamic optimal learning which can accelerate the learning process. The minimum energy requirement in Hopfield NN plays the key role in the back-propagation training of Hopfield NN as a real-time controller. Finally the inverted pendulum system and aircraft control system are illustrated as the plants to be controlled by Hopfield NN. Excellent results are obtained and the implementation of Hopfield NN is also discussed.

ACKNOWLEDGEMENT

I feel external gratitude to my advisor, Chi-Hsu Wang for teaching me many things about how to do the research in the two years. When I get some problems in my research, he always gives me a hand at the right moment such that my thesis can finish in time.

And I am grateful to everyone in ECL. I am very happy to get along with all of you. Finally, I appreciate my family's support and encouragement; therefore I can finish my master degree smoothly.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

Neural network has increasing applications in many fields: pattern recognition, identification and control of dynamical systems, system modeling. The most interesting character of neural network is that it can learn how to achieve the goal by learning algorithm and training data sets. There are many kinds of neural network such as single-layer network, multilayer feed-forward network, radial basis function network, Hopfield network, ... etc [1]. Each kind of network has different applications in many fields. Among those neural networks, the Hopfield neural network will be discussed in this thesis. The Hopfield neural network is first proposed by Hopfield J.J. in 1984 [2]. The Hopfield neural network has applied on many fields: optimization [3, 4], system identification, [5, 6], and image processing [7, 8]. In this thesis, we want to use the Hopfield neural network as a controller. The Hopfield neural network is trained by one most popular algorithm of neural network which is the back propagation algorithm [9, 10]. The well-known back propagation algorithm for training multilayer feed-forward network was proposed by Rumelhart in 1986 [11]. By using the same basis, it also can be applied on the Hopfield neural network. But in back propagation algorithm, there is an important problem about the choice of the learning rate. For smaller learning rate, we may have a convergent result. But the speed of the output convergence is very slow and need more time to train the network. For larger learning rate, the speed of training can be accelerated, but it will cause the training result to fluctuate and even leads to divergent result. The dynamic optimum learning rate algorithm proposed in [12, 13] can help us to solve the learning rate problem. The basic theme in [12, 13] is to find a stable and optimal learning rate for the next iteration in back propagation algorithm such that the neural network can maintain in convergence. Thus, we use the back propagation algorithm with

optimum learning rate to train the Hopfield neural network as a controller. The inverted pendulum system and the aircraft control system are used to verify the algorithm in the end of this thesis.

CHAPTER 2
The Hopfield Neural Network

In this chapter, the Hopfield neural network will be reviewed. First the structure of recurrent neural network which the overall Hopfield neural network belong to will be discussed. The Hopfield neural network can be divided into continuous part and discrete part, and the two parts will be discussed in Section 2.2 and Section 2.4. We will also talk about the energy function in the Hopfield neural network in Section 2.3 of this chapter.

2.1 The Recurrent Neural Network

Figure 2-1 shows a kind of recurrent neural network, which consists of a set of neurons form a multiple-loop feedback system. The output of each neuron is fed back to each of all the neurons in the neural network.

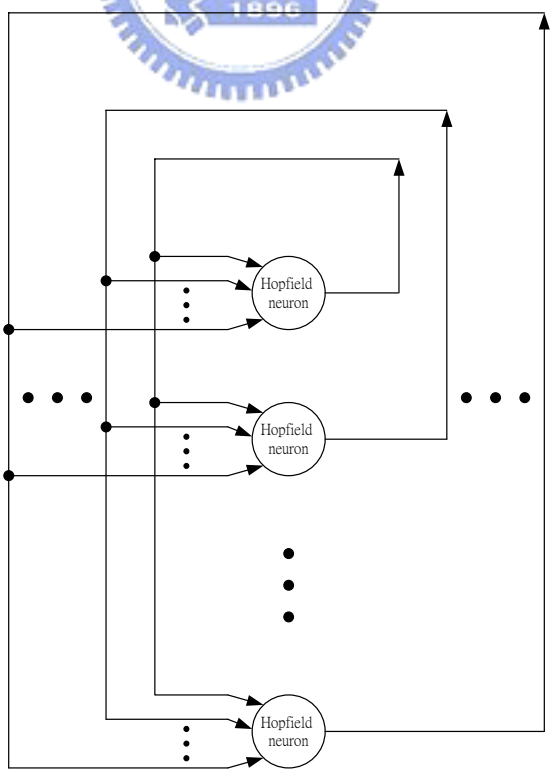


Figure 2-1. The continuous Hopfield neural network

When each neuron in Figure 2-1 is a Hopfield neuron (which will be discussed in the next section), Figure 2-1 is the so-called Hopfield neural network. It is actually a nonlinear closed-loop feedback system which will have dynamic responses in each of the output signals. The stability analysis of the Hopfield neural network plays a major role in the applicability of Hopfield neural network to engineering fields.

2.2 The Hopfield Neuron with Closed-Loop Dynamics

As mentioned in Section 2.1, we use the additive model of a neuron to form the continuous Hopfield neural network [14]. The Hopfield neuron is defined in Figure 2-2 as a continuous RC electrical network with a nonlinear activation function $\varphi(\cdot)$ to confine v_j to yield the final output signal x_j. The dotted line ellipse in Figure 2-2 is the Hopfield neuron in Figure 2-1.

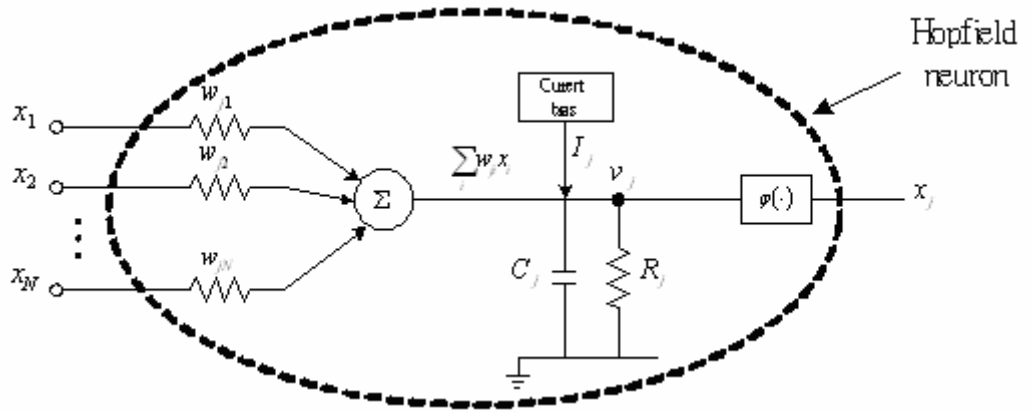


Figure 2-2. The additive model of a single Hopfield neuron

In Figure 2-2, the inputs $x_i(t)$ ($i = 1, ..., N$) are fed-back from the outputs $x_j(t)$ ($j = 1, ..., N$). The inputs $x_i(t)$ are represented by potentials, and the synaptic weighting factors w_{ji} are represented by conductance. The summing junction is a unit current gain

summing junction with low input resistance and high output resistance. We also may have a bias current I_j in the additive model. The $\varphi(\cdot)$ in this additive model is a nonlinear sigmoid function which is defined by hyperbolic tangent function [2, 14]:

$$x_j = \varphi(v_j) = \tanh\left(\frac{a_j v_j}{2}\right) = \frac{1-\exp(-a_j v_j)}{1+\exp(-a_j v_j)} \tag{2-1}$$

which has a slope of $a_j/2$ at the origin as shown by

$$\frac{a_j}{2} = \left.\frac{d\varphi}{dv_j}\right|_{v_j=0} \tag{2-2}$$

Hence we can say that a_j is the gain parameter of neuron j. Figure 2-3 shows a plot of standard sigmoidal nonlinearity $\varphi(v)$.

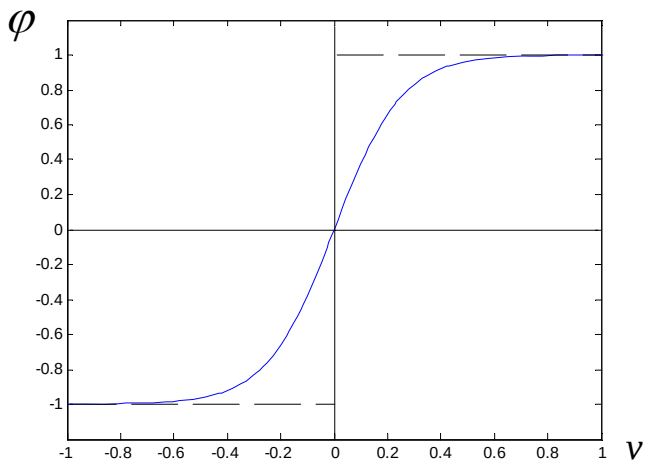


Figure 2-3. The hyperbolic tangent function

Now, we should also investigate the inverse function of $\varphi(\cdot)$. The inverse input- output relation of (2-1) may be written as:

$$v_j = \varphi^{-1}(x_j) = \frac{1}{a_j}\log\left(\frac{1-x_j}{1+x_j}\right) \tag{2-3}$$

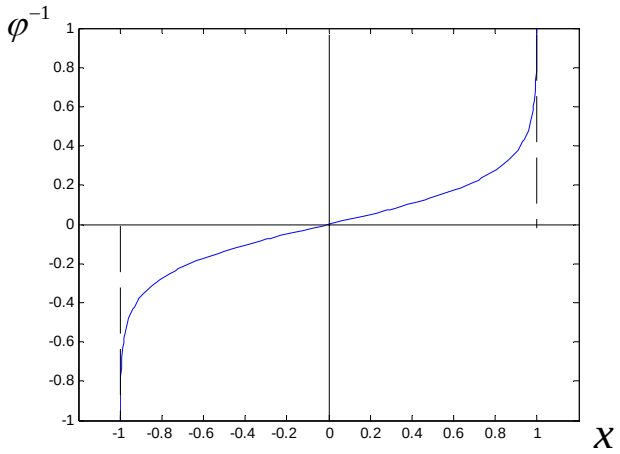


Figure 2-4. The inverse of hyperbolic tangent function

Figure 2-4 shows the corresponding plot of the inverse nonlinearity $\varphi^{-1}(x)$.From Figure 2-2 with the closed-loop configuration in Figure 2-1, we can get the neural dynamics in the overall Hopfield neural network as follows [14]. By using the Kirchhoff's law which states that the total current entering a junction is equal to that leaving the same junction, we can obtain the following dynamic node equation in this model:

$$C_j \frac{d}{dt} v_j(t) + \frac{v_j(t)}{R_j} = \sum_{i=1}^{N} w_{ji} x_i(t) + I_j \ , \quad j = 1, ..., N \tag{2-4}$$

The input $x_i(t)$ is the feedback of the output of the nonlinear sigmoid function $\varphi(\cdot)$, so the dynamic equation becomes:

$$C_j \frac{d}{dt} v_j(t) = -\frac{v_j(t)}{R_j} + \sum_{i=1}^{N} w_{ji} \varphi(v_i(t)) + I_j \ , \quad j = 1, ..., N \tag{2-5}$$

Eq. (2-5) completely describes the time evolution of the system. If each node is given an initial value $v_j(0)$, then the value $v_j(t)$ and the nonlinear activation function output $x_j(t) = \varphi(v_j(t))$ at time t can be known by solving the differential equation in (2-5).

The stability analysis of the above continuous Hopfield neural network can be discussed via the energy (or Lyapunov) function of the Hopfield neural network, which will be introduced in the next section.

2.3 Stability Analysis of Hopfield Neural Network

The energy (Lyapunov) function [2, 14] of the continuous type Hopfield neural network can be defined by

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ji}x_i x_j + \sum_{j=1}^{N}\frac{1}{R_j}\int_0^{x_j}\varphi^{-1}(x)dx - \sum_{j=1}^{N} I_j x_j \tag{2-6}$$

In order for the Hopfield neural network to be asymptotically stable, the time derivative of the above energy function E must be negative. By differentiating the energy function E with respect to time and using the relationship of x_j and v_j in (2-3), we can get

$$\frac{dE}{dt} = -\sum_{j=1}^{N}\left(\sum_{j=1}^{N} w_{ji}x_i - \frac{v_j}{R_j} + I_j\right)\frac{dx_j}{dt} \tag{2-7}$$

The term inside the parentheses in (2-7) is actually equal to $C_j\frac{dv_j}{dt}$ via (2-4). We may thus simplify (2-7) to

$$\frac{dE}{dt} = -\sum_{j=1}^{N} C_j\left(\frac{dv_j}{dt}\right)\frac{dx_j}{dt} \tag{2-8}$$

Since $v_j = \varphi^{-1}(x_j)$ from (2-3), the above (2-8) becomes

$$\frac{dE}{dt} = -\sum_{j=1}^{N} C_j\left(\frac{d\varphi^{-1}(x_j)}{dt}\right)\frac{dx_j}{dt} \tag{2-9}$$

By using the chain rule, (2-9) can be further simplified as:

$$\frac{dE}{dt} = -\sum_{j=1}^{N} C_j\left(\frac{dx_j}{dt}\right)^2\left(\frac{d}{dx_j}\varphi^{-1}(x_j)\right) \tag{2-10}$$

It is obvious from Figure 2-4 that that $\varphi^{-1}(x_j)$ is an increasing function of x_j. It follows therefore that

$$\frac{d}{dx_j}\varphi^{-1}(x_j) > 0, \quad \text{for all} \quad x_j \tag{2-11}$$

It is also true that

$$\left(\frac{dx_j}{dt}\right)^2 \geq 0 \quad \text{for all} \quad x_j \tag{2-12}$$

Therefore, according to (2-11) and (2-12), we have the final fact:

$$\frac{dE}{dt} = -\sum_{j=1}^{N} C_j \left(\frac{dx_j}{dt}\right)^2 \left(\frac{d}{dx_j}\varphi_j^{-1}(x_j)\right) \leq 0 \tag{2-13}$$

Eq. (2-13) says that if the nonlinear activation function is defined as the hyperbolic tangent function shown in Figure 2-3, then the set of nonlinear differential equations defined in (2-5), which represents the dynamical equations of the continuous Hopfield neural network, is asymptotically stable. From (2-13), we also know that $\frac{dE}{dt} = 0$ if $\frac{dx_j}{dt} = 0$. The points where $\frac{dx_j}{dt} = 0$ are defined as the fixed points in the trajectory space, where the energy function $E(t)$ of those fixed points will remain still. These fixed points are also sometimes called "attractors" due to the fact that the surrounding states will sometimes be attracted to the fixed points as the stable states. Thus, we use a second order example which means only x_1 and x_2 to explain the attractor trajectory of Hopfield neural network in Figure 2-5 [2].

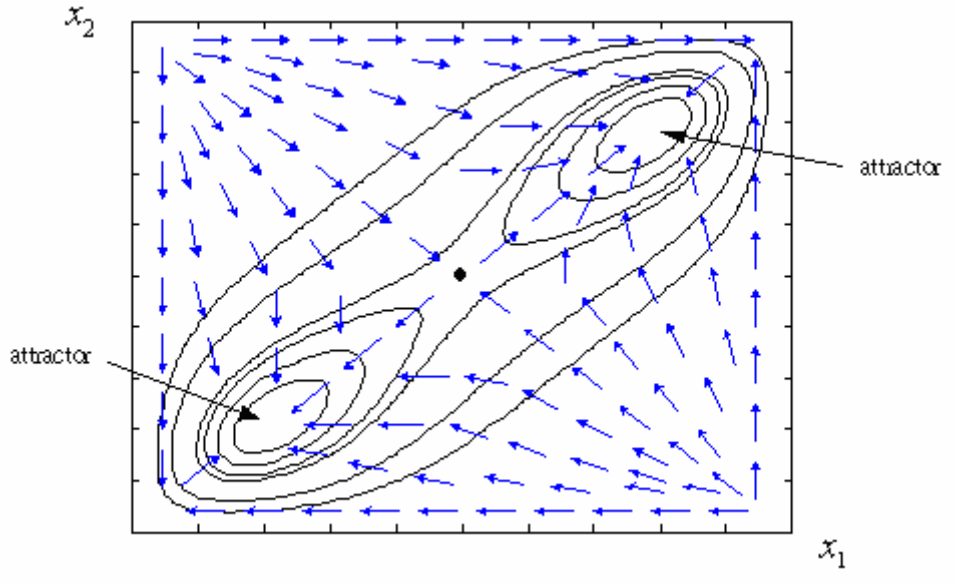


Figure 2-5. The attractor trajectory of second order Hopfield neural network

The contours are the energy contour of Hopfield neural network, and the arrows are the state trajectory. The attractors are located near the lower left and upper right corners. All the other unstable states will be attracted to the fixed points. The arrows show the motion of the states.

2.4 The Discrete Hopfield Neural Network

The continuous mode of Hopfield neural network is based on an additive model, as previously discussed. If the nonlinear activation function $\varphi(\cdot)$ in Figure 2-2 is replaced by the following ***sgn*** function [1, 14]:

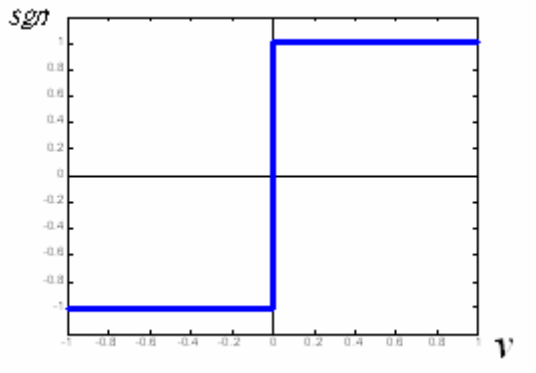


Figure 2-6. The ***sgn*** function

Thus we can basically have a discrete Hopfield neural network, like Figure 2-7. The discrete Hopfield neural network eliminates the self-loop feedback as shown in Figure 2-1. In the continuous mode, the nonlinear activation sigmoid function is the hyperbolic tangent function ((2-1)). The gain parameter a_j in (2-2) is the slope of the hyperbolic tangent function. If we let $a_j \to \infty$, then the input-output relation in a neuron of discrete mode becomes

$$x_j = \begin{cases} +1 & \text{for } v_j > 0 \\ -1 & \text{for } v_j < 0 \end{cases} \tag{2-14}$$

$$\varphi_j(0) = 0 \tag{2-15}$$

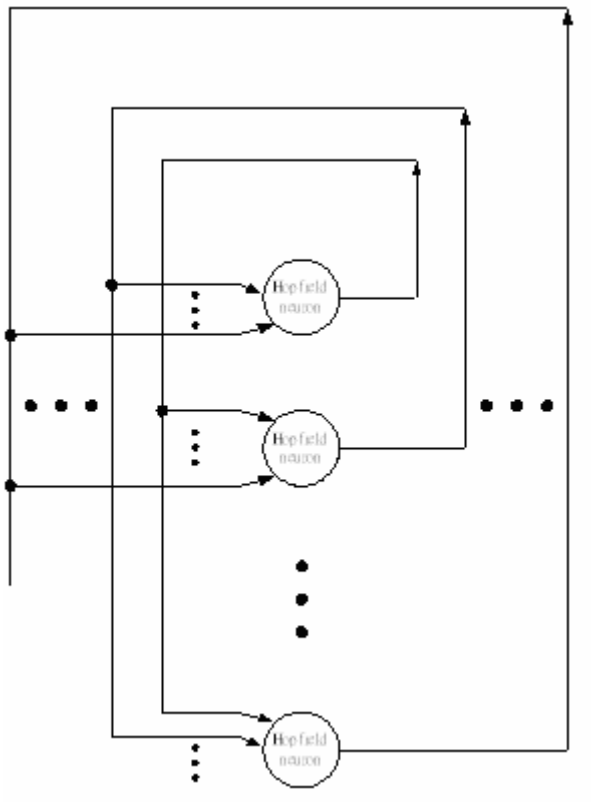


Figure 2-7. The discrete Hopfield neural network without self-loop feedback

The energy function $E(t)$ for discrete Hopfield neural network can be derived from that in

(2-6), i.e., the $E(t)$ for continuous Hopfield neural network. In discrete case, the gain parameter a_j is infinite and this will make the term $\sum_{j=1}^{N}\frac{1}{R_j}\int_0^{x_j}\varphi^{-1}(x)dx$ very small (from (2-3) and Figure 2-4). There is also no bias current I_j in discrete case. Thus, the energy function $E(t)$ of discrete Hopfield neural network becomes:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ji}x_i x_j \tag{2-16}$$

The most important application of the discrete model is working as a content-addressable memory (CAM) [14]. In this application, a content-addressable memory is error-correcting in the sense to retrieve a stored pattern, given a reasonable subset of the information content of that pattern. However, the synaptic weighting factors of the network that produce the desired fixed points are unknown, and the problem is how to determine them. There are two phases to the operation of the discrete Hopfield network as a CAM, namely the storage phase and the retrieval phase. Now we introduce the process of the discrete Hopfield model as CAM:

1. Storage phase:

   There are M N-dimensional vectors called fundamental memories denoted by $\xi_\mu = \left[\xi_{\mu,1} \quad \xi_{\mu,2} \quad .... \quad \xi_{\mu,N}\right]$, $\mu = 1,2,...,M$. And each $\xi_{\mu,i}$ is a binary code, it means that $\xi_{\mu,i} = +1$ or -1. The fundamental memories are the stable states to be stored and memorized by the network. Then we use the following Hebb's learning rule to define the weighting matrix of the discrete Hopfield network[14]:

$$\mathrm{W} = \frac{1}{N}\left(\sum_{\mu=1}^{M}\xi_\mu \xi_\mu{}^T - MI\right) \tag{2-17}$$

   where I is the identity matrix with N dimensions. We need to minus MI because the discrete model doesn't have self-loop feedback. We also find that the weighting matrix is

symmetric, i.e., $W^T = W$. The following theorem will explain why the weighting matrix in discrete Hopfield neural network be decided by (2-17).

Theorem 1

The updating rule

$$\mathrm{W} = \frac{1}{N}\left(\sum_{\mu=1}^{M} \xi_\mu \xi_\mu{}^T - MI\right)$$

will minimize the energy function $E(t)$ of discrete Hopfield neural network when the state X is the fundamental memory ξ_μ.

Proof:

We use the vector and matrix form to express the energy function in (2-16).

$$E = -\frac{1}{2} X^T W X \tag{2-18}$$

where X is a N dimensional column vector. We use the definition of weighting matrix in (2-17) to rewrite (2-18).

$$\begin{aligned} E &= -\frac{1}{2} X^T \frac{1}{N}\left(\sum_{\mu=1}^{M} \xi_\mu \xi_\mu{}^T - MI\right) X \\ &= -\frac{1}{2N}\left(\sum_{\mu=1}^{M} X^T \xi_\mu \xi_\mu{}^T X - M X^T X\right) \end{aligned} \tag{2-19}$$

Because X and ξ_μ are both N dimensional column vectors with $+1$ or -1 elements, we can get the result of (2-19).

$$\begin{aligned} E &= -\frac{1}{2N}\left(\sum_{\mu=1}^{M} \left(\xi_\mu{}^T X\right)^2 - MN\right) \\ &= -\frac{M}{2} - \frac{1}{2N}\sum_{\mu=1}^{M} \left(\xi_\mu{}^T X\right)^2 \end{aligned} \tag{2-20}$$

The reason for using Hebb's learning rule to decide the weighting matrix is right here. When

the state X is equal to the fundamental memory ξ_μ, $\left(\xi_\mu{}^T X\right)^2$ will reach the maximum. It means that the energy function E would reach the minimum if we choose the weighting matrix as (2-17).

2. Retrieval phase:

Now we start the algorithm to stabilize the system and retrieve the pattern. We get a initial wrong state $x(0)$ called "probe", which is a noisy version of the correct stable state ξ_μ. The initial probe has elements equal to ± 1. Then we use the following formula to update $x(n)$:

$$x(n+1) = \boldsymbol{sgn}\left[W \cdot x(n)\right] \tag{2-21}$$

where the ***sgn*** function is defined by:

$$\boldsymbol{sgn}[u] = \begin{cases} +1, & u > 0 \\ -1, & u < 0 \\ \text{previous state}, & u = 0 \end{cases} \tag{2-22}$$

If $W \cdot x(n)$ is greater than zero, neuron j will switch its states to $+1$ or remain in that state if it is already there. Similarly, if $W \cdot x(n)$ is less than zero, neuron j will switch its states to -1 or remain in that state if it is already there. If $W \cdot x(n)$ is exactly zero, neuron j is left in its previous state, regardless of whether it is $+1$ or -1. When $x(n+1) = x(n)$, the system reaches the stable state. It means that the state $x(n)$ is transformed to one of the stable states ξ_μ. Let us use an example to verify this process.

Example: $\xi_1 = \begin{bmatrix} 1 & -1 & 1 \end{bmatrix}'$ and $\xi_2 = \begin{bmatrix} -1 & 1 & -1 \end{bmatrix}'$ are the two stable states (fundamental

memories). Only the two states will be stable, the other unstable states will converge to the two stable states after some iteration. Then we compute the weighting matrix by (2-17):

$$W=\frac{1}{3}\left(\begin{bmatrix}1\\-1\\1\end{bmatrix}\begin{bmatrix}1 & -1 & 1\end{bmatrix}+\begin{bmatrix}-1\\1\\-1\end{bmatrix}\begin{bmatrix}-1 & 1 & -1\end{bmatrix}-2\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\right)=\frac{1}{3}\begin{bmatrix}0&-2&2\\-2&0&-2\\2&-2&0\end{bmatrix}$$

Now suppose the initial wrong unstable states is $x(0)=\begin{bmatrix}1 & 1 & 1\end{bmatrix}'$, we can get,

$$x(1)=\operatorname{sgn}\left(W\cdot x(0)\right)=\operatorname{sgn}\left(\frac{1}{3}\begin{bmatrix}0\\-4\\0\end{bmatrix}\right)=\begin{bmatrix}1\\-1\\1\end{bmatrix}$$

$$x(2)=\operatorname{sgn}\left(W\cdot x(1)\right)=\operatorname{sgn}\left(\frac{1}{3}\begin{bmatrix}4\\-4\\4\end{bmatrix}\right)=\begin{bmatrix}1\\-1\\1\end{bmatrix}=x(1)$$

Because $x(1)=x(2)$, the system reaches its stable condition $x(n+1)=x(n)$. From $x(0)$ to $x(2)$, the initial unstable states $\begin{bmatrix}1 & 1 & 1\end{bmatrix}'$ converges to the fundamental memory $\xi_1=\begin{bmatrix}1 & -1 & 1\end{bmatrix}'$.

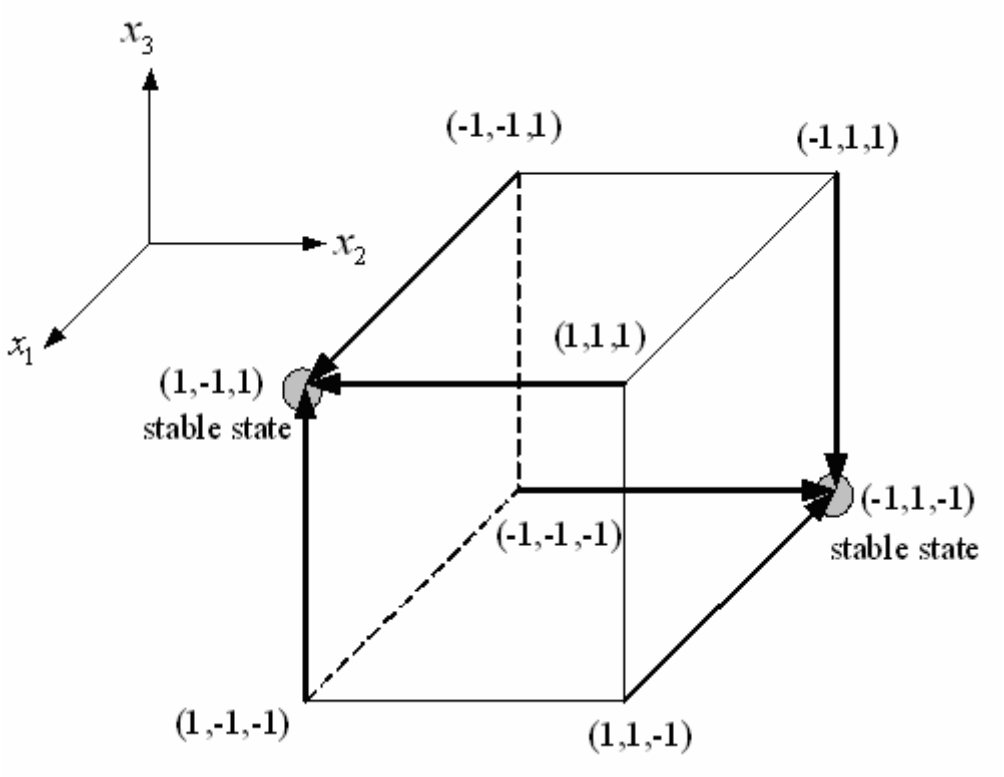


Figure 2-8. Example of the discrete Hopfield model

In Figure 2-7, the left upper corner $\xi_1 = [1 \quad -1 \quad 1]'$ and the right down corner $\xi_2 = [-1 \quad 1 \quad -1]'$ are the two fundamental memories. The nearby unstable wrong states like $[1 \quad 1 \quad 1]'$ or $[1 \quad 1 \quad -1]'$ will converge to the two stable states.

CHAPTER 3

Training Algorithm of Continuous Hopfield Neural Network for Control Applications

We have discussed how to find the weighting factors of discrete Hopfield neural network (NN) in Chapter 2. In this chapter, the training algorithm to find the weighting factors of continuous Hopfield NN will be proposed. The back-propagation skill is mainly adopted for this problem. Further the optimal training algorithm is also proposed to accelerate the learning speed. We assume the following unity feedback Single Input Single Output (SISO) control system using Hopfield NN as a controller:

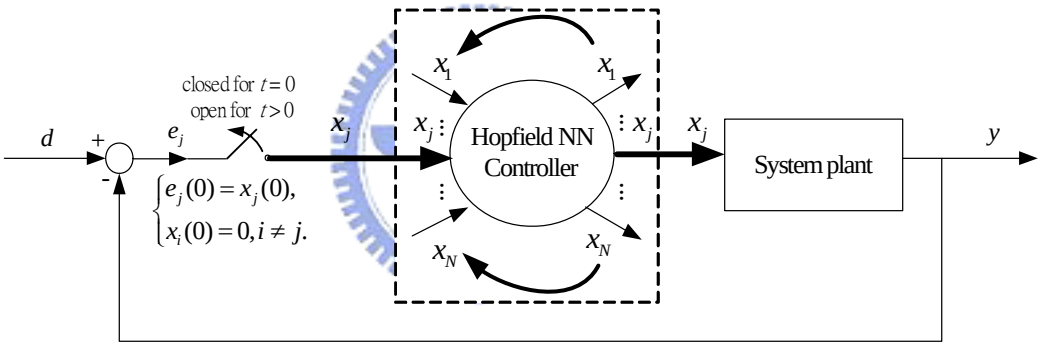


Figure 3-1. The Hopfield NN as a real-time SISO controller

The control goal is to design a Hopfield NN as a controller so that the output y will follow the input d. The design process will include the training of weighting factors in Hopfield NN. Note that it can be shown in Figure 3-1 that the output x_j of the Hopfield NN is the control input to the system plant and there is a switch placed between the error signal $e_j = d - y$ and the input x_j of Hopfield NN. This implies that $\{ x_j(0) = e_j(0)$, $x_i(0) = 0$ for $i \neq j \}$. The reason for this initial arrangement is that Hopfield NN will need non-zero initializations for

all x_i to serve as a real-time controller. This will be explained in more details in later section.

3.1 Back-propagation of continuous Hopfield neural network

The back-propagation learning algorithm is one of the most important historical developments in neural networks [14, 15]. The back-propagation learning algorithm is originally applied to multilayer feed-forward networks consisting of processing elements with continuous differentiable activation functions. Given a training set of input-output pairs $\{d(n), y(n)\}$, the algorithm provides a procedure for the finding of weighting factors in back-propagation network. Now, we use the popular back-propagation algorithm in the continuous Hopfield NN to find the way for the training weighting factors. Let the output of neuron j (i.e., x_j) be the control input to the plant (see Figure 3-1). We define the error signal $e_j(n)$ of the output neuron j at step n as the difference between $y(n)$ and $d(n)$:

$$e_j(n) = d(n) - y(n) \tag{3-1}$$

The error signal $e_j(n)$ can be used to decide the cost function. The cost function is a measure of learning performance which can be defined as

$$J = \frac{1}{2} e_j^{\,2}(n) = \frac{1}{2}\{d(n) - y(n)\}^2 \tag{3-2}$$

To minimize the cost function J, the update of the weighting factors can be obtained from the following rule:

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji} \tag{3-3}$$

where $\Delta w_{ji}(n)$ is defined by:

$$\Delta w_{ji} = -\eta \frac{\partial J}{\partial w_{ji}} \tag{3-4}$$

The parameter η is the learning rate or step size of our continuous Hopfield NN back-propagation algorithm. In (3-3) and (3-4), we can find that the back-propagation algorithm applies a correction $\Delta w_{ji}(n)$ to the synaptic weighting factor $w_{ji}(n)$, which is proportional to the partial derivative $\frac{\partial J}{\partial w_{ji}}$. Thus, we have the following update rule:

$$w_{ji}(n+1) = w_{ji}(n) - \eta \frac{\partial J}{\partial w_{ji}} \tag{3-5}$$

According to the chain rule, we can get the gradient of $\frac{\partial J}{\partial w_{ji}}$ in the following form:

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial w_{ji}} \tag{3-6}$$

Then (3-5) can be further expressed as

$$w_{ji}(n+1) = w_{ji}(n) - \eta \frac{\partial J}{\partial y}\frac{\partial y}{\partial w_{ji}} \tag{3-7}$$

We can have

$$\frac{\partial J}{\partial y} = \frac{\partial}{\partial y}\left[\frac{1}{2}(d-y)^2\right] = -(d-y) \tag{3-8}$$

Since we can not find the relation between y and w_{ji} directly, we have to apply the chain rule again for $\frac{\partial y_j}{\partial w_{ji}}$ as follows:

$$\frac{\partial y}{\partial w_{ji}} = \frac{\partial y}{\partial x_j}\frac{\partial x_j}{\partial v_j}\frac{\partial v_j}{\partial w_{ji}} \tag{3-9}$$

This implies that (3-7) can be further expressed as

$$w_{ji}(n+1) = w_{ji}(n) - \eta \frac{\partial J}{\partial y}\frac{\partial y}{\partial x_j}\frac{\partial x_j}{\partial v_j}\frac{\partial v_j}{\partial w_{ji}} \tag{3-10}$$

In (3-9), x_j is the continuous Hopfield NN output, and it is also the control signal to control the system plant. The signal y is just the system plant output due to the control signal x_j. Because the system plant is uncertain, the first part $\frac{\partial y}{\partial x_j}$ in (3-9) can be approximated by $\frac{\Delta y}{\Delta x_j}$ as follows:

$$\frac{\partial y}{\partial x_j} \cong \frac{\Delta y}{\Delta x_j} = \frac{y(n+1)-y(n)}{x_j(n+1)-x_j(n)} \tag{3-11}$$

For $\frac{\partial x_j}{\partial v_j}$ and $\frac{\partial v_j}{\partial w_{ji}}$, (2-1) and (2-4) in Chapter 2 are repeated as follows as (3-12) and (3-13):

$$x_j = \varphi(v_j) = \tanh\left(\frac{a_j v_j}{2}\right) = \frac{1-\exp(-a_j v_j)}{1+\exp(-a_j v_j)} \tag{3-12}$$

$$C\frac{d}{dt}v_j + \frac{v_j}{R} = \sum_{i=1}^{N} w_{ji}x_i \tag{3-13}$$

From (3-12), it is obvious that

$$\frac{\partial x_j}{\partial v_j} = \varphi'(v_j) = \frac{2a_j \exp(-a_j v_j)}{\left(1+\exp(-a_j v_j)\right)^2} \tag{3-14}$$

For $\frac{\partial v_j}{\partial w_{ji}}$ in (3-9), we may obtain some hints from (3-13), which is the continuous Hopfield NN dynamic equation. From (3-12), we can rewrite (3-13) as:

$$C\frac{dv_j}{dt} + \frac{v_j}{R} = \sum_{i=1}^{n} w_{ji}\varphi(v_i) \tag{3-15}$$

The above (3-15) is a nonlinear differential equation, which describes the relationship between v_j and w_{ji}. Thus $\frac{\partial v_j}{\partial w_{ji}}$ can not be found in (3-15) directly.

However, the basic theme of all training algorithms is to force the system state to have minimum energy. Theorem 1 in Chapter 2 shows the weighting vector to get the minimum energy of a discrete Hopfield NN. This minimum energy weighting vector is then applied for the training of discrete Hopfield NN. The same philosophy can also be applied in the training of continuous Hopfield NN. From Chapter 2, we have the derivate of energy function $E(t)$ for continuous Hopfield NN as

$$\frac{dE}{dt} = -\sum_{j=1}^{N} C\left(\frac{dv_j}{dt}\right)\frac{dx_j}{dt} \tag{3-16}$$

For minimum energy, we must let $\frac{dE}{dt} = 0$. From (3-16), it is obvious that if $\frac{dv_j}{dt} = 0$ for

$j = 1, ..., N$; then $\frac{dE}{dt} = 0$. Based on this, (3-15) can be rewritten as

$$\frac{v_j}{R} = \sum_{i=1}^{n} w_{ji} \varphi(v_i) \tag{3-17}$$

Therefore we can have

$$v_j = R \cdot \sum_{i=1}^{n} w_{ji} \varphi(v_i) \tag{3-18}$$

Up to this stage, we can find $\frac{\partial v_j}{\partial w_{ji}}$ easily from the above (3-18) as follows:

$$\frac{\partial v_j}{\partial w_{ji}} = R \cdot \varphi(v_i) \tag{3-19}$$

Form (3-12), we can further express (3-19) as a more complete form:

$$\frac{\partial v_j}{\partial w_{ji}} = R \cdot \varphi(v_i) = R \cdot \frac{1 - \exp(-a_i v_i)}{1 + \exp(-a_i v_i)} \tag{3-20}$$

Now the three terms $\frac{\partial y}{\partial x_j}$, $\frac{\partial x_j}{\partial v_j}$, and $\frac{\partial v_j}{\partial w_{ji}}$ in (3-9) have been found, we can express $\frac{\partial y}{\partial w_{ji}}$ in the following form:

$$\begin{aligned} \frac{\partial y}{\partial w_{ji}} &= \frac{\partial y}{\partial x_j} \frac{\partial x_j}{\partial v_j} \frac{\partial v_j}{\partial w_{ji}} \\ &= \frac{\Delta y}{\Delta x_j} \varphi'(v_j) \cdot R\varphi(v_i) \\ &= \frac{y(n+1) - y(n)}{x_j(n+1) - x_j(n)} \cdot \frac{2a_j \exp(-a_j v_j(n))}{\left(1 + \exp(-a_j v_j(n))\right)^2} \cdot R \frac{1 - \exp(-a_i v_i(n))}{1 + \exp(-a_i v_i(n))} \end{aligned} \tag{3-21}$$

Finally we can find the $\frac{\partial J}{\partial w_{ji}}$ in (3-6) as:

$$\begin{aligned} \frac{\partial J}{\partial w_{ji}} &= \frac{\partial J}{\partial y} \frac{\partial y}{\partial w_{ji}} \\ &= \frac{\partial J}{\partial y} \frac{\partial y}{\partial x_j} \frac{\partial x_j}{\partial v_j} \frac{\partial v_j}{\partial w_{ji}} \\ &= -(d - y) \cdot \frac{y(n+1) - y(n)}{x_j(n+1) - x_j(n)} \cdot \frac{2a_j \exp(-a_j v_j(n))}{\left(1 + \exp(-a_j v_j(n))\right)^2} \cdot R \frac{1 - \exp(-a_i v_i(n))}{1 + \exp(-a_i v_i(n))} \end{aligned} \tag{3-22}$$

Therefore the final step to perform the back-propagation algorithm of the continuous Hopfield NN is to combine all the above equations to get

$$\Delta w_{ji} = -\eta \frac{\partial J}{w_{ji}}$$

$$= -\eta \left[-(d-y) \cdot \frac{y(n+1)-y(n)}{x_j(n+1)-x_j(n)} \cdot \frac{2a_j \exp\left(-a_j v_j(n)\right)}{\left(1+\exp\left(-a_j v_j(n)\right)\right)^2} \cdot R \frac{1-\exp\left(-a_i v_i(n)\right)}{1+\exp\left(-a_i v_i(n)\right)} \right] \quad (3\text{-}23)$$

$$= \eta (d-y) \cdot \frac{y(n+1)-y(n)}{x_j(n+1)-x_j(n)} \cdot \frac{2a_j \exp\left(-a_j v_j(n)\right)}{\left(1+\exp\left(-a_j v_j(n)\right)\right)^2} \cdot R \frac{1-\exp\left(-a_i v_i(n)\right)}{1+\exp\left(-a_i v_i(n)\right)}$$

The complete back propagation equation for the training of continuous Hopfield NN is therefore

$$w_{ji}(n+1) = w_{ji}(n) +$$

$$\eta\left(d(n)-y(n)\right) \cdot \frac{y(n+1)-y(n)}{x_j(n+1)-x_j(n)} \cdot \frac{2a_j \exp\left(-a_j v_j(n)\right)}{\left(1+\exp\left(-a_j v_j(n)\right)\right)^2} \cdot R \frac{1-\exp\left(-a_i v_i(n)\right)}{1+\exp\left(-a_i v_i(n)\right)} \quad (3\text{-}24)$$

Up to this stage, we still need to find the v_j for next iteration. This can be easily obtained from (3-18) (repeated here for convenience, as (3-25)), as we assume minimum energy requirement.

$$v_j(n+1) = R \cdot \sum_{i=1}^{n} w_{ji}(n+1)\varphi\left(v_i(n)\right) \quad (3\text{-}25)$$

The following algorithm summarizes the above steps to complete the training process.

Algorithm 3-1: Back-propagation learning for continuous Hopfield neural network

Step 0: Given training pair $\{ d(n), y(n), \quad n = 0,1,...,N \}$, with the following initial conditions:

1. a_j : The slope of hyperbolic tangent function
2. $x_j(0)$: Initial value of continuous Hopfield NN
3. $w_{ji}(0)$: Initial synaptic weighting factors of the continuous Hopfield NN.

Step 1: Define the error signal as (3-1).

$$e_j(n) = d(n) - y(n)$$

Step 2: Define the cost function J using the error signal as (3-2).

$$J = \frac{1}{2} e_j^2(n) = \frac{1}{2}\{d(n) - y(n)\}^2$$

Step 3: Decide the back-propagation update rule in (3-5).

$$w_{ji}(n+1) = w_{ji}(n) - \eta \frac{\partial J}{\partial w_{ji}}$$

Step 4: Separate the partial derivative $\frac{\partial J}{\partial w_{ji}}$ in (3-6).

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y} \frac{\partial y}{\partial w_{ji}}$$

Step 5: Get the result of $\frac{\partial J}{\partial y_j}$ in (3-8).

$$\frac{\partial J}{\partial y} = \frac{\partial}{\partial y}\left[\frac{1}{2}(d-y)^2\right] = -(d(n) - y(n))$$

Step 6: Separate the partial derivative term $\frac{\partial y}{\partial w_{ji}}$ in (3-9).

$$\frac{\partial y}{\partial w_{ji}} = \frac{\partial y}{\partial x_j} \frac{\partial x_j}{\partial v_j} \frac{\partial v_j}{\partial w_{ji}}$$

Step 7: Get the approximation form of $\frac{\partial y}{\partial x_j}$ in (3-11).

$$\frac{\partial y}{\partial x_j} = \frac{\Delta y}{\Delta x_j} = \frac{y(n+1) - y(n)}{x_j(n+1) - x_j(n)}$$

Step 8: Get the result of $\frac{\partial x_j}{\partial v_j}$ in (3-14).

$$\frac{\partial x_j}{\partial v_j} = \varphi'(v_j) = \frac{2a_j \exp(-a_j v_j(n))}{\left(1+\exp(-a_j v_j(n))\right)^2}$$

Step 9: Get the result of $\frac{\partial v_j}{\partial w_{ji}}$ in (3-20).

$$\frac{\partial v_j}{\partial w_{ji}} = R \cdot \varphi(v_i) = R \cdot \frac{1-\exp(-a_i v_i(n))}{1+\exp(-a_i v_i(n))}$$

Step 10: Define the correction $\Delta w_{ji}(n)$ as (3-23).

$$\Delta w_{ji} = -\eta \frac{\partial J}{w_{ji}} =$$

$$\eta\left(d(n)-y(n)\right) \cdot \frac{y(n+1)-y(n)}{x_j(n+1)-x_j(n)} \cdot \frac{2a_j \exp(-a_j v_j(n))}{\left(1+\exp(-a_j v_j(n))\right)^2} \cdot R \frac{1-\exp(-a_i v_i(n))}{1+\exp(-a_i v_i(n))}$$

Step 11: Complete the back-propagation in (3-24).

$$w_{ji}(n+1) =$$

$$w_{ji}(n) + \eta\left(d(n)-y(n)\right) \cdot \frac{y(n+1)-y(n)}{x_j(n+1)-x_j(n)} \cdot \frac{2a_j \exp(-a_j v_j(n))}{\left(1+\exp(-a_j v_j(n))\right)^2} \cdot R \frac{1-\exp(-a_i v_i(n))}{1+\exp(-a_i v_i(n))}$$

Step 12: Solve the differential equation with minimum energy condition:

$$v_j(n+1) = R \cdot \sum_{i=1}^{n} w_{ji}(n+1) x_i(n)$$

Step 13: $n = n+1$, if $n \neq N$, GOTO Step 1.

Step 14: **Stop.**

3.2 Optimum learning rate for the training of continuous Hopfield NN

In Section 3.1, we have developed the back-propagation algorithm for the training of weighting factors in continuous Hopfield NN. The back-propagation algorithm will train the continuous Hopfield NN to generate the control signal to control the plant such that the output signal of the plant will be as close as possible to the desired input signal. But the training speed is very slow in Algorithm 3.1 with a arbitrary fixed learning rate. It is better if we can find a better dynamic learning rate in each iteration so that the convergent speed can be increased [12, 13].

It is very important to choose an appropriate learning rate (or step size) in the training process. If the learning rate is too large, the learning process will jump to the next iteration in a larger step and this may cause divergence in the training process. If the learning is too small, the learning process will be slow, and yet the convergence is not guaranteed. Thus, how to choose a suitable learning rate is important. With the concept of dynamic optimal training in [12, 13], we will develop a similar optimal training algorithm for the training of Hopfield NN. This process is to decide the learning rate η in each iteration so that the error energy will be reduced as much as possible. This will not yield a stable training process, but the convergence speed is also the fastest. The error energy function (cost function) at n step has been defined in (3-2) and is listed here as (3-32):

$$J = \frac{1}{2}\left\{d(n) - y(n)\right\}^2 \tag{3-26}$$

The cost function in the next step $(n+1)$ is:

$$J(n+1) = \frac{1}{2}\left\{d(n+1) - y(n+1)\right\}^2 \tag{3-27}$$

Now, it is our purpose to find an optimum learning rate η_{opt} such that $J(n+1) - J(n) \le 0$ and $\left|J(n+1) - J(n)\right|$ is maximized. Suppose the function $G(\cdot)$ represents an input dependent function so that we can have the output $y = G(x_j)$. This implies $G(\cdot)$ may be a

linear transfer function, a nonlinear differential equation, or even a look-up table. Thus we can get the following equation:

$$\begin{aligned} & J(n+1)-J(n) \\ &= \frac{1}{2}\left\{d(n+1)-y(n+1)\right\}^2-\frac{1}{2}\left[d(n)-y(n)\right]^2 \\ &= \frac{1}{2}\left\{d(n+1)-G\left[x_j(n+1)\right]\right\}^2-\frac{1}{2}\left[d(n)-y(n)\right]^2 \\ &= \frac{1}{2}\left\{d(n+1)-G\left[\varphi\left(v_j(n+1)\right)\right]\right\}^2-\frac{1}{2}\left[d(n)-y(n)\right]^2 \\ &= \frac{1}{2}\left\{d(n+1)-G\left[\varphi\left(R\cdot\sum_{i=1}^{N}w_{ji}(n+1)x_i\right)\right]\right\}^2-\frac{1}{2}\left[d(n)-y(n)\right]^2 \end{aligned} \tag{3-28}$$

We have already known that $v_j(n+1)$ is a function of the weighting factors $w_{ji}(n+1)$. So we know that (3-28) is also a function of the uncertain weighting factors $w_{ji}(n+1)$. Further we may rewrite the update of weighting factors in Step 10 of **Algorithm 3-1** as follows:

$$\begin{aligned} & \Delta w_{ji}(n)= \\ & \eta_{opt}\cdot\left(d(n)-y(n)\right)\cdot\frac{y(n+1)-y(n)}{x_j(n+1)-x_j(n)}\cdot\frac{2a_j\exp\left(-a_jv_j(n)\right)}{\left(1+\exp\left(-a_jv_j(n)\right)\right)^2}\cdot R\frac{1-\exp\left(-a_iv_i(n)\right)}{1+\exp\left(-a_iv_i(n)\right)} \end{aligned} \tag{3-29}$$

Where η_{opt} is the dynamical optimal learning rate to be decided. Thus, we have the uncertain correction $\Delta w_{ji}(n)$ with an uncertain learning rate η_{opt}. According to (3-24), we also get the uncertain new weighting factors $w_{ji}(n+1)$. As we have discussed, $J(n+1)-J(n)$ is a function of the uncertain weighting factors $w_{ji}(n+1)$. Combine all the above relations, we may express $J(n+1)-J(n)$ as a function of the uncertain variable η_{opt}. It means that what we have to do is to find an optimum value for the variable η_{opt} such that $J(n+1)-J(n)\le 0$ and $\left|J(n+1)-J(n)\right|$ is maximized. Now, we know that $J(n+1)-J(n)$ is a function of uncertain variable η_{opt}.

$$J(n+1)-J(n)=H(\eta_{opt}) \tag{3-30}$$

In other neural networks, the function $H(\cdot)$ may be a simple parabolic curve, thus we can find η_{opt} easily. But in this continuous Hopfield NN optimum learning rate algorithm, the function $H(\cdot)$ is very complicated, we can apply the Matlab routine ***fminbnd*** [13] to find the optimal learning rate η_{opt} from $J(n+1)-J(n)$. After using Matlab routine to find η_{opt}, we have finished the process of optimum learning rate. At the end of this section, we combine the important result in Section 3.1 and 3.2 to perform the training algorithm in continuous Hopfield NN.

Algorithm 3-2: Back-propagation learning of continuous Hopfield network with optimal learning rate

Step 0: Given training pair $\{d(n), y(n), n=0,1,...,N\}$, with the following initial conditions:

1. a_j: The slope of hyperbolic tangent function
2. $x_j(0)$: Initial value of continuous Hopfield NN
3. $w_{ji}(0)$: Initial synaptic weighting factors of the continuous Hopfield NN.

Step 1: Use the hyperbolic tangent function $x_j(n)=\varphi(v_j(n))=\dfrac{1-\exp(-a_j v_j(n))}{1+\exp(-a_j v_j(n))}$ to get initial Hopfield controller output $x_j(n)$.

Step 2: Use control signal $x_j(n)$ to control system plant $G(\cdot)$ and get output $y(n)$.

Step 3: $J(n+1)-J(n)$ is the function of η_{opt}, $J(n+1)-J(n)=H(\eta_{opt})$. Apply the

Matlab routine ***fminbnd*** to find the optimal learning rate η_{opt}.

Step 4: Use the **Algorithm 3-1** to get the new weighting factors:

$$\begin{cases} \Delta w_{ji} = \eta_{opt} \cdot \left(d(n) - y(n)\right) \cdot \dfrac{y(n+1) - y(n)}{x(n+1) - x(n)} \cdot \dfrac{2a_j \exp\left(-a_j v_j(n)\right)}{\left(1 + \exp\left(-a_j v_j(n)\right)\right)^2} \cdot R \dfrac{1 - \exp\left(-a_i v_i(n)\right)}{1 + \exp\left(-a_i v_i(n)\right)} \\ w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji} \end{cases}$$

Step 5: Use the new weighting factors to get the solution of the Hopfield NN differential equation with minimum energy requirement:

$$v_j(n+1) = R \cdot \sum_{i=1}^{n} w_{ji}(n+1) x_i(n)$$

Step 6: $n = n+1$, if $n \neq N$, GOTO Step 1.

Step 7: **Stop.**

3.3 Hopfield NN as a Real-Time Controller

The above mentioned back propagation training for Hopfield NN can only be performed in a batch off-line processing. Once the training process is completed, the Hopfield must be connected in a real-time environment to serve as a real-time controller. However, we know that the Hopfield NN is a recurrent dynamic system. It has dynamic response from initial excitations. The Hopfield NN dynamic equation is in (3-13) (repeated here for convenience as (3-31)),

$$C\frac{d}{dt}v_j + \frac{v_j}{R} = \sum_{i=1}^{N} w_{ji} x_i \qquad (3\text{-}31)$$

If the Hopfield NN is with zero initial conditions, i.e., $x_i(0) = 0$ for all i, then it is obvious that Hopfield NN will not have any response at all for $t > 0$. Therefore we need to set a non-zero initial condition for the control configuration in Figure 3-1. In Figure 3-1, x_j is connected to e_j at $t = 0$, and disconnected for $t > 0$, where x_j is also the control input to

the plant. This implies that $\{x_j(0)=e_j(0),\ x_i(0)=0 \text{ for } i\neq j\}$.To see the effect of this initial configuration, we have the following Theorem 2.

Theorem 2:

If w_{jj} is negative, then the initial excitation to the Hopfield NN in Figure 3-1 will have a stable response in terms of the maximum decay of $v_j(t)$ with respect to $e_j(0)$.

Proof:

The following (3-32) is the solution of (3-31):

$$v_j = R\cdot\sum_{i=1}^{N} w_{ji}x_i + \left(v_j(0) - R\cdot\sum_{i=1}^{N} w_{ji}x_i\right)\cdot e^{\frac{-1}{RC}t} \tag{3-32}$$

Further, substituting $v_j(0)=\sum_{i=1}^{N} w_{ji}x_i(0)$ into (3-32), we have

$$v_j = R\cdot\sum_{i=1}^{N} w_{ji}x_i + \left(\sum_{i=1}^{N} w_{ji}x_i(0) - R\cdot\sum_{i=1}^{N} w_{ji}x_i\right)\cdot e^{\frac{-1}{RC}t} \tag{3-33}$$

Since the output of Hopfield NN controller is a particular neuron output x_j. Thus, we can rewrite (3-33) as:

$$v_j = R\cdot\sum_{i=1}^{N} w_{ji}x_i + \left(w_{jj}x_j(0) + \sum_{\substack{i=1\\ i\neq j}}^{N} w_{ji}x_i(0) - R\cdot\sum_{i=1}^{N} w_{ji}x_i\right)\cdot e^{\frac{-1}{RC}t} \tag{3-34}$$

Now, the initial excitations of the Hopfield NN are $x_j(0)=e_j$ and $x_i(0)=0$ for $i\neq j$. Therefore (3-34) becomes

$$v_j = R\cdot\sum_{i=1}^{N} w_{ji}x_i + \left(w_{jj}e_j - R\cdot\sum_{i=1}^{N} w_{ji}x_i\right)\cdot e^{\frac{-1}{RC}t} \tag{3-35}$$

Eq. (3-35) shows that v_j is a function of e_j. Differentiating (3-35) with respect to e_j, we can get:

$$\frac{dv_j}{de_j} = e^{\frac{-1}{RC}t}\cdot w_{jj} \tag{3-36}$$

The first term $e^{\frac{-1}{RC}t}$ in (3-36) is always positive, and the second term w_{jj} is the self-feedback of Hopfield neural network. In order for the function v_j to decay with increasing e_j, w_{jj} needs to be a negative number. Thus, if w_{jj} is a negative number, then v_j in (3-35) is a decreasing function of e_j. It means that we should choose a large e_j to get the fastest response of v_j. Ii is obvious that the initial value $e_j = e_j(0)$ is the largest, since the initial output is always zero.

Q.E.D.

CHAPTER 4

Experimental Results

In this chapter, we use the algorithm introduced in Chapter 3 to control the inverted pendulum system in Section 4.1, and the aircraft attitude control in Section 4.2.

4.1 Example 1: The Inverted Pendulum System (IPS)

Inverted pendulum is a nonlinear and unstable system, and it is the most popular benchmark for new control algorithms to verify their advantages [16].

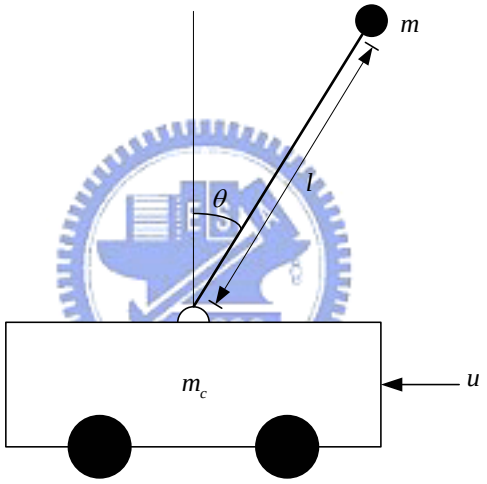


Figure 4-1. The inverted pendulum system

According to Figure 4-1, we let the angle θ which is between the pendulum and the vertical line be equal to the first state θ_1, and the angular speed $\dot{\theta}$ be the second state θ_2. Then we can get the following state equation:

$$\begin{aligned}\dot{\theta}_1 &= \theta_2 \\ \dot{\theta}_2 &= \frac{g\sin\theta_1 - \dfrac{ml\theta_2^2\cos\theta_1\sin\theta_1}{m_c+m}}{l\left(\dfrac{4}{3}-\dfrac{m\cos^2\theta_1}{m_c+m}\right)} + \frac{\dfrac{\cos\theta_1}{m_c+m}}{l\left(\dfrac{4}{3}-\dfrac{m\cos^2\theta_1}{m_c+m}\right)}u\end{aligned} \tag{4-1}$$

Where

g : *The constant of acceleration of gravity*

m : *The mass attached to the pendulum*

m_c : *The mass of based car*

l : *The length of pendulum arm*

θ_1 : *Angle between pendulum and vertical line*

θ_2 : *Angular speed*

u : *Control signal*

In our example, the parameters of the inverted pendulum are defined in the following values:

$$g = 9.8\ m/\sec^2; \qquad m = 0.1\ kg$$
$$m_c = 1\ kg; \qquad l = 0.5\ m$$

Thus, we can further express (4-1) as

$$\dot{\theta}_1 = \theta_2$$
$$\dot{\theta}_2 = \frac{215.6\sin\theta_1 - x_2^2\cos\theta_1\sin\theta_1}{14.667 - \cos^2\theta_1} + \frac{2\cos\theta_1}{14.667 - \cos^2\theta_1}u \qquad (4\text{-}2)$$

The control objective is to stabilize the inverted pendulum with initial angle between 10° and -10°. Note that the state equations in (4-2) are only used for simulation purpose. It is by no means used to design the Hopfield NN controller. For training data set, we generate 6 data set which correspond to six initial angles, i.e., {10°, 7°, 3°, -3°, -7°, -10°}. The six data sets are shown in Figure 4-2.

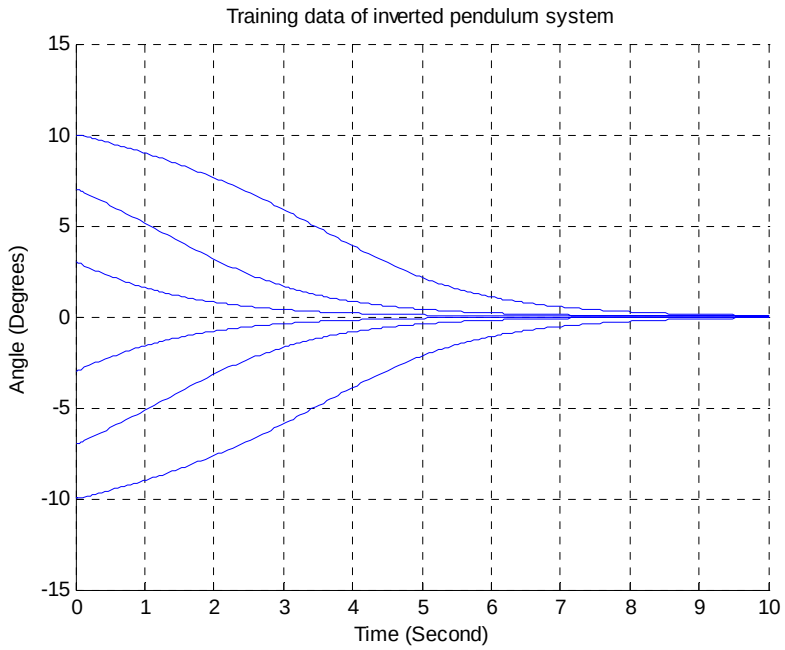


Figure 4-2. The six training curves for the control of IPS using Hopfield NN

The overall real control architecture of Hopfield NN controller is shown in Figure 4-3. The Hopfield NN in this application has two neurons and the first neuron output x_1 is the control signal.

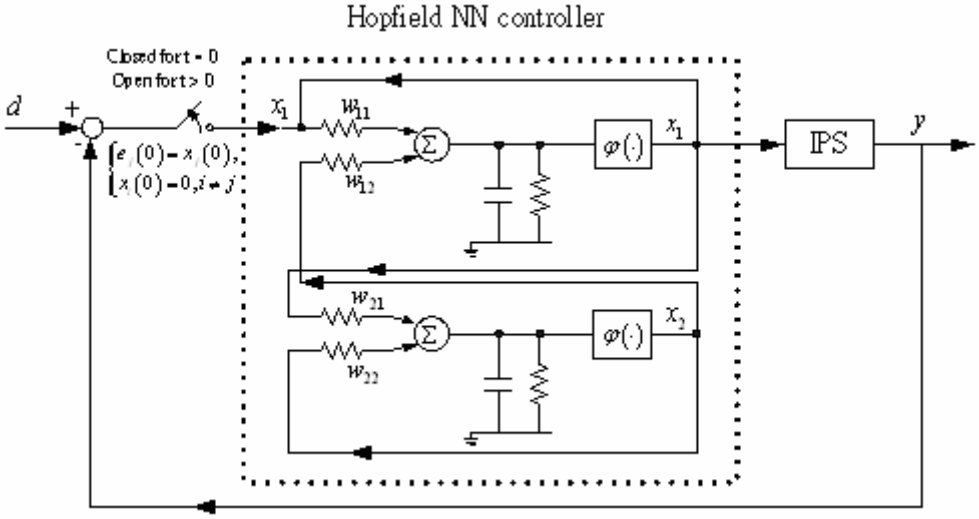


Figure 4-3. Real-time SISO control architecture of Hopfield NN controller

After training by the Algorithm 3-2 developed in Chapter 3, the trained weighting factors in Hopfield NN can be used to generate control signal. The values of capacitor and resistance are

10 uF and 1K Ohms respectively. The final four weighting factors in this application are shown in the following Table 4-1. Table 4-1 also shows the resistance values for the weighting factors.

Weighting factor	w_{11}	w_{12}	w_{21}	w_{22}
Trained value	-3.4275	-0.012878	-2.1702	0.2845
| Resistance |	0.2918Ω	77.6518Ω	0.4608Ω	3.5149Ω

Table 4-1. Trained weighting factors of IPS

The first weighting factor w_{11} is negative. It is the fact from **Theorem 2** discussed in Chapter 3. Three cases with different initial angles and different initial angular speed will be adopted to test the Hopfield NN controller. There are two figures in each of those cases. The first one is about the final angle output θ_1 in actual real time control case, which is a stable situation in this inverted pendulum system. The second figure is the control signal generated from the trained Hopfield NN.

Case 1. Initial angle $\theta_1(0)=10^{\circ}$, initial angular speed $\theta_2(0)=0(^{\circ}/\sec)$

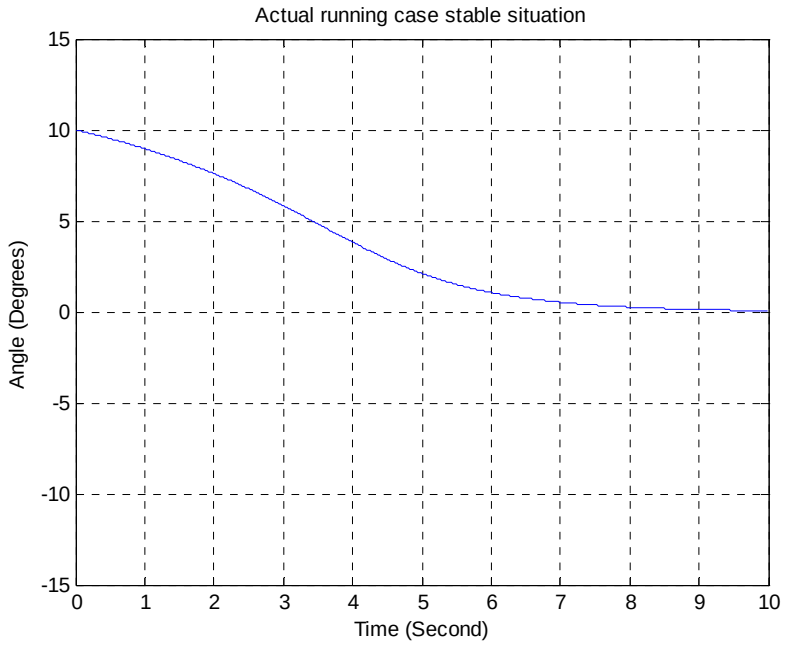


Figure 4-4. The actual running stable situation for Case 1

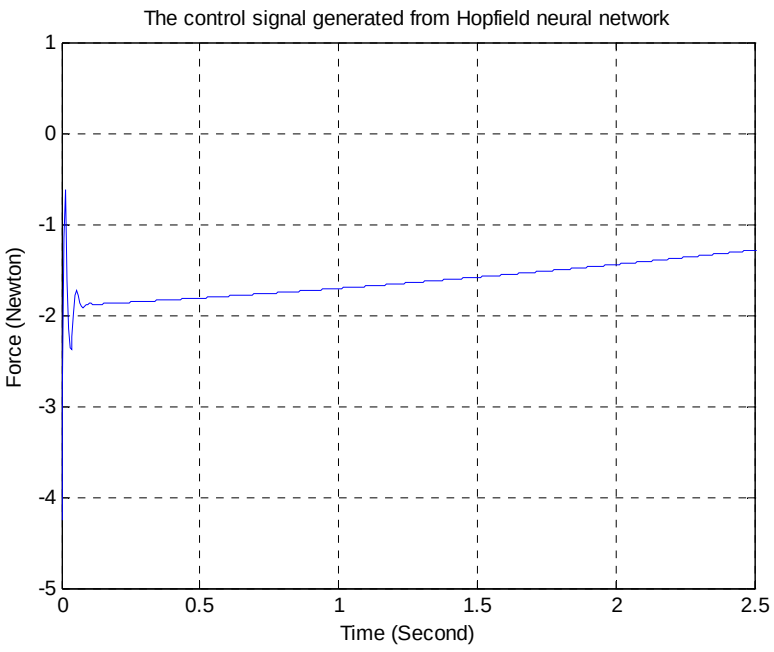


Figure 4-5. The control signal of the inverted pendulum system for Case 1

Case 2. Initial angle $\theta_1(0) = -8^\circ$, initial angular speed $\theta_2(0) = 4(^\circ/\sec)$

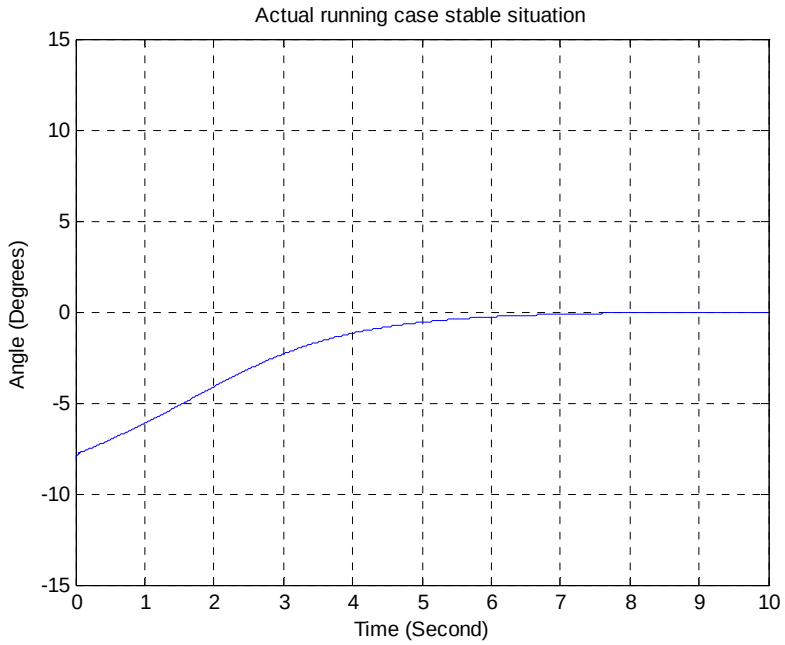


Figure 4-6. The actual running stable situation for Case 2

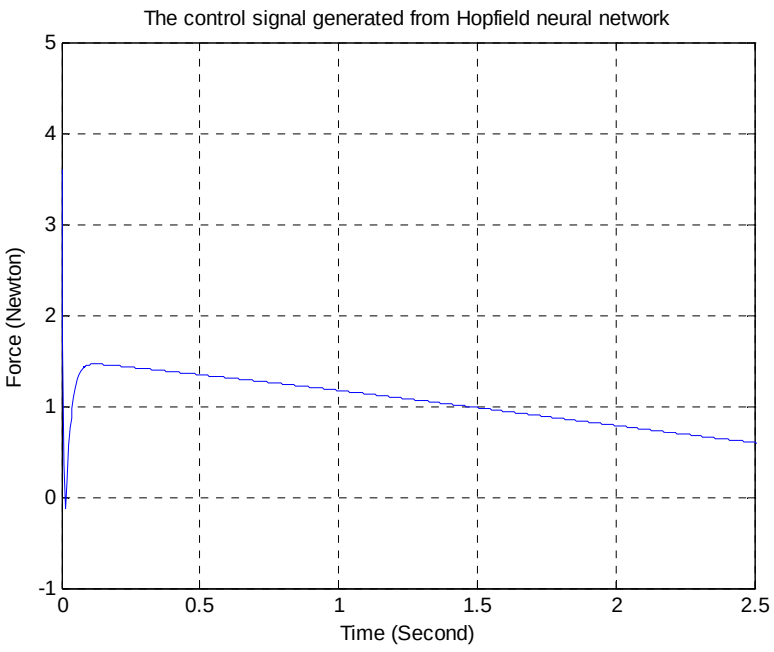


Figure 4-7. The control signal of the inverted pendulum system for Case 2.

Case 3. Initial angle $\theta_1(0) = -5^\circ$, initial angular speed $\theta_2(0) = -3(^\circ/\sec)$

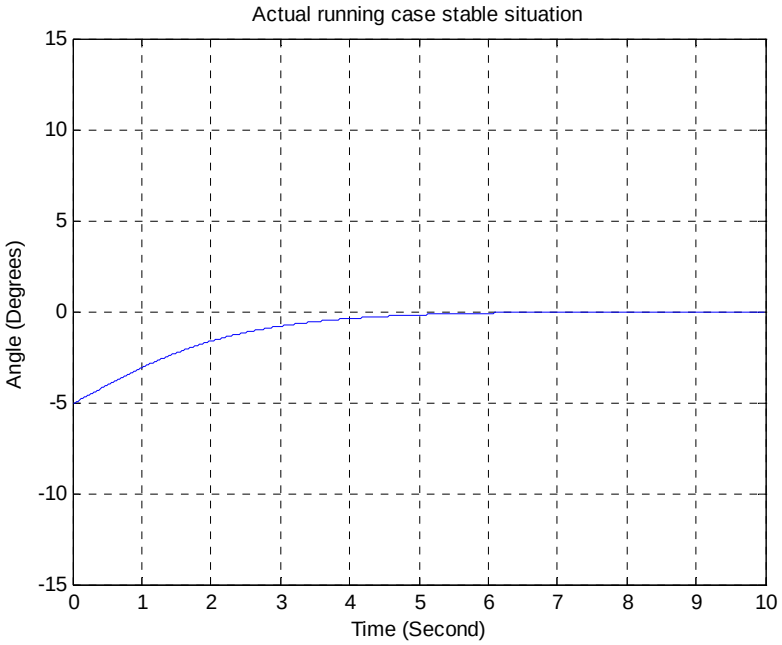


Figure 4-8. The actual running stable situation for Case 3

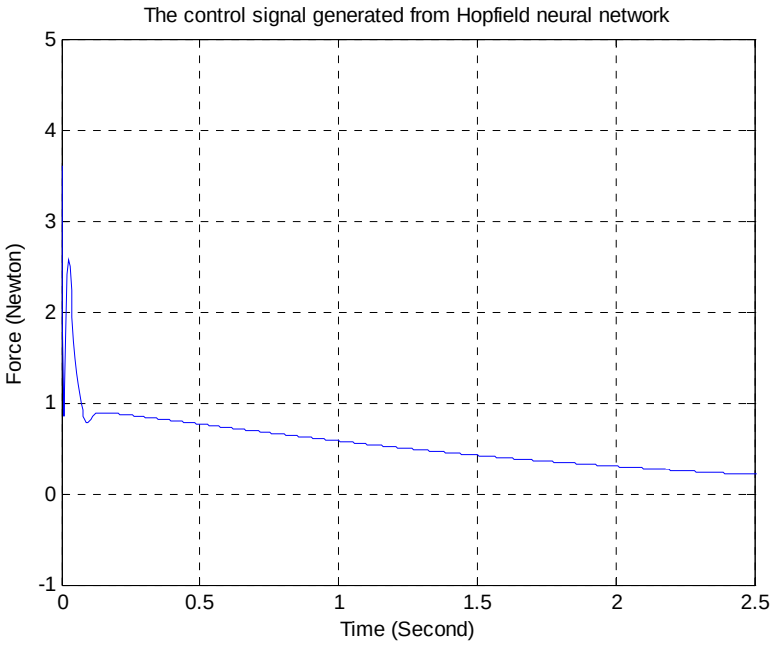


Figure 4-9. The control signal of the inverted pendulum system for Case 3

In above three cases, we can find that the system output angles θ will be stable in our desired range. It means that the algorithm discussed in Chapter 3 is useful to control the nonlinear inverted pendulum system.

4.2 Example 2: The aircraft attitude control system

The aircraft attitude control system is different from the inverted pendulum system. It is a simplified linear system [17]. The control surfaces of modern aircraft are controlled by electric actuators with electronics controls. Figure 4-10 illustrates the control block diagram of one axis of such a position control system.

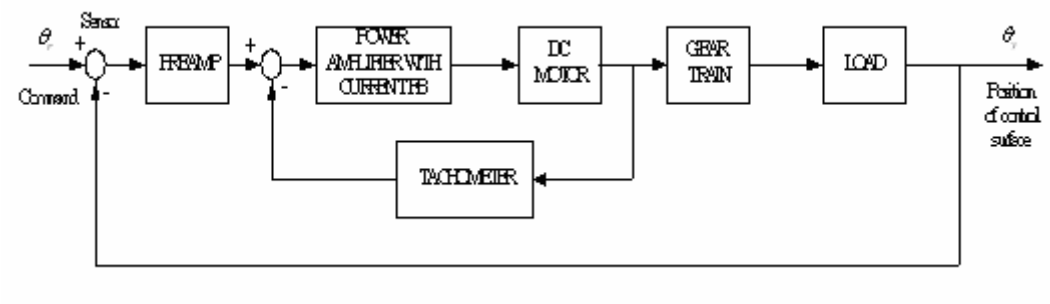


Figure 4-10. Block diagram of an attitude-control system of an aircraft

The objective of the system is to have the output of the system, $\theta_y(t)$, follow the unit step input $\theta_r(t)$. The transfer function is defined as

$$G(s)=\frac{\theta_y(s)}{\theta_r(s)}=\frac{4500K}{s(s+361.2)} \tag{4-3}$$

Where K is gain parameter of preamplifier. When we have introduced the aircraft attitude control system, we must set the specifications as follows:

1. Steady state error due to unit ramp input ≤ 0.000443
2. Maximum overshoot ≤ 5 percent
3. Rise time ≤ 0.005 sec

4. Settling time ≤ 0.007 sec

Now, we will design the Hopfield NN controller for this problem so that the above specifications can be satisfied. The first specification is different from the other specification, so we discuss it first. It is the specification about steady state error due to unit ramp input. It must be satisfied by using preamplifier K. The system transfer function in (4-3) has a s term in denominator, so it is a type 1 system. For type 1 system, the steady state error due to unit step input is zero, but it is a constant for unit ramp input. Thus we must choose a proper gain of preamplifier K to reach the first specification. Now, we compute the velocity error constant k_v of transfer function in (4-3):

$$k_v = \lim_{s\to 0} sG(s) = \lim_{s\to 0} s\frac{4500K}{s(s+361.2)} = 12.45847K \tag{4-4}$$

As we get the velocity error constant k_v, we can compute the steady state error due to unit ramp input $e_{ss}(ramp)$:

$$e_{ss}(ramp) = \frac{1}{k_v} = \frac{1}{12.45847K} = 0.08026\frac{1}{K} \tag{4-5}$$

According to the first specification, we should let the result of (4-5) be smaller than 0.000443. Thus we calculate the minimum value of gain of preamplifier *K*:

$$0.08026\frac{1}{K} \le 0.000443 \Rightarrow K \ge 181.17 \tag{4-6}$$

In order to reach the first specification, we choose the gain of preamplifier K equal to 181.17. Thus the transfer function $G(s)$ becomes to

$$G(s) = \frac{815265}{s(s+361.2)} \tag{4-7}$$

Now we hope the output of the system can reach the specifications in actual real time control case. The overall real control architecture of Hopfield NN controller is shown in Figure 4-3. The Hopfield NN in this application has two neurons and the first neuron output x_1 is the control signal.

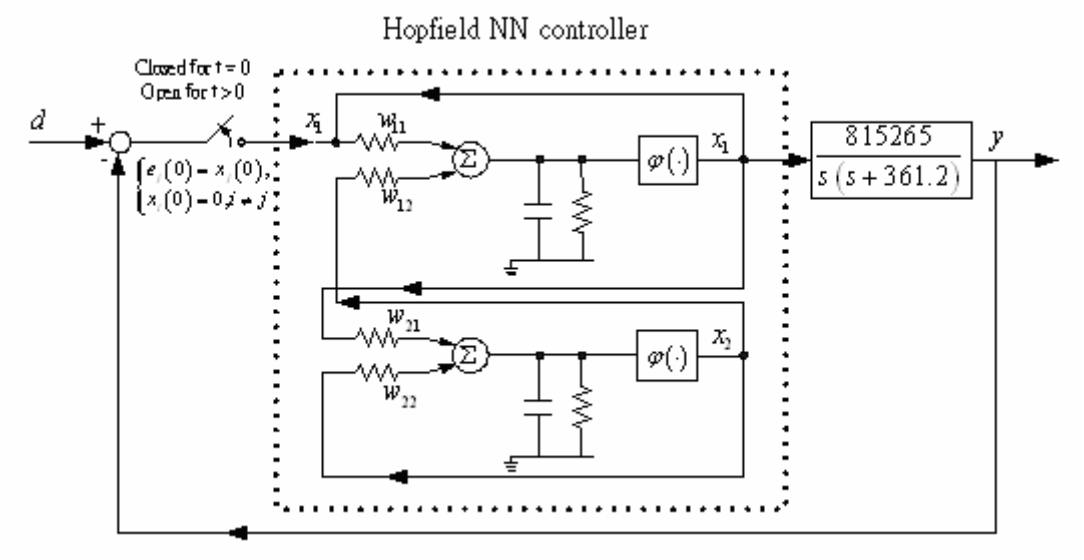


Figure 4-11. Real control architecture of Hopfield NN controller

For training data set, we can generate time response curve as shown in Figure 4-12 so that the above 2, 3, 4 specifications. That is, maximum overshoot = 0.91 percent, rise time = 0.00284 sec and settling time = 0.00424 sec in Figure 4-12.

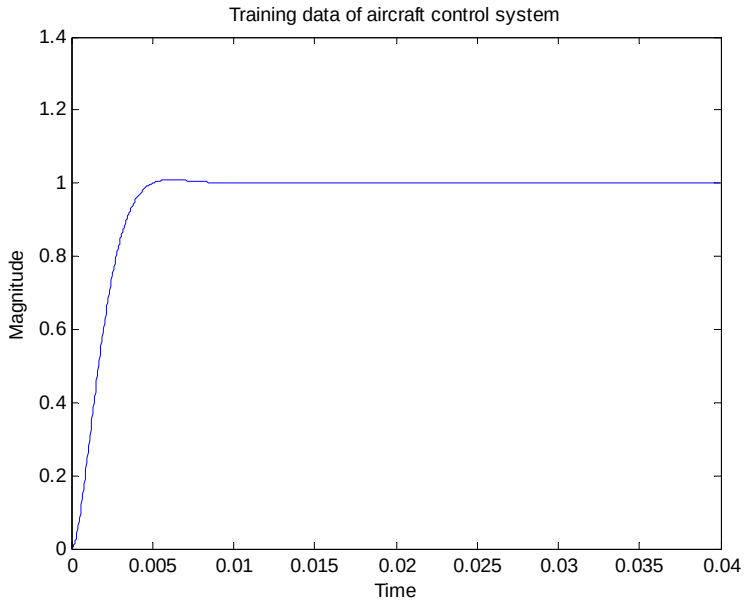


Figure 4-12. The training data of the aircraft control system

After training by Algorithm 3-2 developed in Chapter 3, the trained weighting factors in

Hopfield NN can be used to generate control signal. The final four weighting factors in this application are:

Weighting factor	w_{11}	w_{12}	w_{21}	w_{22}
Trained value	-47.552	-0.000032	-44.98	-0.048
| Resistance |	0.021Ω	31250Ω	0.0222Ω	20.833Ω

Table 4-2. Trained weighting factors of aircraft system

The following Figure 4-13 is the output when the Hopfield NN is applied to the closed-loop control of the attitude control of aircraft.

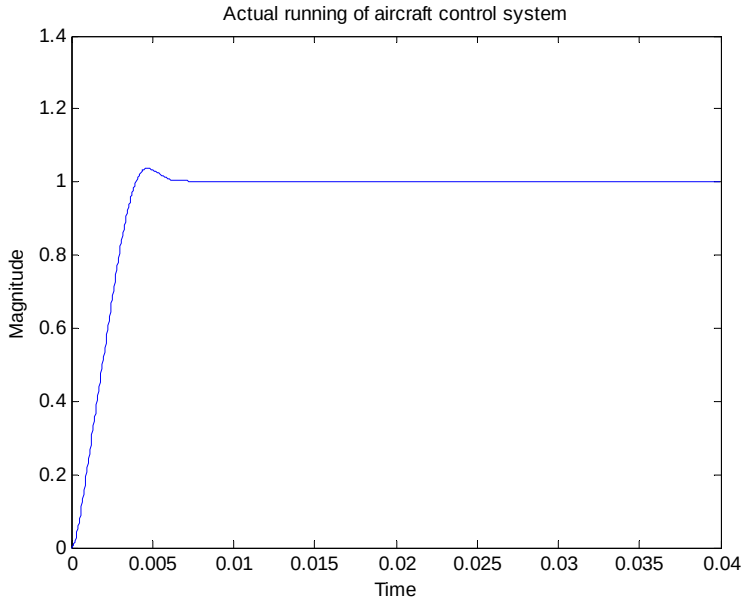


Figure 4-13. The output of the aircraft control system by Hopfield NN controller

Figure 4-14 shows the response of Figure 4-13 in transient period (0 ~ 0.01 sec). It is obvious from Figure 4-14 that maximum overshoot is 3.77% ≤ 5 percent, rise time is 0.0028sec ≤ 0.005 sec and settling time is 0.0052 sec ≤ 0.007 sec. These data shows that the Hopfield NN controller indeed can be applied as a controller in this linear control system.

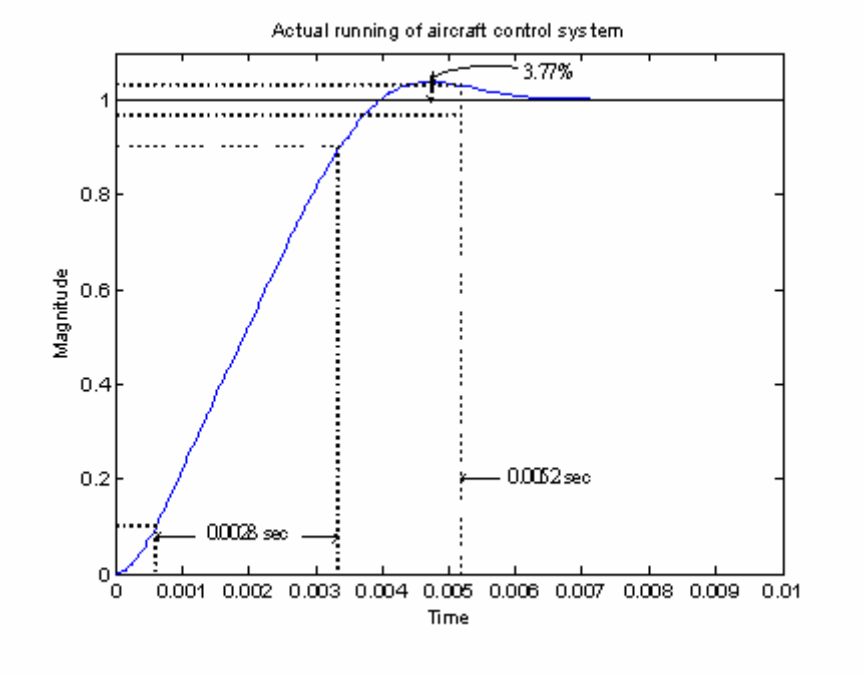


Figure 4-14. The plot of Figure 4-13 in transient period

Figure 4-15 shows the control signal generated from the Hopfield NN controller.

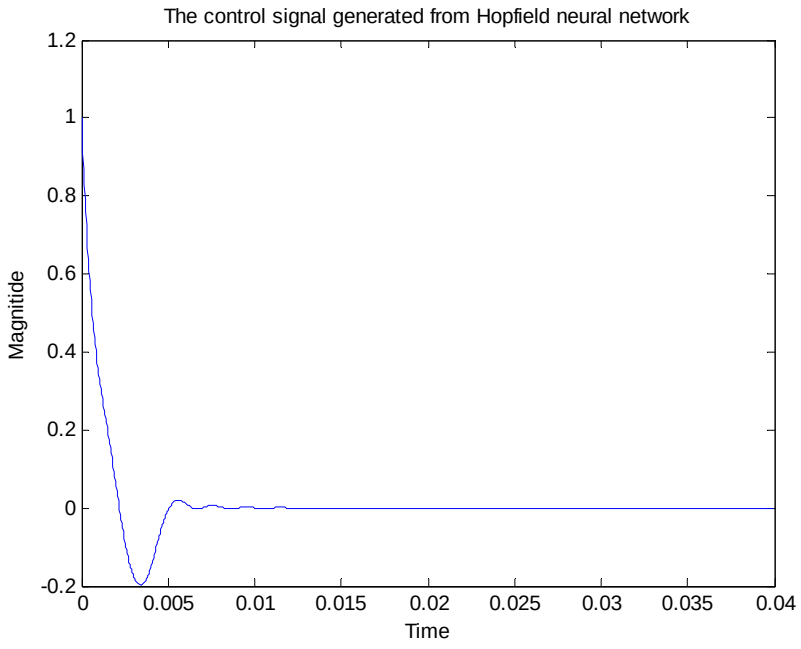


Figure 4-15. The control signal the aircraft control system

In this problem, we have shown that the Hopfield NN controller can be used to control the linear system like the aircraft control system.

CHAPTER 5
Conclusions

The application of Hopfield NN is extended to control systems in this thesis. The problem of deciding weighting factors for continuous Hopfield NN is solved in this thesis using back propagation approach. To improve the convergence rate of the back propagation training of Hopfield NN, a dynamic optimal training algorithm is also proposed to speed up the convergence speed. A new architecture of using Hopfield NN as a real-time controller is also proposed in this thesis. The popular nonlinear inverted pendulum system is first illustrated and controlled by Hopfield NN. An aircraft attitude linear control system is also illustrated and controlled by Hopfield NN. Excellent results have been obtained. In comparison with classical approaches, the Hopfield NN is very easy to implement using simple RC circuit. For higher order systems, it is also suitable for VLSI implementation. The problem of negative resistance in the trained weighting factors can be easily solved by placing a multiplexer in front of every resistance to allow for negative resistance. The following Figure 5-1 shows the inclusion of multiplexer for input signal of Hopfield NN.

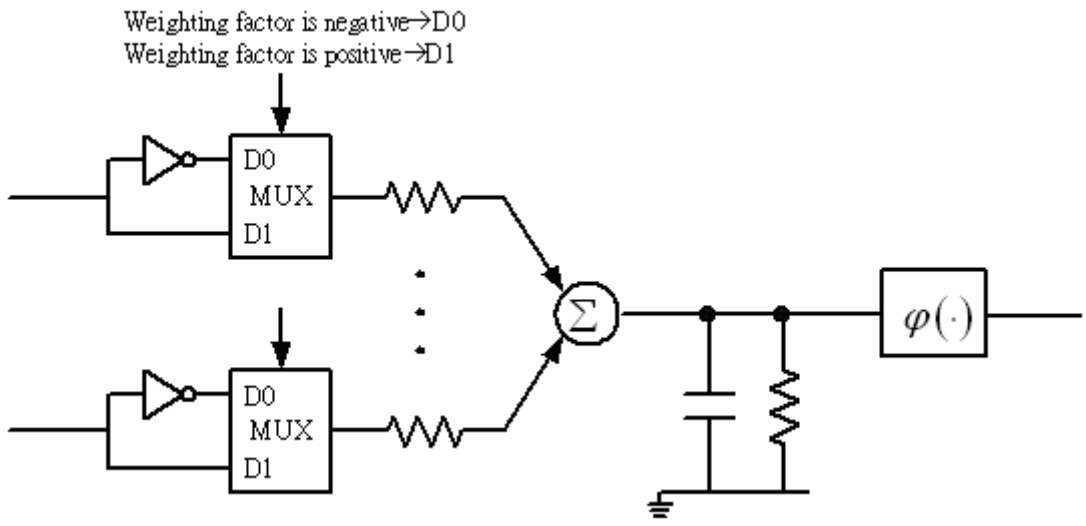


Figure 5-1 The inclusion of multiplexer in Hopfield NN

REFERENCES

[1] C. T. Lin and C. S. George Lee, "Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems," *Prentice-Hall* , 1999.

[2] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two states neurons," *Proceedings of National Academy of sciences, USA*, vol.81, pp3088-3092, 1982.

[3] M. A. Li and X. G. RUAN, "Optimal Control with Continuous Hopfield Neural Network," *IEEE. Proceedings of International Conference on Robotics, Intelligent Systems and Signal*, pp758-762 , 2003.

[4] J. J. Hopfield and D.W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biol., Cyber.*, Vol.52, pp1-25, 1985.

[5] L. Wang, Y. S. Xiao, G. Zhou, and Q. Wu, "Further Discussion of Hopfield Neural Network based DC Drive System Identification and Control," *IEEE. Proceedings of the 4'" World Congress on Intelligent Control and Automation*, pp1990-1993, 2002.

[6] C. C. Chen, "Nonlinear System Identification Using Gaussian-Hopfield Neural Networks," *MS Thesis, Department of Electronic Engineering, FJU, Taipei, Taiwan*, 2004.

[7] J. S. Lin, K. S. Chen, and C. W. Mao, "A Fuzzy Hopfield Neural Network for Medical Image Segmentation," *IEEE. Transactions On Nuclear Science*, VOL. 43, NO. 4, AUGUST 1996.

[8] G. Pajares, "A Hopfield Neural Network for Image Change Detection," IEEE *Transactions On Nuclear Science*, VOL. 17, NO. 5, SEPTEMBER 2006.

[9] R. P. Lippmann, "An introduction to computing with neural networks," *IEEE ASSP Magazine*, 1987.

[10] D. E. Rumelhart et al., Learning representations by back propagating error," *Nature*, Vol. 323, pp. 533-536, 1986

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing, Exploration in the Microstructure of Cognition*, Vol. 1, D. E. Rumelhart and J. L. McClelland, eds. Cambridge, MA: MIT Press, 1986.

[12] C. H. Wang, H. L. Liu, and C. T. Lin, "Dynamic Optimal Learning Rates of a Certain Class of Fuzzy Neural Networks and its Applications with Genetic Algorithm," *IEEE Trans. Syst., Man, Cybern. Part* B, Vol. 31, pp. 467-475, June 2001.

[13] Y. Y. Chi, "Dynamic Optimal Training of A Three Layer Neural Network with Sigmoid Function," *MS Thesis, Department of Electrical and Control Engineering, NCTU, Hsin-Chu, Taiwan*, 2004.

[14] S. Haykin, "Neural Networks: A Comprehensive Foundation," *Prentice-Hall*, second edition, 1999.

[15] M. T. Hagan, H. B. Demuth, and M. Beale, "Neural Network Design," *PWS Publishing Company*, 1996.

[16] S. Y. Cavalcanti Catunda and J. H. Feitosa Cavaltanti, "Adaptive Hopfield Neural Controller," *Industrial Electronics, 1997.* Proceedings of the IEEE International Symposium on 7-11, vol.3, pp.1206-1210, July 1997.

[17] B. C. Kuo, "Automatic Control Systems," *John Wiley and Sons, Inc.*, seventh edition, 1995.