On-line Adaptive Control of Dynamic Systems Preceded by Hysteresis via Neural Networks

Weihua Chen

A Thesis

in

The Department

of

Mechanical & Industrial Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Mechanical & Industrial Engineering at
Concordia University
Montreal, Quebec, Canada

April 2004

Your file Votre référence
ISBN: 0-612-91010-5
Our file Notre référence
ISBN: 0-612-91010-5

ABSTRACT

On-line Adaptive Control of Dynamic Systems Preceded by Hysteresis via Neural Networks

Weihua Chen

This thesis deals with on-line adaptive control of a class of dynamic systems preceded by backlash-like hysteresis nonlinearities via dynamic neural networks. A three layer recurrent neural network called the diagonal recurrent neural network (DRNN) is applied to construct the hysteresis inverse compensator (DRNNC) to remove the effect of hysteresis. An on-line learning algorithm called the dynamic back propagation (DBP) algorithm is developed to train the DRNN. Based on the cancellation of hysteresis effect, an adaptive tracking control architecture, which is constructed through the combination of sliding mode and Gaussian network (GNNC), is then proposed. The diagonal recurrent neural network compensator (DRNN) and Gaussian network controller (GNNC) are trained at the same time since DRNN requires fewer weights, and less training time, and still preserves the dynamic characteristics, which allow the DRNN model to be used for on-line application. The performance of this control structure is illustrated through simulations with example system.

Table of Contents

List of Figure

List of Table

Nomenclature

O_{net}	Output of the neuron
$\mathbf{W}$	Weight vector
W^{o}	Weights output layer
W^{I}	Weights output layer
Θ	Constant for evaluate input, output of neural network
Ω_{I}	Adjustment parameter for the learning rate
n_{I}	The neuron number of input layer
n_{h}	The neuron number of input layer
$\mathbf{X}$	Input vector
η	Learning rate
$\overline{\eta}$	Modified learning rate
η^{O}	Learning rate of output layer
η^{I}	Learning rate of input layer
Φ	Arbitrary positive constant
β	Smoothness constraint parameter
θ	Over sampling parameter
Δ	Sampling mesh
σ_{v}^{2}	Gaussian variance
l	Truncation radius

N	Number of nodes in Gaussian networks
k_d	Constant gain of feedback
k_a	Positive constant determining the adaptation rate
k_{sl}	The gain of the slide controller

Chapter 1

Introduction

In this chapter, a brief introduction of hysteresis and a literature survey regarding hysteresis modeling are presented. Contributions and organization of this thesis subsequently follow.

1.1 The phenomenon of hysteresis

Hysteresis is a property of a wide range of physical systems and devices, such as electro-magnetic fields, mechanical gear transition, electronic relay circuits, thermostats, and shape memory alloy [4]-[13]; numerous other examples could be added. The term hysteresis typically refers to the input-output relation between two time-dependent quantities that cannot be expressed as a single-valued function [20], see the loops as depicted in figure 1.1.

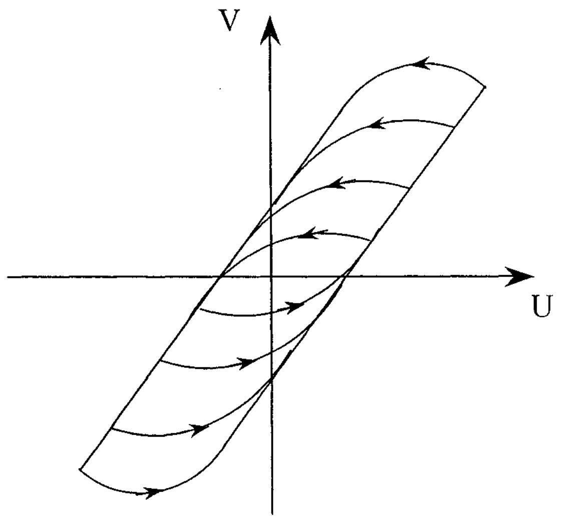


Figure 1.1 Hysteresis loops of inputs versus outputs

Hysteresis nonlinearities are among the key factors limiting both static and dynamic performance of systems that are responsible for undesirable inaccuracy or oscillations, and even leading to instability [6]. For example, harmful effects of backlash in gears prevent accurate positioning and may lead to chattering and limit-cycle-type instabilities. This increases wear and tear on the gears, which, in turn, increases backlash. Escapes from this problem are various designs of anti-backlash gears. However, their cost is high and they introduce extra weight and friction. On the other hand, in a non mechanical fashion, real-time computations can be employed to remove the harmful effects of backlash, without cumbersome and expensive anti-backlash gears [20]. There is an increasing usage of piezoelectric actuators in precision machining, because they have fast response, high stiffness, and no friction. However, because it presents a highly nonlinear input/output behavior, a piezoelectric actuator shows hysteresis behavior. For high-accuracy position and tracking systems, hysteresis will cause undesired chatter and error. Hysteresis in these systems is a consequence of the impurities properties of the materials.

Generally, common types of hysteresis nonlinearity include relay, backlash and hysteron, as illustrated bellow.

- **Relay**

The simplest example for hysteresis nonlinearity is given by a switch or relay in electronic circuits, as depicted in figure 1.2.

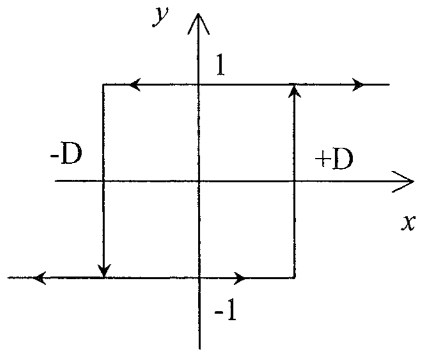


Figure 1.2 Relays

In figure 1.2, x represents input and y represents output.

- **Backlash**

Backlash is also one of the simplest forms of hysteresis. Typical examples for this kind of hysteresis are given by gear trains and position tables. A widely accepted characteristic of backlash is shown in figure 1.3, where x is the input and y is the output.

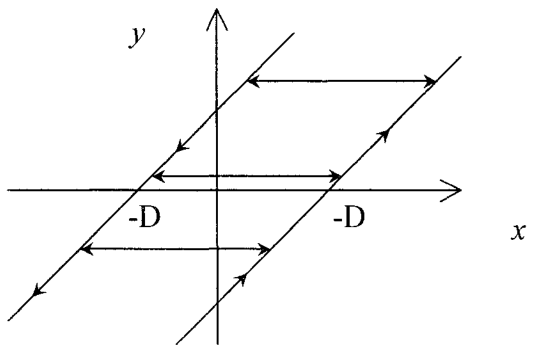


Figure 1.3 Backlashes

- **Hysteron**

Examples of more complex hysteresis phenomena are given by electro-magnetic, piezoelectric actuators and shape memory alloys, etc. The nonlinear characteristic of hysteron is shown in figure 1.4, where x is the input and y is the output.

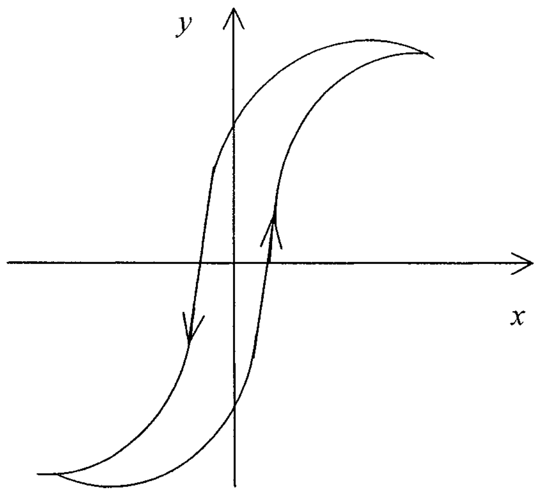


Figure 1.4 Hysteron

1.2 Properties of hysteresis

It is well known that a property of hysteresis is “memory behavior” [3], [9]. The so called “memory behavior”, implies that the outcome of a hysteresis is not only based on the current value, but is also related to the previous history. This can be shown in figure 1.1. When the input value alternates between increasing and decreasing, the response curve does not continue to follow the original path; instead, it draws a new effect-delayed curve.

In this thesis, we only focus on hysteresis with the rate-independence, says that the form of the hysteresis diagrams is independent of the speed with which they are traversed. In other words, only the previous extreme input values determine the hysteresis branches. The speed of input variations is not an influential factor. Figure 1.5 illustrates this property.

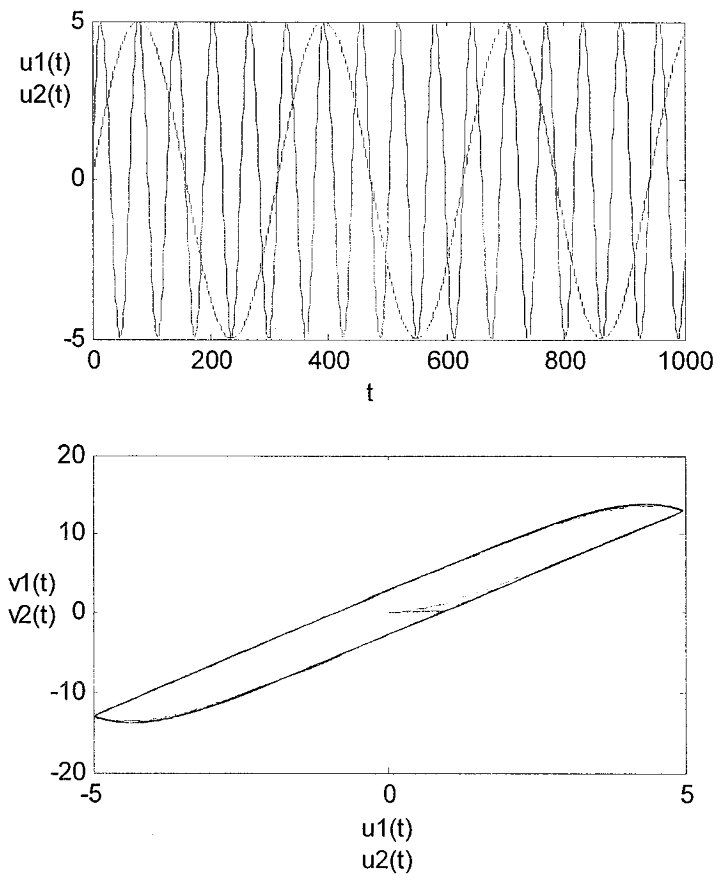


Figure 1.5 Rate-independent effect of hysteresis

In plotting of Figure 1.5, two sin inputs $u_1(t)$ and $u_2(t)$, with the same successive extreme values but different frequencies are employed as shown in the upper part of Figure 1.5. The same hysteresis responses are obtained as shown in the lower part of Figure 1.5, where the rate-independent memory effect can be shown.

1.3 Mathematical models for hysteresis

Hysteresis is a phenomenon common to a broad spectrum of physical systems. As such, it is often present in plants for which controllers are being designed, where it introduces a problematic nonlinear multi-valued behavior. Continuing research into nonlinear systems is generating an increasing interest in the control of hysteretic plants. Herein, there is a broad twofold focus: to obtain general and tractable models of hysteresis capable of accurately describing a variety of hysteretic phenomena, and then to derive corresponding control designs and supporting analytical results [6]. According to past research on hysteresis, many types of mathematical models for hysteresis have been presented, such as Duhelm, Chua-Stromsmoe, Preisach, Ishlinskii, and Mohammed models, etc [6], [8]. Among these, the Duhelm and Preisach models are most popular in recent research because both of them are capable of representing a great many forms of hysteretic behavior and they are also mathematically tractable enough for design control [6]. Here, we give a brief illustration for these two models.

– Duhelm Model

The Duhelm model for active hysteresis dates from 1897 [8] and focuses on the fact that the output changes its character when the input changes its direction. This model uses a phenomenological approach, postulating a differential equation to model the relation of input and output. Bouc-Wen differential equation is described by [8]

$$\frac{dw}{dt} + a\left|\frac{dv}{dt}\right| g(v,w) = b\frac{dv}{dt} \tag{1.1}$$

A typical choice for g is $g(w,v) = cw - v$. When $v(\cdot)$ is a sinusoid input, $w(\cdot)$ forms a classical hysteresis loop as shown in Figure 1.6.

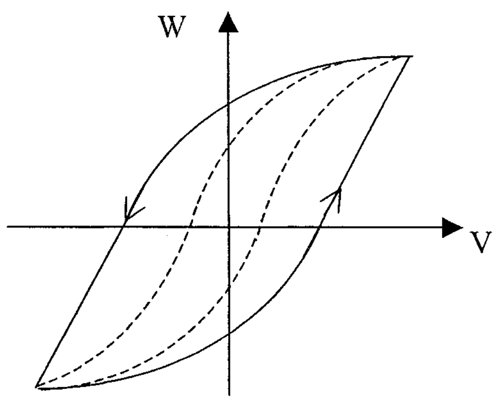


Figure 1.6 The shape of the hysteresis curve can be changed

by modifying a, b, and c

The real parameters a, b, and c control the scale and shape of the hysteresis curve. Their effects are illustrated in Figure 1.6.

Such a model is useful in applied electromagnetics because the function and parameters can be fine-tuned to match experimental results [8].

– The Preisach Model

The Preisach Model of electromagnetic hysteresis dates from 1935 [8]. It was investigated in the 1950s by Everett and collaborators and by Biorci and Pescetti, and has been studied extensively in recent times. The basic idea of this model is to represent a large class of hysteresis operators as an average of relays. That is [8]

$$v(t) = \iint_S \mu(\alpha,\beta)\gamma_{\alpha\beta}[u](t)d\alpha\, d\beta \tag{1.2}$$

where $\mu(\alpha,\beta) \geq 0$ is a weight function usually with support on a bounded set in the (α,β) plane, $\gamma_{\alpha\beta}[u](t)$ is a relay hysteresis operator with thresholds $\alpha < \beta$, and

$$\gamma_{\alpha\beta}[u](t) = \begin{cases} +1, & u(t) > \beta \\ \varsigma, & \alpha \le u(t) \le \beta \\ -1, & u(t) \le \alpha \end{cases} \qquad (1.3)$$

where ς can be +1 or -1, depending on the history of $u(t)$. See Figure 1.7

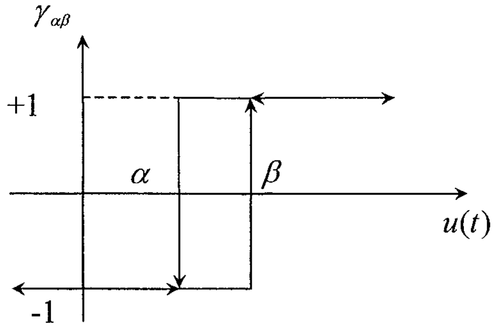


Figure 1.7 Relay hysteresis operator

Figure 1.8 will be used to illustrate how the Preisach model works. In Figure 1.8(a), S represents the limiting triangle corresponding to maximal and minimal inputs, and, while $u_{\max}$ and $u_{\min}$ represent the upper and lower limits of the inputs respectively. The region S+ and S- represents the hysteresis operators that are in 'down' and 'up' positions respectively. Assume initially the input u_0 at time t_0 is 0, and all the hysteresis operators are in the 'down' position (with an output of 0) (see Figure 1.8(a)). When the input increases from 0 to u_1, all the hysteresis operators, whose up switching β value is less than the input u_1, are turned to the 'up' position. Geometrically, this switch to the 'up' position leads to the subdivision of the limited triangle S into two areas that are S+1 and S-1 (see figure 1.8(b)). This subdivision is made by the horizontal line $\beta = u(t)$, which moves upwards when the input is increased. When the input decreases from u_1 to u_2, all the hysteresis operators, whose down switching α value is larger than the current input value u_2, are turn to the 'down' position. Geometrically, this is illustrated by a vertical line moving from right to left that, thus, reduces the S+1 area (see figure 1.8(c)).

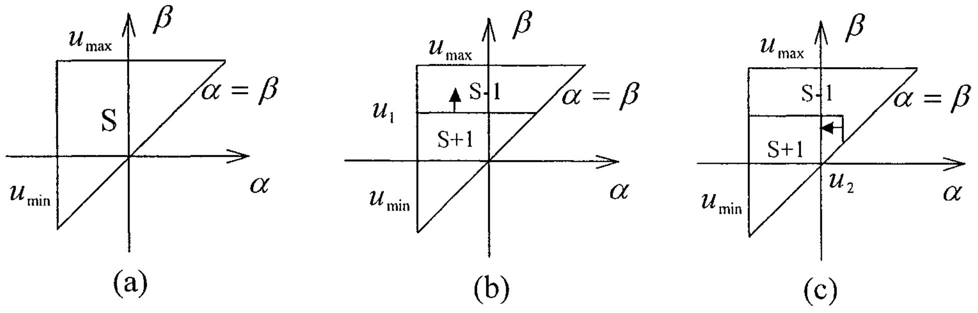


Figure 1.8 Illustration of how the Preisach model works

According to the above illustration, the double integral in equation (1.2) can be subdivided into two double integrals over S+1 and S-1, respectively,

$$v(t) = \iint_{S+1} \mu(\alpha,\beta)\gamma_{\alpha\beta}[u](t)d\alpha\, d\beta + \iint_{S-1} \mu(\alpha,\beta)\gamma_{\alpha\beta}[u](t)d\alpha\, d\beta \tag{1.4}$$

Let us consider an input history consisting of a few cycles of increase and the decrease as illustrated in Figure 1.9(a). At time t_1, the subdivision area with the bound line 'a-f-g-h-i-j-k' is illustrated in figure 1.9(b). Subsequently the input is further decreased to some value u_8 whose absolute value is bigger than that of u_2, u_3, u_4, and u_5. Because of the wipeout property of the Preisach model, the history value of u_2, u_3, u_4, and u_5 will not affect the output. At time t_2, the subdivision area with the bound line 'a-b-c-d-e' is illustrated in Figure 1.9(c).

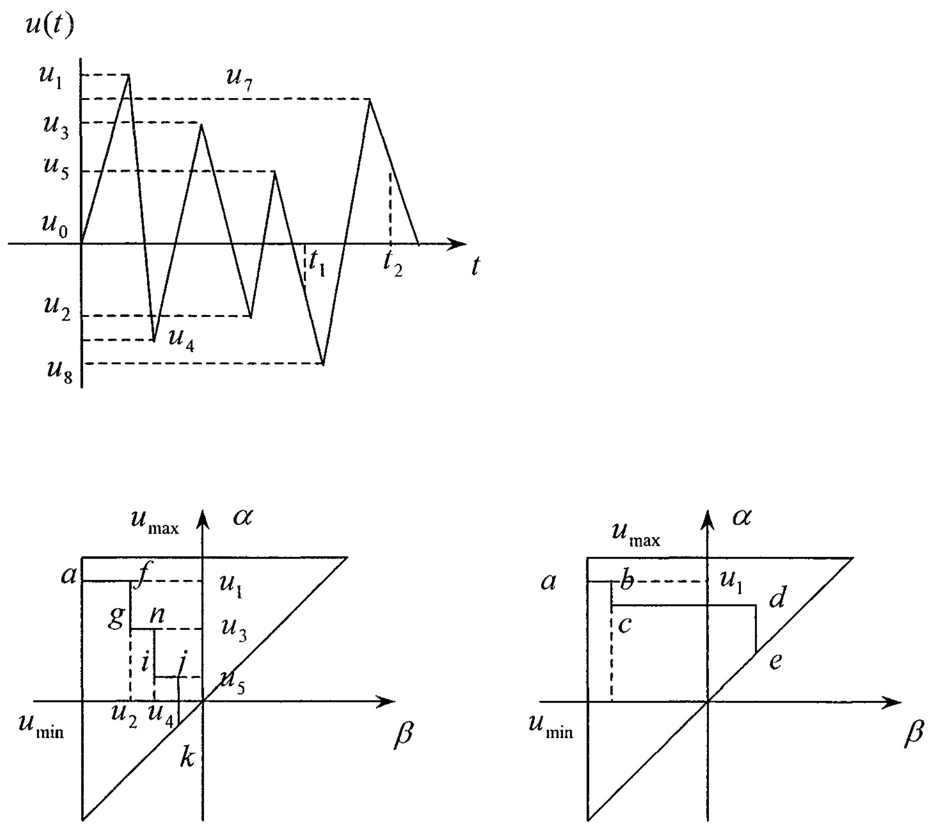


Figure 1.9 Illustration of how the Preisach model works using example

1.4 Neural network approximation for hysteresis

Neural network is a powerful and flexible tool for the identification of nonlinear systems and it can be used in the identification of systems with hysteresis [3], [13]. Hysteresis modeling based on neural networks has indeed been receiving increasing attention in the last few years [13]. In 2000, Visone et al proposed a hysteresis identification technique using Neural-Preisach-Type models [13], which was based on Multi-layer neural networks (NN). In 2002, Tiansun Lu developed a hysteresis modeling method based on diagonal recurrent networks. (DRNN) [3]. A reference model for hysteresis identification based on the neural network is displayed in Figure 1.10.

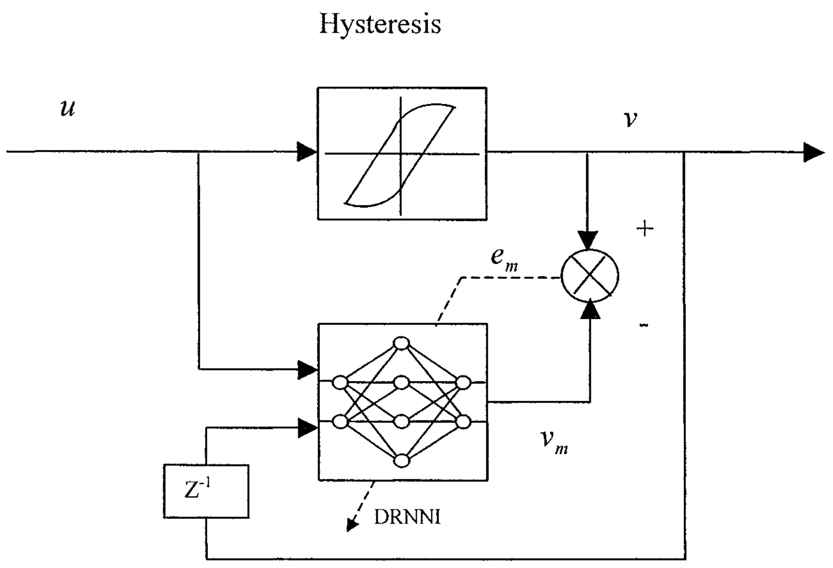


Figure 1.10 Block diagram of Hysteresis learning structure

Compared to traditional methods, hysteresis modeling based on neural networks can achieve higher approximation accuracy since neural networks are capable of approximating any nonlinear continuous function to arbitrary accuracy in a compact subset if the number of connection weights is sufficiently large. In addition, it can approximate some hysteresis nonlinearities that are difficult to fuse with controller design.

1.5 Objectives

In this thesis, the hysteretic system is considered as a dynamical nonlinear plant preceded by a hysteretic actuator. The proposed control strategy will track both cyclic and acyclic reference input signals. The two main objectives of this study are:

(1) to develop a hysteresis inverse compensator using Recurrent Neural Networks.

(2) to develop a controller based on the combination methodology of sliding mode component and Gaussian network adaptive component.

1.6 Thesis Organization

Chapter 2 gives a brief introduction of hysteretic system. And a brief survey of the control approach for such a system is also included in this chapter.

Chapter 3 gives a brief introduction of diagonal recurrent neural networks and sigmoid self-feedback neurons, by which the diagonal neural networks are constructed.

Chapter 4 proposes two hysteresis learning methods based on diagonal recurrent neural networks. One of the hysteresis learning methods is based on the dynamic network with adaptive input weights. The other method is based on the dynamic network with fixed input weights. Simulation studies show that both of these two methods perform very high accuracy for hysteresis learning and, therefore, they can be used for hysyeresis identification, especially the latter method.

Chapter 5 proposes a hysteresis inverse online learning method based on diagonal recurrent neural networks without hysteresis identification. Simulation studies show that this method is effective for mitigating the effect of hysteresis and can be employed in the control system designs described in chapter 6.

Chapter 6 gives the control scheme based on the Gaussian network controller with a diagonal neural network compensator. Controller design is derived from a continuous system taking into consideration dynamic error caused by the mis-match of hysteresis inverse compensation. Based on the results, a controller design procedure for the discrete system is proposed.

Chapter 7 describes the simulation studies and presents two examples. One is a second order linear dynamic plant preceded by the hysteresis and another is a third order linear dynamic plant preceeded by the hysteresis. The hysteresis inverse learning happens on-line rather than by obtaining a hysteresis inverse model through offline training. Simulation results are given based on the proposed control methodology so as to evaluate the whole system performance by using matlab programs. The simulation results show that the proposed control methodology is effective for the hysteretic system control.

Chapter 8 presents the conclusion of this thesis and gives recommendations for future work. The references follow Chapter 8.

1.7 Contribution

(1) One dynamic neural network for hysteresis (or hysteresis inverse) learning is used rather than two as in past research. Great uncertainties caused by switching from one neural network to another are avoided, thus learning accuracy can be dramatically improved.

(2) Since very high precision is required in hysteresis inverse learning, the hysteresis inverse compensator removes most of the hysteresis nonlinearity. Therefore, more control methodologies can be applied after hysteresis effects have been removed.

Chapter 2

Literature survey of hysteretic system control

2.1 Introduction of hysteretic system

As has been mentioned in Chapter 1, the hysteresis actuator output $v(t)$ depends not only on the current input $u(t)$, but also on its previous trajectory. Control of a system is typically challenging in the presence of hysteresis nonlinearities. They severely limit system performance in such manners as giving rise to undesirable inaccuracies or oscillations, which can even lead to instability.

Actuators made of some smart material, such as magnetostrictives, piezoelectrics, and shape memory alloys, etc, can be built into structures that have the ability to respond to environmental change to achieve desired control objective. A general class of industrial systems has the structure of a nonlinear plant preceded by hysteretic nonlinearity in the actuator [1], [3]. A block diagram of this structure is shown in Figure 2.1 for reference.

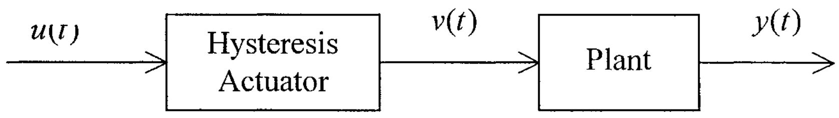


Figure 2.1 Plant with actuator hysteresis

where $v(t)$ represents the output of hysteresis actuator, $u(t)$ stands for the input to hysteresis actuator and $y(t)$ is the output of the plant. Hysteresis nonlinearity can be denoted as:

$$v(t) = H[u(t)] \tag{2.1}$$

The nonlinear dynamic system being proceeded by above hysteresis can be described in the formula:

$$x^{(n)}(t) + f(x(t), \dot{x}(t), \cdots, x^{(n-1)}(t)) = bv(t) \tag{2.2}$$

where f is continuous, linear, or nonlinear function. Control gain b is unknown but a positive constant. It should be noted that more general classes of nonlinear systems can be transformed into this structure [1].

2.2 Control strategy for system with hysteresis nonlinearity

Researchers have tried different methods to deal with the control problems of hysteretic systems. Early approaches, which have lasted for decades, to the control strategy for such a system simply ignore the effects of hysteresis. It is extremely difficult to prove stability with this method. Besides, the control performances of the hysteretic system without considering the effects of hysteresis often lead to oscillations or even instabilities [3]. A fundamental idea in coping with hysteresis has been to formulate the mathematical model and use inverse compensation to cancel out the hysteretic effect. This idea can be found in [16]-[20]. Recently, many control schemes have been proposed to deal with unknown backlash hysteresis. A common feature of such schemes is that they rely on the construction of an inverse hysteresis to mitigate the effects of the hysteresis.

Inspired by the above research, Su, et al. propose a new approach for controller synthesis by using the properties of the hysteresis model rather than constructing an inverse hysteresis nonlinearity to mitigate the effects of the hysteresis in 2000 [1]. This method can be thought of as a preliminary step to the fusion of complicated general hysteresis models with controller design.

In summary, four common approaches for the control problem of hysteretic systems are:

1. Ignoring the effects of hysteresis (See Fig 2.2):

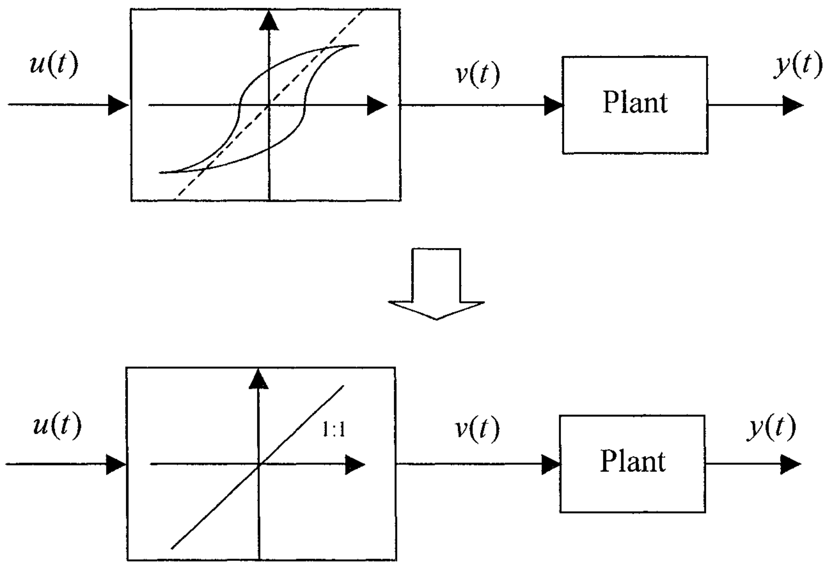


Figure 2.2 Illustration of ignoring hysteresis effects

As has been mentioned above, this approach is extremely difficult to prove stability and often leads to oscillations or even instabilities.

2. Finding the inverse model to compensate the effect of hysteresis:

In order to remove hysteresis effects, some researchers construct a hysteresis inverse compensator in the feedforward path. This idea can be observed in Figure 2.3.

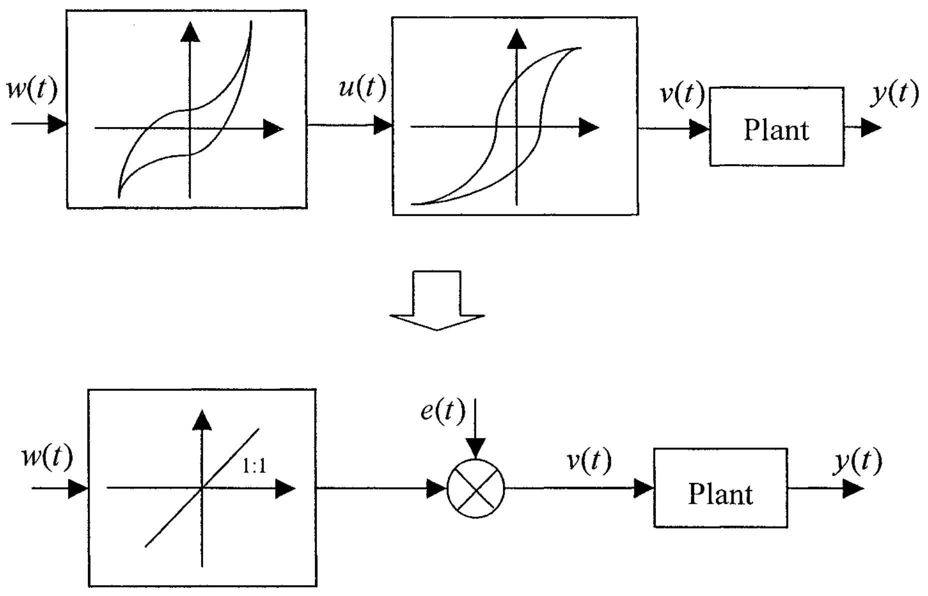


Figure 2.3 Illustration of hysteresis inverse compensation

In Figure 2.3, the plant input can be expressed as follows:

$$v(t) = w(t) + e(t)$$

where $e(t)$ represents residue error caused by the mismatching of the inverse model. The residue error can be further reduced by a closed loop control design.

3. Partitioning the characteristics of hysteresis:

Some researchers partition the characteristics of hysteresis into two parts. The first part is linear in input signals and the second part can be thought of as a disturbance. This idea is illustrated in Figure 2.4.

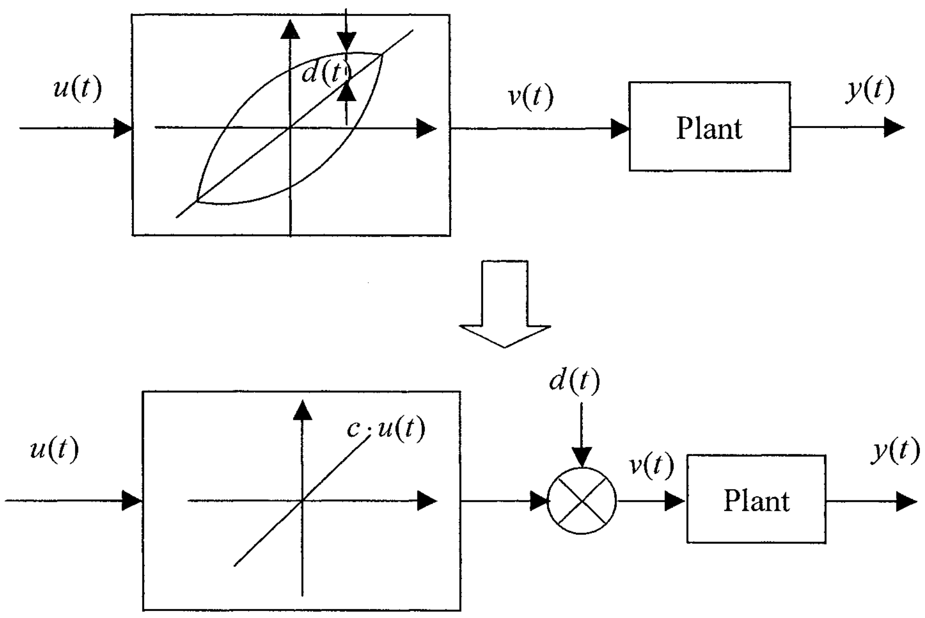


Figure 2.4 Illustration of hysteresis linearization

In Figure 2.4, the plant input can be express as follows:

$$v(t) = cu(t) + d(v)$$

where c represents a constant, $d(v)$ can be treated as dynamical disturbance.

If the second part $d(v)$ is bounded, then some conventional control methods can be applied, such as robust control [13].

4. Direct control without constructing a hysteresis inverse compensator:

By using the properties of the hysteresis model, controllers can be synthesized directly rather than by constructing an inverse hysteresis compensator to mitigate the effect of the hysteresis. The control system has a very simple structure, as presented in Figure 2.5.

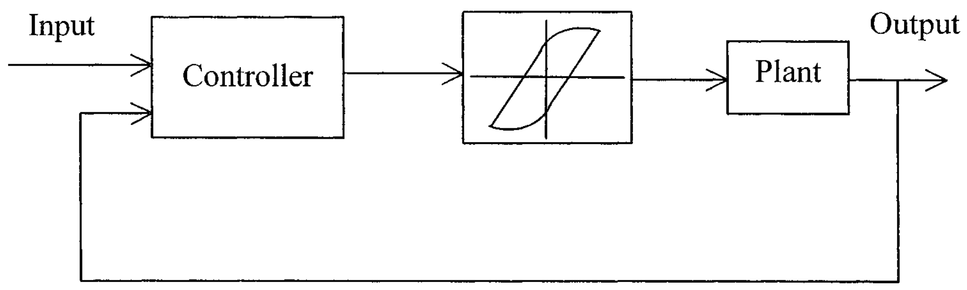


Figure 2.5 Direct control by using the properties of hysteresis

2.3 Neural network application on dealing with the hysteresis effect

In the traditional method for mitigating the effects of hysteresis, a fundamental idea is to formulate the mathematical model and then find its inverse to compensate hysteretic effect [16]-[21]. Thus, many kinds of mathematical models of hysteresis have been developed for control design over the past decades. The development of general models capable of accurately representing large classes of diverse hysteretic phenomena has been a subject of increasing interest in recent years. Whereas, when dealing with some hysteretic systems using the traditional method, one is often faced with difficulties. Some of these difficulties are: the unavailability of mathematical models for hysteresis nonlinearity, the mathematical models used for descript hysteresis nonlinearity are not accruable enough in some high accuracy applications, and that some of hysteresis models are so complicated that their inverse models are not available [3], [24], and, therefore, present a challenge for the control design engineer.

In contrast, hysteresis modeling based on neural networks can achieve a higher approximation accuracy since neural networks are capable of approximating any nonlinear continuous function to arbitrary accuracy in a compact subset if the number of connection weights is sufficiently large [3], [13]. In addition, it can approximate some hysteresis nonlinearities that are difficult to describe with mathematical models, such as some types of piezoelectric actuators. Hysteresis (or its inverse) modeling based on neural networks has been receiving increasing attention in the last year [13]. For example, in 2000, Visone, et al. proposed a hysteresis identification technique using Neural-Preisach-Type models which was based on Multi-layer neural networks (NN) [13]. Rastko and Lewis proposed a dynamic inversion compensation scheme based on the neural networks [24]. In 2002, Lu developed a hysteresis inverse modeling scheme based on diagonal recurrent networks (DRNN) [3]. A reference model for hysteresis inverse modeling based on the neural network is presented in Figure 2.6.

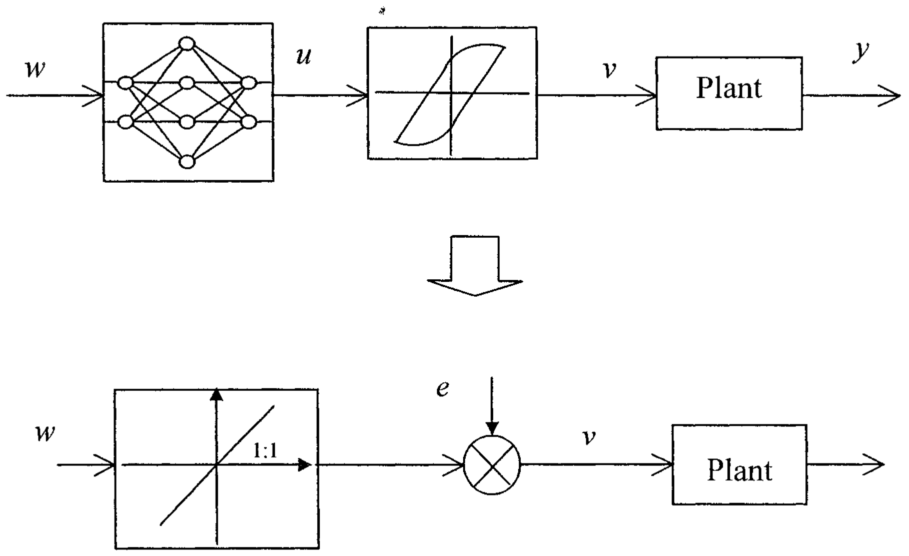


Figure 2.6 Compensated hysteresis effect using neural network

2.4 Problem Formulation

Consider the following unknown systems proceeded by hysteresis, as illustrated in Figure 2.7.

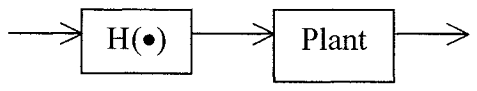


Figure 2.7 Plant proceed by hysteresis

Mathematically, this system can be described by equation (2.3) and (2.4) [1]

$$x^{(n)} + f(x, \dot{x}, \cdots, x^{(n-1)}) = bv \tag{2.3}$$

$$v = H(u) \tag{2.4}$$

where f is an unknown function, which can be linear or nonlinear, H(•) represents hysteresis nonlinearity with input u, v is the input to the plant, and b a nonzero unknown constant.

To mitigate the effects of hysteresis, the hysteresis inverse compensator is built into the feedforward path of system, as illustrated in Figure 2.8

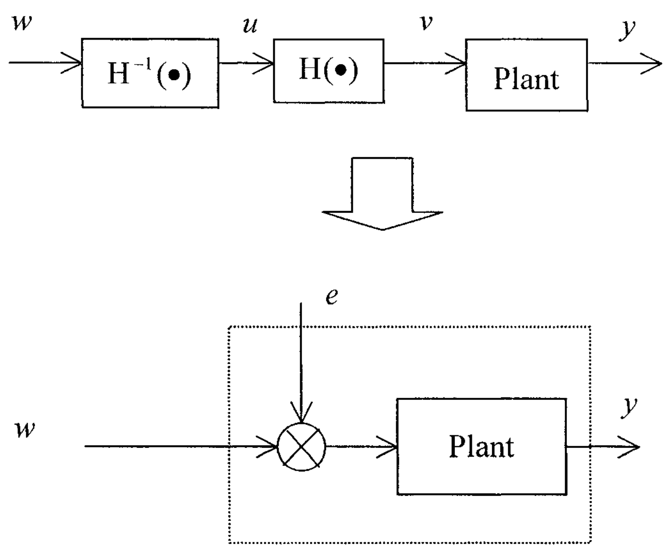


Figure 2.8 Hysteresis compensate effect

Then, the mathematical expression of the system can be rewritten as:

$$x^{(n)} + f(x, \dot{x}, \cdots, x^{(n-1)}) = b(w+e) \tag{2.5}$$

where e represents residue error caused by hysteresis inverse compensation

Again, we modify equation (2.5)

$$x^{(n)} + f(x, \dot{x}, \cdots, x^{(n-1)}) - be = bw \tag{2.6}$$

If the residue error e is bounded, equation (2.6) can be approximated by the following equation:

$$x^{(n)} + \sum_{i=1}^{N} c_i G_i(x, \dot{x}, \cdots, x^{(n-1)}) = bw \tag{2.7}$$

where G_i is a Gaussian function.

Thus, a Gaussian network controller can be applied for such a system.

Chapter 3

Diagonal Recurrent Neural Network (DRNN)

3.1 Description of Sigmoid Neuron

An artificial neuron is designed to mimic the first-order characteristics of a biological neuron. A set of inputs is applied, each representing the output of another neuron. Each input is multiplied by a corresponding weight, analogous to a synaptic strength, and all of the weighted inputs are then summed up to determine the activation level of the neuron. In the block diagram of Figure 3.1, showing the model of a neuron, an input vector X, consisting of a set of inputs $x_1, x_2, \cdots, x_n$ is applied to the artificial neuron. Each signal is multiplied by an associated weight $w_1, w_2, \cdots, w_n$ in the weight vector W before it is applied to a summation block, labeled Σ. Each weight corresponds to the strength of a single biological synaptic connection. The neuron's output is produced by a nonlinear activation function, denoted by $f(\cdot)$, acting on the summed output as shown in Figure 3.1.

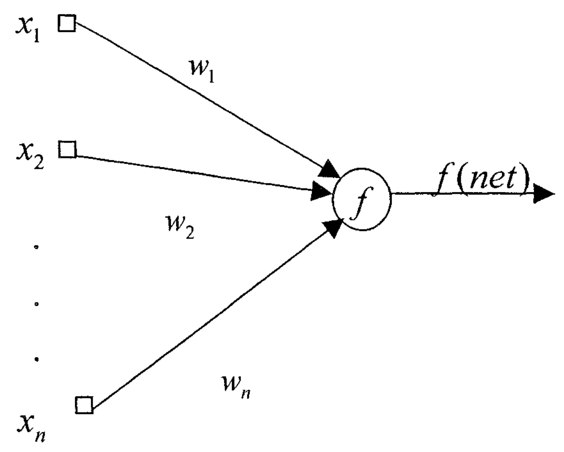


Figure3.1 General Neuron model

In mathematical terms, we may describe a neuron with the flowing pairs of equations:

$$net = \sum_{i=1}^{n} w_i \cdot x_i \tag{3.1}$$

$$O_{net} = f(net) \tag{3.2}$$

The active function, denoted by $f(\cdot)$, defines the type of neuron. If $f(net) = net$, the neuron is defined as *linear neuron*, and its model can be simplified as shown in Figure 3.2

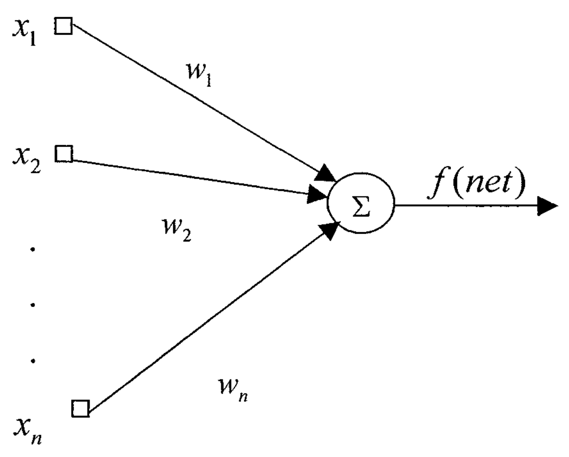


Figure 3.2 Linear neuron models

In this case, the equation (3.2) can be rewritten as follows:

$$O_{net} = \sum_{i=1}^{n} w_i \cdot x_i \tag{3.3}$$

Consider the equation (3.4) shown as below. It is actually a form of the hyperbolic tangent function that is commonly applied as an activation function. If we apply this function as an activation function of a neuron, the neuron can be defined as *sigmoid neuron*. Its detailed model is shown in Figure 3.3.

$$f(net) = \gamma \tanh[\alpha(net + \beta)] + \rho \tag{3.4}$$

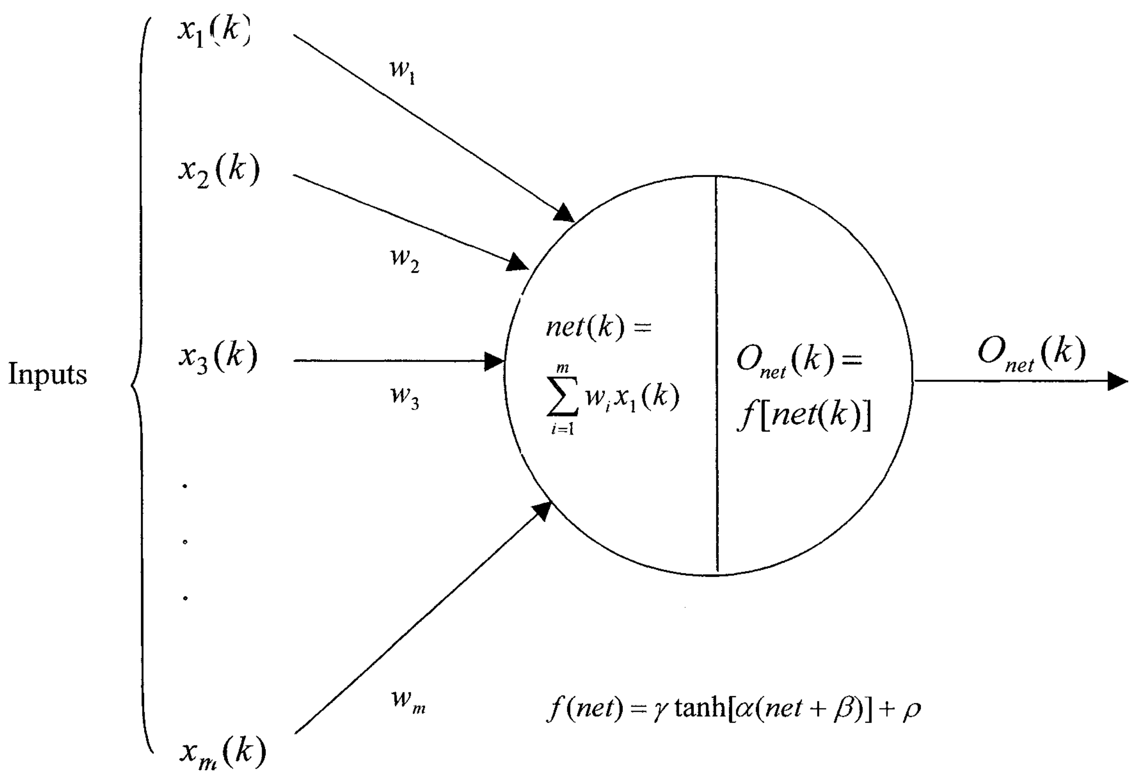


Figure 3.3 Sigmoid neuron models

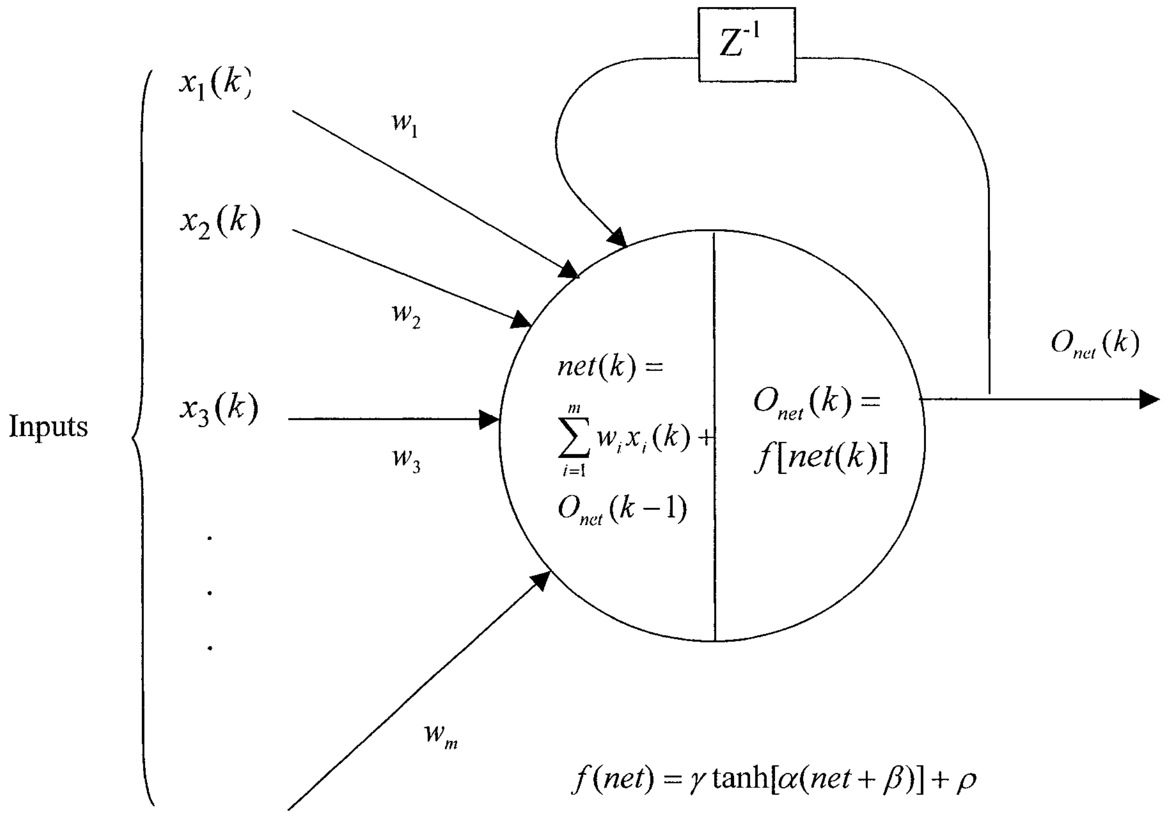
Correspondingly, we may describe this sigmoid neuron by writing the flowing pairs of equations:

$$net(k) = \sum_{i=1}^{m} w_i x_i(k) \tag{3.5}$$

$$O_{net}(k) = f[net(k)] \tag{3.6}$$

$$f[net(k)] = \gamma \tanh[\alpha(net(k) + \beta)] + \rho \tag{3.7}$$

The model of a sigmoid neuron with self-feedback is presented in figure 3.4.



Inputs
$x_1(k)$
$x_2(k)$
$x_3(k)$
w_1
w_2
w_3
w_m
Z^{-1}
$net(k) = \sum_{i=1}^{m} w_i x_i(k) + O_{net}(k-1)$
$O_{net}(k) = f[net(k)]$
$O_{net}(k)$
$f(net) = \gamma \tanh[\alpha(net + \beta)] + \rho$

Figure 3.4 Model of sigmoid neuron with self-feedback loop

Mathematically, we may describe this sigmoid neuron by writing the flowing pairs of equations:

$$net(k) = \sum_{i=1}^{m} w_i x_i(k) + O_{net}(k-1) \tag{3.8}$$

$$O_{net}(k) = f[net(k)] \tag{3.9}$$

$$f[net(k)] = \gamma \tanh[\alpha(net(k) + \beta)] + \rho \tag{3.10}$$

The characteristics of equation (3.10) are illustrated in Figure 3.5(a) and 3.5(b). Figure 3.5(a) demonstrates that varying the value of parameter β will change the saturation point but doesn't affect the saturation values (1, 2). Figure 3.5(b) demonstrates that varying the value of slope parameter α will cause a change in the slope of the curve. The activation function may be thought of as a nonlinear gain of the artificial neuron, which may be calculated by finding the ratio of the change in O_{net} to a small change in *net*. Thus, gain is the slope of the curve at a specific excitation level. It varies from a low value at large negative excitations, where the curve is nearly flat, to a high value at zero exciation, and it drops back as exciation becomes very large and positive.

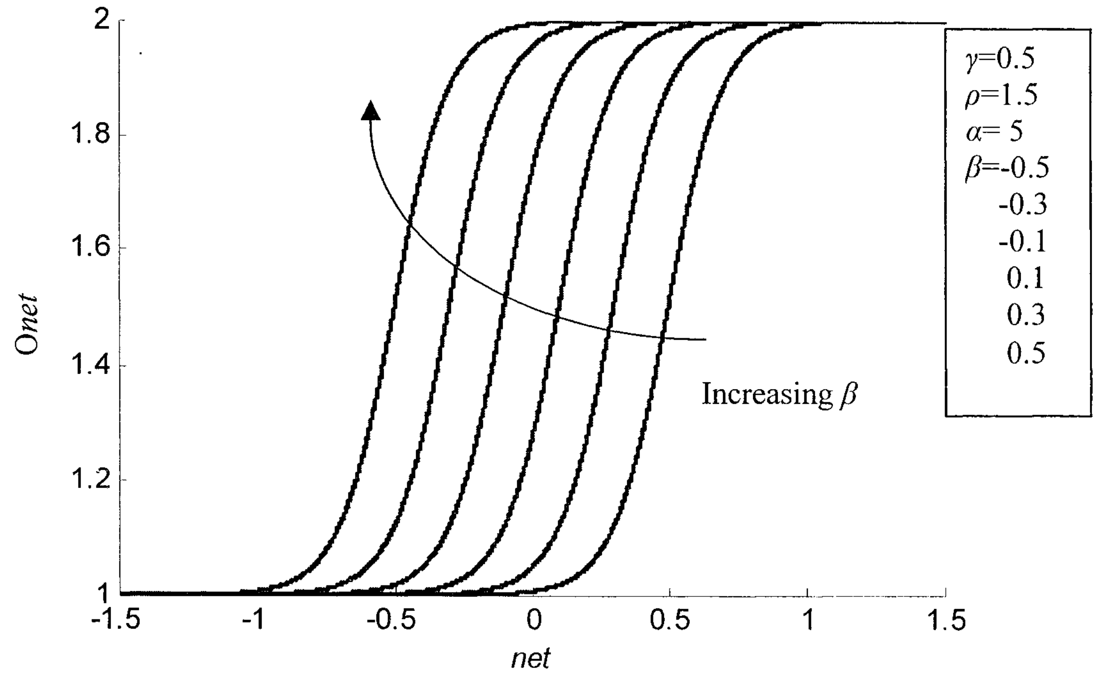


Figure 3.5(a) Sigmoid function curves for varying saturation parameter β

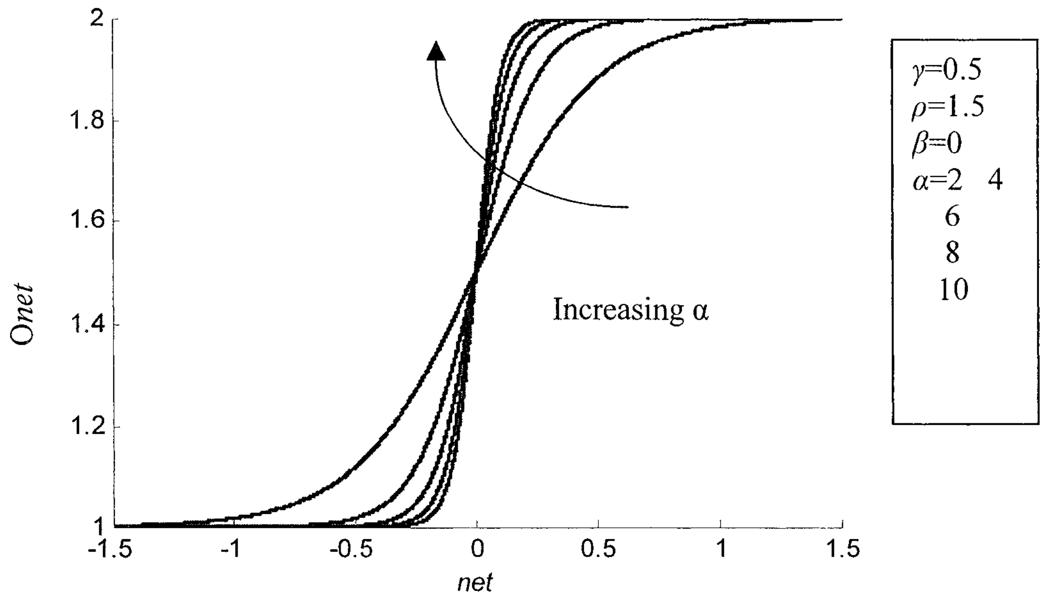


Figure 3.5(b) Sigmoid function curves for varying slope parameter α

In this thesis, a specific sigmoid function is applied, which is described as follows:

$$f(net) = 0.5\tanh(net) + 1.5 \tag{3.11}$$

It satisfies: (See Figure 3.6)

$$1 < f(net) < 2 \tag{3.12}$$

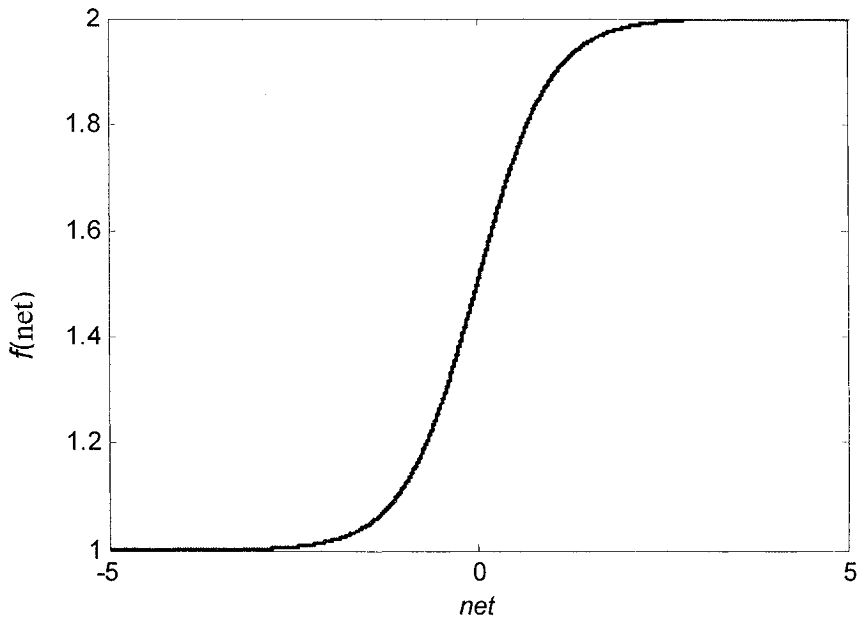


Figure 3.6 The curve of equation (3.11)

The derivative of equation (3.11) is given as follows:

$$f'(net) = \frac{0.5}{\cosh^2(net)} \tag{3.13}$$

which satisfies: (See Figure 3.7)

$$0 < f'(net) < 0.5 \tag{3.14}$$

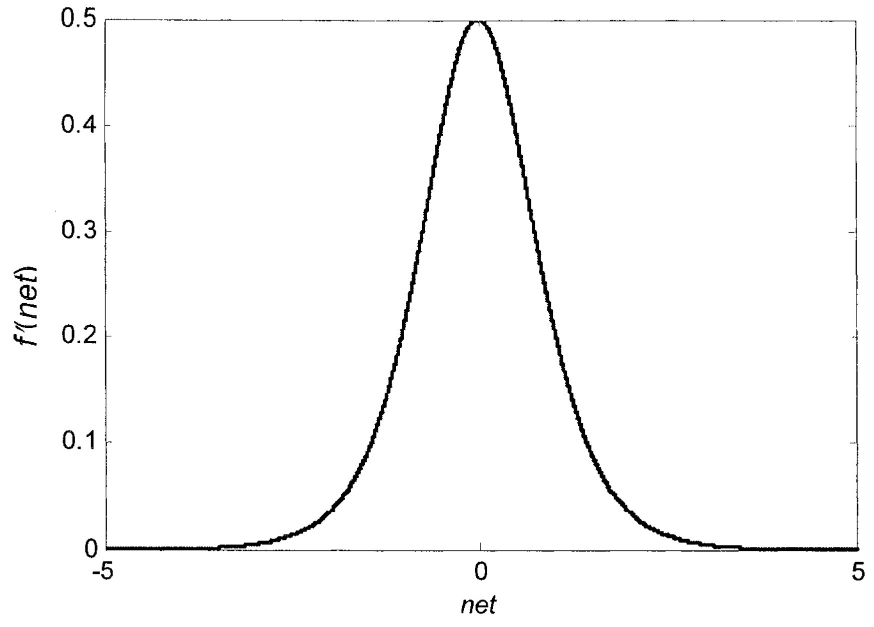
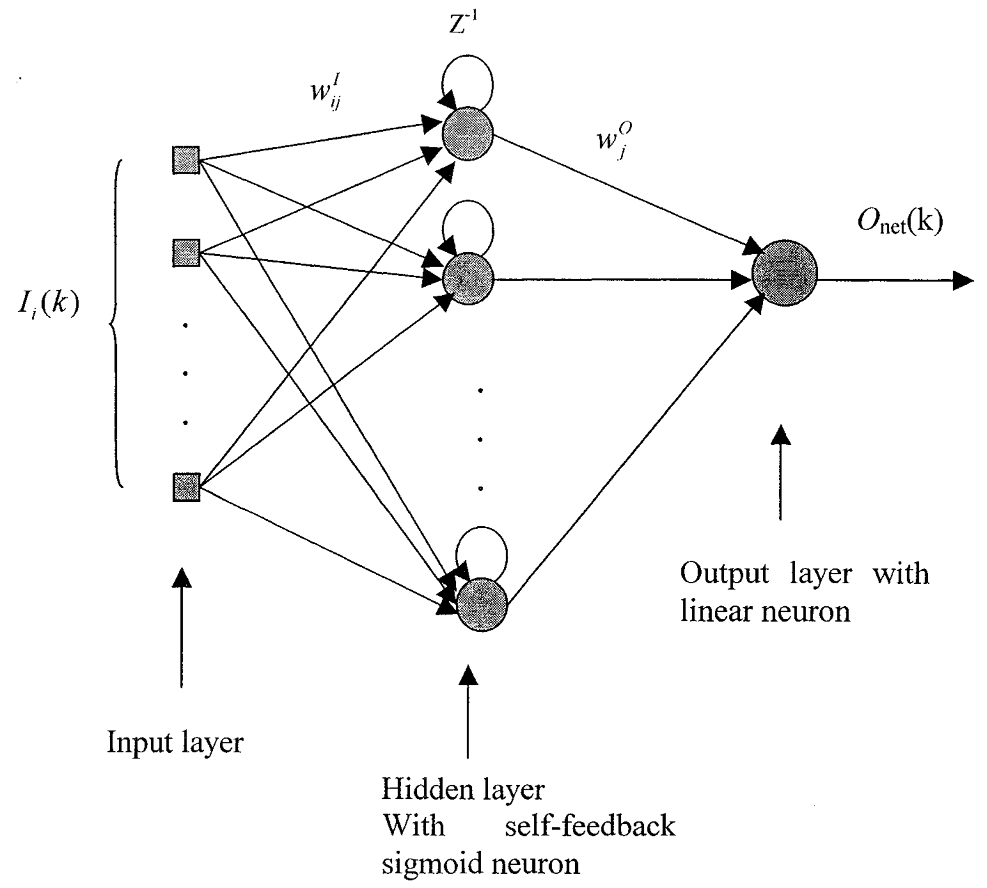


Figure 3.7 The curve of equation (3.13)

3.2 Diagonal Recurrent Neural Network (DRNN)

The recurrent neural network was introduced by Hopfield [39]. It distinguishes itself from a feed forward neural network in that it has at least one feedback loop [64]. A typical type of recurrent neural network called the "Diagonal recurrent neural network" is illustrated in the architectural graph in Figure 3.8. In the structure depicted in this figure there are self-feedback loops in the hidden layer; that is, every node in each layer is connected to every other node in the adjacent forward layer without interlinks among neurons in the same layer.



Z^{-1}
w_{ij}^{I}
w_{j}^{O}
$O_{net}(k)$
$I_i(k)$
Output layer with linear neuron
Input layer
Hidden layer With self-feedback sigmoid neuron

Figure 3.8 Diagonal recurrent neuron networks

The presence of feedback loops has a profound impact on the learning capability of the network and on its performance. For this reason, recurrent neural networks have been shown to be more powerful than pure feedforward neural networks. For example, the feedforward neural network is a static mapping, hence it is unable to represent a dynamic system mapping. Although many people use the feedforward neural network together with tapped delays to deal with dynamic systems, the feedforward neural network requires a large number of neurons to represent dynamic responses in the time domain.

On the other hand, the recurrent neural network consists of both feedforward and feedback connections between lays and neurons forming complicated dynamics and is able to deal with time-varying input or output through its own natural temporal operation. Thus the recurrent neural network is a dynamic mapping system and is more appropriate than the feedforward neural network when applied to a dynamic system [37], [41], [45].

3.3 Mathematical representation of DRNN

The mathematical model for the DRNN in Figure 3.8 is shown below:

$$O_{net}(k) = \sum_j w_j^O X_j(k) \tag{3.13}$$

$$X_j(k) = f(S_j(k)) \tag{3.14}$$

$$S_j(k) = X_j(k-1) + \sum_i w_{ij}^I I_i(k) \tag{3.15}$$

where $I_i(k)$ represents the *ith* input to the DRNN, $S_j(k)$ is the sum of inputs to the *j*th sigmoid neuron, $X_j(k)$ is the output of the *j*th sigmoid neuron and $O_{net}(k)$ is the output of the DRNN. Here $f(\cdot)$ is the usual sigmoid function representing the nonlinear threshold function, and $w_{ij}{}^I$, $w_j{}^O$ are input, and output weights respectively, in R^{n_i} and R^{n_o}, which are Euclidean spaces with appropriate dimensions.

Chapter 4

Hysteresis Learning Based on the DRNN

4.1 Description of Hysteresis

It is well known that hysteresis exists in a wide range of physical systems [4]-[13]. Throughout this chapter, a typical class of hysteresis, which is described by a modified Duhelm Model [1], is given for the purpose of simulation to test the performance of hysteresis modeling based on the DRNN model.

A typical class of hysteresis is described by [1]:

$$\frac{dv}{dt} = \alpha\left|\frac{du}{dt}\right|(cu - v) + B_1\frac{du}{dt} \tag{4.1}$$

where v represents the input and u represents response of hysteresis. The parameter α, B_1, and c represent real parameters, which control the scale and shape of the hysteresis curve.

The discrete form of (4.1), based on the forward difference method, is described below:

$$v(k) = v(k-1) + \alpha|u(k) - u(k-1)|(cu(k-1) - v(k-1)) + B_1(u(k) - u(k-1)) \tag{4.2}$$

The characteristics of hysteresis described by equation (4.1) with α=1, c=3.1635, B$_1$=0.345, $v(0) = 0$ for $u(k) = \sin(k * ts)$ is presented in Figure 4.1.

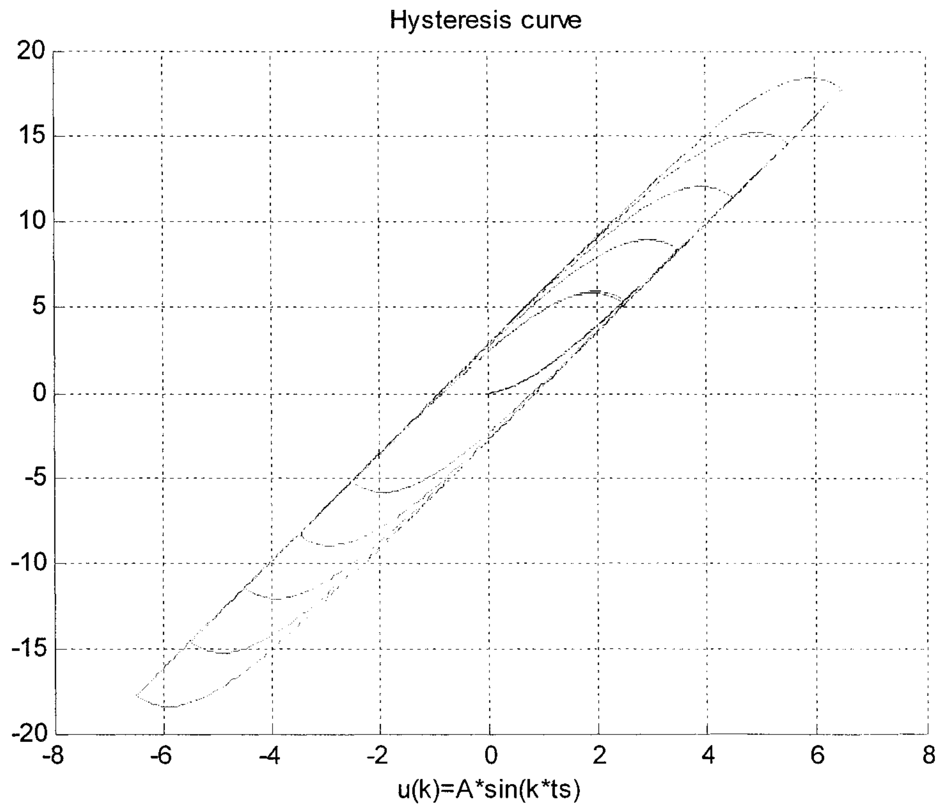


Figure 4.1 Hysteresis curves with $\alpha=1$, c=3.1635, and $B_1=0.345$ for u(k)=A*sin(k*ts) with A=2.5, 3.5, 4.5, 5.5, 6.5.

4.2 DRNN based Hysteresis Learning Structure

Neural networks are a known method for universal function approximation. They are capable of approximating any nonlinear continuous function to arbitrary accuracy in a compact subset if the number, in relation to the connection weight, is sufficiently large [3]. In this thesis, a dynamic neural network named diagonal recurrent neural network (DRNN) is employed for hysteresis learning. The block diagram of the learning structure is presented in Figure 4.2, and this neural network is denoted as DRNNI, the abbreviation of diagonal recurrent neural network identifier. The dynamic learning algorithm for DRNNI will be developed in the next section.

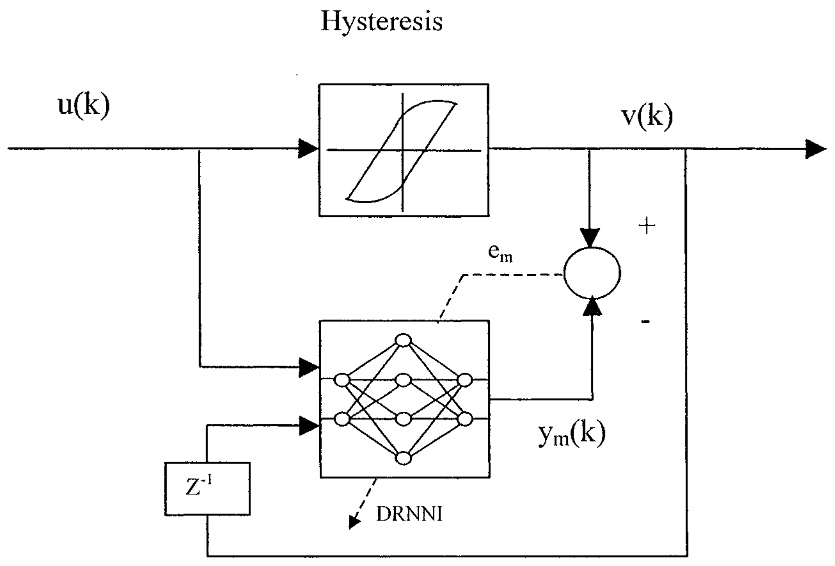


Figure 4.2 Block diagram of Hysteresis learning structure

4.3 Hysteresis Learning Algorithm for DRNNI with Adaptive Input and Output Weights

4.3.1 Description of Weights Update Rule

Let $u(k)$ and $y_m(k)$ be the input and output of the DRNNI. The error function can be defined as:

$$E = \frac{1}{2} e_m{}^2(k) = \frac{1}{2}(v(k) - y_m(k))^2 \tag{4.3}$$

The gradient of error in (4.3) with respect to an arbitrary weight vector $W \in R^n$ is represented by

$$\frac{\partial E}{\partial W} = -e_m(k)\frac{\partial y_m(k)}{\partial W} = -e_m(k)\frac{\partial O_{net}(k)}{\partial W} \tag{4.4}$$

where $e_m(k) = v(k) - y_m(k)$ is the error between the reference input and output response of the DRNNI, and $O_{net}(k)$ is the output of DRNNI. Respectively, the output gradients with reference to output and input weights can be computed using the following equations:

$$\frac{\partial O_{net}(k)}{\partial W_j^o} = X_j(k) \tag{4.5a}$$

$$\frac{\partial O_{net}(k)}{\partial W_{ij}^I} = W_j^O Q_{ij}(k) \tag{4.5b}$$

where $Q_{ij}(k) \equiv \dfrac{\partial X_j(k)}{\partial W_{ij}^I}$, which satisfies:

$$Q_{ij}(k) = f'(S_j)[I_i(k) + Q_{ij}(k-1)] \qquad Q_{ij}(0) = 0 \tag{4.6}$$

Note: The equations in (4.6) are nonlinear dynamic recursive equations for the gradients $\frac{\partial X_j(k)}{\partial W}$, and can be solved recursively with given initial conditions.

<u>Proof of (4.5) and (4.6):</u>

From (3.13), the gradient with respect to the output weight is found as

$$\frac{\partial O_{net}(k)}{\partial W_j^O} = \frac{\partial\left[\sum_j W_j^O X_j(k)\right]}{\partial W_j^O} = X_j(k)$$

Hence, (4.5a) follows.

Similarly, from (3.13), the gradient with respect to the input weight is

$$\frac{\partial O_{net}(k)}{\partial W_{ij}^I} = \frac{\partial O_{net}(k)}{\partial X_j(k)}\frac{\partial X_j(k)}{\partial W_{ij}^I} = W_j^o \frac{\partial X_j(k)}{\partial W_{ij}^I} = W_j^o Q_{ij}(k)$$

Thus, (4.5b) follows.

Finally, from (3.13), (3.14) and (3.15),

$$\frac{\partial X_j(k)}{\partial W_{ij}^I} = \frac{\partial X_j(k)}{\partial S_j(k)}\frac{\partial S_j(k)}{\partial W_{ij}^I} = f'(S_j(k))\frac{\partial S_j(k)}{\partial W_{ij}^I} \tag{4.7}$$

$$\frac{\partial S_j(k)}{\partial W_{ij}^I} = \frac{\partial}{\partial W_{ij}^I}[X_j(k-1) + \sum_i W_{ij}^I I_i(k)]$$

$$= I_i(k) + Q_{ij}(k-1) \tag{4.8}$$

Substitute (4.8) into (4.7), then (4.6) follows.

From (4.4) and (4.5), the negative gradient of error with respect to a weight is:

$$-\frac{\partial E}{\partial W_j^O} = e_m(k)X_j(k) \tag{4.9}$$

$$-\frac{\partial E}{\partial W_{ij}^I} = e_m(k)W_j^O Q_{ij}(k) \tag{4.10}$$

The weight can now be adjusted following the steepest descent method, i.e., the update rule of the weights is described as follows:

$$W(k+1) = W(k) + \eta(-\frac{\partial E}{\partial W}) \tag{4.11}$$

where the weight W can be W^o or W^I, and η is the learning rate.

The update rule equations for individual weights can be described as follows:

$$W_j^O(k+1) = W_j^O(k) + \eta^O e_m(k) \cdot X_j \tag{4.12a}$$

$$W_{ij}^I(k+1) = W_j^I(k) + \eta^I e_m(k) \cdot W_j^O(k) Q_{ij}(k) \tag{4.12b}$$

4.3.2 Learning Rate Estimate for DRNNI

In this section, we give a rule for choosing the learning rate under the guarantee of network stable.

If W is an arbitrary weight vector in R^n, η is the learning rate for the correspond weight of DRNNI, and $\|\cdot\|$ is the Euclidean norm in R^n, then the convergence is guaranteed if η is chosen as

$$0 < \eta < \frac{2}{\left\| \frac{\partial O_{net}(k)}{\partial W} \right\|^2} \tag{4.13}$$

Proof of (4.13):

Give a Lyapunov function as

$$\Delta V(k) = V(k+1) - V(k)$$

$$= \frac{1}{2}[e_m{}^2(k+1) - e_m{}^2(k)]$$

$$= \frac{1}{2}\Delta e_m(k)[2e_m(k) + \Delta e_m(k)] \tag{4.14}$$

If ΔW represents a change in an arbitrary weight vector in R'', the error difference due to the learning can be presented as below:

$$\Delta e_m(k) = e_m(k+1) - e_m(k) = \left[\frac{\partial e_m(k)}{\partial W}\right]^T \Delta W = -\left[\frac{\partial O_{net}(k)}{\partial W}\right]^T \Delta W \tag{4.15}$$

From the updated rules (4.4) and (4.11),

$$\Delta W = \eta e_m(k) \frac{\partial O_{net}(k)}{\partial W} \tag{4.16}$$

Substitute (4.15) and (4.16) into (4.14), we obtain:

$$\Delta V(k) = -\frac{1}{2}\left[\frac{\partial O_{net}(k)}{\partial W}\right]^T \eta \cdot e_m(k) \frac{\partial O_{net}(k)}{\partial W} \cdot \left\{2e_m(k) - \left[\frac{\partial O_{net}(k)}{\partial W}\right]^T \eta \cdot e_m(k) \frac{\partial O_{net}(k)}{\partial W}\right\}$$

$$= -\eta \cdot e_m^2(k) \cdot \left\|\frac{\partial O_{net}(k)}{\partial W}\right\|^2 + \frac{1}{2}\eta^2 e_m^2(k) \left\|\frac{\partial O_{net}(k)}{\partial W}\right\|^4$$

$$= -\eta \cdot e_m^2(k) \cdot \left\|\frac{\partial O_{net}(k)}{\partial W}\right\|^2 \left(1 - \frac{1}{2} \cdot \eta \cdot \left\|\frac{\partial O_{net}(k)}{\partial W}\right\|^2\right) \tag{4.17}$$

If the convergence must be guaranteed, then $\Delta V(k) < 0$, thus,

$$1 - \frac{1}{2} \cdot \eta \cdot \left\|\frac{\partial O_{net}(k)}{\partial W}\right\|^2 > 0 \tag{4.18}$$

and (4.13) follows.

Theorem 4.1:

Let η^O and η^I be the learning rates of the DRNNI weights W^O and W^I respectively. Then the dynamic back propagation algorithm converges if the learning rates are chosen as:

$$0 < \eta^O < \frac{1}{2 \cdot n_h} \tag{4.19a}$$

$$0 < \eta^I < \frac{2}{(n_I + n_h) \cdot {W^O_{\max}}^2 \cdot {I_{\max}}^2} \tag{4.19b}$$

where n_I represents the neuron number of input layer, and n_h represents the neuron number of output layer.

Here, we define $W^O{}_{\max} := \max\left|W_j^O\right|$, and $I_{\max} := \max\left|I_i\right|$.

Proof of (4.19a)

From (4.5a):

$$\frac{\partial O_{net}(k)}{\partial W^O} = X(k)$$

where $X = \left[X_1, X_2, \cdots, X_{n_h}\right]^T$, and X_j is the output value of the jth neuron in the hidden layer, and n_h is the neuron number in the hidden layer.

Since $1 < X_j < 2$, $j = 1, 2, \cdots, n_h$, then we have:

$$\left\|\frac{\partial O_{net}(k)}{\partial W^O}\right\|^2 < 4 \cdot n_h \tag{4.20}$$

From (4.13) and (4.20), (4.19a) can be obtained.

Proof of (4.19b)

In equation (4.5), we define $Q_{ij}(k) \equiv \dfrac{\partial X_j(k)}{\partial W_{ij}^I}$, and from (4.6)

$$Q_{ij}(k) = f'(S_j(k))[I_i(k) + Q_{ij}(k-1)]$$

Thus the solution of the above equation can be written as:

$$Q_{ij}(k) = \sum_{m=1}^{k}\left(\left[\prod_{n=0}^{k-m} f'(S_j(k-n))\right] I_i(m)\right) + \left[\prod_{n=0}^{k-1} f'(S_j(k-n))\right] Q_{ij}(0)$$

Note that $Q_{ij}(0) = 0$

$$Q_{ij}(k) = \sum_{m=1}^{k}\left(\left[\prod_{n=0}^{k-m} f'(S_j(k-n))\right] I_i(m)\right)$$

$$\left|Q_{ij}(k)\right| \le \sum_{m=0}^{k}\left(\left[\prod_{n=0}^{k-m} \left|f'(S_j(k-n))\right|\right] \left|I_i(m)\right|\right)$$

Since $f'(S_j(k-n)) < 0.5 \quad (0 \le n < k < +\infty)$,

$$\left|Q_{ij}(k)\right| \le \sum_{m=0}^{k} 0.5^{(k-m)} I_i(m) \le \sum_{m=0}^{k} 0.5^{(k-m)} I_{\max} = (0.5 + 0.5^2 + \cdots) I_{\max} \le I_{\max}$$

where $I_{\max} := \max\left[bias,\ |u_{\max}|,\ |v_{\max}|\right]$

Thus,

$$\|Q(k)\|^2 \le (n_I + n_h) I_{\max}{}^2$$

From the definition, $\dfrac{\partial O_{net}(k)}{\partial W_{ij}^I} = W_j^o Q_{ij}(k)$, we can obtain:

$$\left|\frac{\partial O_{net}(k)}{\partial W_{ij}^I}\right| \le W_{\max}^O \cdot \left|Q_{ij}(k)\right|$$

Here, we define $W^{O}{}_{\max} := \max\left|W_j^O\right|$. Thus

$$\left\|\frac{\partial O_{net}(k)}{\partial W^I}\right\|^2 \le W_{\max}^{O}{}^2 \cdot (n_I + n_h) I_{\max}{}^2 \tag{4.21}$$

Hence, from (4.13) and (4.20), (4.19b) follows.

Remarks:

In the fixed-learning-rate learning process, the two uncertain terms, $W_{\max}^O$ and $I_{\max}$ in (4.19b), need to be further evaluated. And this can be achieved under the following conditions:

1. The input signals to the system are bounded.
2. The DRNNI is fully connected, and the same initial value for all weights is set.

Clearly, condition 1 constrains the term $I_{\max}$ in (4.19b), and condition 2 will make it possible to estimate the value of $W_{\max}^O$. The detailed process is presented as follows:

In the learning process of fully connected DRNNI, if we set the same initial value for all weights and apply fixed learning rates for weights adaptive, we can achieve such properties: $W_1^O = W_2^O = \cdots W_{n_h}^O, X_1 = X_2 = \cdots = X_{n_h}$. Thus, equation (3.13) can be rewritten as:

$$O_{net}(k) = \sum_j^{n_h} W_j^O X_j(k) = n_h \cdot W_j X_j(k)$$

$$W_{\max}^O = \left|W_j^O\right| = \frac{\left|O_{net}(k)\right|}{n_h \cdot X_j(k)}$$

where n_h is the neuron number in the hidden layer, $O_{net}(k)$ is output of DRNNI, and $X_j(k)$ is the output of *j*th neuron in hidden layer.

If we choose a constant, Θ, which satisfies:

$$\Theta \geq \max\{O_{net}(k), I_{\max}\},$$

and this condition is held through the whole learning process then together with the condition: $1 < X_j(k) < 2$, (18a) and (18b) can be modified as below:

$$0 < \overline{\eta}^{O} < \frac{1}{2 \cdot n_h} \tag{4.22}$$

$$0 < \overline{\eta}^{I} < \frac{(n_h)^2}{(n_I + n_h) \cdot \Theta^4} \tag{4.23}$$

If we choose the learning rate as follows:

$$\overline{\eta} = \Omega_I \times \min\{\sup(\overline{\eta}^{O}), \sup(\overline{\eta}^{I})\} \qquad 0 < \Omega_I < 1 \tag{4.24}$$

then, the convergence will be guaranteed.

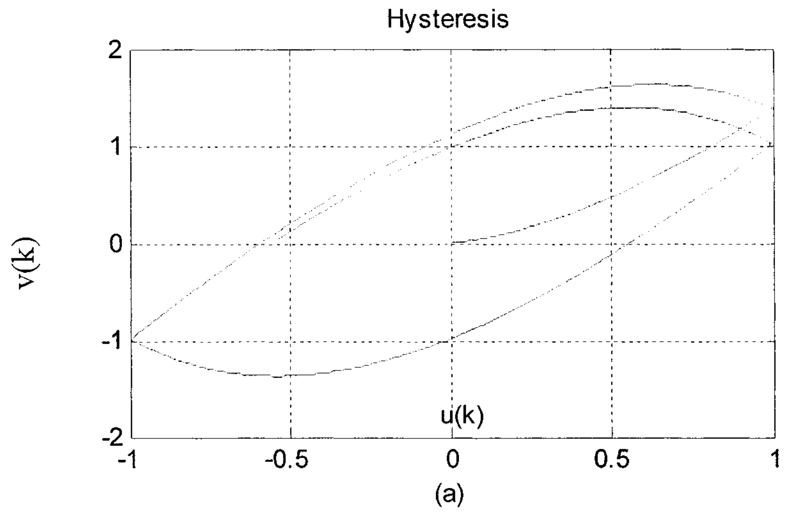
4.3.3 Simulation

In this simulation study, 2x5×1 DRNN is employed to act as DRNNI with the uniform initial weight value of 0.00001. The learning rate is chosen according to (4.24).

For the purpose of simulation to test the performance of hysteresis inverse modeling, a typical class of hysteresis described in section 4.1 is used.

The input signal is: $u(k) = A * \sin(k * ts)$ with ts=0.001, A=1, 2, 3, 4 and initial value $v(0) = 0$.

Figs 4.3- 4.10 show the hysteresis curves given by Equation (4.2), the learning curves, and learning errors with A=1,2,3,4. From all these Figures, it clearly shows the DRNN is capable of learning the hysteresis.



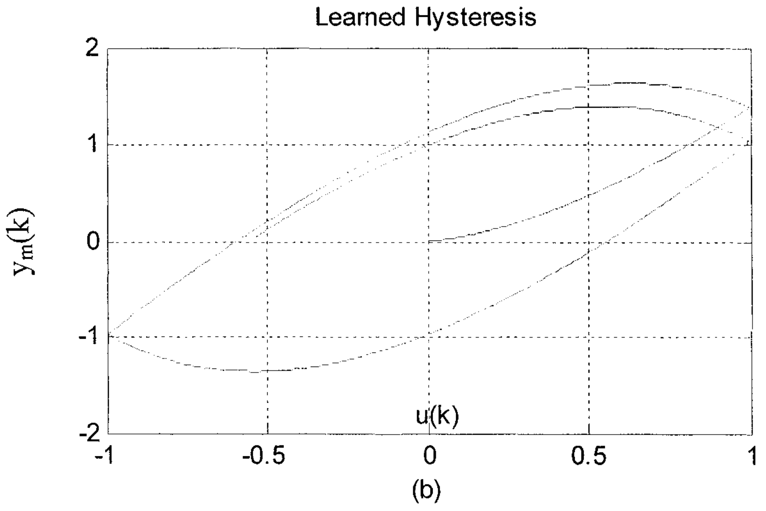
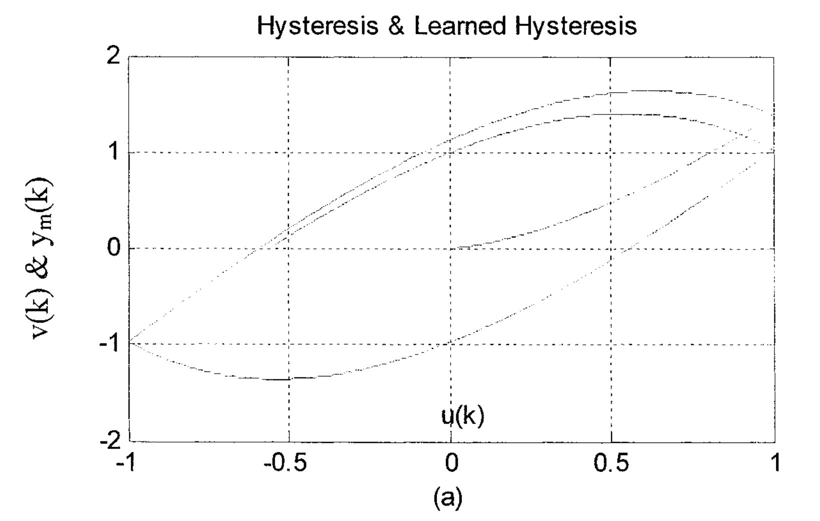


Figure 4.3 (a) Hysteresis curve (b) Learned hysteresis curve

For u(k)=sin(k*ts)



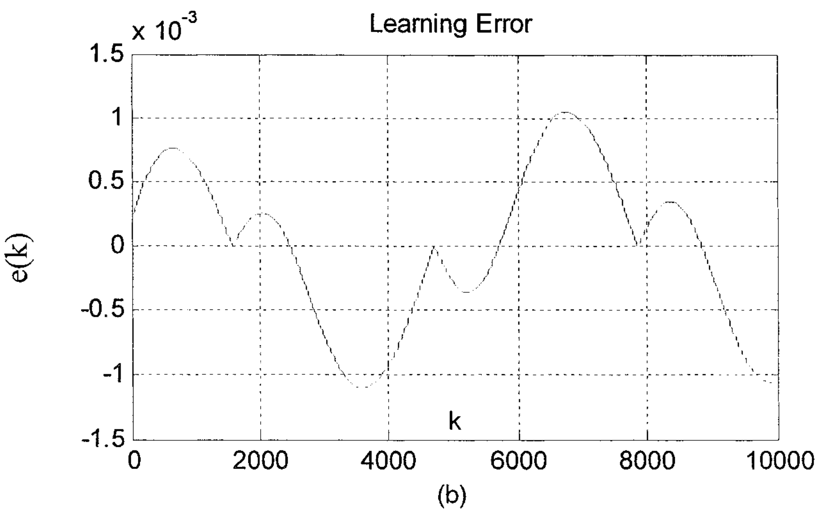
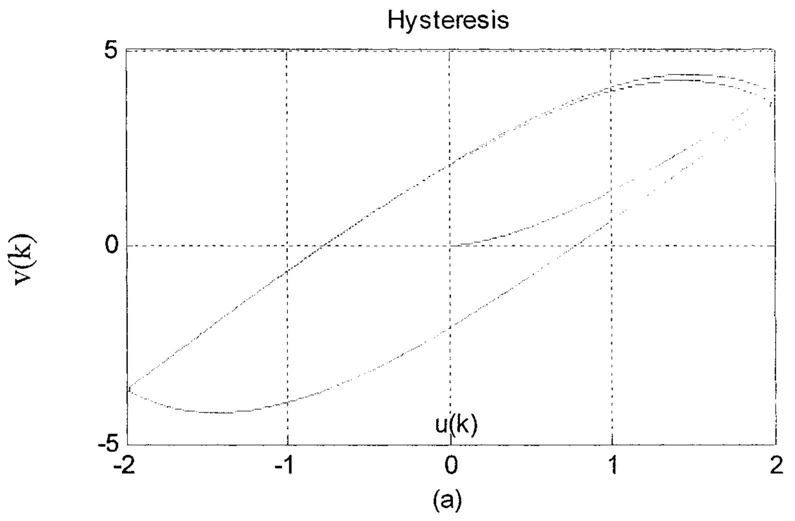


Figure 4.4 (a) Hysteresis curve & Learned hysteresis curve (b) Learning error

For u(k)=sin(k*ts)



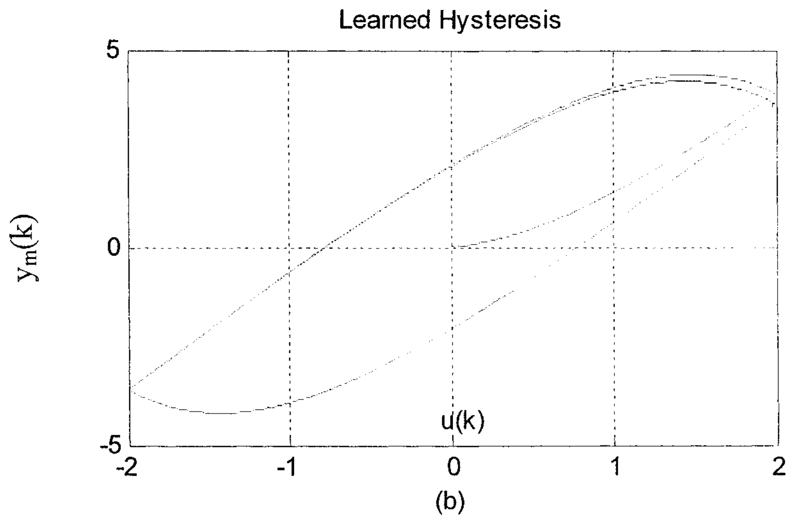


Figure 4.5 (a) Hysteresis curve (b) Learned hysteresis curve

For u(k)=2*sin(k*ts)

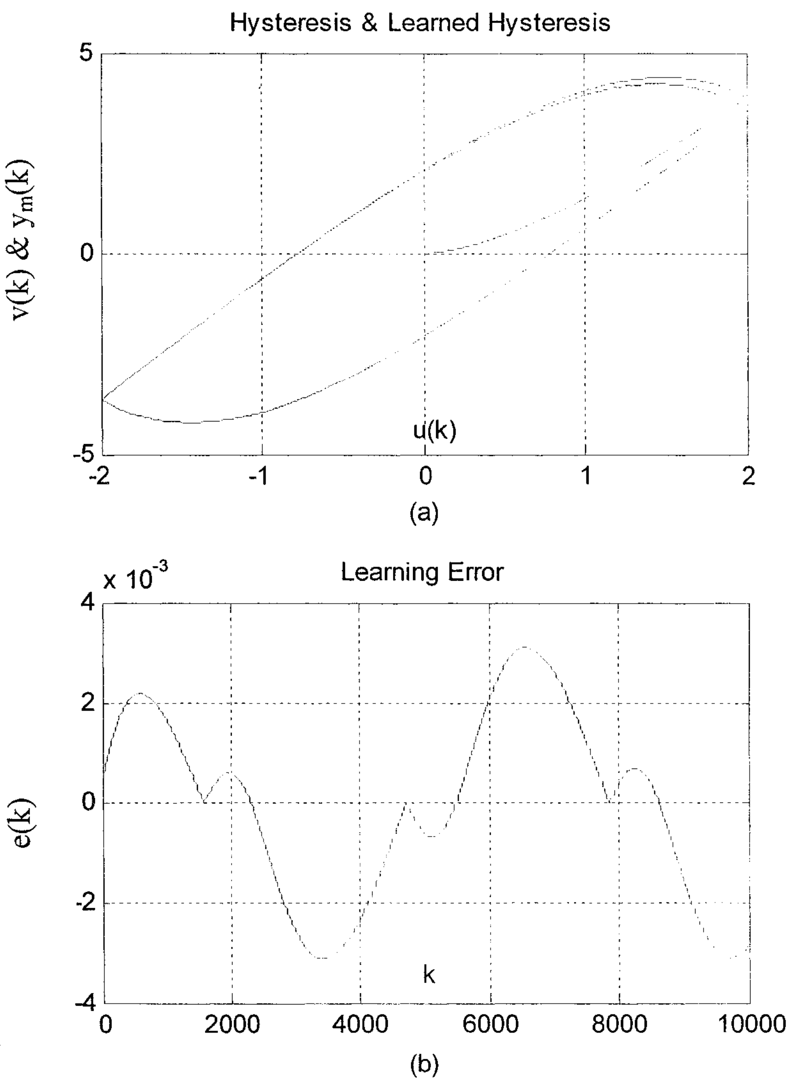
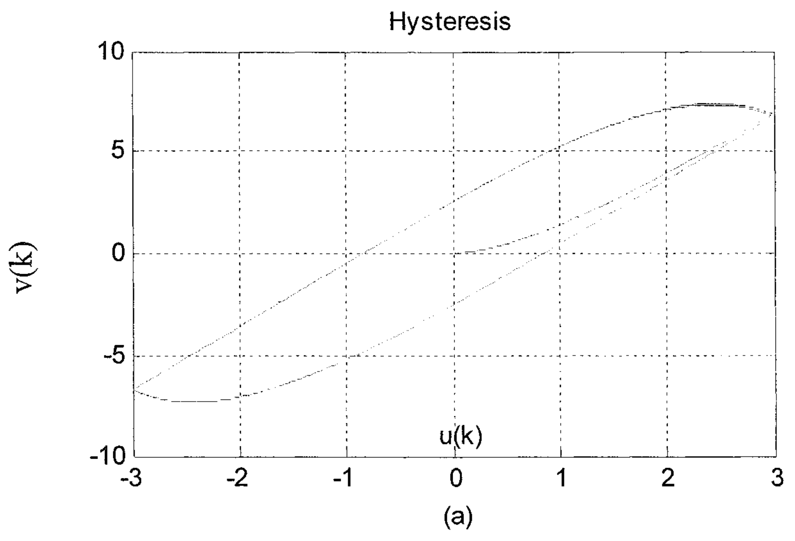


Figure 4.6(a) Hysteresis curve & Learned hysteresis curve (b) Learning error

For u(k)=2*sin(k*ts)



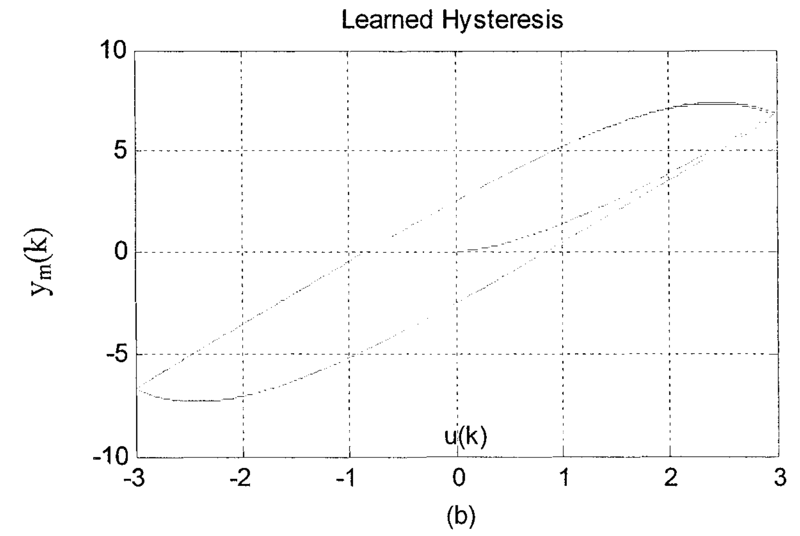
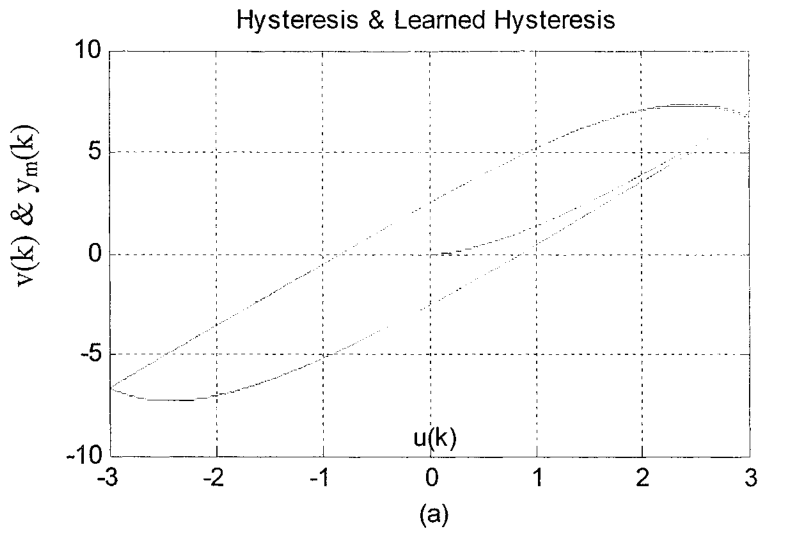


Figure 4.7 (a) Hysteresis curve (b) Learned hysteresis curve

For u(k)=3*sin(k*ts)



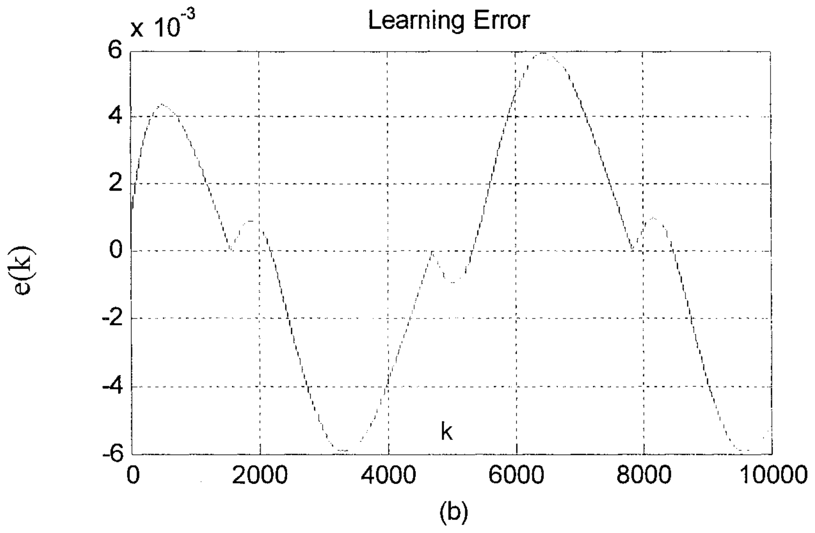
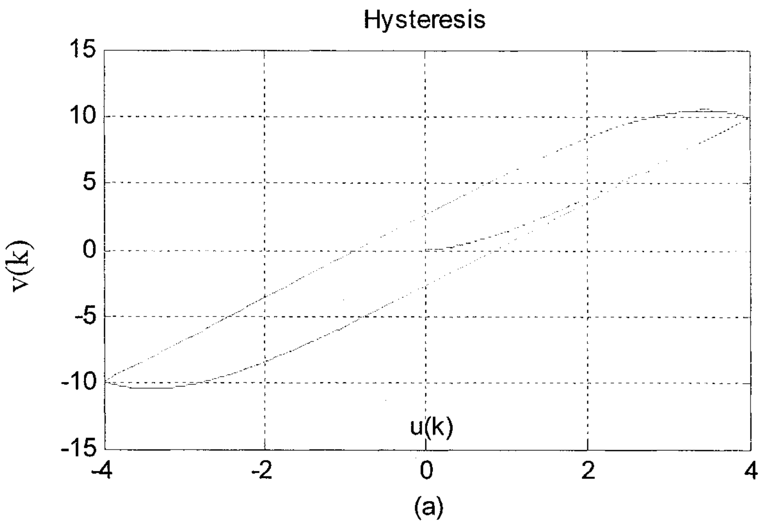


Figure 4.8 (a) Hysteresis curve & Learned hysteresis curve (b) Learning error

For u(k)=3*sin(k*ts)



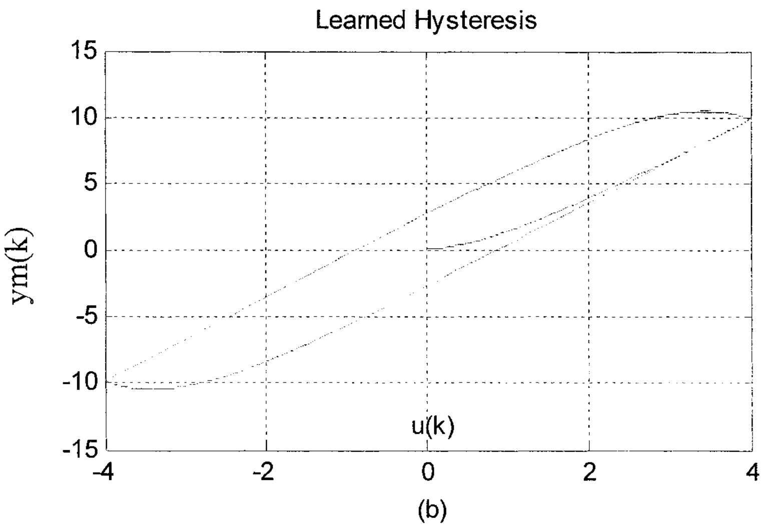
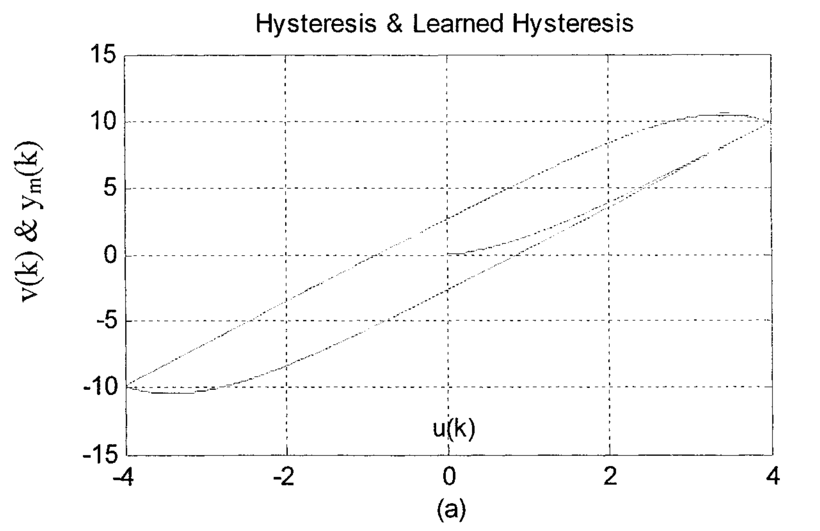


Figure 4.9 (a) Hysteresis curve (b) Learned hysteresis curve

For u(k)=4*sin(k*ts)



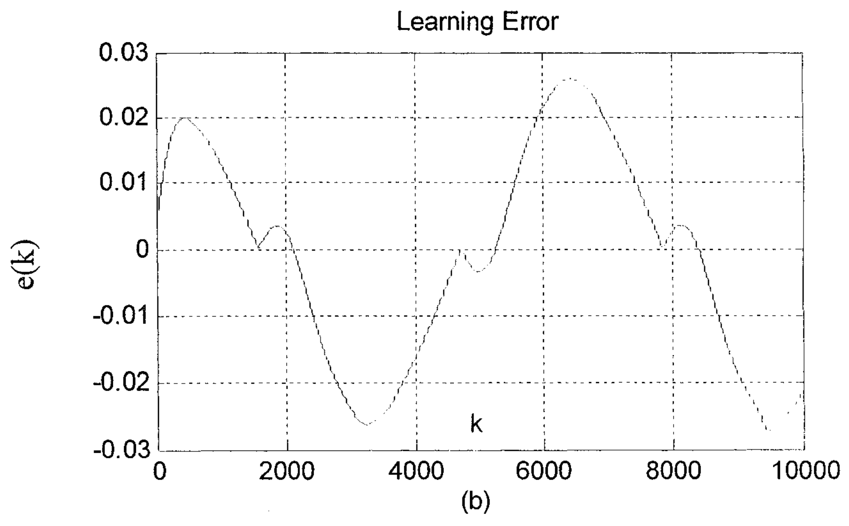


Figure 4.10 (a) Hysteresis curve & Learned hysteresis curve (b) Learning error

For u(k)=4*sin(k*ts)

4.4 Hysteresis Learning Algorithm for DRNNI with Fixed Input weights and Adaptive Output Weights

In section 4.3, we developed hysteresis learning algorithm for DRNN with adaptive input weights and adaptive output weights. For this learning algorithm, input weights and output weights need to be updated during the learning process. However, in the online application, we hope less parameter to be updated because less parameter to be trained less computation is required. So, in this section, hysteresis learning algorithm for DRNN with fixed input weights and adaptive output weights will be developed.

4.4.1 Description of weights update rule

Let $u(k)$ and $y_m(k)$ be the input and output of the DRNNI respectively. The error function can be defined as:

$$E = \frac{1}{2}{e_m}^2 = \frac{1}{2}(v(k) - y_m(k))^2 \tag{4.25}$$

The gradient of error in (4.3) with respect to $W^O \in R^{n_h}$ is represented by:

$$\frac{\partial E}{\partial W^O} = -e_m(k)\frac{\partial y_m(k)}{\partial W^O} = -e_m(k)\frac{\partial O_{net}(k)}{\partial W^O} \tag{4.26}$$

where $e_m(k) = v(k) - y_m(k)$ is the error between the reference input and output response of the DRNNI. And $O_{net}(k)$ is the output of DRNNI.

From (3.15), the gradient, with respect to the output weight, can be computed as:

$$\frac{\partial O_{net}(k)}{\partial W_j^O} = \frac{\partial\left[\sum_j W_j^O X_j(k)\right]}{\partial W_j^O} = X_j(k) \tag{4.27}$$

From (4.26) and (4.27), the negative gradient of the error with respect to a weight is:

$$-\frac{\partial E}{\partial W_j^O} = e_m(k) X_j(k) \tag{4.28}$$

The weight can now be adjusted following the steepest descent method, i.e., the update rule of the weights is described as follows:

$$W(k+1) = W(k) + \eta\left(-\frac{\partial E}{\partial W}\right) \tag{4.29}$$

The update rule equations for W^O can be described as:

$$W_j^O(k+1) = W_j^O(k) + \eta^O e_m(k) \cdot X_j \tag{4.30}$$

4.4.2 Learning Rate Estimate for DRNNI

Let W^O be a weight vector in R^{n_h}, η^O be the learning rate for the correspond weight of DRNNI, and $\|\cdot\|$ be the Euclidean norm in R^{n_h}. Then the convergence is guaranteed if η^O is chosen as:

$$0 < \eta^O < \frac{2}{\left\| \frac{\partial O_{net}(k)}{\partial W^O} \right\|^2} \tag{4.31}$$

<u>*Theorem 4.2*</u>:

Let η^O be the learning rate of the fully connected DRNNI's weight W^O. In the fixed-learning-rate learning process, if fixed input weights W^I are set at the same value and all adaptive weights W^O are set same initial value, then the dynamic back propagation algorithm converges if the learning rates are chosen as:

$$0 < \eta^{O} < \frac{1}{2 \cdot n_h} \tag{4.32}$$

Remark:

The proof of (4.31) and (4.32) are similar to that of (4.13) and (4.19a), and, therefore, will not be given in this section.

4.4.3 Choice of Input weights

Figure 4.11 shows the *j*th sigmoid neuron model

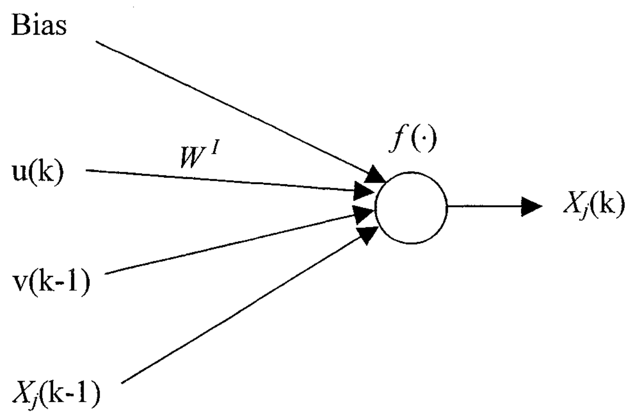


Figure 4.11 The *j*th sigmoid neuron model

From equation (3.17), we can obtain:

$$S_j(k) = X_j(k-1) + bias + W^I(u(k) + v(k-1)) \tag{4.33}$$

$$X_j(k) = f(S_j(k)) \tag{4.34}$$

$$f(S) = 0.5\tan(S) + 1.5 \tag{4.35}$$

The curve of equation (4.35) is shown in Figure 4.12.

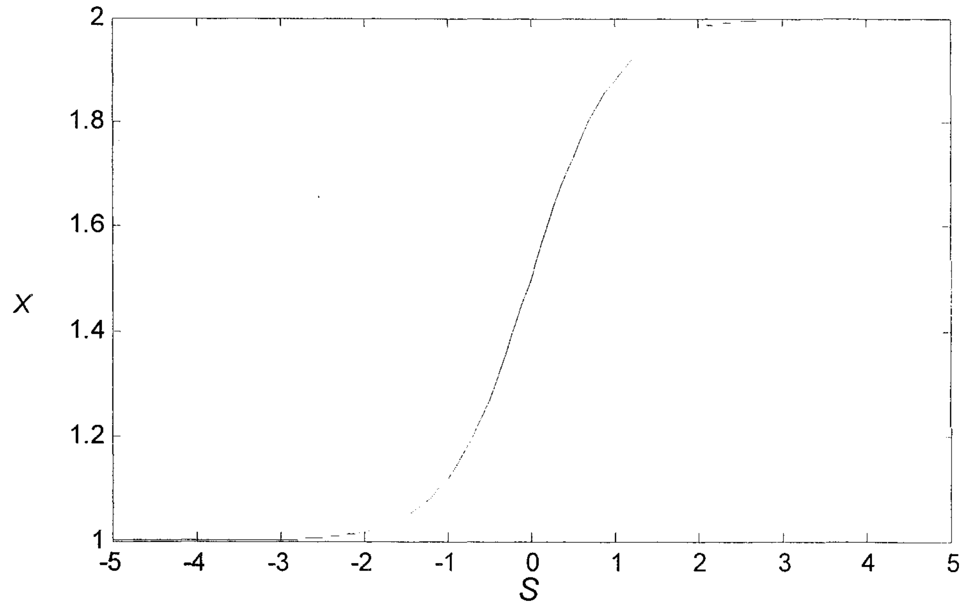


Figure 4.12 The curve of equation (4.35)

Figure 4.12 shows that the instauration range for S is (-4 - 4). In the learning process, if the input signal to the sigmoid neuron exceeds this range, it may result in a large learning error. So we can constrain S in a slight larger range (-5 5). Thus, from (4.33), we can obtain:

$$\left|X(k-1) + bias + W^I(u(k) + v(k-1))\right| \le 5 \tag{4.36}$$

Since $bias = 1$, from (4.36), we can choose W^I within the range as below:

$$\left|W^I\right| \le \frac{4 - X(k-1)}{\left|u(k) + v(k-1)\right|} \tag{4.37}$$

Since $\left|u(k)\right| + \left|v(k-1)\right| > \left|u(k) + v(k-1)\right|$, (4.37) can be modified as:

$$\left|W^I\right| \le \frac{4 - X(k-1)}{|u(k)| + |v(k-1)|} \tag{4.38}$$

Since $1 < X < 2$, and if we choose a parameter Θ which satisfies:

$\Theta \ge \max(|u(k)|, |v(k-1)|, bias)$, then, (4.38) can be rewritten as:

$$\left|W^I\right| \le \frac{1}{\Theta} \tag{4.39}$$

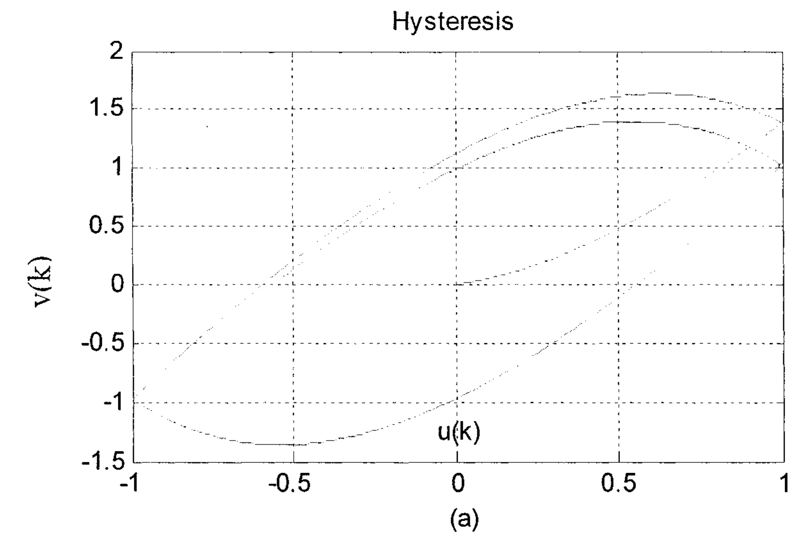
Since $\Theta \ge 1$, the input weights should be:

$$\left|W^I\right| \le 1 \tag{4.40}$$

4.4.4 Simulation

In this simulation study, 2x5x1 DRNN with fixed input weights is employed to act as DRNNI with the uniform initial weight value of 0.00001. The learning rate is chosen according to (4.22).

For the purpose of simulation to test the performance of hysteresis inverse modeling, a typical class of hysteresis described in section 4.1 is used. The input signal is $u(k) = A * \sin(k * ts)$ with ts=0.001, A=1, 2, 3, 4 and initial value $v(0) = 0$. The simulation results are shown in Figs 4.13-4.20. it also shows that the DRNN with fixed input weights can learn the hysteresis very well.



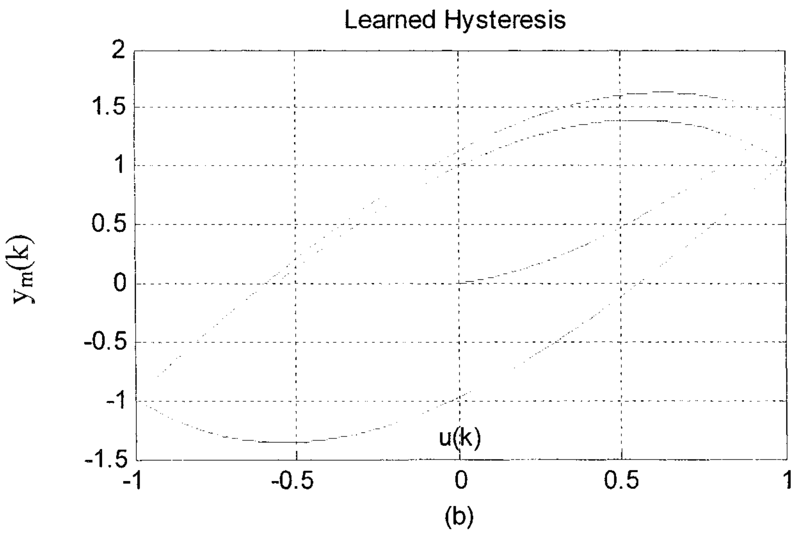


Figure 4.13 (a) Hysteresis curve (b) Learned hysteresis curve

For u(k)=sin(k*ts)

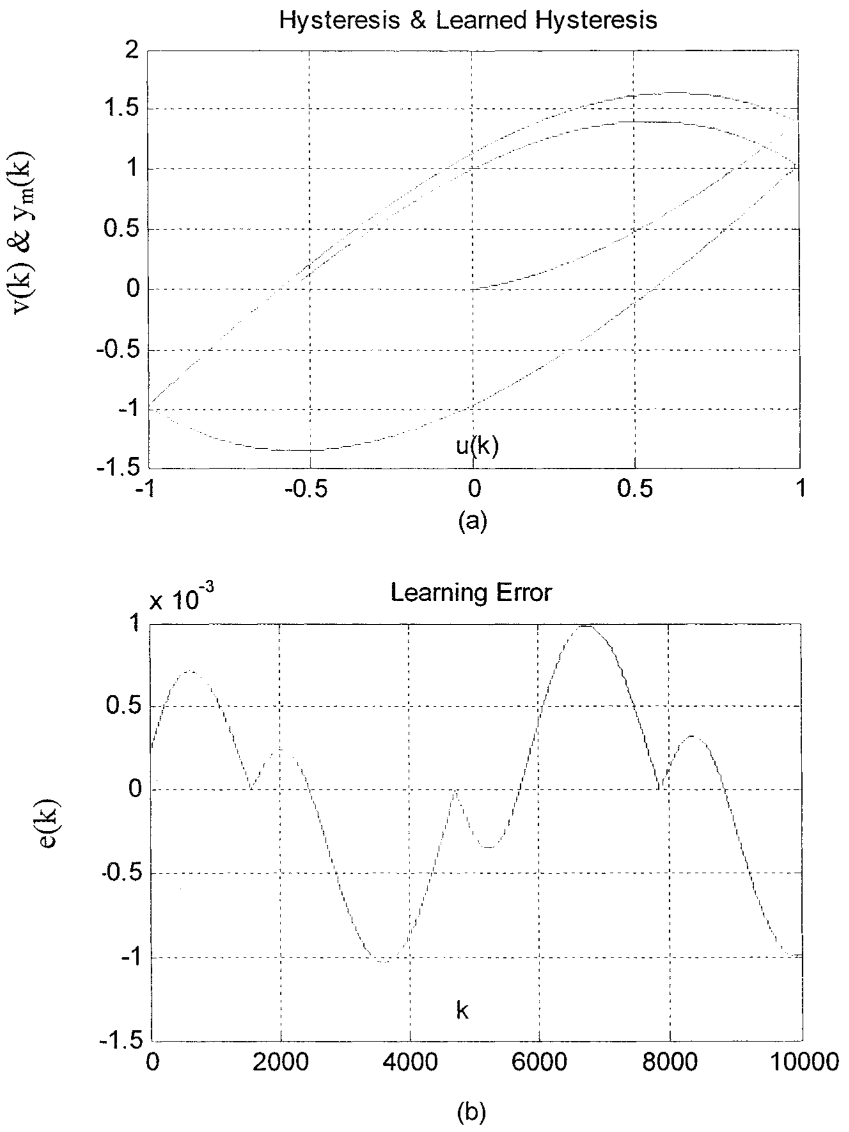
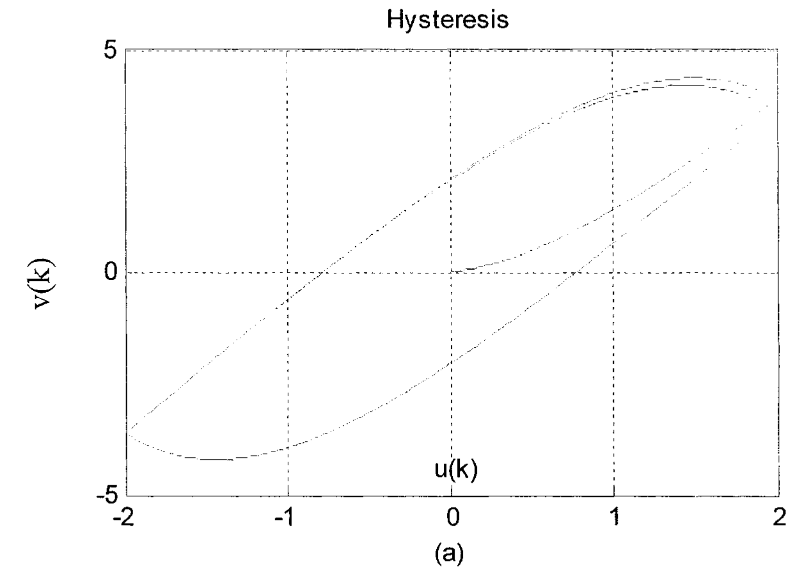


Figure 4.14 (a) Hysteresis curve & Learned hysteresis curve (b) Learning error

For u(k)=sin(k*ts)



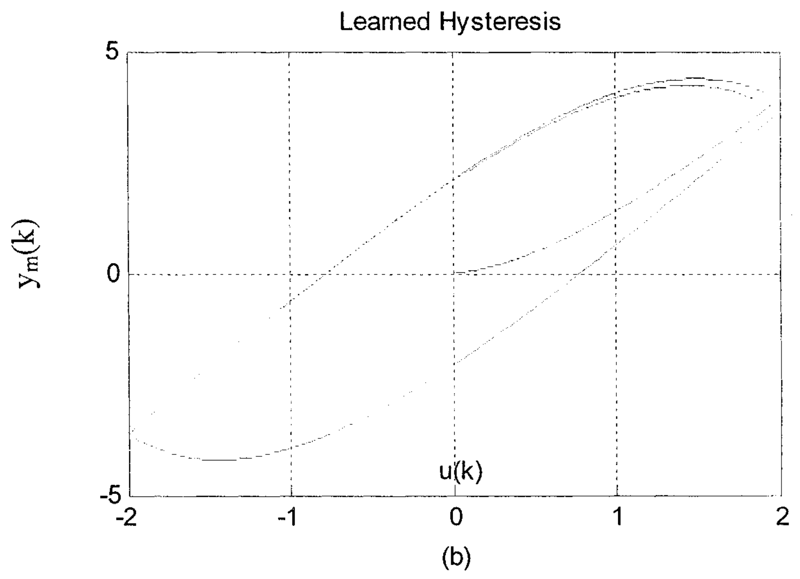


Figure 4.15 (a) Hysteresis curve (b) Learned hysteresis curve

For u(k)=2*sin(k*ts)

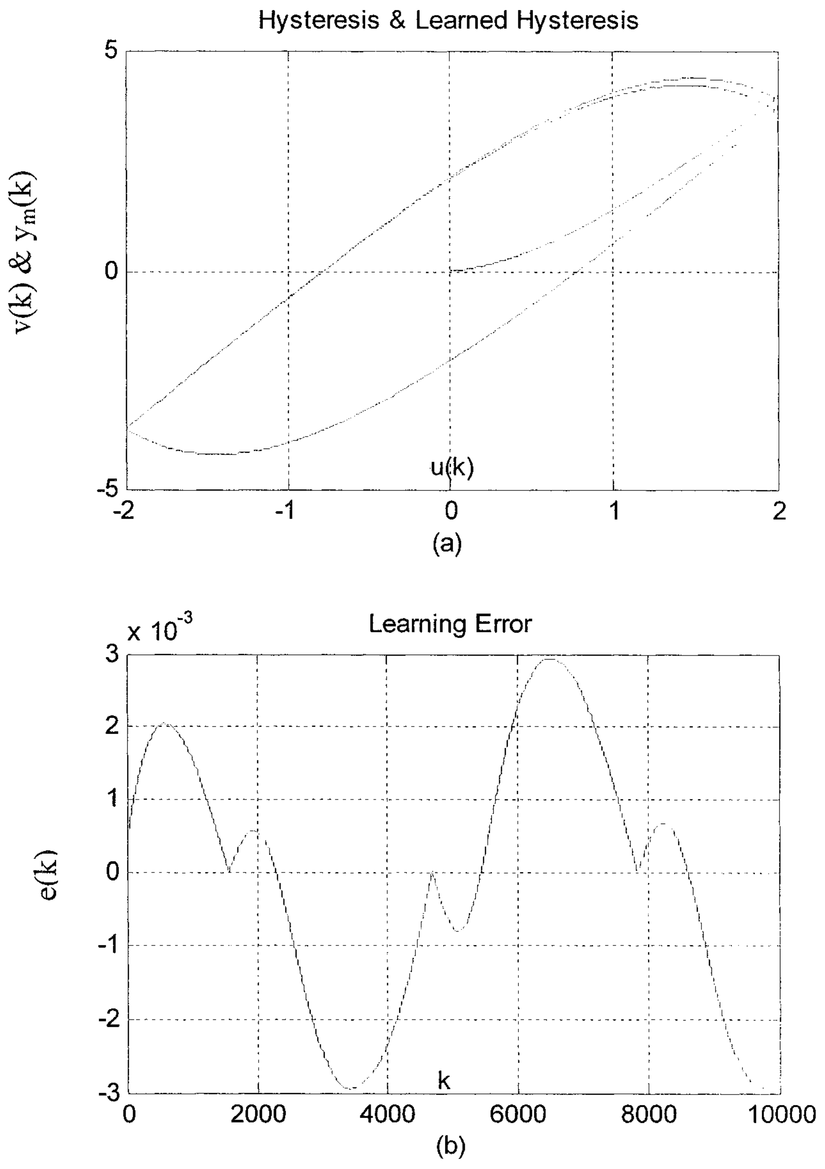
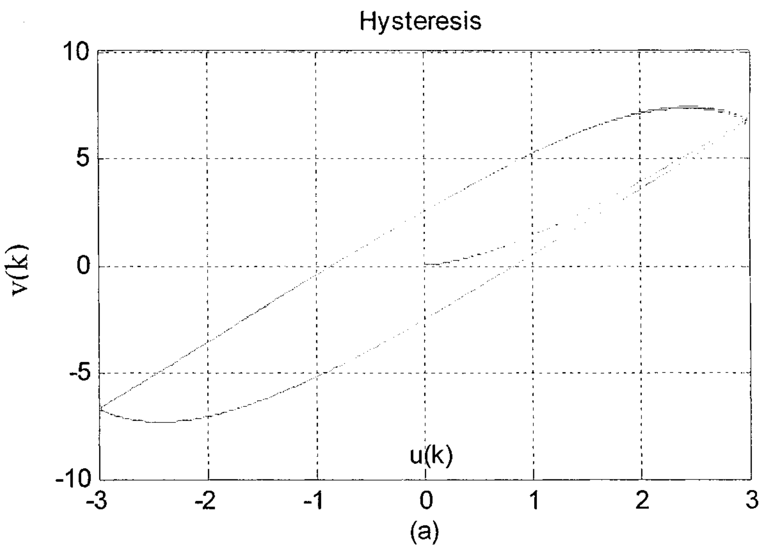


Figure 4.16 (a) Hysteresis curve & Learned hysteresis curve (b) Learning error

For u(k)=2*sin(k*ts)



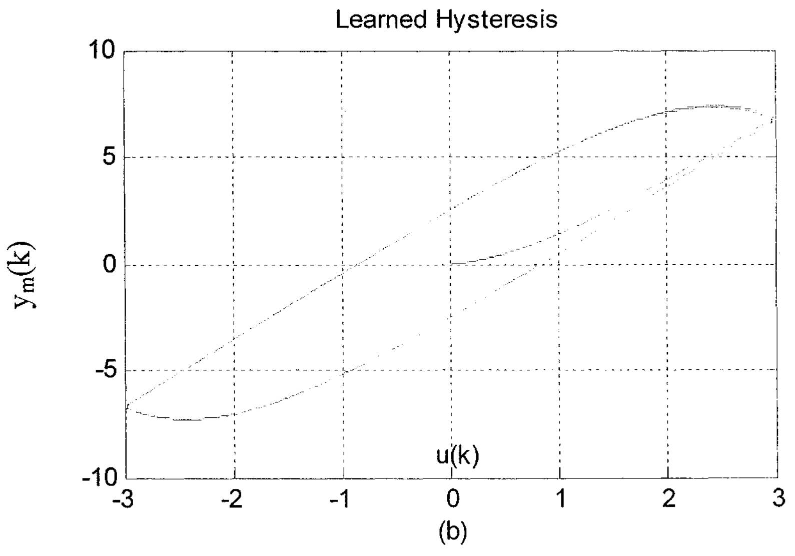
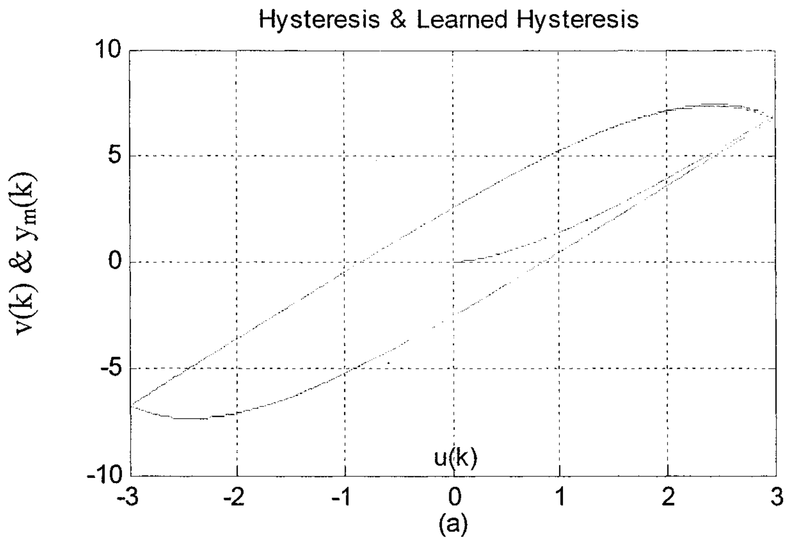


Figure 4.17 (a) Hysteresis curve (b) Learned hysteresis curve

For u(k)=3*sin(k*ts)



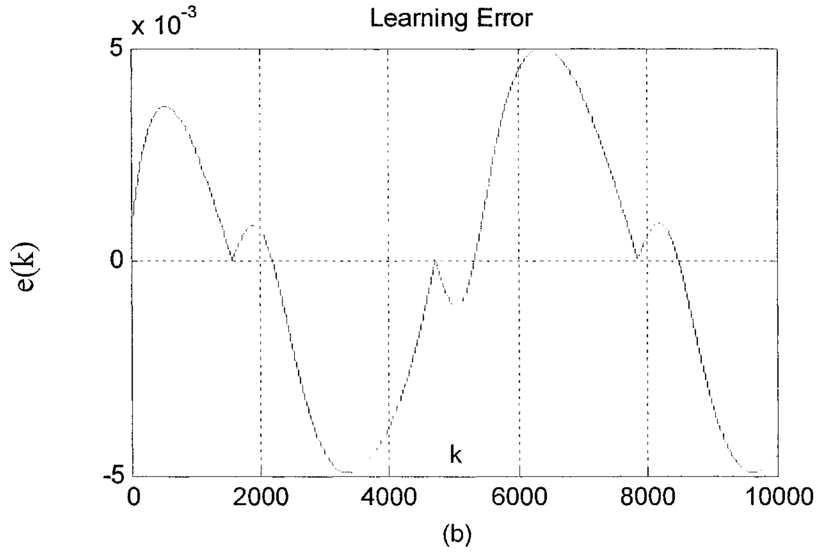
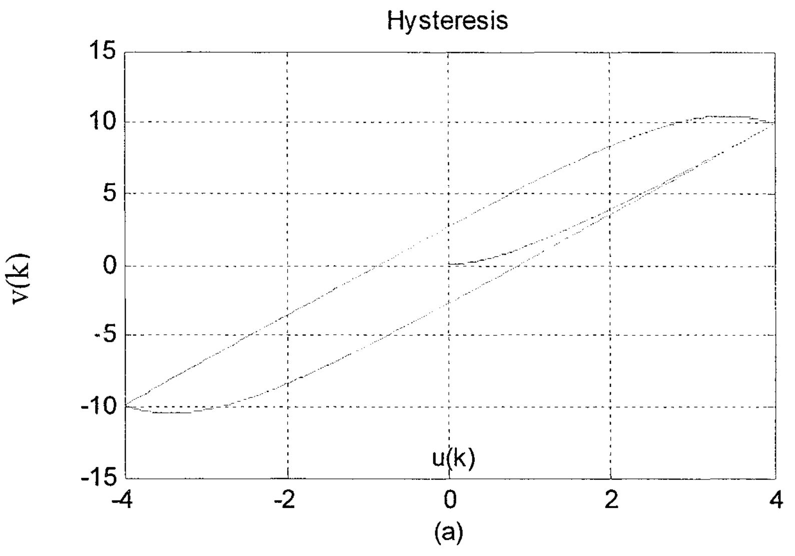


Figure 4.18 (a) Hysteresis curve & Learned hysteresis curve (b) Learning error

For u(k)=3*sin(k*ts)



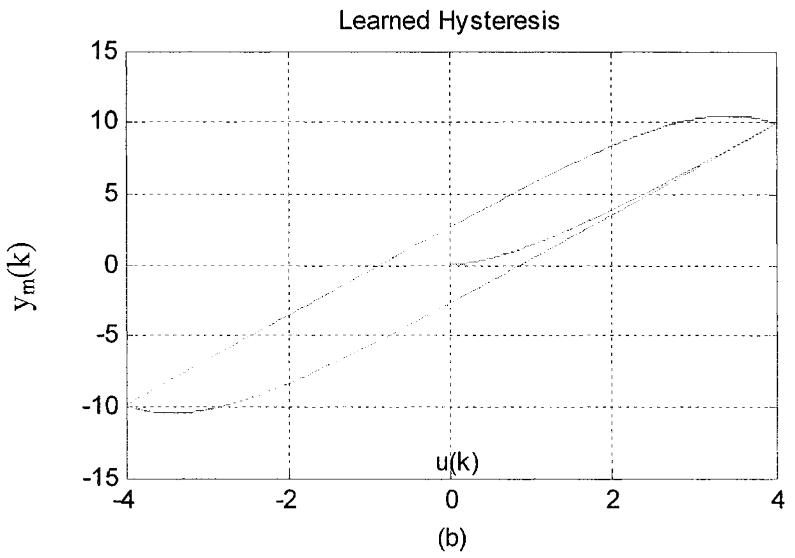
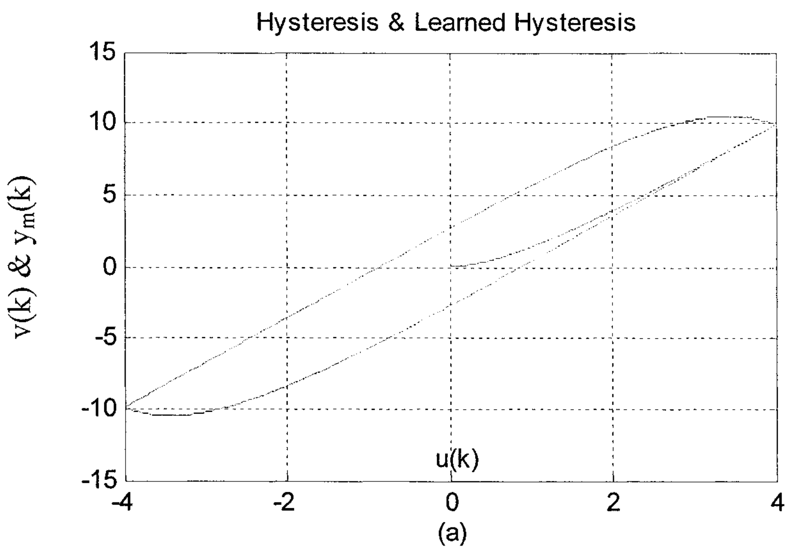


Figure 4.19 (a) Hysteresis curve (b) Learned hysteresis curve

For u(k)=4*sin(k*ts)



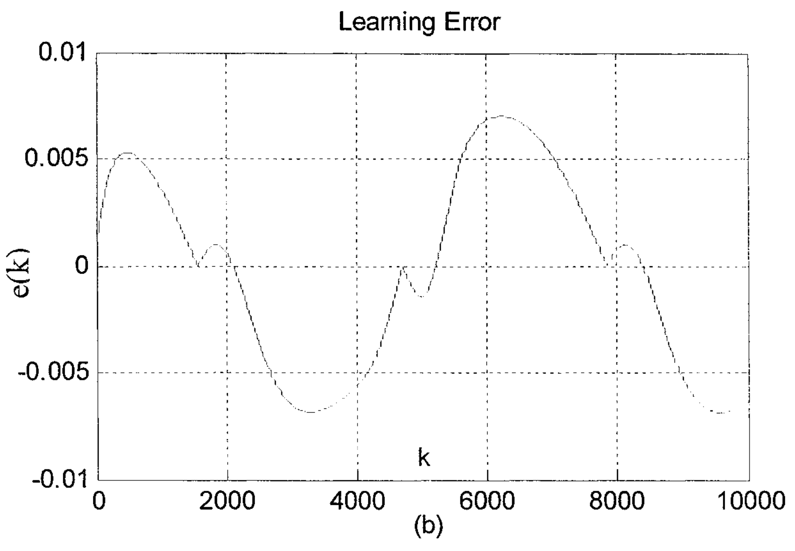


Figure 4.20 (a) Hysteresis curve & Learned hysteresis curve (b) Learning error

For u(k)=4*sin(k*ts)

4.5 Summary

In this chapter, two types of DRNN identifiers have been presented; one is a $2 \times 5 \times 1$ *Diagonal Recurrent Neural Networks* with adaptive input and output weights, and the other is a $2 \times 5 \times 1$ *Diagonal Recurrent Neural Networks* with fixed input weights and adaptive output weights. Both of identifiers perform high learning precise, but the second one performs better than the first for the hysteresis learning.

Table 4.1 Hysteresis learning accuracies with fixed input weights and adaptive input weights

Max Learning Error corresponding reference input:	DRNNI with adaptive input and output weights:	DRNNI with fixed input weights and adaptive output weights:
sin(k*ts)	0.11%	0.11%
2*sin(k*ts)	0.16%	0.15%
3*sin(k*ts)	0.2%	0.17%

4*sin(k*ts)	0.65%	0.18%

Chapter 5

Hysteresis Inverse Learning Without Hysteresis Identifier

5.1 General structure for Hysteresis Inverse Learning

The block diagram of the diagonal recurrent neural network (DRNN) based hysteresis inverse learning system is shown in Figure 5.1. In this diagram, the hysteresis compensator is denoted as DRNNC, the abbreviation of "diagonal recurrent neural network based hysteresis compensator."

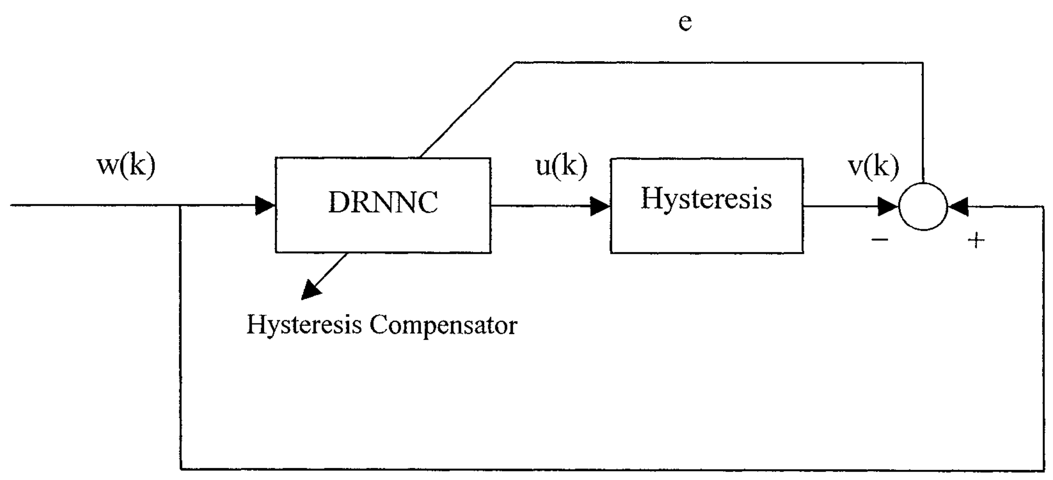


Figure 5.1 Block diagram of inverse learning of Hysteresis

5.2 Hysteresis inverse learning with identifier

The block diagram of neural networks based hysteresis inverse learning structure with identifier is shown in Figure 5.2. The hysteresis identifier is denoted as DRNNI, the abbreviation of “diagonal recurrent neural network based hysteresis identifier.” In Chapter 4, it has been proven that multilayer DRNN with fixed input weights can achieve precision, that is, low error, in hysteresis learning. So, throughout this chapter, DRNNI refers to three-layer DRNN with fixed input weights.

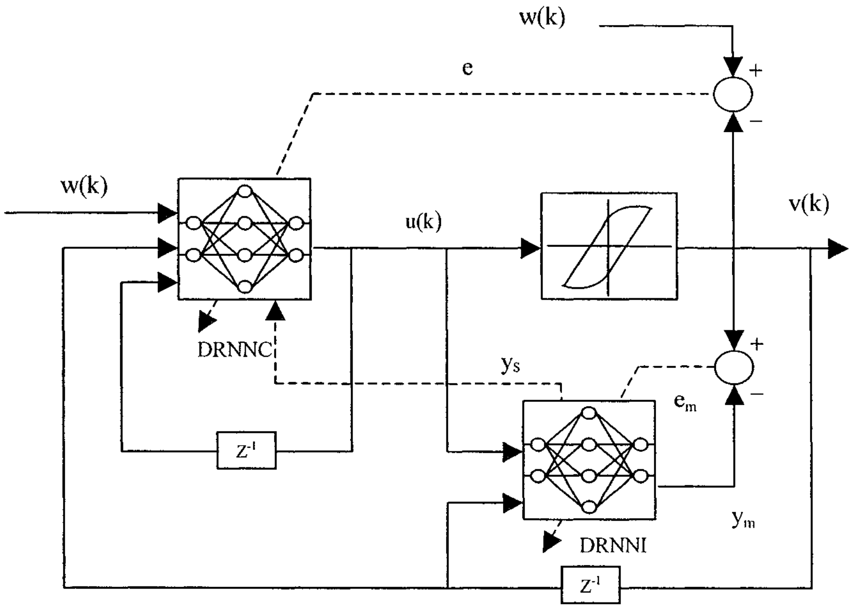


Figure 5.2 Block diagram of Hysteresis inverse learning with identifier

5.2.1 Learning Algorithm

Let *w(k)* and *v(k)* be the input of DRNNC and output of the hysteresis model. The error function can be defined as:

$$E = \frac{1}{2}e^2 = \frac{1}{2}(w(k) - v(k))^2 \tag{5.1}$$

The gradient of error in (5.1) with respect to an arbitrary weight vector $W \in R^n$ is represented by

$$\frac{\partial E}{\partial W} = -e(k)\frac{\partial v(k)}{\partial W} = -e(k)\frac{\partial v(k)}{\partial u(k)}\frac{\partial u(k)}{\partial W} = -e(k)y_S\frac{\partial u(k)}{\partial W}$$

$$= -e(k) y_S \frac{\partial O_{net}(k)}{\partial W} \tag{5.2}$$

where $e(k) = v(k) - w(k)$ represents the error between the reference input and output response of the hysteresis model, and the factor $y_S \equiv \frac{\partial v(k)}{\partial u(k)}$ represents the sensitivity of the hysteresis with respect to its input, and $O_{net}(k)$ is the output of DRNNC. Since the hysteresis is normally unknown, the sensitivity needs to be estimated by DRNNI.

The output gradients with respect to output and input weights can be computed using the following equations respectively:

$$\frac{\partial O_{net}(k)}{\partial W_j^o} = X_j(k) \tag{5.3a}$$

$$\frac{\partial O_{net}(k)}{\partial W_{ij}^I} = W_j^o Q_{ij}(k) \tag{5.3b}$$

where $Q_{ij}(k) \equiv \frac{\partial X_j(k)}{\partial W_{ij}^I}$, which satisfies:

$$Q_{ij}(k) = f'(S_j)[I_i(k) + Q_{ij}(k-1)] \qquad Q_{ij}(0) = 0 \tag{5.4}$$

Note: Equations (5.4) are nonlinear dynamic recursive equations for the gradients $\frac{\partial X_j(k)}{\partial W}$, and can be solved recursively with given initial conditions.

Remark:

The proof of (5.3a), (5.3b) and (5.4) are similar to that of (4.5a), (4.5b) and (4.6), and, therefore, will not be given in this section.

From (5.2) and (5.3), the negative gradient of the error with respect to a weight are

$$-\frac{\partial E}{\partial W_j^o} = e(k) y_S X_j(k) \tag{5.7a}$$

$$-\frac{\partial E}{\partial W_{ij}^I} = e(k) y_S W_j^o Q_{ij}(k) \tag{5.7b}$$

The weight can now be adjusted following the steepest descent method, i.e., the update rule of the weights is described as follows:

$$W(k+1) = W(k) + \eta(-\frac{\partial E}{\partial W}) \tag{5.8}$$

where the weight W can be W^o or W^I, and η is the learning rate respectively.

The update rule equations for individual weights can be described respectively as follows:

$$W_j^O(k+1) = W_j^O(k) + \eta^O \cdot y_S \cdot e(k) \cdot X_j \tag{5.9a}$$

$$W_{ij}^I(k+1) = W_j^I(k) + \eta^I \cdot y_S \cdot e(k) \cdot W_j^O(n) Q_{ij} \tag{5.9b}$$

5.2.2 Learning Rate Estimate for DRNNC

Let W be an arbitrary weight vector in R^n, η be the learning rate for the corresponding weight of DRNNC, and $\|\cdot\|$ be the Euclidean norm in R^n. Then the convergence is guaranteed if η is chosen as:

$$0 < \eta < \frac{2}{\left\| \frac{\partial O_{net}(k)}{\partial W} \right\|^2 {y_S}^2} \tag{5.10}$$

Proof of (5.10):

Give a Lyapunov function as:

$$\Delta V(k) = V(k+1) - V(k)$$

$$= \frac{1}{2}[e^2(k+1) - e^2(k)]$$

$$= \frac{1}{2}\Delta e(k)[2e(k) + \Delta e(k)] \qquad (5.11)$$

If ΔW represents a change in an arbitrary weight vector in R'', the error difference due to the learning can be presented as below:

$$\Delta e(k) = e(k+1) - e(k) = \left[\frac{\partial e(k)}{\partial W}\right]^T \Delta W \qquad (5.12)$$

From update rules (5.2) and (5.8),

$$\Delta W = \eta \cdot e(k) \cdot y_S \cdot \frac{\partial O_{net}(k)}{\partial W} \qquad (5.13)$$

Substitute (5.12) and (5.13) into (5.11), we obtain:

$$\Delta V(k) = \frac{1}{2}\left[\frac{\partial e(k)}{\partial W}\right]^T \eta \cdot e(k) \cdot y_S \cdot \frac{\partial O_{net}(k)}{\partial W} \cdot \left\{2e(k) + \left[\frac{\partial e(k)}{\partial W}\right]^T \eta \cdot e(k) \cdot y_S \frac{\partial O_{net}(k)}{\partial W}\right\}$$

(5.14)

Since $\frac{\partial e(k)}{\partial W} = -y_S \frac{\partial O_{net}(k)}{\partial W}$, equation (5.14) can be rewritten as:

$$\Delta V = -\eta \cdot e^2(k) \cdot y_S{}^2 \left\|\frac{\partial O_{net}(k)}{\partial W}\right\|^2 + \frac{1}{2}\eta^2 e^2(k) \cdot y_S{}^4 \left\|\frac{\partial O_{net}(k)}{\partial W}\right\|^4$$

$$= -\eta \cdot e^2(k) \cdot y_S{}^2 \left\|\frac{\partial O_{net}(k)}{\partial W}\right\|^2 \left(1 - \frac{1}{2} \cdot \eta \cdot y_S{}^2 \left\|\frac{\partial O_{net}(k)}{\partial W}\right\|^2\right) \qquad (5.15)$$

If the convergence must be guaranteed, then $\Delta V(k) < 0$, thus:

$$1-\frac{1}{2}\cdot\eta\cdot {y_S}^2\left\|\frac{\partial O_{net}(k)}{\partial W}\right\|^2>0 \tag{5.16}$$

Thus (4.10) follows.

Theorem 5.1:

Let η^O and η^I be the learning rates of the DRNNI weights W^O and W^I respectively. Then the dynamic back propagation algorithm converges if the learning rates are chosen as:

$$0<\eta^O<\frac{1}{2\cdot n_h\cdot {y_S}^2} \tag{5.17a}$$

$$0<\eta^I<\frac{2}{(n_I+n_h)\cdot {W^O_{\max}}^2\cdot {I_{\max}}^2\cdot {y_S}^2} \tag{5.17b}$$

where n_I is the neuron number of input layer, n_h is the neuron number on hidden layer in DRNNC, $W^O{}_{\max}:=\max_k\|W_i^O(k)\|$, $\|W_i^O(k)\|:=\max_k|W_j^O(k)|$, $I_{\max}=\max_k\|I(k)\|$, $I(k)$ is the input vector of DRNN at time k, and $\|\cdot\|$ is the sup-norm.

Proof of (5.17a)

From (5.3a),

$$\frac{\partial O_{net}(k)}{\partial W^O}=X(k)$$

where $X=\left[X_1,X_2,\cdots,X_{n_h}\right]^T$, and X_j is the output value of the jth neuron in the hidden layer, and n_h is the neuron number in the hidden layer.

Since $1<X_j<2$, $j=1,2,\cdots,n_h$, then we have

$$\left\|\frac{\partial O_{net}(k)}{\partial W^O}\right\|^2 < 4 \cdot n_h \tag{5.18}$$

From (5.18) and (5.10), (5.17a) can be obtained.

Proof of (5.17b)

In equation (5.3b), we define $Q_{ij}(k) \equiv \frac{\partial X_j(k)}{\partial W_{ij}^I}$, and from (5.4)

$$Q_{ij}(k) = f'(S_j(k))[I_i(k) + Q_{ij}(k-1)]$$

Thus the solution of the above equation can be written as:

$$Q_{ij}(k) = \sum_{m=1}^{k}\left(\left[\prod_{n=0}^{k-m} f'(S_j(k-n))\right] I_i(m)\right) + \left[\prod_{n=0}^{k-1} f'(S_j(k-n))\right] Q_{ij}(0)$$

Note that $Q_{ij}(0) = 0$

$$Q_{ij}(k) = \sum_{m=1}^{k}\left(\left[\prod_{n=0}^{k-m} f'(S_j(k-n))\right] I_i(m)\right)$$

$$\left|Q_{ij}(k)\right| \le \sum_{m=0}^{k}\left(\left[\prod_{n=0}^{k-m} \left|f'(S_j(k-n))\right|\right] \left|I_i(m)\right|\right)$$

Since $f'(S_j(k-n)) < 0.5 \quad (0 \le n < k < +\infty)$,

$$\left|Q_{ij}(k)\right| \le \sum_{m=0}^{k} 0.5^{(k-m)} I_i(m) \le \sum_{m=0}^{k} 0.5^{(k-m)} I_{\max} = (0.5 + 0.5^2 + \cdots) I_{\max} \le I_{\max}$$

where $I_{\max} := \max\left[bias,\ |w_{\max}|,\ |u_{\max}|,\ |v_{\max}|\right]$

Thus,

$$\|Q(k)\|^2 \le (n_I + n_h) I_{\max}{}^2$$

From the definition, $\frac{\partial O_{net}(k)}{\partial W_{ij}^I} = W_j^o Q_{ij}(k)$, we can obtain:

$$\left| \frac{\partial O_{net}(k)}{\partial W_{ij}^I} \right| \leq W_{\max}^O \cdot \left| Q_{ij}(k) \right|$$

Here, we define $W^O{}_{\max} := \max \left| W_j^O \right|$. Thus,

$$\left\| \frac{\partial O_{net}(k)}{\partial W^I} \right\|^2 \leq W_{\max}^{O\ 2} \cdot (n_I + n_h) I_{\max}{}^2 \tag{5.19}$$

Hence, from (5.19) and (5.10), (5.17b) follows.

5.2.3 Sensitivity Estimate for Hysteresis

In order to make a distinction, we put a star "*" on the left upper corner of parameters of DRNNI.

From the definition

$$y_S(k) = \frac{\partial v(k)}{\partial u(k)} \cong \frac{\partial y_m(k)}{\partial u(k)} \tag{5.20}$$

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

Applying the chain rule to (5.20), and noting that $y_m(k) = {}^*O_{net}(k)$ of (3.13),

$$\frac{\partial y_m(k)}{\partial u(k)} = \frac{\partial^* O_{net}(k)}{\partial u(k)} = \sum_j \frac{\partial^* O_{net}(k)}{\partial^* X_j(k)} \frac{\partial^* X_j(k)}{\partial u(k)} = \sum_j {}^*W_j^o \frac{\partial^* X_j(k)}{\partial u(k)} \tag{5.21}$$

From (3.14),

$$\frac{\partial^* X_j(k)}{\partial u(k)} = f'(^*S_j(k))\frac{\partial^* S_j(k)}{\partial u(k)} \tag{5.22}$$

Since inputs to the DRNNI are $u(k)$ and $v(k-1)$ from (3.15), we can obtain:

$$^*S_j(k) = ^*X_j(k-1) + ^*W_{1j}^I u(k) + ^*W_{2j}^I v(k-1) + ^*W_{3j}^I b_I \tag{5.23}$$

where b_I is the bias input of DRNNI. Thus,

$$\frac{\partial^* S_j(k)}{\partial u(k)} = ^*W_{1j}^I \tag{5.24}$$

from (5.21), (5.22), and (5.24),

$$y_S(k) \cong \sum_j {}^*W_j^O f'(^*S_j(k))^*W_{1j}^I \tag{5.25}$$

If we set same initial value for all output weights $^*W_j^O$, and apply fixed learning rates for weights adaptive, then the output weights have the properties: $^*W_1^O = ^*W_2^O = \cdots = ^*W_{n_h}^O = ^*W^O$. Moreover, if input weights are set at the same value as: $^*W_{ij}^I = ^*W^I$, together with the condition: $0 < f'(^*S_j) < 0.5$, then from (5.25), we can obtain:

$$|y_S(k)| \le \frac{^*n_h}{2}\left|^*W^O\right| \cdot \left|^*W^I\right| \tag{5.26}$$

where $^*n_h$ represents the number of neurons in the hidden layer of DRNNI.

5.3 Inverse Learning Algorithm without Identifier

During the hysteresis inverse learning process, which applies the learning structure with the hysteresis identifier, two neural networks are trained simultaneously. Information transfers between them while they are training which allows the two networks to affect each other and results in several disadvantages. A high level of randomness and uncertainty will arise, especially at the beginning of networks training. This learning structure depends almost entirely on the choice of initial value. Therefore, if we can not choose the correct initial value, the hysteresis inverse learning based on this structure may fail. Another disadvantage of this learning structure is that computation will rise since two networks are being trained at same time. Thus a simple structure is desired in the real-time application. In this section, we will propose a simple structure, by which hysteresis inverse learning is done without using an identifier.

From (5.17), we can make a modification of:

$$0 < \left|\eta^O y_S\right| < \frac{1}{2 \cdot n_h \cdot \left|y_S\right|} \tag{5.27a}$$

$$0 < \left|\eta^I y_S\right| < \frac{2}{(n_I + n_h) \cdot W^{O\,2}_{\max} \cdot I_{\max}{}^2 \cdot \left|y_S\right|} \tag{5.27b}$$

Since (5.26), (5.27a) and (5.27b) can be further modified as:

$$0 < \left|\eta^O y_S(k)\right| < \frac{1}{n_h{}^* n_h} \cdot \frac{1}{\left|{}^*W^O\right| \cdot \left|{}^*W^I\right|} \tag{5.28a}$$

$$0 < \left|\eta^I y_S(k)\right| < \frac{4}{(n_I + n_h) \cdot{}^* n^h} \cdot \frac{1}{W^{O\,2}_{\max} \cdot I_{\max}{}^2 \cdot \left|{}^*W^O\right| \cdot \left|{}^*W^I\right|} \tag{5.28b}$$

In the training process of DRNNC, if we set the same initial value for all weights and apply fixed learning rates for weights adaptive, we can have the property: $W_1^O = W_2^O = \cdots W_{n_h}^O . X_1 = X_2 = \cdots = X_{n_h}$. Thus, from equation (3.13) we can obtain:

$$O_{net}(k) = \sum_j^{n_h} W_j^O X_j(k) = n_h \cdot W_j X_j(k)$$

$$W_{\max}^O = \left|W_j^O\right| = \frac{\left|O_{net}(k)\right|}{n_h \cdot X_j(k)} \tag{5.29a}$$

where n_h is the neuron number in the hidden layer, $O_{net}(k)$ is the output of DRNNC, and $X_j(k)$ is the output of *j*th neuron in hidden layer.

Similarly,

$$\left|{}^*W_j^O\right| = \frac{\left|{}^*O_{net}(k)\right|}{{}^*n_h \cdot {}^*X_j(k)} \tag{5.29b}$$

where ${}^*n_h$ is the neuron number in the hidden layer, ${}^*O_{net}(k)$ is the output of DRNNI, and ${}^*X_j(k)$ is the output of *j*th neuron in hidden layer.

Substitute (5.29a), (5.29b) into (5.28a) and (5.28b) can be rewritten as:

$$0 < \left|\eta^O y_S(k)\right| < \frac{1}{n_h} \cdot \frac{{}^*X(k)}{\left|{}^*O_{net}(k)\right| \cdot \left|{}^*W^I\right|} \tag{5.30a}$$

$$0 < \left|\eta^I y_S(k)\right| < \frac{4 \cdot (n^h)^2}{(n_I + n_h)} \cdot \frac{X(k)^2 \cdot {}^*X(k)}{\left|O_{net}(k)\right|^2 \cdot I_{\max}{}^2 \cdot \left|{}^*O_{net}(k)\right| \cdot \left|{}^*W^I\right|} \tag{5.30b}$$

If we choice a constant, Θ, which satisfies:

$$\Theta \geq \max\left\{O_{net}(k), I_{\max}, \left|{}^*O_{net}(k)\right|\right\},$$

and this condition is hold through the whole process. Together with the condition: $1 < X(k) < 2$, we can obtain two constants, $\overline{\eta}^{O}$ and $\overline{\eta}^{I}$, which satisfy:

$$0 < \overline{\eta}^{O} < \frac{1}{n_h} \cdot \frac{1}{\Theta \cdot \left| {}^{*}W^{I} \right|} \tag{5.31a}$$

$$0 < \overline{\eta}^{I} < \frac{4 \cdot (n_h)^2}{(n_I + n_h)} \cdot \frac{1}{\Theta^5 \cdot \left| {}^{*}W^{I} \right|} \tag{5.31b}$$

From (4.39), we can obtain: $\left| {}^{*}W^{I} \right| \le \frac{1}{\Theta}$, and modify (5.31) as:

$$0 < \overline{\eta}^{O} < \frac{1}{n_h} \tag{5.32a}$$

$$0 < \overline{\eta}^{I} < \frac{4 \cdot n_h^2}{(n_I + n_h)} \cdot \frac{1}{\Theta^4} \tag{5.32b}$$

Thus, we choice the learning rate as follows:

$$\overline{\eta} = \Omega \times \min\{\sup(\overline{\eta}^{O}), \sup(\overline{\eta}^{I})\} \;,\; \forall \;\; 0 < \Omega < 1 \tag{5.33}$$

then, (5.9a) and (5.9b) can be modified as follows:

$$W_j^O(k+1) = W_j^O(n) + \overline{\eta} e(k) X_j \tag{5.34a}$$

$$W_{ij}^I(k+1) = W_j^I(n) + \overline{\eta} e(k) W_j^O(k) Q_{ij} \tag{5.34b}$$

From (5.32), (5.33) and (5.34), it is clear that the modified weights updating rule of DRNNC given in (5.34) do not need the information from the hysteresis identifier. Thus, the part that is the hysteresis identifier can be moved from figure (5.2), and a simplified hysteresis inverse learning structure can be obtained as follows:

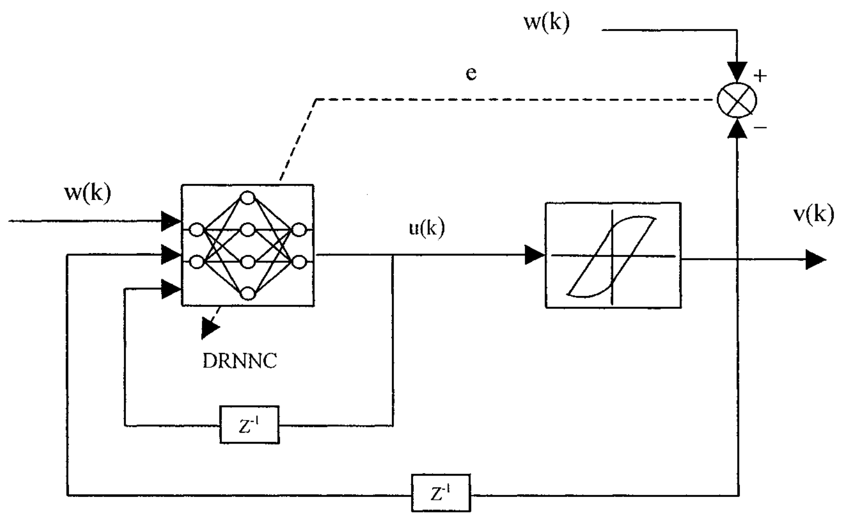


Figure 5.3 Block diagram of Hysteresis inverse learning without identifier

5.4 Simulation study for Hysteresis Inverse Learning

In this simulation study, the hysteresis inverse learning structure, which is shown in Fig 5.3, is applied. 3x7x1 DRNN is employed to act as DRNNC with the uniform initial weight value of 0.00001. The learning rate is chosen according to (5.33).

For the purpose of simulation to test the performance of hysteresis inverse modeling, a typical class of hysteresis described in section 4.1 is used.

The input signal is $u(k) = A * \sin(k * ts) / u(k) = A * [0.6\sin(k * ts) + 0.6\sin(k * ts / 3)]$ with ts=0.001, A=1, 2, 3, 4 and initial value $v(0) = 0$.

The simulation results are shown in Figs 4.3- 4.10. From these Figs, it shows that hysteresis effect can be mostly removed by DRNNC. The residue error caused by hysteresis inverse compensation is small and bounded.

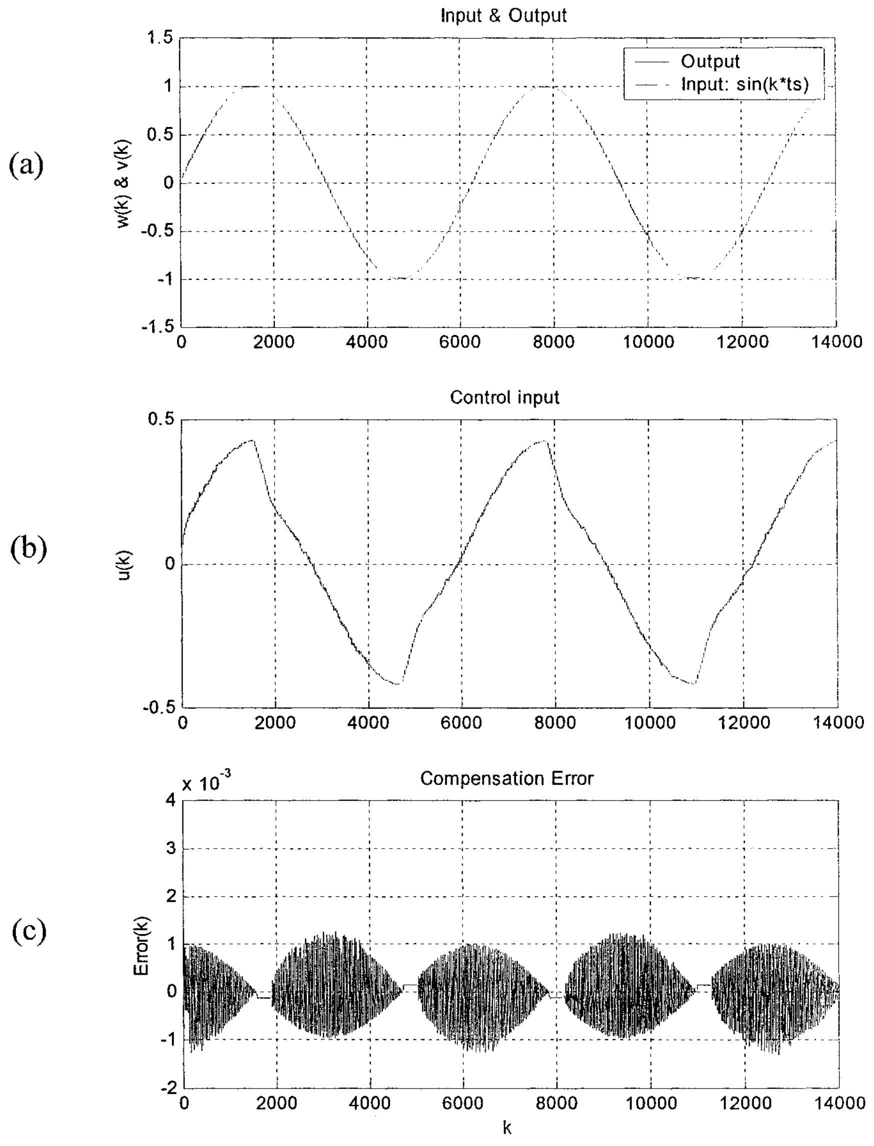
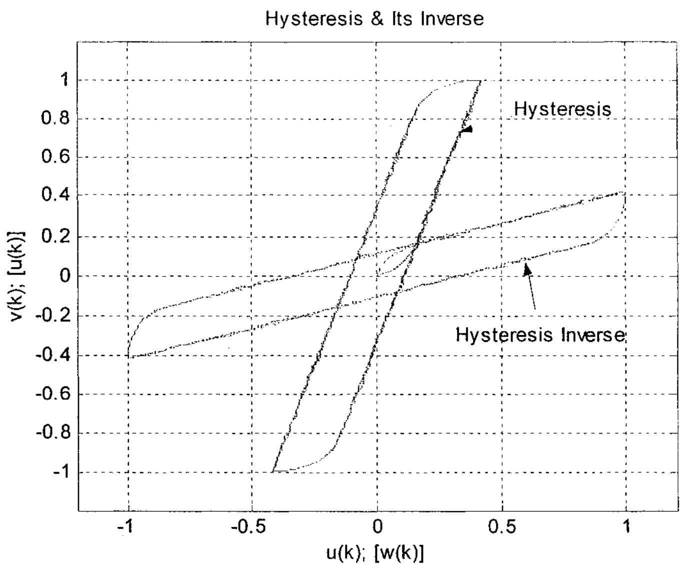


Figure 5.4 (a) Input & output (b) Control input (c) Learning error

For w(k)=sin(k*ts)

(d)



(e)

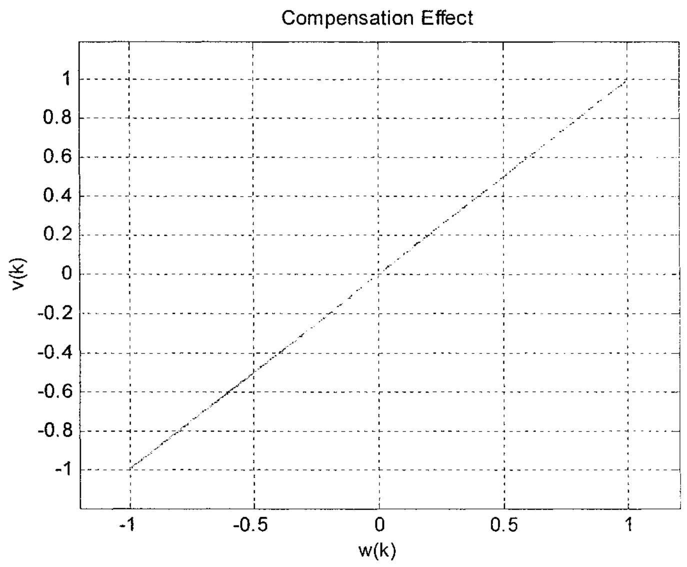
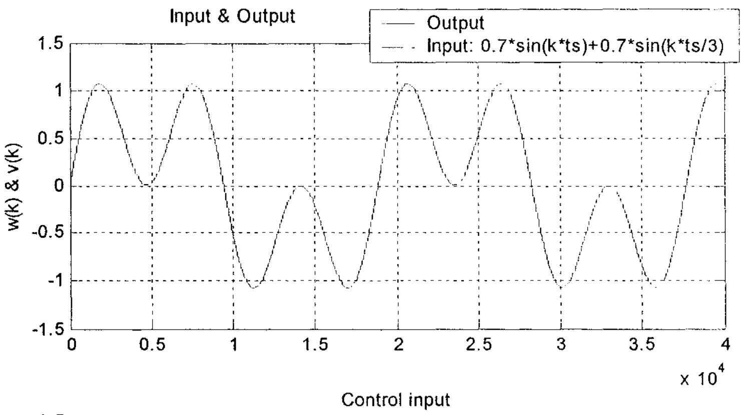


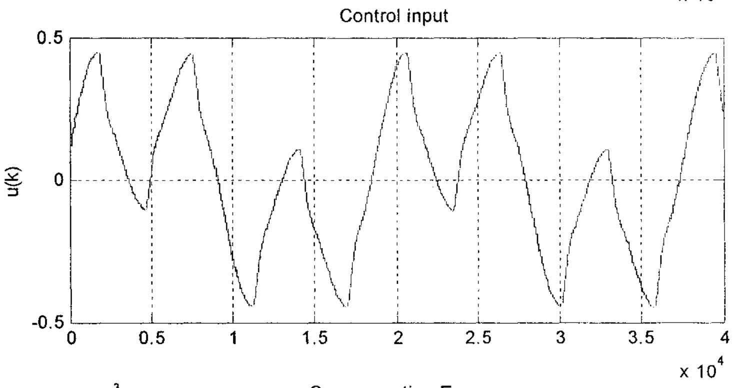
Figure 5.4 (d) Hysteresis & Its inverse (e) Compensation effect

For w(k)=sin(k*ts)

(a)



(b)



(c)

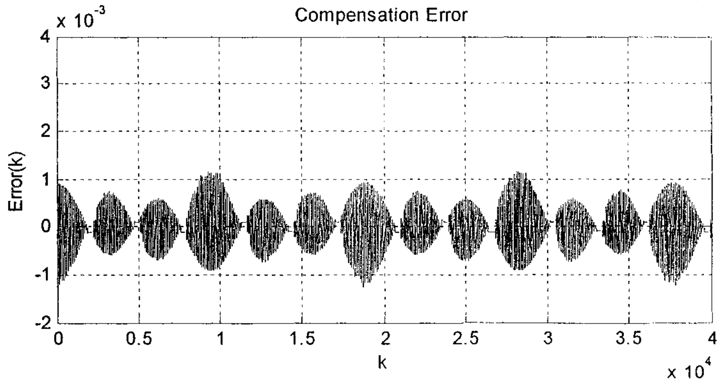
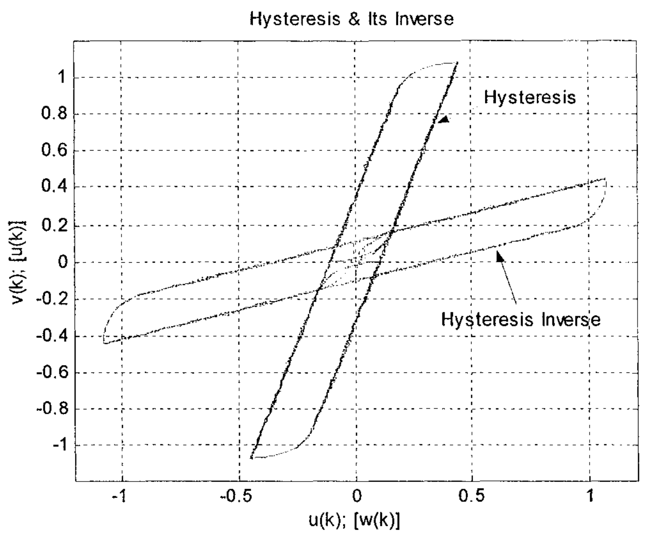


Figure 5.5 (a) Input & output (b) Control input (c) Learning error

For w(k)=sin(k*ts)

(d)



(e)

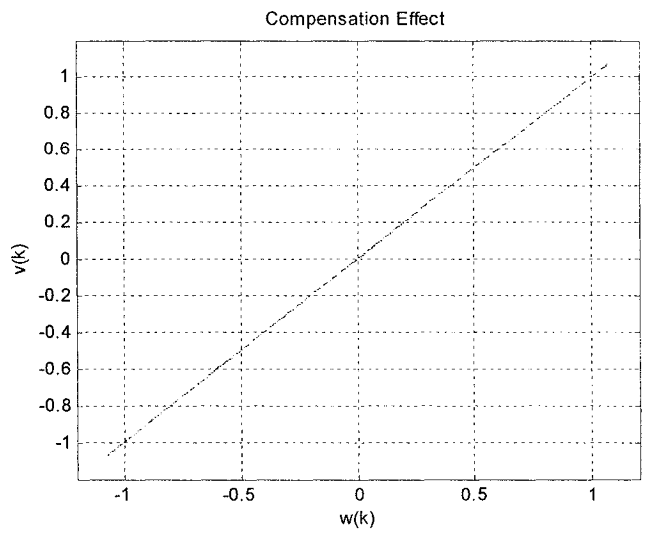
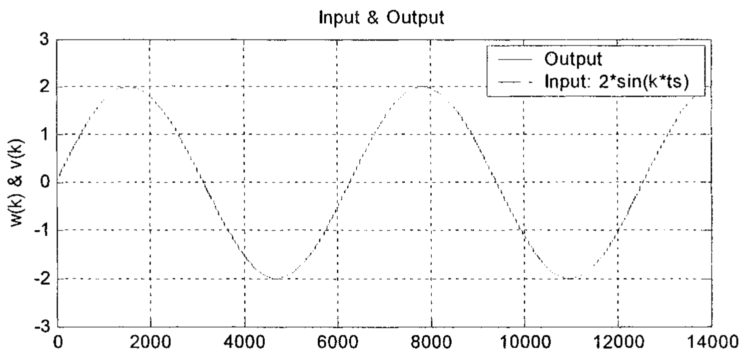


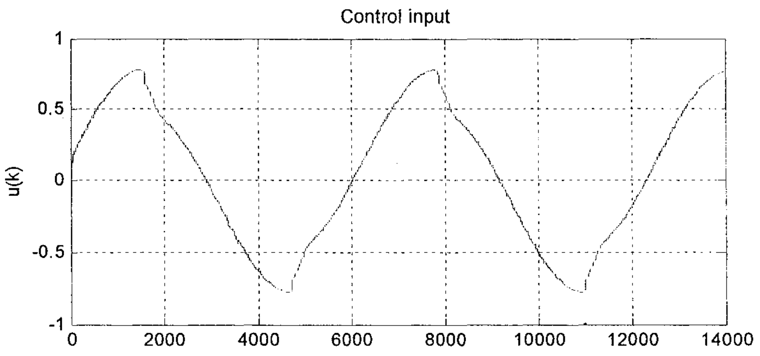
Figure 5.5 (d) Hysteresis & Its inverse (e) Compensation effect

For w(k)=sin(k*ts)

(a)



(b)



(c)

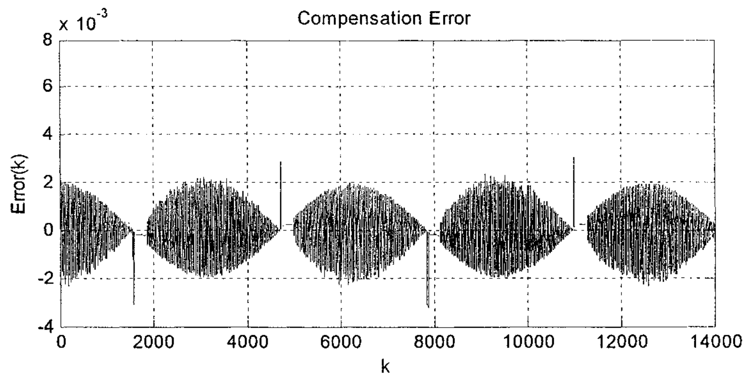
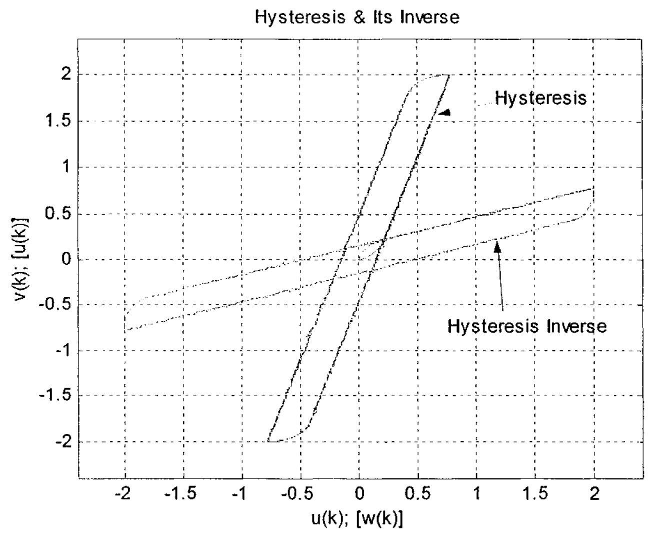


Figure 5.6 (a) Input & output (b) Control input (c) Learning error

For w(k)=2*sin(k*ts)

(d)



(e)

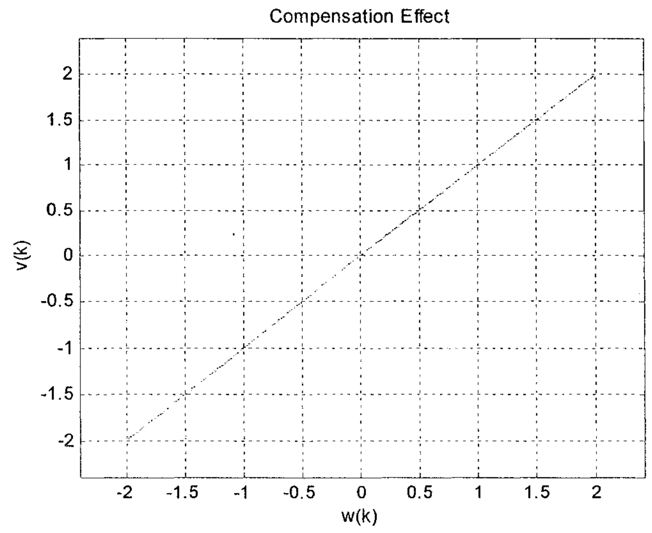


Figure 5.6 (d) Hysteresis & Its inverse (e) Compensation effect

For w(k)=2*sin(k*ts)

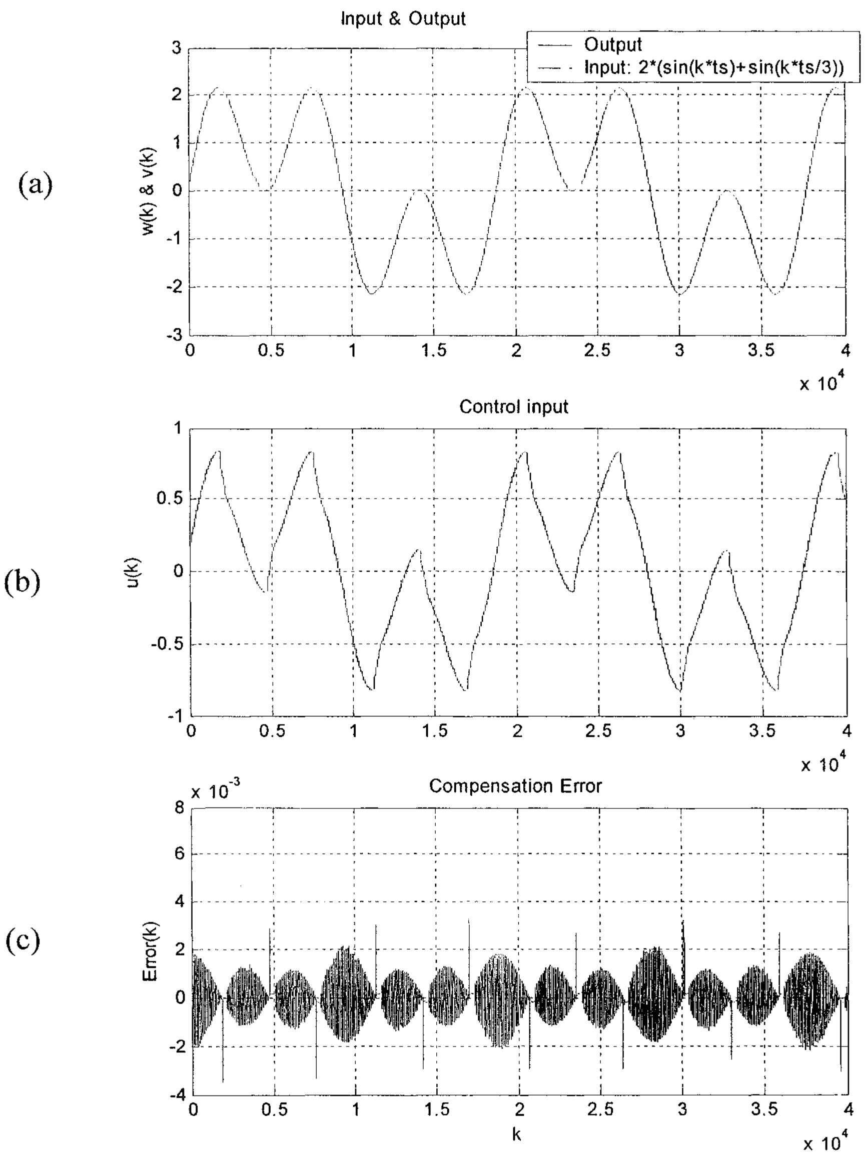
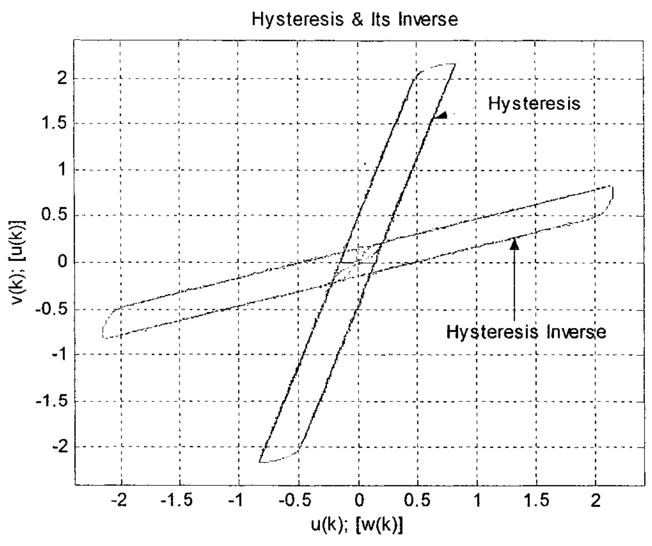


Figure 5.7 (a) Input & output (b) Control input (c) Learning error

For w(k)=2*sin(k*ts)

(d)



(e)

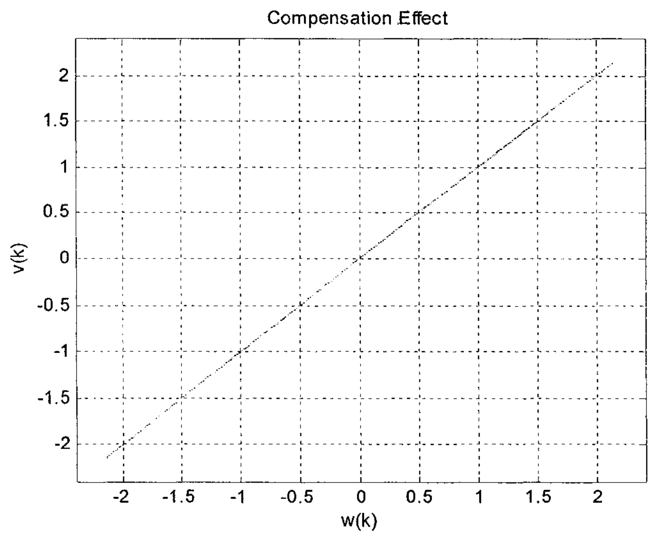
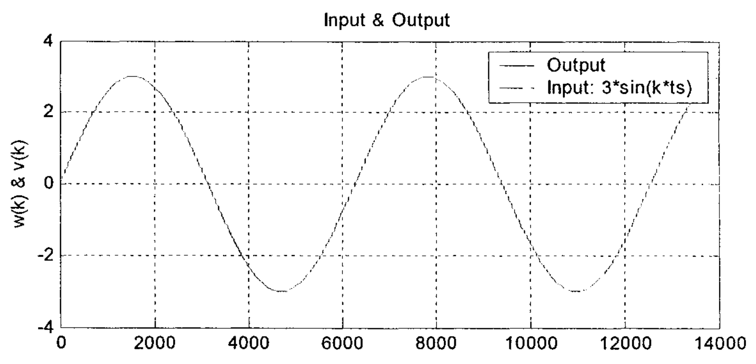


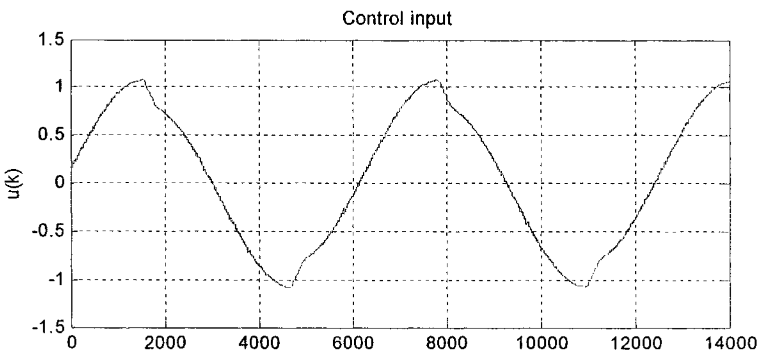
Figure 5.7 (d) Hysteresis & Its inverse (e) Compensation effect

For w(k)=2*sin(k*ts)

(a)



(b)



(c)

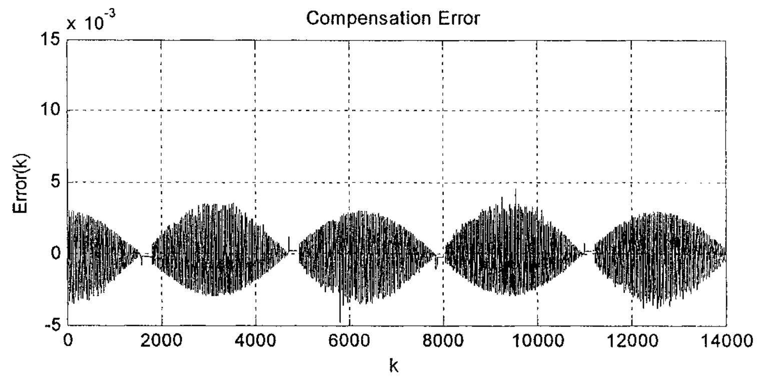
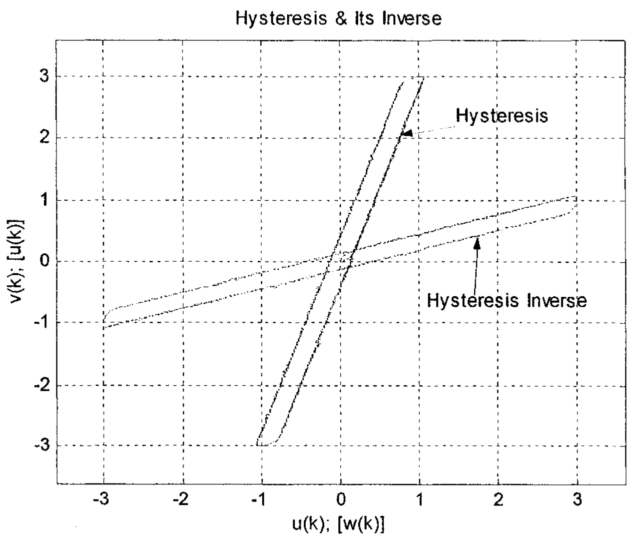


Figure 5.8 (a) Input & output (b) Control input (c) Learning error

For w(k)=3*sin(k*ts)

(d)



(e)

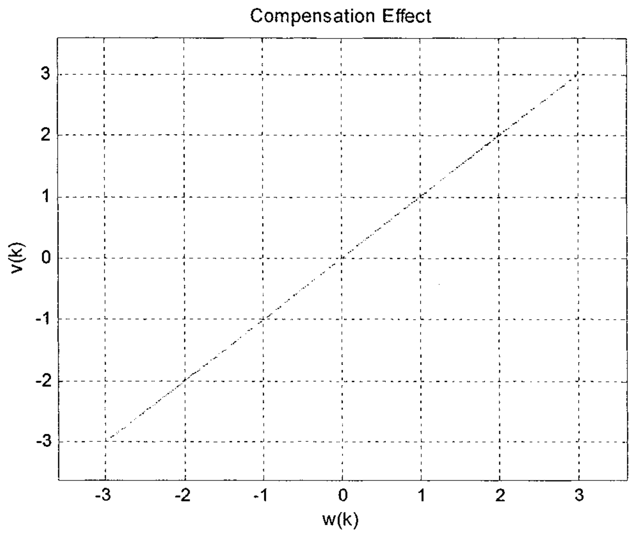
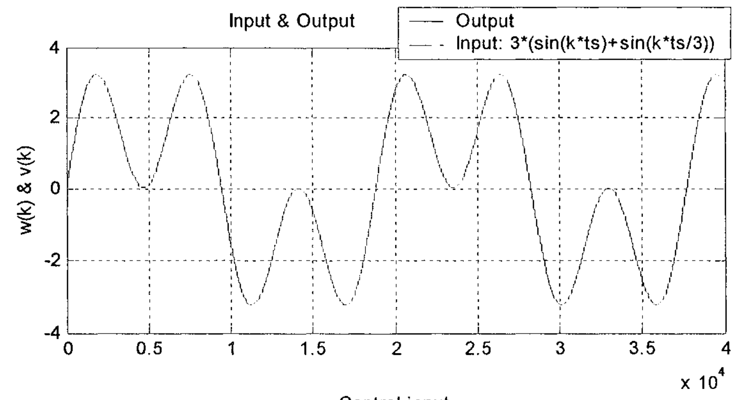


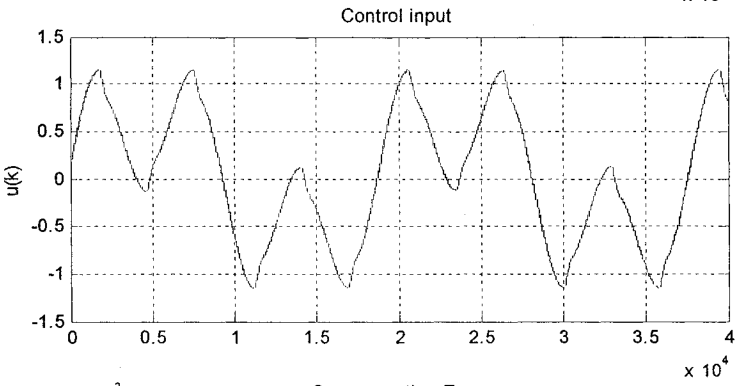
Figure 5.8 (d) Hysteresis & Its inverse (e) Compensation effect

For w(k)=3*sin(k*ts)

(a)



(b)



(c)

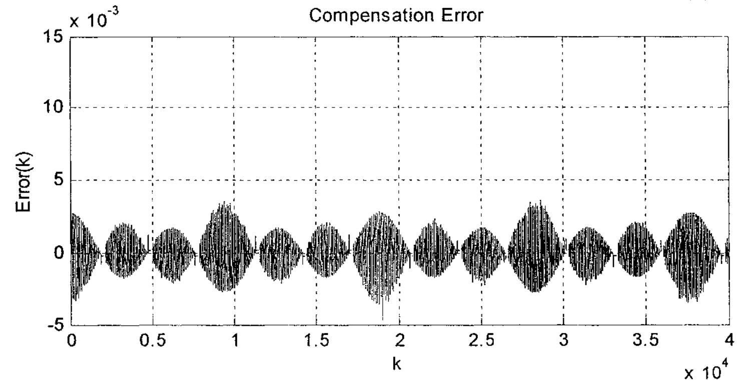
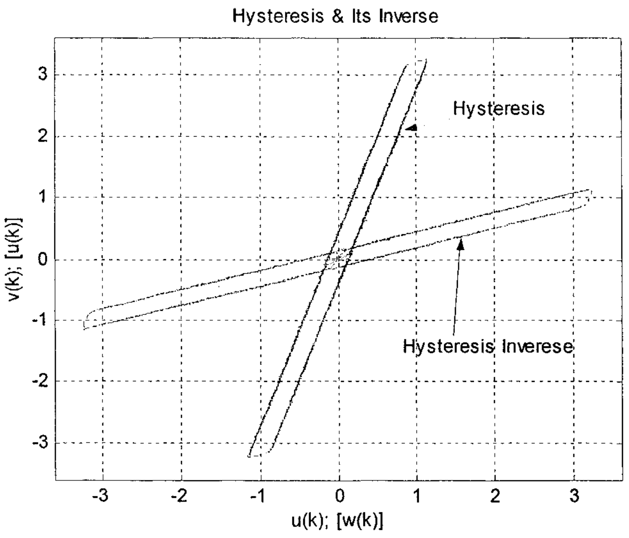


Figure 5.9 (a) Input & output (b) Control input (c) Learning error

For w(k)=3*sin(k*ts)

(d)



(e)

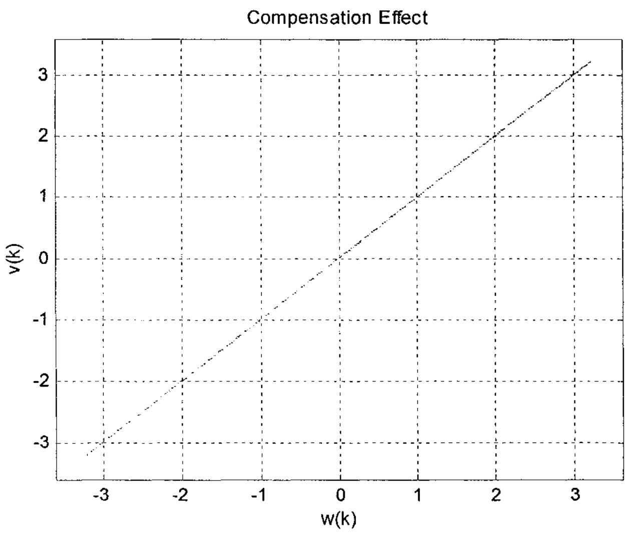
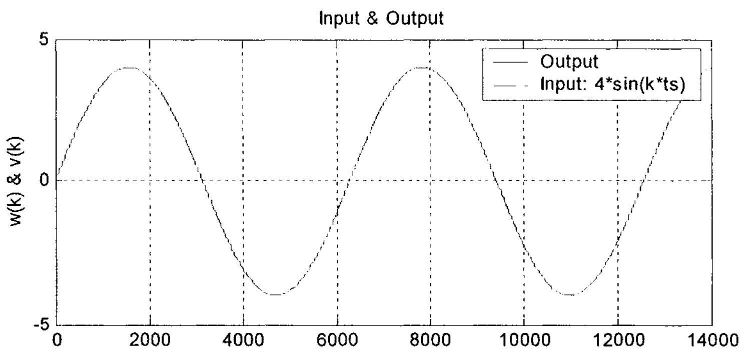


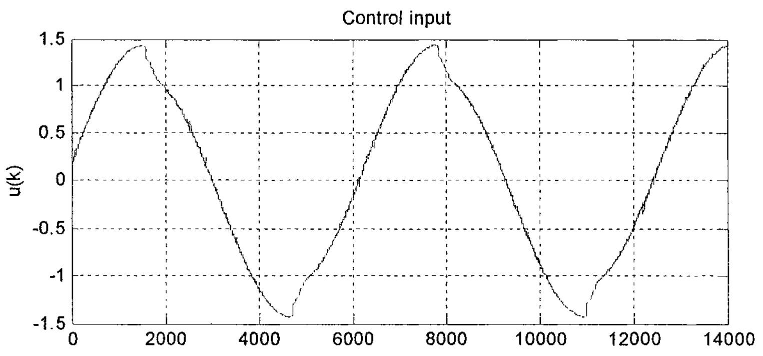
Figure 5.9 (d) Hysteresis & Its inverse (e) Compensation effect

For w(k)=3*sin(k*ts)

(a)



(b)



(c)

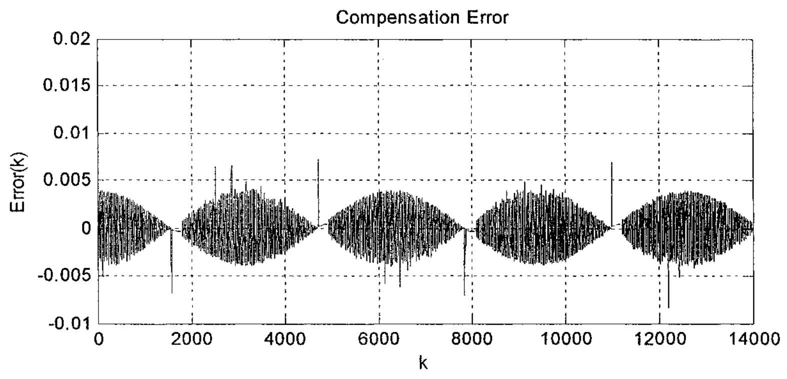
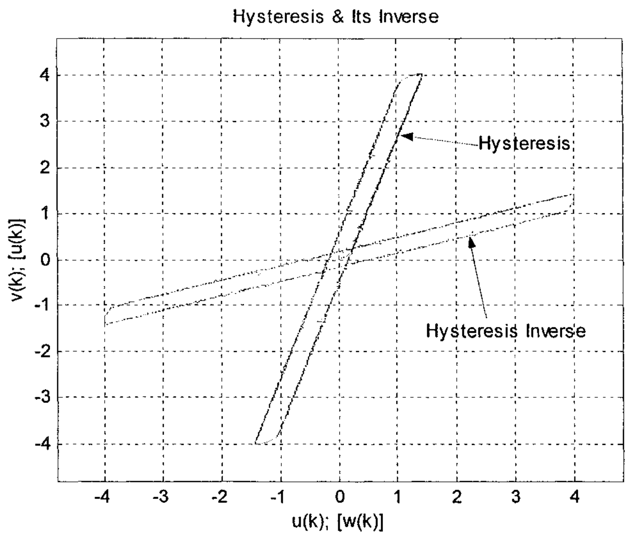


Figure 5.10 (a) Input & output (b) Control input (c) Learning error

For w(k)=4*sin(k*ts)

(d)



(e)

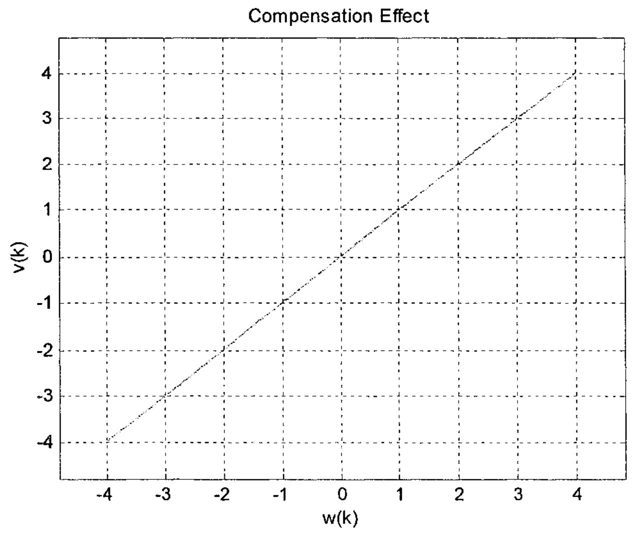


Figure 5.10 (d) Hysteresis & Its inverse (e) Compensation effect

For w(k)=4*sin(k*ts)

(a)

(b)

(c)

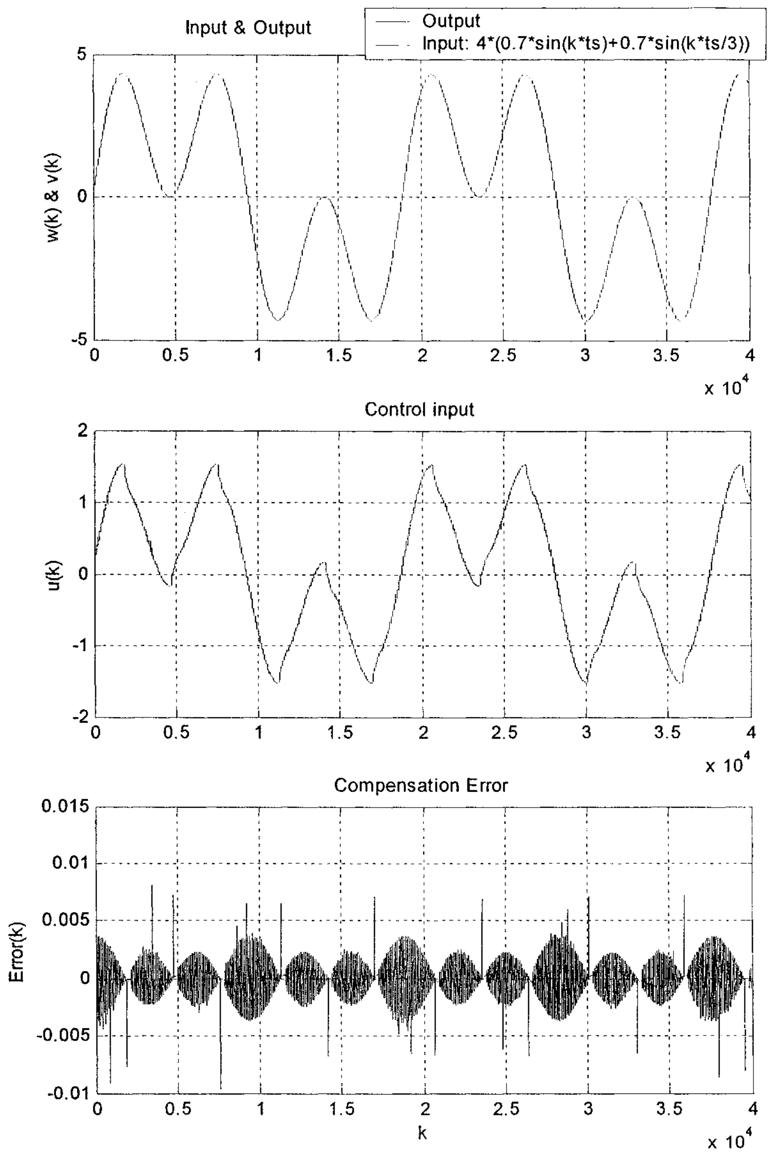
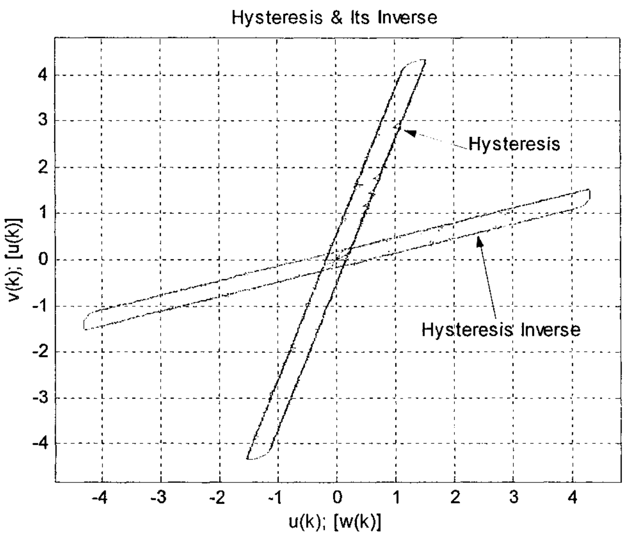


Figure 5.11 (a) Input & output (b) Control input (c) Learning error

For w(k)=4*sin(k*ts)

(d)



(e)

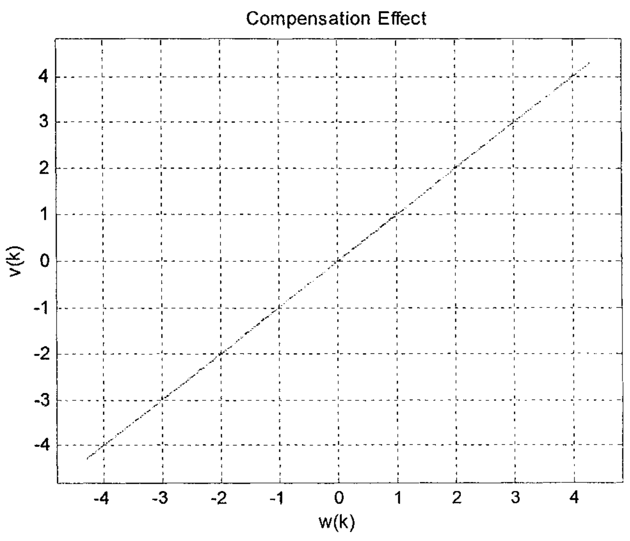


Figure 5.11 (d) Hysteresis & Its inverse (e) Compensation effect

For w(k)=4*sin(k*ts)

Chapter 6

Gaussian Networks Controller Design

6.1 Choice of control law

If the effect of hysteresis can be mostly removed by DRNNC, control problems will be simplified and thus many of control methods can be applied for such a system. For a dynamic system control, neural network based control strategies have been receiving increasing interest in recent years. Gradient decent optimization methods are usually employed for adjustment of the weights of neural networks and have proven to be quite effective in practice. However, these methods are limited in their use in high precision applications and stable analysis becomes very complicated when learning and control are attempted simultaneously [52]. To overcome these drawbacks, a specified controller, which is synthesized with Gaussian networks and a sliding mode component, is proposed in this thesis. The parameters of Gaussian networks are tuned under the guarantee of stability, and desired tracking performance can be achieved by properly setting the free parameters of the controller.

Consider a nonlinear dynamic system in the canonical form:

$$x^{(n)} + \sum_{i=1}^{n} a_i f_i(x, \dot{x}, \cdots, x^{(n-1)}) = bv \tag{6.1}$$

$$v = \mathrm{H}(w) \tag{6.2}$$

where f is an unknown function which can be linear or nonlinear, v is the input to the plant, w is the output of the hysteresis, and b is a nonzero unknown constant.

To facilitate the adaptive controller derivation, let us rewrite equation (6.1) as

$$h \cdot x^{(n)} + \sum_{i=1}^{n} \alpha_i f_i(\mathrm{x}) = v \tag{6.3}$$

where $\mathbf{x} = [x, \dot{x}, \cdots, x^{(n-1)}]$, $h = 1/b$, and $\alpha_i = a_i / b$.

Using hysteresis inverse scheme developed in the above Chapters, v can be rewritten as follows:

$$v = w + e \tag{6.4}$$

where w is the control input, and e represents residue error of hysteresis inverse learning, which are illustrated in figure 6.1

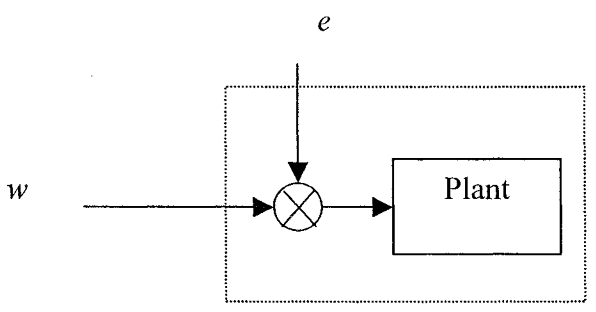


Figure 6.1 Illustrate the inputs of the plant

Clearly, the equation can be written as:

$$h \cdot x^{(n)} + \sum_{i=1}^{r} \alpha_i f_i(\mathbf{x}) - e = w \tag{6.5}$$

Applying the sliding control approach, let us define the combined error as:

$$s = \tilde{x}^{(n-1)} + \lambda_{n-2} \tilde{x}^{(n-2)} + \cdots + \lambda_0 \tilde{x} = \Delta(p) \tilde{x} \tag{6.6}$$

where $\tilde{x} = x - x_d$ and $\Delta(p) = p^{(n-1)} + \lambda_{n-2}p^{(n-2)} + \cdots + \lambda_0$ is a stable (Hurwitz) polynomial in the Laplace variable p. Note that s can be rewritten as:

$$s = x^{(n-1)} + a_r^{(n-1)} \tag{6.7}$$

where $a_r^{(n-1)}$ is defined as:

$$a_r^{(n-1)} = -x_d^{(n-1)} + \lambda_{n-2}\tilde{x}^{(n-2)} + \cdots + \lambda_0\tilde{x} \tag{6.8}$$

From equation (6.7), we can obtain:

$$h\dot{s} = w - \sum_{i=1}^{r}\alpha_i f_i(\mathrm{x}) + ha_r^{(n)} \tag{6.9}$$

With this expression, a control law can be obtained as:

$$w = -ha_r^{(n)} - k_D s + \sum_{i=1}^{r}\alpha_i f_i(\mathrm{x},t) \tag{6.10}$$

where k_D is a positive constant, and $a_r^{(n)}$ is the derivative of $a_r^{(n-1)}$, i.e.,

$$a_r^{(n)} = -x_d^{(n)} + \lambda_{n-2}\tilde{x}^{(n-1)} + \cdots + \lambda_0\dot{\tilde{x}} \tag{6.11}$$

since h and c_i are usually unknown parameters. Thus for the adaptive control, the control law (6.3) is replaced by:

$$w = -\hat{h}a_r^{(n)} - k_D s + \sum_{i=1}^{r}\hat{\alpha}_i f\,(\mathrm{x},t) + e \tag{6.12}$$

The term in equation (6.4), that is $\sum_{i=1}^{r}\hat{\alpha}_i f_i(\mathrm{x},t)$, can be approximated using the Gaussian function expressed as $\sum_{i=1}^{N}\hat{c}_i g_i(\mathbf{x},\xi_i)$, thus (6.12) can be rewritten as:

$$w = -\hat{h}a_r^{(n)} - k_D s + \sum_{I\in I_0}\hat{c}_I g(\mathrm{x},\xi_I) + e \tag{6.13}$$

To overcome the effects of e, this control law can be modified as [52]:

$$w = -\hat{h}a_r^{(n)} - k_D s + (1 - m(t))\sum_{i=1}^{N} \hat{c}_i g_i(\mathbf{x}, \xi_i) + m(t) \cdot u_{sl} \qquad (6.14)$$

where, $m(t)$ will be specified in the following development.

This control law expression can be rewritten in more general form as:

$$w(t) = u_{pd}(t) + (1 - m(t))u_{ad}(t) + m(t)u_{sl}(t) \qquad (6.15)$$

Here $u_{pd}(t)$ is a negative feedback term including a weighted combination of the measured tracking error states. The term $u_{sl}(t)$, represented the sliding component of the control law to overcome the effects of e, and $u_{ad}(t)$ represented the adaptive component to compensate the system dynamics. The function $m(t)$ is a continuous, state dependent modulation that allows the controller a smooth transition between sliding and adaptive modes of operation. The block diagram of this control system structure is shown in figure 6.2. From Figure 6.2, detailed structure of GNNC is presented in Figure 6.3.

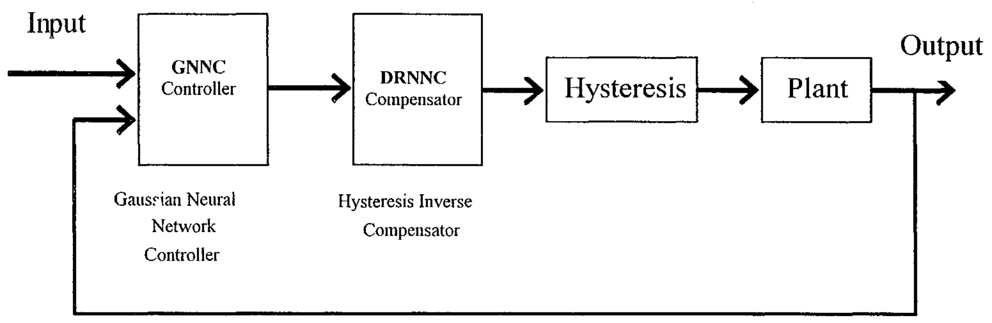

Input
GNNC Controller
DRNNC Compensator
Hysteresis
Plant
Output
Gaussian Neural Network Controller
Hysteresis Inverse Compensator

Figure 6.2 Block diagram of control system structure

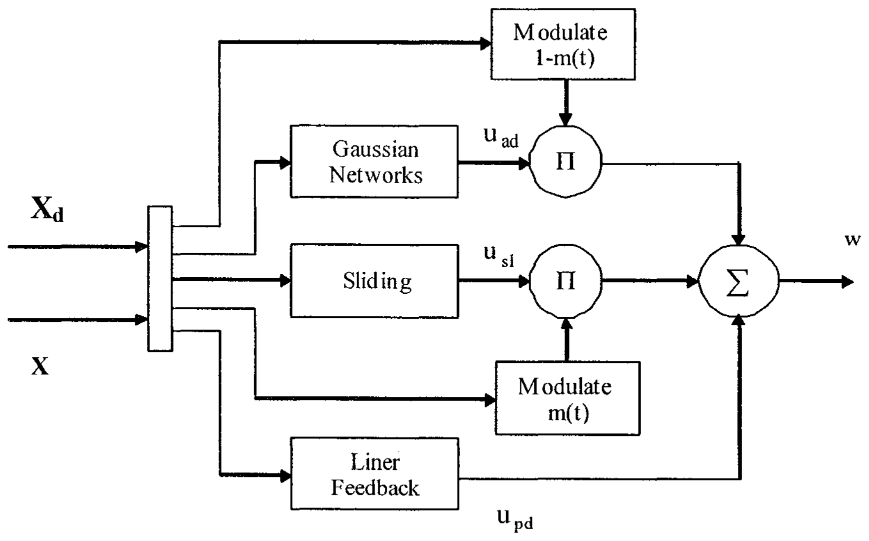


Figure 6.3 Block diagram of controller (GNNC) structure

□ $u_{pd}(t)$ **- A negative feedback term given as below:**

$$u_{pd}(t) = -k_d s(t) - \hat{h} a_r^{(n)} \tag{6.16}$$

where k_d is the constant gain of feedback, and $a_r^{(n)}$ is defined in equation (6.11)

The update rule of $\hat{h}$ is given as below

$$\dot{\hat{h}} = -\gamma \operatorname{sgn}(h) \cdot s \cdot a_r^{(n)} \tag{6.17}$$

□ u_{sl} **- The sliding component of control law defined as:**

$$u_{sl}(t) = -k_{sl}(t) sat(s(t)/\Phi) \tag{6.18}$$

where $sat(\cdot)$ is the saturation function (sat(y)=y if $|y|<1$, and sat(y)= sat(y) otherwise), Φ is an arbitrary positive constant, and k_{sl} is the gain of the slide controller.

$m(t)$ - **A continuous, state dependent modulation which allows the controller having a smooth transition between sliding and adaptive modes of operation, which is defined as follows:**

$$m(t) = \max(0, sat(\frac{r(t)-1}{\Psi})) \tag{6.19}$$

where $\max(\cdot)$ is the maximum function, $sat(\cdot)$ is the saturation function, and Ψ is a positive constant representing the width of the transition region, and $r(t)$ is defined as:

$$r(t) = \|\mathbf{x}(t) - \mathbf{x}_0\|_{p,w} = \left\{ \sum_{i=1}^{n} \left(\frac{| x_i - x_{0,i} |}{w_i} \right)^p \right\}^{1/p} \tag{6.20}$$

where $\mathbf{x}_0$ fixes the absolute location of sets in the state space of the plant, and w_i represents the weight.

We define two sets as:

$$A_d = \left\{ \mathbf{x} \middle| \|\mathbf{x} - \mathbf{x}_0\|_{p,\mathrm{w}} \leq 1 \right\}$$

and

$$A = \left\{ \mathbf{x} \middle| \|\mathbf{x} - \mathbf{x}_0\|_{p,\mathrm{w}} \leq 1 + \Psi \right\}$$

when

i) $r(t) \leq 1$, meaning that $\mathbf{x} \in A_d$ and $m(t) = 0$

ii) $1 < r(t) < 1 + \Psi$, mean that $\mathbf{x} \in A - A_d$ and $0 < m(t) < 1$

iii) $r(t) \geq 1 + \Psi$, mean that $\mathbf{x} \in A^c$ and $m(t) = 1$

In the limiting case $p = \infty$,

$$r(t) = \|\mathbf{x}(t) - \mathbf{x}_0\|_{p,w} = \max\left(\frac{|x_1 - x_{0,1}|}{w_1}, \frac{|x_2 - x_{0,2}|}{w_2}, \cdots, \frac{|x_n - x_{0,n}|}{w_n} \right) \tag{6.21}$$

□ $u_{ad}(t)$ - **the adaptive component in control law**:

$$u_{ad}(t) = \sum_{i=1}^{N} \hat{c}_i(t) g_i(\mathbf{x}, \xi_i) \tag{6.22}$$

where $\mathbf{x} = [x, \dot{x}, \cdots, x^{(n-1)}]$, ξ_i encode the sampling mesh, which represents the input weights to neuron *i*, and $\hat{c}_i$ is the estimate of the true coefficient c_i is the output weight for that neuron.

The structure of this component is illustrated in figure 6.4.

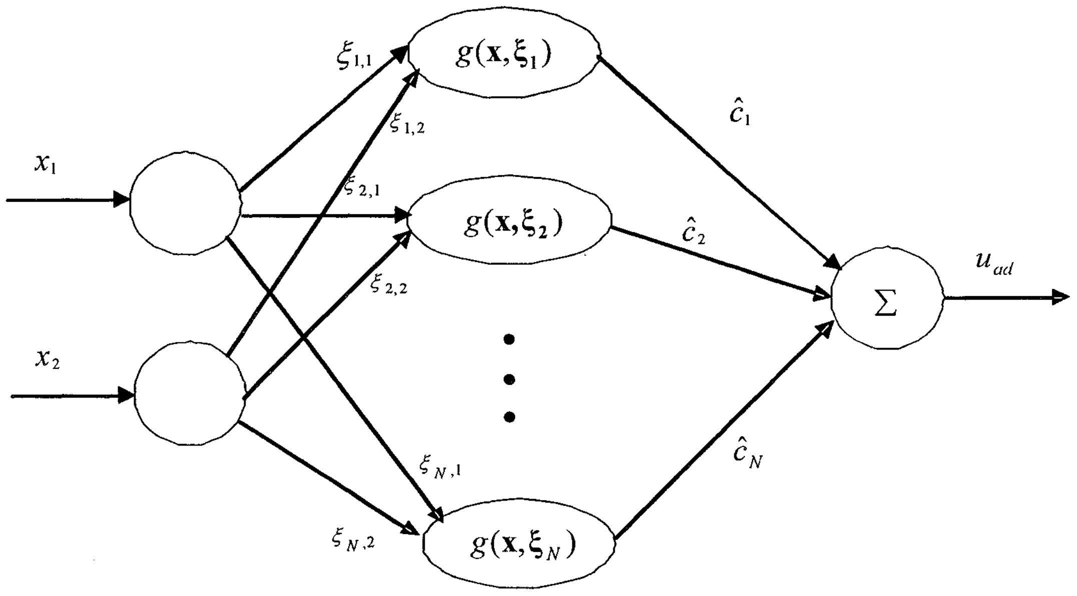


Figure 6.4 Structure of the adaptive component of the control law

Here, the Gaussian function g is given by:

$$g(\mathbf{x}, \xi) = g(\mathbf{x} - \xi) = \exp(-\pi \sigma_v^{\,2} d^2) \tag{6.23}$$

where σ_v is a positive constant. $d^2 = \|\mathbf{x}-\xi\| = (\mathbf{x}-\xi)^T(\mathbf{x}-\xi)$. This can be illustrated in figure 6.5.

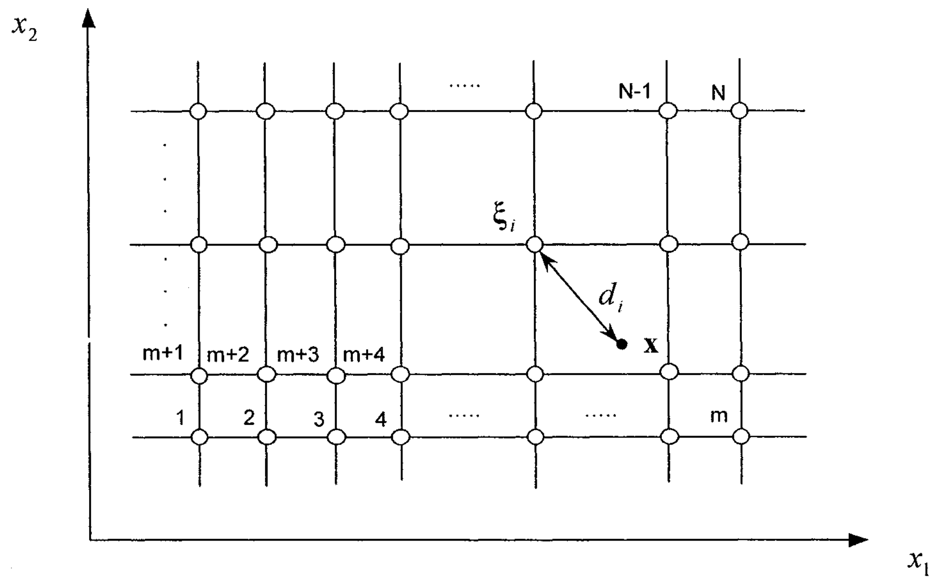


Fig 6.5 Illustration of sampling mesh

The updated rules of c_i are given as [52]:

$$\dot{c}_i(t) = -k_a(1-m(t))s_\Delta(t)g_i(\mathbf{x}(t)-\xi_i) \tag{6.24}$$

where $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots\ldots, x_n(t)]$, $\xi_i = [\xi_{i,1}, \xi_{i,2}, \ldots, \xi_{i,n}]$, and $s_\Delta(t)$ is defined as

$$s_\Delta(t) = s(t) - \Phi sat(s(t)/\Phi) \tag{6.25}$$

where Φ is an arbitrary positive constant which is chosen as the deadband of s, and $sat(\cdot)$ is the saturation function.

The function $s_\Delta(t)$ has two properties useful in the design of adaptive laws:

i) If $|s| < \Phi$, then $s_\Delta = 0$

ii) While if $|s| > \Phi$, then $|s_\Delta| = |s| - \Phi$

6.2 Choice of Gaussian network parameters

It is well known that three layer feed forward neural networks with one hidden layer of nonlinear nodes can uniformly approximate functions. The approximation implemented by such three layer networks can be represented mathematically as:

$$f_A(\mathbf{x}) = \sum_{i=1}^{N} c_i g_i(\mathbf{x}, \xi_i)$$

where $\mathbf{x} = [x, \dot{x}, \cdots, x^{(n-1)}]$, $\xi = [\xi_1, \xi_2, \cdots \xi_n]$, and g_i is the elementary function implemented by node *i*, often taken as a logic, or sigmoid function. In this thesis, g_i represents a Gaussian function, which is defined in equation (6.23). The coefficient c_i represents the output weight of node *i*, and ξ_i represents the input weight for that node.

In the Gaussian network design, the three parameters that should be determined are: the lattice mesh size Δ, the variance $\sigma_v^{\;2}$, and the truncation radius l. If we make suitable choices for these three parameters, then we can obtain:

$$|f(\mathbf{x}) - f_A(\mathbf{x})| \le \varepsilon_f \qquad \forall \;\; \mathbf{x} \in A \tag{6.26}$$

The error sources can be given by:

$$\varepsilon_f = \varepsilon_1 + \varepsilon_2 + \varepsilon_3 \tag{6.27}$$

In above equation,

ε_1 - represents the error introduced by approximating f on a compact set A using only frequencies of an absolute value less than β.

ε_2 - represents the error introduced into the cardinal series by the fact that the radial Gaussian is not an ideal spatial low-pass filter. The mesh size Δ and variance $\sigma_v^{\,2}$ can be chosen so as to reduce ε_2 as much as is required. The rule that yields the best results for low dimensional networks is to take:

$$\sigma_v^{\,2} = \pi\beta^2 \tag{6.28}$$

$$\Delta = \frac{1}{2\theta\beta} \tag{6.29}$$

where θ is the over sampling parameter.

ε_3 - represents the error introduced by truncating the cardinal series to a finite number of terms. The truncation parameter l can be chosen to reduce the error ε_3. The rule for choosing l is given by [49]:

$$l \geq \sqrt{\frac{n-1}{2\pi\sigma_v^{\,2}}} \tag{6.30}$$

where n is the dimension **X.**

6.3 Determining the number of nodes in the networks

After the appropriate prior choice of network parameters, we can finally determine the number of nodes in the networks. Here, a new set, Γ, is defined as below:

$$\Gamma = \left\{ \mathbf{x} \,\middle|\, \|\mathbf{x}\text{-}\mathbf{y}\|_\infty \leq l\Delta \;\; \forall\, \mathbf{y} \in A \right\} \tag{6.31}$$

that is, Γ is the rectangle containing all lattice points within a distance $\rho = l\Delta$ of A.

Since l and A define the set Γ, this set plus the choice of Δ together determine the number of nodes in the network. For example, if

$$A = [-m_1\Delta,\ m_1\Delta] \times [-m_2\Delta,\ m_2\Delta],$$

then

$$\Gamma = [-(m_1\Delta + \rho),\ (m_1\Delta + \rho)] \times [-(m_2\Delta + \rho),\ (m_2\Delta + \rho)]$$

Thus, the total number of nodes in the network should be:

$$N = [2(m_1 + l) + 1] \times [2(m_2 + l) + 1]$$

For reference, Figure 6.6 illustrates the relationships among the A and Γ.

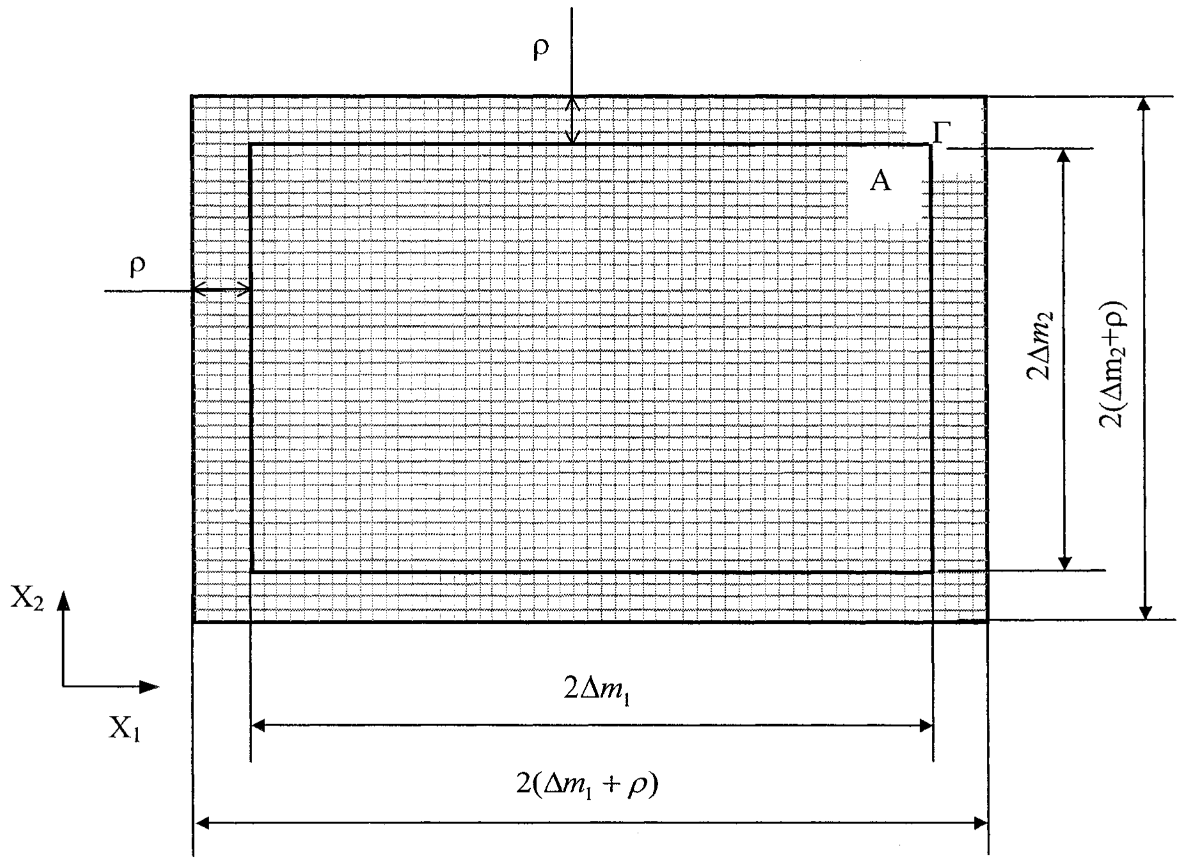


Figure 6.6 Illustration of the relationship of state space A and Γ.

6.4 Outlining the implementation procedure of discrete Gaussian Network Controller

Generally, the design procedure of the Gaussian network controller includes two basic steps:

Step 1 Properly choosing the parameters for Gaussian network controller according the rules that have been presented in previous section.

Step 2 Base on these parameters, constructing a Gaussian network controller and implementing it into the control system to train the network while adapting output weights "*c*".

In step 1, the parameters that should be determined are listed as follows:

Φ — Arbitrary positive constant

β — Smoothness constraint parameter

θ — Over sampling parameter

Δ — Sampling mesh

σ_v^2 — Gaussian variance

l — Truncation radius

N — Number of nodes in Gaussian networks

k_d — Constant gain of feedback

k_a — Positive constant determining the adaptation rate

k_{sl} — The gain of the slide controller

After properly choosing the parameters of the network, the network can be trained and its output weights can be tuned.

At step k

1) Compute the error state, $\tilde{\mathbf{x}}(k)$:

$$\tilde{\mathbf{x}}(k) = \mathbf{x}(k) - \mathbf{x}_d(k)$$

where $\mathbf{x}(k) = [x(k), \frac{x(k)-x(k-1)}{ts}, \cdots, \frac{x(k-n+2)-x(k-n+1)}{ts}]^T$,

$$\mathbf{x}_d(k) = [x_d(k), \frac{x_d(k)-x_d(k-1)}{ts}, \cdots, \frac{x_d(k-n+2)-x_d(k-n+1)}{ts}]^T$$

2) Compute the filtered error $s(t)$:

$$s(k) = \Lambda^T \tilde{\mathbf{x}}(k)$$

where $\Lambda^T = [\lambda^{(n-1)}, (n-1)\lambda^{(n-2)}, \cdots, 1]$; $\tilde{\mathbf{x}}(k) = \mathbf{x}(k) - \mathbf{x}_d(k)$

3) Compute the tuning error $s_\Delta(t)$ using equation (6.25):

$$s_\Delta = s - \Phi \cdot sat(s(k)/\Phi)$$

where Φ is an arbitrary positive constant and $sat(\cdot)$ is the saturation function.

4) Compute the parameter, $m(k)$, using the modulation function (6.19):

$$m(k) = \max(0, sat(\frac{r(k)-1}{\Psi}))$$

where $\max(\cdot)$ is the maximum function, $sat(\cdot)$ is the saturation function, Ψ is a positive constant, and $r(k)$ has been defined in section (6.1).

5) Compute the sliding control component, $u_{sl}(k)$, using function (6.18):

$$u_{sl}(k) = -k_{sl}(k) sat(s(k)/\Phi)$$

where $sat(\cdot)$ is the saturation function, Φ is an arbitrary positive constant, and

k_{sl} is the gain of the slide controller.

6) Compute the adaptive output of network, $u_{ad}(k)$, using function (6.22):

$$u_{ad}(k) = \sum_{i=1}^{N} \hat{c}_i(k) g_i(\mathbf{x}(k) - \xi_i)$$

where ξ_i represents the input weights to neuron *i*, and $\hat{c}_i$ represents the output weight for that neuron.

7) Compute the negative feedback term, $u_{pd}(k)$, using function (6.16):

$$u_{pd}(k) = -k_d s(k) - h(k) a_r^{(n)}(k)$$

where k_d is the constant gain of feedback, $a_r^{(n)}(k) = \boldsymbol{\lambda}_v^{\mathrm{T}} \tilde{\mathbf{x}}(k) - x_d^{(n)}(k)$ with $\boldsymbol{\lambda}_v^T = [0,\ \lambda^{n-1}, (n-1)\lambda^{n-2}, \cdots, (n-1)\lambda]$, $x_d^{(n)}(k)$ is the nth derivative of the desired trajectory, and $s(t)$ is the filtered tracking error which has be defined in section (6.1).

8) Compute the general controller output, $w(k)$, using function (6.15):

$$w(k) = u_{pd}(k) + (1 - m(k)) u_{ad}(k) + m(k) u_{sl}(k)$$

9) Compute system output from $w(k)$.

10) Update Gaussian network output weight $\hat{c}$:

$$\hat{c}_i(k+1) = \hat{c}_i(k) - ts \cdot k_a (1 - m(k)) s_\Delta(k) g(\mathbf{x}(k) - \xi_i) \tag{6.32}$$

where k_a is a positive constant determining the adaptation rate

11) Update input gain h using function (6.16):

$$\hat{h}(k+1) = \hat{h}(k) - ts \cdot \gamma\, \mathrm{sgn}(\hat{h}(k)) \cdot s \cdot a_r^{(n)}$$

12) Wait for the last step to elapse and loop back to step 1)

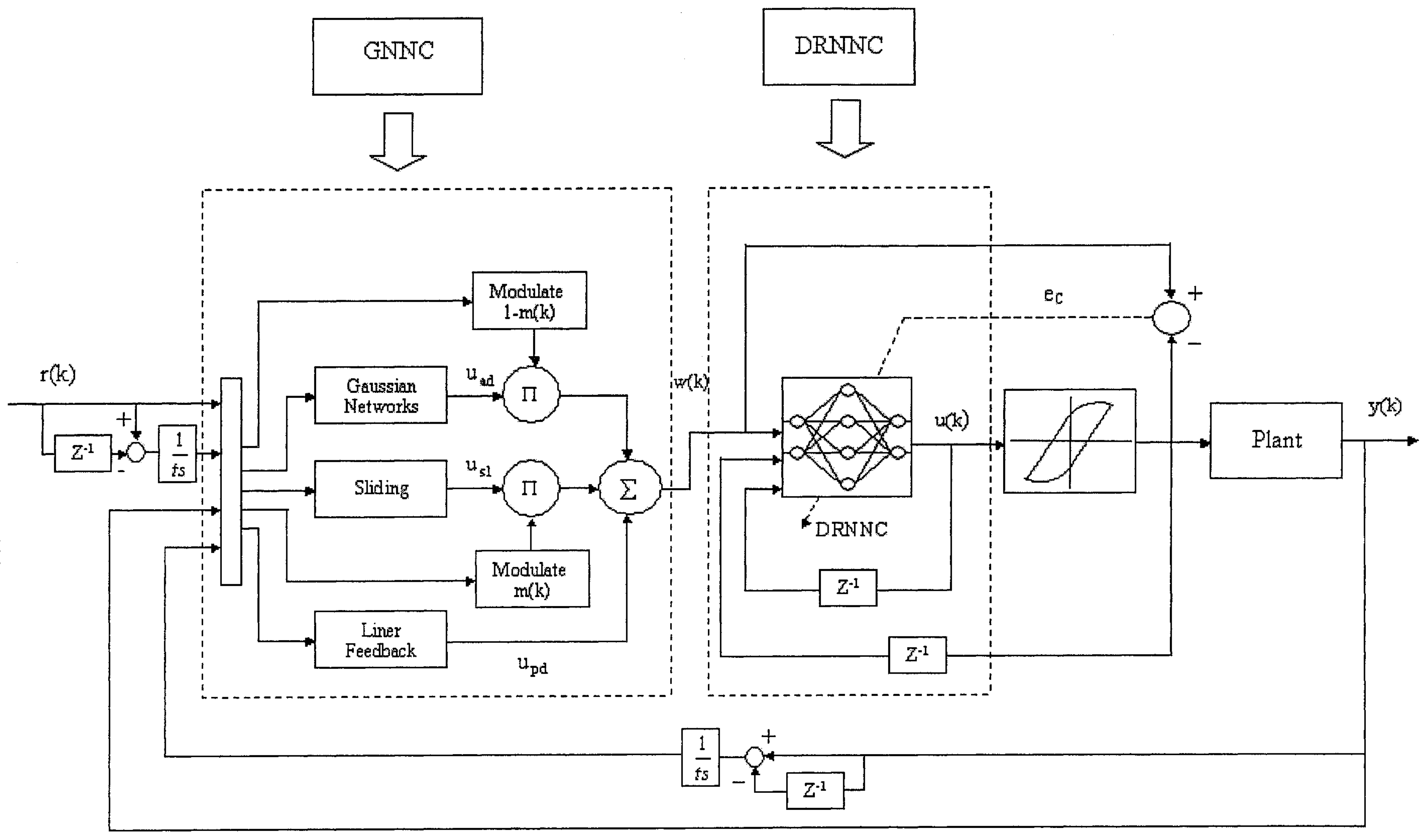


Figure 6.7 Block diagram of control system with hysteresis compensator

Chapter 7

Simulation study

In this section, the preceding theoretical development is applied to four examples. In the simulations that follow, two different plants are used to track a set of desired trajectories, and the whole control system which is applied is descript in Fig 6.7.

The simulation results are shown in Figs 4.3- 4.10. Figure 4.3(a)-4.10(a) show the curves of reference signal and tracking signal, and Figure 4.3(b)-4.10(b) show the tracking error. From these Figs, it clearly shows that the proposed control system can tracking the reference signal very well, and tracking accuracy can be achieved within desired goal.

<u>Example 7.1</u>:

A second order plant is applied to track a sin wave signal $\sin(k * ts)$. The goal of the controller is to obtain asymptotic position tracking accuracy within 0.005 units. For this design target, the parameters of Gaussian networks controller are chosen as below:

$\Phi = 0.005$	$l = 4$
$\beta = 2$	$\mathrm{N} = 961$
$\theta = 2$	$k_d = 100$
$\Delta = 0.125$	$k_a = 50$
$\sigma_v^{\ 2} = 4\pi$	$k_{sl} = 10$

The plant that is illustrated in discrete form is given as follows:

$$z^{-1} \cdot \frac{5.4735 + 5.464z^{-1}}{1 - 1.9948z^{-1} + 0.99478z^{-2}} \times 10^{-7}$$

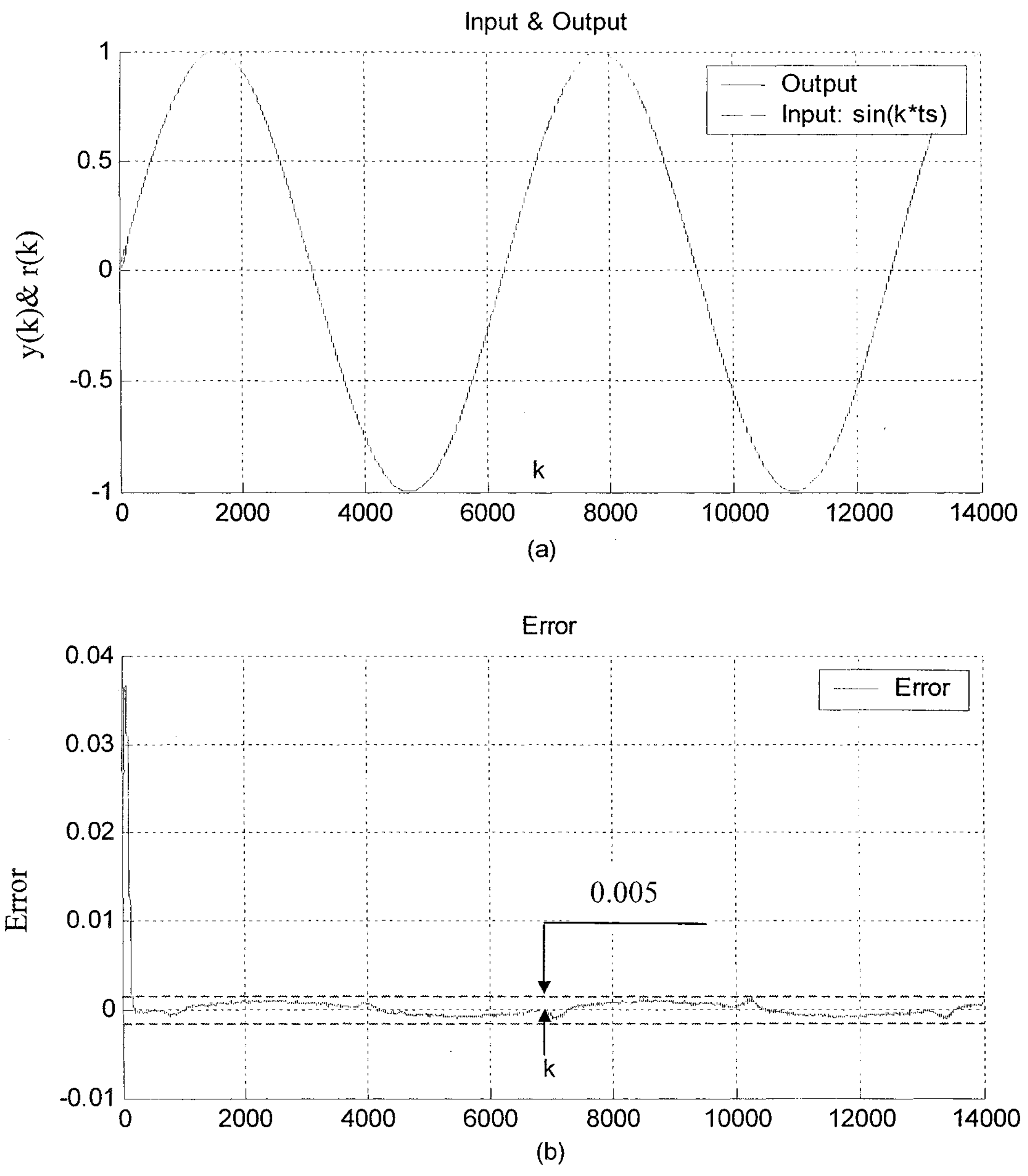


Figure 7.1 (a) Input & output (b) Tracking error

for the fist order plant with the input: sin(k*ts)

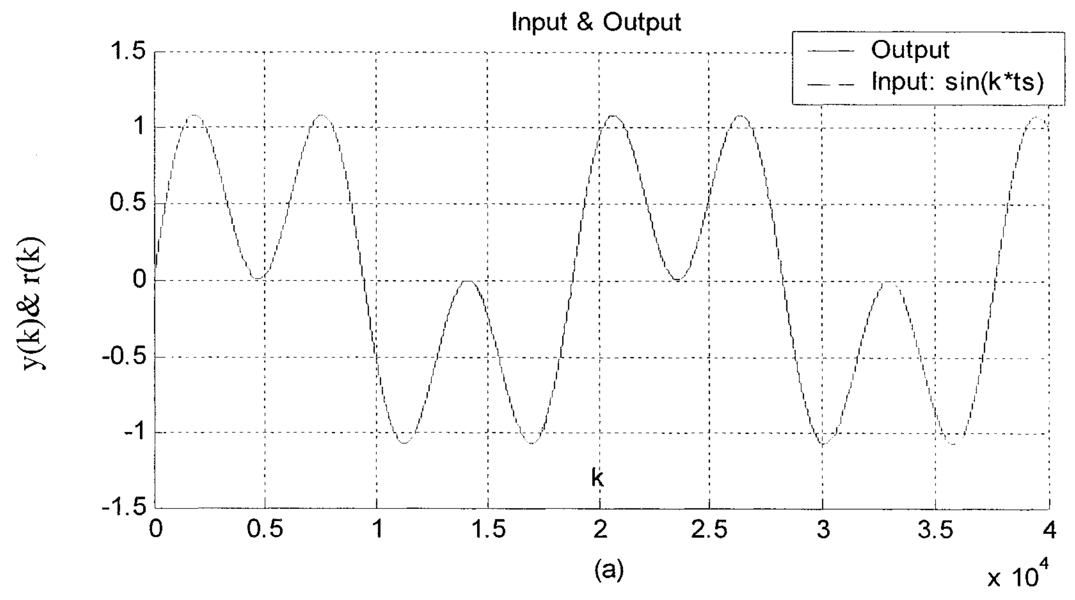
Example 7.2:

A second order plant is applied to track a sin wave signal $0.5\sin(k * ts) + 0.5\sin(k * ts/3)$. The goal of the controller is to obtain asymptotic position tracking accuracy within 0.005 units. For this design target, the parameters of Gaussian networks controller are chosen as below:

$\Phi = 0.005$	$l = 4$
$\beta = 2$	$\mathrm{N} = 961$
$\theta = 2$	$k_d = 100$
$\Delta = 0.125$	$k_a = 50$
$\sigma_v^2 = 4\pi$	$k_{sl} = 10$

The plant that is illustrated in discrete form is given as follows:

$$z^{-1} \cdot \frac{5.4735 + 5.464z^{-1}}{1 - 1.9948z^{-1} + 0.99478z^{-2}} \times 10^{-7}$$



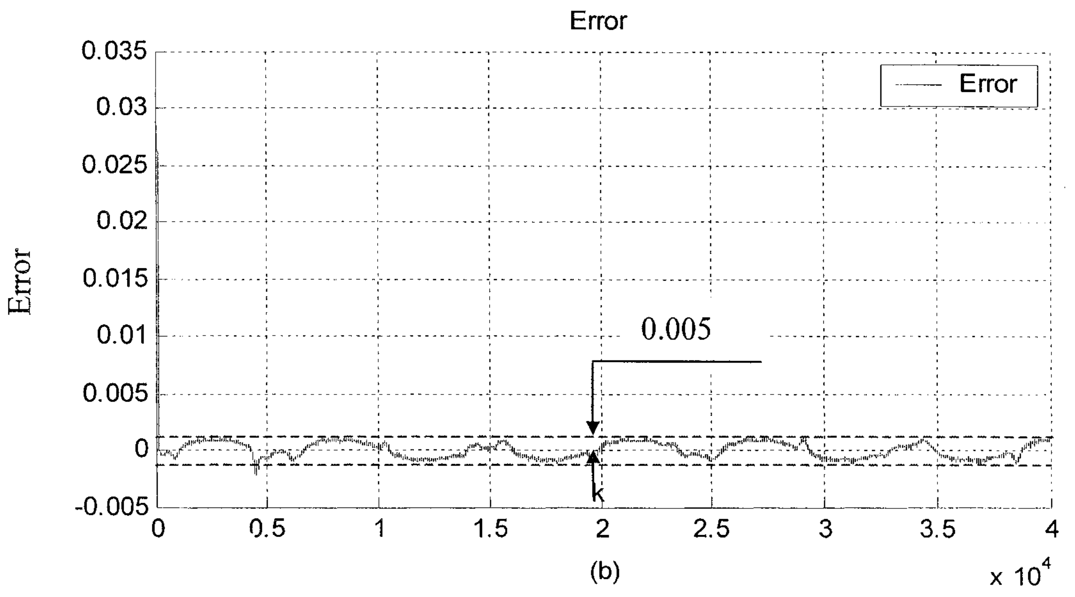


Figure 7.2 (a) Input & output (b) Tracking error

for the fist order plant with the input: 0.6sin(k*ts)+0.6sin(k*ts/3)

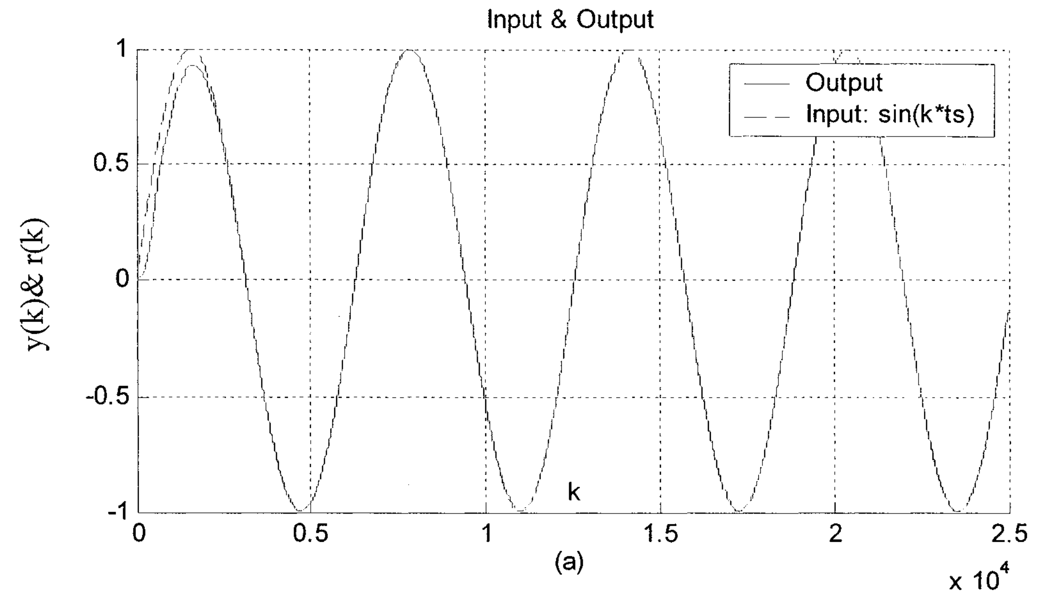
Example 7.3:

A third order plant is applied to track a sin wave signal, $\sin(t)$. The goal of the controller is to obtain asymptotic position tracking accuracy within 0.005 units. For this design target, the parameters of Gaussian networks controller are chosen as below:

$\Phi = 0.005$	$l = 4$
$\beta = 2$	$\mathrm{N} = 961$
$\theta = 2$	k_d=220
Δ= 0.125	k_a=0.01
$\sigma_v^2 = 4\pi$	k_{sl}=20

The plant that is illustrated in discrete form is given as follows:

$$z^{-1} \cdot \frac{0.0333 + 0.1330z^{-1} + 0.0332z^{-2}}{1 - 2.9950z^{-1} + 2.9900z^{-2} - 0.9950z^{-3}} \times 10^{-8}$$



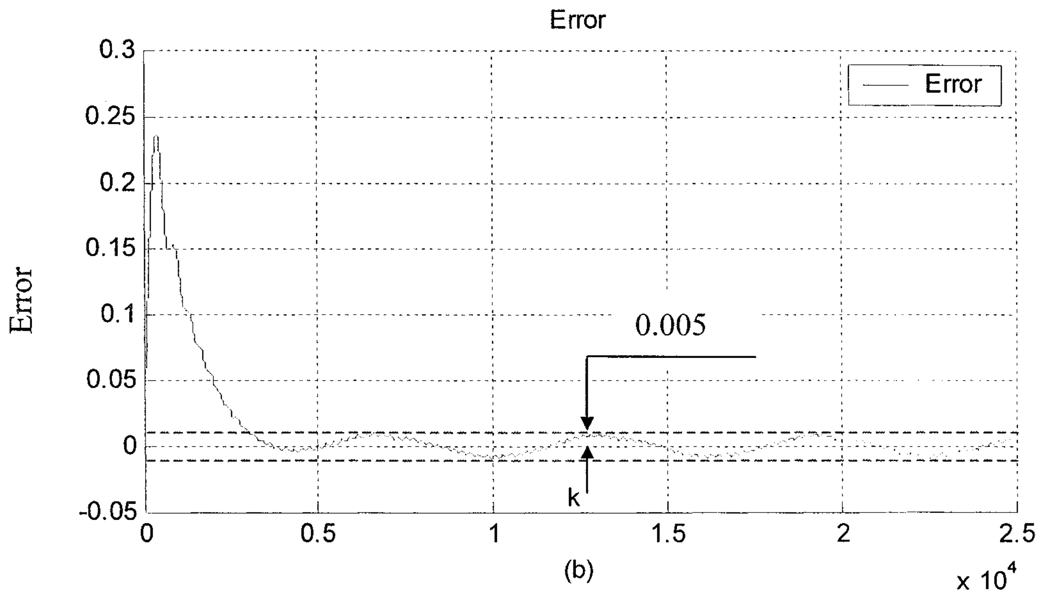


Figure 7.3 (a) Input & output (b) Tracking error

for the second order plant with the input: sin(k*ts)

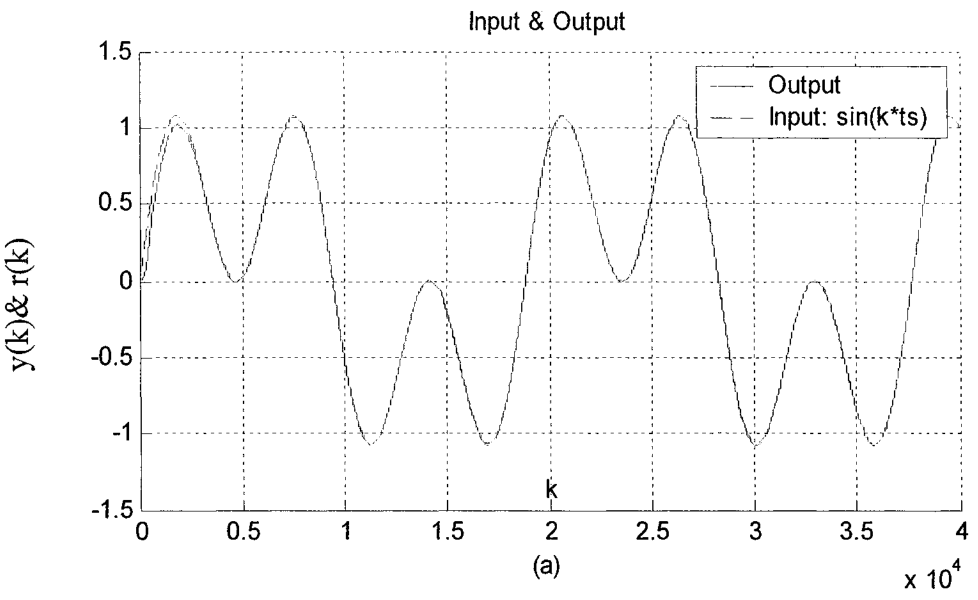
Example 7.4:

A second order plant is applied to track a sin wave signal: $0.5\sin(t)+0.5\sin(t/3)$. The goal of the controller is to obtain asymptotic position tracking accuracy within 0.005 units. For this design target, the parameters of Gaussian networks controller are chosen as below:

$\Phi = 0.005$	$l = 4$
$\beta = 2$	$\mathrm{N} = 961$
$\theta = 2$	$k_d = 220$
$\Delta = 0.125$	$k_a = 10$
$\sigma_v^2 = 4\pi$	$k_{sl} = 30$

The plant that is illustrated in discrete form is given as follows:

$$z^{-1} \cdot \frac{0.0333 + 0.1330z^{-1} + 0.0332z^{-2}}{1 - 2.9950z^{-1} + 2.9900z^{-2} - 0.9950z^{-3}} \times 10^{-8}$$



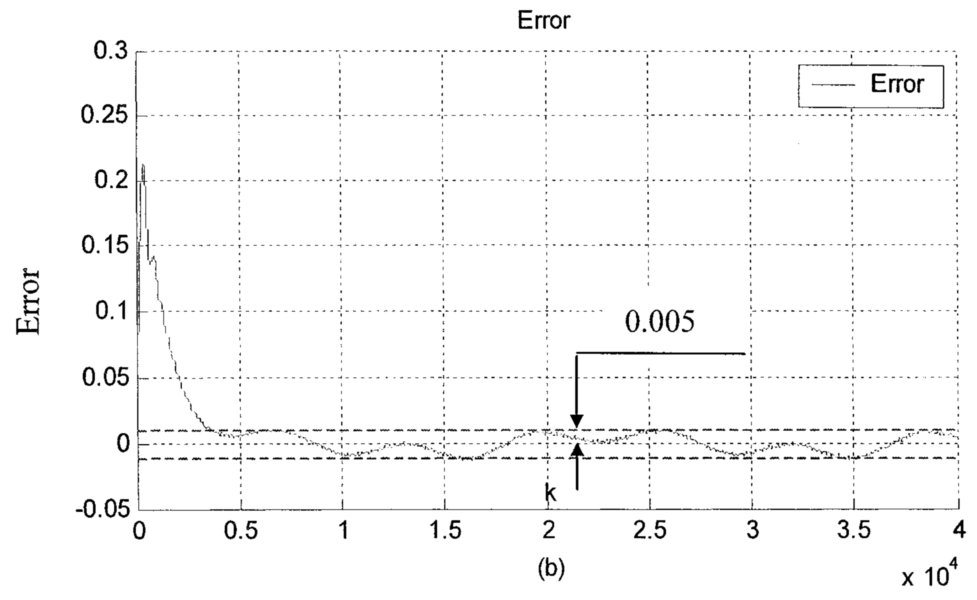


Figure 7.4 (a) Input & output (b) Tracking error

for the second order plant with the input: 0.6sin(k*ts)+0.6sin(k*ts/3)

Chapter 8

Conclusion and future work

Conclusion:

For a system with hysteresis, we have developed a neural network based hysteresis inverse compensator to remove the effect of hysteresis. Based on the cancellation of hysteresis effect, we propose an adaptive tracking control architecture, which can be constructed via the combination of sliding mode and Gaussian networks.

A specialized recurrent neural network called the diagonal recurrent neural network (DRNN) is applied to construct the hysteresis inverse compensator because DRNN requires fewer weights, less training time, and still preserves the dynamic characteristics, which allow the DRNN model to be used for on-line applications. An on-line learning algorithm called the dynamic back propagation (DBP) algorithm has been developed to train the DRNN.

We propose a simplified on-line hysteresis inverse learning architecture by which the hysteresis inverse modeling can be obtained without using a hysteresis identifier. A simple structure is desired in the real-time application because simple structure implies that less parameters need to be updated and therefore less computation is required. Moreover, simple structure will allow a reduction in the random influence that may result in learning structure failure. Simulation results indicate that this structure can achieve high inverse modeling precision.

The update rule for weights adaption requires choosing a proper value for the learning rate. With a small learning rate, the convergence of DRNN is guaranteed but the learning speed is very slow; on the other hand, if the learning rate is too big, the algorithm becomes unstable. In this thesis we have developed a guideline for selecting the learning rate properly, which ensures the fastest possible learning rate while maintaining the stability of DRNN.

Under the guarantee that the hysteresis effect can be mostly removed by the DRNN compensator on-line, a hybrid controller, which is combined with sliding mode and tree-layer Gaussian network (GNNC), has been developed. A unique feature of this control law is its ability allow the smooth transition from adaptive to nonadaptive modes of operation; becoming a), a sliding controller in the regions of the state space where the network has poor approximating capability, b), a purely adaptive controller where the network approximating power is good, or c), a stabilizing blend of the two modes in an intermediate transition region.

In conclusion, the proposed control strategy, which constructs GNNC controller and DRNN compensator in the close loop system, is successful for the control of a class of dynamic systems preceded by hysteresis. The simulation results show that the control system can track cyclic and acyclic signal very well.

Future work:

Even though the proposed control strategy is efficient for a class of dynamic system preceded by hysteresis. However, this kind of control method limits its use in more general hysteric system. It is suppose that a neural network based identifier should be used to acquire the information of the hysteric system, and then accordingly control methodology will be generated.

References:

[1] Su, C.Y., Stepanenko, Y., Svoboda, J., Leung, T.P., "Robust Adaptive Control of a Class of Nonlinear Systems with Unknown Backlash-Like Hysteresis," IEEE Transactions on Automatic Control, vol. 45, no. 12, December 2000.

[2] Stepanenko, Y., Su, C.Y., "Intelligent Control of Piezoelectric Actuator," 37th IEEE CDC, Tampa, Florida, USA, pp.4234-4239, 1998.

[3] Lu, Tianshun, Tan, Yonghong, Su, C.Y., "Pole Placement Control of Discrete Time Systems Preceded with Hysteresis via Dynamic Neural Network Compensator," Proceedings of the 2002 IEEE International Symposium on Intelligent Control, pp. 228 –233, 2002.

[4] Del Vecchio, P., Salvini, A., "Neural Network and Fourier Descriptor Macromodeling Dynamic Hysteresis," IEEE Transactions on Magnetics, Volume: 36 Issue: 4, pp. 1246 –1249, Jul 2000.

[5] Kuczmann, M., Ivanyi, A., "A New Neural-network-based Scalar Hysteresis Model," IEEE Transactions on Magnetics, Volume: 38 Issue: 2, pp. 857 –860 Mar 2002.

[6] Sain, P.M.,. Sain, M.K., Spencer, B.F., "Models for Hysteresis and Application to Structural Control," Proceedings American Control Conference, pp. 16-20, June1977.

[7] Chua, Leon O., Bass, Steven C., "A Generalized Hysteresis Model," IEEE Transactions on Circuit Theory, Vol., CT-19, No.1, PP.36-48, Jan.1972

[8] Han J.M., Adriaens, T.A., de Koning, Willem L., Banning, Reinder., "Modeling Piezoelectric Actuators," IEEE/ASME Transactions on Mechatronics. Vol.5, No. 4, Dec. 2000.

[9] Visintin, Augusto, "On Hysteresis in Elasto-plasticity and in Ferromagnetism," International Journal of Non-Linear Mechanics 37, pp.1283-1298, 2002.

[10] Cruz-Hernandez, Juan Manuel, Hayward, Vincent, "Reduction of Major and Minor Hysteresis Loops in a Piezoelectric," Proceedings of the 37th IEEE Conference on Decision & Control, Tampa, Florida USA. Dec. 1998.

[11] Majima, Sumiko, Kodama, Kazuyuki, Hasegawa, Tadahiro, "Modeling of Shape Memory Alloy Actuator and Tracking Control System with the Model," IEEE Transactions on Control Systems Technology, Vol.9, No. 1, PP. 54-59, Jan. 2001.

[12] Visone, C., Serpico, C., "Hysteresis Operators for the Modeling of Magnetostrictive Materials," Physica B 306 (2001), pp.78-83, 2001.

[13] Visonea, C., Serpicob, C., Mayergoyzb, J.D., Huangb, M.W., Adly, A.A., "Neural-Preisach-type Models and their Application to the Identification of Magnetic Hysteresis from Noisy Data," Physica B 275, pp.223-227, 2000.

[14] Subbarao, Varigonda, "Periodic Oscillations in Systems with Hysteresis," Proceedings of the 38th Conference on Decision & Control Phoenix, Arizona USA, Dec. 1999.

[15] Venkataraman, R., Krishnaprasad, P.S., "Approximation Inversion of Hysteresis: Theory and Numerical Results," Proceedings of the 39th IEEE Conference on Decision & Control, Sydney, Australia, December 2000.

[16] Tao, G., Kokotovic, P.V., "Adaptive Control of Plants with Unkown Hystereses," IEEE Transactions on Automatic Control, vol. 40, no. 2, pp. 200-212, 1995.

[17] Smith, R.C., "Inverse Compensation for Hysteresis in Magnetostrictive Transducers," CRSC Technical Report, North Carolina State University, CRSC- TR98-36, 1998.

[18] Smith, Ralph C., Bouton, Chad., "Partial and Full Inverse Compensation for Hysteresis in Smart Material System," CRSC Technical Report, North Carolina State University, CRSC- TR00-06, 2000.

[19] Kuhnen, K., Janocha, H., "Compensation of the Creep and Hysteresis Effects Piezoelectric Actuators with Inverse System," available on line at http://www.lpa.uni-saarland.de/pdf/act98.pdf

[20] Natale, C., Velardi, F., Visone, C., "Modelling and compensation of hysteresis for magnetostrictive actuators," in Proceedings of IEEE/ ASME International Conference on Advanced Intelligent Mechatronics, pp. 744-749, 2001.

[21] Tan, Xiaobo., Baras, John S., "Modeling and Control of a Magnetostrictive Actuator," Proceedings of the 41st IEEE Conference on Decision and Control, Las Vegas, Nevada USA. December 2002.

[22] Ahmad, N.J., Khorrami, F., "Adaptive Control of Systems with Backlash Hysteresis at the Input," Proceeding of American Control Conference, pp. 3018-3022, 1999.

[23] Cruz-Hernandez, Juan Manuel: Member, IEEE, Hayward, Vincent: Member, IEEE, "Phase Control Approach to Hysteresis Reduction," IEEE Transactions on Control Systems Technology, Vol. 9, No. 1, pp. 17-26, Jan. 2001.

[24] Selmic, Rastko R., Lewis, Frank L., "Backlash Compensation in Nonlinear Systems Using Dynamic Inversion by Neural Networks," Proceedings of the IEEE, International Conference on Control Applications, 1991.

[25] Nealis, J.M., Smith, R.C., "An Adaptive Control Method for Magnetostrictive Transducers with Hysteresis" Proceedings of the 4th IEEE Conference on Decision and Control, Orando, Florida USA, December 2001.

[26] Mittal, S., Meng, C.H., "Hysteresis Compensation in Electromagnetic Actuators Though Preisach Model Inversion," IEEE/ASME Transactions on Mechatronics, 101. 5, no. 4, December 2000.

[27] Ku, C.C., K. Y. Lee, K.Y., "Diagonal Recurrent Neural Networks for Nonlinear System Control," International Joint Conference on Neural Network, Vol. 1, pp. 315-320, Baltimore, 1992.

[28] Ku, C.C., Lee, Y., "Diagonal Recurrent Neural Network Based Control Using Adaptive Learning Rates", IEEE Conference on Decision and Control, Tucson, Arizona, Dec. 18-18,1992.

[29] Ku, C.C., Lee, K.Y., "Diagonal Recurrent Neural Networks for Dynamic System Control", IEEE Transactions on Neural Networks, Vol. 5, No.4, July 1994.

[30] Ku, C.C., Lee, K.Y., "Diagonal Recurrent Neural Networks for Dynamic Systems Control," IEEE Transactions on Neural Networks, Vol.6, No. 1, JANUARY 1995.

[31] Wang, Pan., Li, Youfeng., Feng, Shan., Wei, Wei., "Convergence of Diagonal Recurrent Neural Networks Learning," IEEE Proceedings of the 4th World Congress on Intelligent Control and Automation, Shanghai. P.R.China, June 10-14. 2002.

[32] Wang, Yongji., Fernholz, Gregor., Engell, Sebastian., "A Second Order Recursive Prediction Error Algorithm for Diagonal Recurrent Neural Networks," Proceedings of the 1998 IEEE International Conference on Control Applications Trieste, Italy 1-4 September 1998.

[33] Ii, Xiao D., Familoni, Babajide O., "A Diagonal Recurrent Neural Network Based Hybrid Direct Adaptive SPSA Control System," IEEE Transactions on Automal1C Control, Vol. 44, NO.7, JULY 1999.

[34] Cho, Jae-Soo., Kim, Yong-Woon., Park, Dong-Jo., "Identification of Nonlinear Dynamic Systems Using Higher Order Diagonal Recurrent Neural Network," IEE Electronics Letters, Vol. 33, No. 25, Dec. 4, 1997.

[35] Vowal, H., Kada, A., RamztM., El-lsmalli, L., "Fast Diagonal Recurrent Neural Networks for Identification and Control of Nonlinear Systems," IEE Control, '94, 21-24 March 1994. Conference Publication No. 389, 1994.

[36] Xiao D., Ji Babaiide., Familnni, O.,"Experimental Study of Direct Adaptive SPSA Control System With Diagonal Recurrent Neural Network Controller," IEEE 1996.

[37] Chow, Tommy W.S., Fang, Yong., “A Recurrent Neural-Network-Based Real-Time Learning Control Strategy Applying to Nonlinear Systems with Unknown Dynamics,” IEEE Transactions on Industrial Electronics, Vol.45. No. 1. Feb.1998.

[38] Cabrera, J. B. D., Narendra, K.S., "Issues in the Application of Neural Networks for Tracking Based on Inverse Control," IEEE Transactions on Automatic Control. Vol. 44, No. 11, November 1999.

[39] Kambhampati, C., Craddock, R., “Inversion of Recurrent Neural Networks for Control of Non-linear Systems,” IEEE 10th International Conference on Global Connectivity in Energy, Computer, Communication and Control, Vol. 2, pp. 642 –647, 1998.

[40] Lu, Xiqun., Chen, Chun., "A New Hybrid Recurrent Neural Network,” Proceedings of IEEE International Symposium on, Vol.5, pp. 616 –618, 1999.

[41] Wai, Rong-Jong., Hong, Chun-Ming., “Adaptive Recurrent-neural-network Control for Linear Induction Motor,” Proceedings of IEEE International Conference , Control Applications, pp. 184 –189, 2000.

[42] Kwan, H.K., Yan, J., “Recurrent Neural Network Design for Temporal Sequence Learning,” Proceedings of the 43rd IEEE, Circuits and Systems, Midwest Symposium on, Vol. 2, pp. 832 –835, 2000.

[43] Liu, D., “Open-loop Training of Recurrent Neural Networks for Nonlinear Dynamical System Identification,” International Joint Conference on, Volume: 2, pp. 1215 –1220, 2001.

[44] Jiang, Danchi., Wang, Jun., "On-line Learning of Dynamical Systems in the Presence of Model Mismatch and Disturbances," IEEE Transactions on Neural Networks, Vol. 11 Issue 6, pp. 1272 –1283, Nov 2000.

[45] Wai, Rong-Jong., Lin, Faa-Jeng., "Adaptive Recurrent-neural-network Control for Linear Induction Motor," IEEE Transactions on Aerospace and Electronic Systems, Vol. 37 Issue 4, pp. 1176 –1192, Oct 2001 .

[46] Aussem, A., "Sufficient Conditions for Error Back Flow Convergence in Dynamical Recurrent Neural Networks," Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on, Vol. 4, pp. 577 –582, 2000.

[47] Mandic, D.P., Chambers, J.A., and Bozic, M.M., "On Global Asymptotic Stability of Fully Connected Recurrent Neural Networks," IEEE International Conference on, Vol. 6, pp. 3406 –3409, 2000.

[48] Wang, Dai-Hu., Liao, Wei-Hsin., "Neural Network Modeling and Controllers for Magnetorheological Fluid Dampers," The 10th IEEE International Conference on Fuzzy Systems, Vol. 3, pp. 1323 –1326, 2001.

[49] Sanner, Robert M., Slotine, Jean-Jacques E., "Stable Adaptive Control and Recursive Identification Using Radial Gaussian Networks," IEEE CDC, Dec. 1991.

[50] Mayosky, Miguel Angel., Cancelo, Gustavo I. E., "Direct Adaptive Control of Wind Energy Conversion Systems Using Gaussian Networks," IEEE Transactions on Neural Networks, Vol.10, No.4, July 1999.

[51] Won, Mooncheol., Choi, Seibum B., Hedrick, J. K., "Air-to-Fuel Ratio Control of Spark Ignition Engines Using Gaussian Network Sliding Control," IEEE Transactions on Control Systems Technology, Vol.6, No.5, Sept.1998.

[52] Sanner, Robert M., Slotine, Jean-Jacques E., "Gaussian Networks for Direct Adaptive Control," IEEE Transactions on Neural Networks, Vol.3, No. 6, Nov.1992.

[53] Onder, Efe M., Kaynak, O., Yu, X., Wilamowski, B. M., "Sliding Mode Control of Nonlinear Systems Using Gaussian Radial Basis Function Neural Networks," Neural Networks, 2001. Proceedings. IJCNN '01, International Joint Conference on, Vol.1, pp. 474 -479, 2001.

[54] Wasserman, P., "Neural Computing Theory and Practice," New York: Van Nostrand Reinhold, 1989.