Intelligent systems for active vibration control in flexible engineering structures

Xiaoxu Ji

A thesis presented to Lakehead University

In partial fulfillment of the requirement for the degree of

Master of Science in Control Engineering

Thunder Bay, Ontario, Canada, 2008

Your file Votre référence
ISBN: 978-0-494-47138-8
Our file Notre référence
ISBN: 978-0-494-47138-8

Abstract

Vibration arises in almost all moving structures. Vibration control is important to many applications such as robotic arms, aircraft wings, buildings in wind, vehicle transmission systems, to name but a few. The objective of this thesis is to develop more efficient intelligent controllers for vibration suppression, mainly for time-varying flexible structures.

At first, based on TS0 and TS1 fuzzy models, novel neural-fuzzy (NF) controllers are developed for active vibration control of the flexible structures. The NF control paradigms are intended to integrate the advantages from both fuzzy logic and neural networks while overcoming their respective limitations. The control reasoning is undertaken by fuzzy logic whereas the fuzzy control system is optimized by neural network related training algorithms. A new strategy is suggested to simplify the architectures of the classical NF controllers so as to make the control process computationally efficient for real-time applications. A recurrent identification network (RIN) is developed to adaptively identify system dynamics of the time-varying flexible structures. When system dynamics (e.g., mass, stiffness, and damping) varies, the proposed RIN and NF controller can effectively recognize the system's new dynamics and perform corresponding control operations. A novel hybrid training technique based on real time

recurrent learning (RTRL) and least square estimate (LSE) is suggested for real-time training of the RIN scheme to optimize its nonlinear input-output mapping.

The effectiveness of the developed intelligent controllers and the related techniques has been verified by online experimental tests of corresponding fixed and time-varying dynamic conditions. Test results have shown that the developed adaptive NF controller outperforms the classical controllers (e.g., PD) and other related intelligent control strategies.

Acknowledgements

I wish to express my sincere appreciation to Dr. Wilson Wang. His guidance and support were essential to the completion of this thesis. I would like to thank my co-supervisor Dr. Abdelhamid Tayebi for his help and suggestions. I would also like to acknowledge Dr. Kefu Liu and Dr. Sultan Siddiqui for their efforts as examiners of this thesis. I would like to thank Dr. Xiaoping Liu for his help in the aspect of linear control.

I thank all my friends at Lakehead University for their friendship, moral support and technical discussion during the past two years.

Most importantly, I want to thank my parents for their encouragement and support on my graduate studies.

Contents

List of figures

List of Tables

Chapter 1 Introduction

1.1 Overview

Vibration control and associated smart structure techniques are critically needed in many engineering applications such as robotic arms in the precise operations, aerospace systems, buildings in wind, vehicle transmission systems, etc. This research will tackle the control problems in engineering systems especially for some more challenging flexible structures. The flexible structures contain infinite number of vibration modes, and have low rigidity and small material damping. Even a small external excitation may lead to large amplitude vibration and/or long vibration decreasing time. The following summarizes the development of vibration control related to flexible structures in the literature.

1.2 Literature review

Generally speaking, vibration control can be undertaken either passively or actively. The passive vibration control methods utilize the passive elements such as masses, dampers and springs, to adjust the characteristics of controlled structures to suppress vibration [1]. Although the passive control is relatively simple in principle, passive control techniques are usually difficult to apply in the low frequency applications [2]. Furthermore, in many applications such as space vehicles, it is desirable to keep the weight as low as possible, correspondingly passive control with extra hardware systems becomes unattractive [3]. Since the last decade, modeling and active control of engineering structures have received more interests in both research and applications [4-6]. Different from the passive control methods, the active vibration control

supplies energy to suppress the vibration; it can provide higher control performance than most passive controllers. The implementation of active vibration control, however, needs systematic system analysis and modeling, vibration measurement, actuator elements and the controller design and implementation. Based on comprehensive comparison studies, active vibration control will be applied in this work.

In the literature, a variety of control strategies have been reported for active vibration control, which are schematically summarized in Figure 1.1.

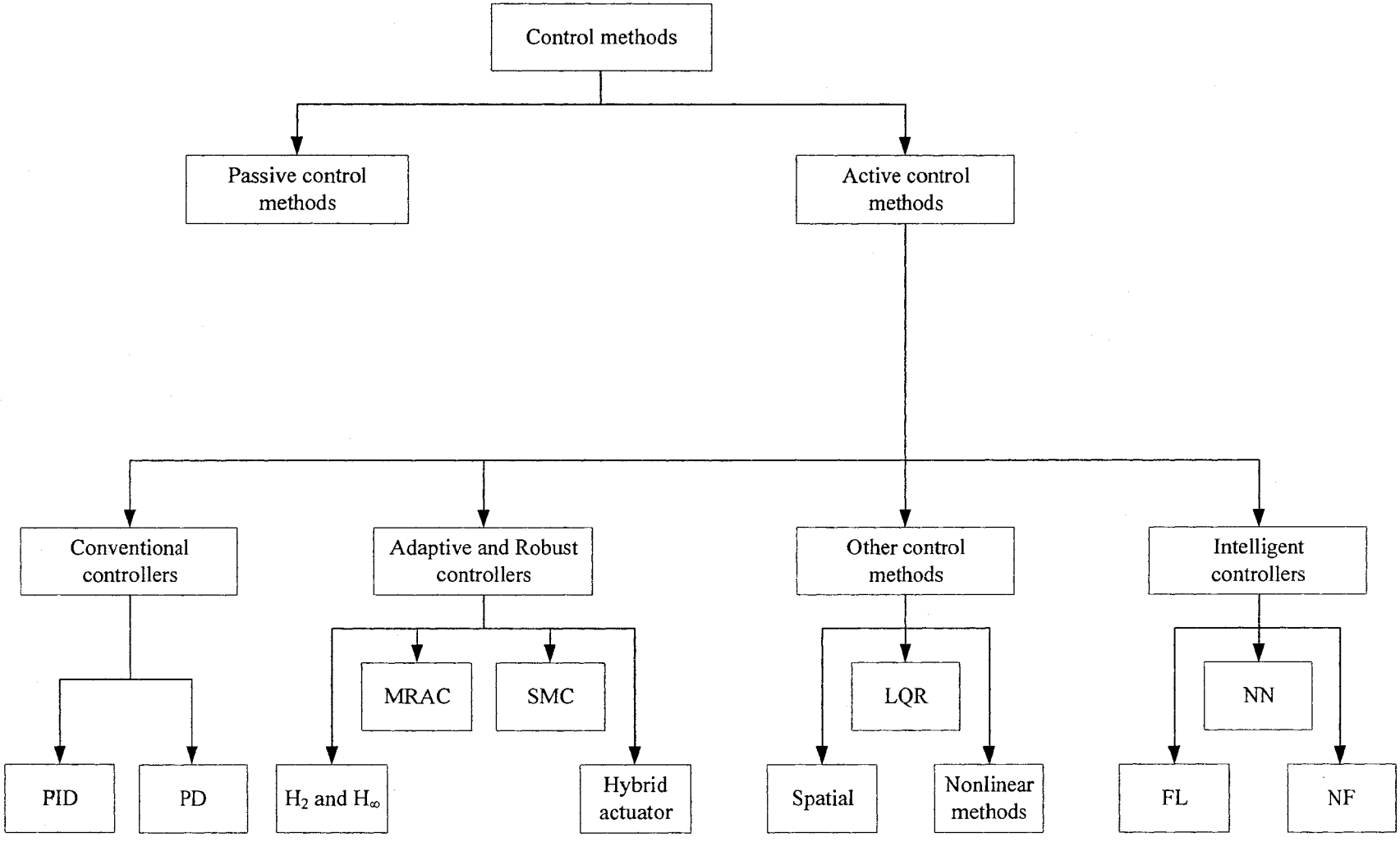


Figure 1.1. General classification of control methods.

1.2.1. Classical Vibration Control

The classical linear controller such as proportional-integral-derivative (PID) [7] or proportional-derivative (PD) [8] has been widely applied to flexible beams or manipulators. However, these linear controllers are sensitive to parameter variations and load disturbance; their performance also depends on operating conditions and controller's gains. It is usually difficult to adjust controller (e.g., PD and PID) gains to tackle the overshoot and load disturbance rejection problems simultaneously. In general, overshoot elimination settling will cause a poor load disturbance rejection, whereas rapid load disturbance rejection setting will cause overshoot or even instability in the system.

1.2.2 Adaptive and robust vibration control

In order to avoid the shortcomings of PD controllers, one important class of controllers are referred as adaptive feed forward controllers [9-11]. The suggested control methods in [9-11] are based on Model Reference Adaptive Control (MRAC), in which the process of system identification is followed by the adaptation of the controller as a function of the identified system parameters [12-15]. A typical approach to adaptive vibration control is to feed back an error signal through an appropriate filter (e.g., a simple low pass filter) and to apply the resulting signal to the plant. The parameters of the filters are tuned automatically by an adaptive algorithm to achieve the best vibration reduction. The most commonly used adaptive algorithm is the Least Mean Square (LMS) algorithm [16-18].

The other main class of controllers as described in the vibration control literature are linear feedback controllers which are designed by using robust control design techniques [19-24].

Another type of active vibration control is based on adaptive and robust control strategies, such as H_2 control [25, 26], H_∞ control [27-29], and sliding mode control (SMC) [30-37]. In paper [38], for example, some hybrid actuator schemes are suggested for the robust control against various modeling uncertainties. However, the performance of these controllers is usually highly dependent on accuracy of system models and their parameters, which limits these controllers' limitations in smart structure applications. Furthermore, calculation in these adaptive and robust controllers is time-consuming, which makes them difficult to implement for real-time control application for time-varying systems.

1.2.3 Other classical vibration control

Other classical vibration control approaches that can be mentioned consist of: a) spatial control where vibration is minimized over an entire structure [39]; b) linear-quadratic-Gaussian (LQG) controller synthesis by which both the small amplitude random vibrations and higher order modes in flexible beams can be suppressed [40, 41]; and c) nonlinear methods for improving the active control efficiency of Smart Structures [42-44].

1.2.4 Intelligent vibration control

In order to overcome the disadvantages of classical PD, adaptive and robust controllers, as well as nonlinear control methods, recently researchers have applied intelligent controllers for flexible beam applications. The main advantages of intelligent controllers are the controllers' independent of the system parameters, and their applications to the nonlinear systems.

The neural network (NN) is well known for its learning ability and approximation to any arbitrary continuous functions. It also possesses the tolerant capability in performing in noisy environments and operating with faulty and missing data. Therefore, NNs can be trained to map

complex input-output mapping functions in various fields, including pattern recognition, identification, classification, speech, vision, and control systems. Some works have been reported on the use of NN controllers for flexible beams [45, 46]. Because the NNs can adjust their inter-connections among network nodes to achieve optimal or near optimal input-output mappings, the major advantage of NN learning is its ability to accommodate even poorly modeled and nonlinear dynamical systems. NN-based controllers may not be suitable for some linear or linearizable systems control, which can result in degradation in performance in terms of computation time and controller convergence [47]. Unfortunately, the resulting distributed knowledge representation by NNs is usually difficult to understand for designers.

In order to overcome the need for precise process representation, Zadeh introduced the soft-computing concept of fuzzy logic (FL), which may be viewed as a parallel representation to probability theory rather than as an alternative. FL aims at modeling the imprecise systems of reasoning, by common sense reasoning, for uncertain and complex processes. The theory of FL controller is based on the linguistic rules with an IF-THEN general structure, simulating the human logic based reasoning. These characters make FL controllers more attractive to flexible beams [48-53]. FL control has proven effective for complex, nonlinear and imprecisely defined systems for which classical model-based control techniques are impractical or impossible. On the other hand, FL control design also involves a number of difficulties. A common bottleneck is how to properly set up fuzzy control rules; this task is usually time consuming and difficult, and relies to a great extent on the inputs from the process experts who, however, may not be able to transcribe their knowledge into the requisite rule form. To optimize control performance, the related fuzzy parameters, membership functions (MFs), and fuzzy sets for the input/output variables have to be optimized by trial and error. Unfortunately there exists no formal framework

for the choice of the fuzzy sets and parameters of fuzzy systems; That is, fuzzy systems lack self-tuning and optimization capabilities.

A solution to integrate the strengths of both FL and NNs but to overcome their respective limitations is to use their synergetic schemes, in which the FL provides the controller a high-level IF-THEN control reasoning framework, whereas the controller structure and parameters are optimized by the NN-based training. In another word, the synergetic schemes can integrate the properties of transparent and linguistic control rules of FL and the learning ability of the NNs. Neural fuzzy (NF) controllers have attracted much attention since the last decade, and have been utilized by researchers for several flexible manipulator applications [54-57]. For example, the author in [58] presents an NF controller for the trajectory tracking of four degree-of-freedom rigid-link flexible and rigid joint Selectively Compliance Assembly Robot Arm (SCARA)-type manipulators; A backpropagation related training algorithm is suggested to fine tune the controller parameters. In paper [59], the authors utilize the parallel processing characteristics of the NF system to solve a trajectory design problem of a robot manipulator. The technique of this NF system replaces the rule base of a traditional FL system with a backpropagation NN. A stable discrete-time adaptive tracking controller is proposed in [60] using an NF dynamic-inversion for a robotic manipulator; it is shown that the NF variable structure control can enhance the stability of the controlled system and improve the system dynamic performance.

Despite the aforementioned advantages of NF controllers, the industry is still reluctant to directly implement these NF controllers for real-world industrial applications due to high computational burden of the controllers. A typical NF controller consists of the following layer functions: inputs, fuzzification, normalization, rules, and defuzzification, which leads to the higher complexity compared to the FL controller and NN controller. The high complexity

(associated with the large number of MFs, rule weights, premise and consequent parameters, etc) causes high computational overhead. High computation burden will lead to low sampling frequency, which makes it difficult to implement an NF controller for real-time applications. Furthermore, in some practice, NF controllers may need more control requests (e.g. voltages) than some classical control strategies (e.g., PD) to achieve the required control performance.

1.2.5 System identification

Most of these above techniques, however, are model based and require an exact knowledge of the flexible structure dynamics; It is usually difficult to derive accurate system models in most real engineering applications where the mechanical systems to be controlled are complex in structures and operate under noisy and/or uncertain environments. One of the solutions to these problems is to apply identification system to the control designing.

Nonlinear structural dynamics has been studied over a relatively long time, but the first contribution to the identification of nonlinear structural models dated back to the 1970s, for example, by Ibanez [61] and Masri *et al.* [62]. Since then, numerous methods have been proposed in the literature because of the highly individualistic nature of nonlinear systems. This research field has also attracted more researchers to concentrate on, especially in recent years. We note that:

- The first textbook Nonlinearity in Structural Dynamics: Detection, Identification and Modelling was written by Worden and Tomlinson (2001 [63]).
- Synthesis of nonlinear system identification in structural dynamics was made in several survey papers (Adams and Allemang, 1998 [64]; Hemez and Doebling, 2000 [65]; Worden, 2000 [66]; Hemez and Doebling, 2001a [67]).

- From 1997 to 2001, a working group in the framework of the European Cooperation in the field of Scientific and Technical Research Action F3 Structural Dynamics was devoted to the Identification of Nonlinear Systems (Golinval et al., 2003b [68]). The researches focused on two benchmarks, namely the Ecole Centrale de Lyon benchmark and the benchmark from the Technical Research Center of Finland (Juntunen, 2003 [69]; Thouverez, 2003 [70]), with different techniques.
- A special issue on Nonlinear System Identification was published in the Nonlinear Dynamics journal (2005 [71]).

System identification is especially useful for modeling system when the model cannot be easily represented in terms of first principles or known physical laws. Thus, one major difficulty of nonlinear system identification is that the functional S [•], which maps the input x (t) to the output y (t), y (t) = S [x (t)], is generally unknown beforehand. Physical insight is most often of great help to select a reasonably accurate model of the nonlinearity. Only if this gives unsatisfactory results or if physical insight is completely lacking, it is then time to move to nonlinear black-box modeling.

Some methods for performing the nonlinear mapping are summarized as follows:

- *Artificial neural networks* (NNs) have come into prominence because of their universal approximation features;
- *Wavelet networks* are attractive because they unify multi-resolution features of wavelet bases and universal approximation features of neural networks;
- *Splines* are interesting functions, because they are computationally very simple, can be made as smooth as desired and are very economic to store;

- *NF models* combine the semantic transparency of rule-based fuzzy systems with the learning capabilities of neural networks; they can be regarded more as grey-box models [72].

Among the different choices for nonlinear black-box modeling, NNs have received the most attention in identifying nonlinear structural dynamics. This method of identifying nonlinear dynamical systems is to estimate a system model from measured input–output data [73-76]. It plays a key role in the model-based prediction of dynamical systems to target a given performance or to satisfy operating constraints. Recurrent neural networks (RNNs) [77-79] are generated by extending the structures of NNs.

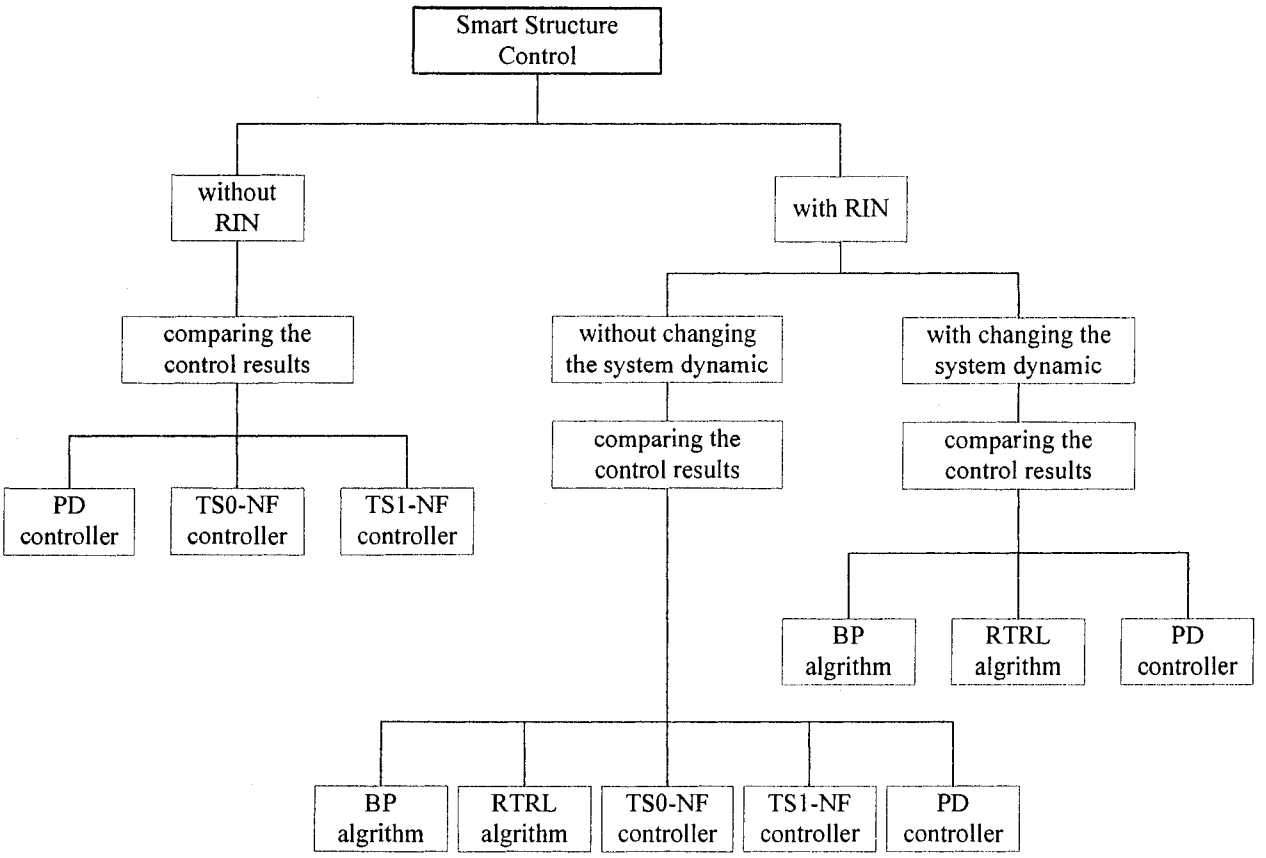
1.3 Objective of research

As discussed in literature review, the high complexity and computational burden have prevented the NF controllers from many practical applications. Correspondingly, the objective of this thesis is to develop low computational NF controllers for active vibration control of the flexible beams. One strategy is suggested to simplify the complicated structure of the conventional NF controllers while maintaining the required system performance. The proposed NF controllers can reduce the computational burden by employing carefully selected input variables. The effectiveness of the developed NF controllers is verified by experimental tests. Test results show that the new NF controllers outperform the classical controls (e.g. PD) and general NN controllers, and exhibit better performance in terms of overshoot, settling time, and the Rotating Angle (control requests).

Another objective of this work is to develop novel recurrent identification networks (RIN) to adaptively identify system models. A new hybrid training technique based on real time recurrent

learning (RTRL) method and least square estimate (LSE) method is adopted to adaptively optimize the control system model and to accommodate time-varying system dynamics. The viability of the suggested RIN and its applications in NF control are verified by experimental tests.

To simplify the illustration in the following chapters, the research outline of this work is summarized in Figure 1.2.



BP: backpropagation
RTRL: real time recurrent learning
TS0-NF: the zeroth-order Takagi-Sugeno NF controller
TS1-NF: the first-order Takagi-Sugeno NF controller

Figure 1.2. Research outline.

1.4 Thesis Organization

After the Introduction as described in this chapter, the remainder of this thesis are organized as follows. In Chapter 2, some fundamental theories related to this work are provided such as the FL systems, NN architectures, the synergetic NF schemes, as well as the related training algorithms.

In Chapter 3, the zeroth-order Takagi-Sugeno (TS0) NF controller and the first-order Takagi-Sugeno (TS1) NF controller are developed, respectively. The objective of this chapter is to simplify implementation of the proposed NF controllers.

In Chapter 4, comparison tests show that the proposed NF controllers can not only simplify control structure, but also outperform the classical controllers (e.g., PD). The related system parameters in the controllers are updated online by the corresponding hybrid training technique.

The RIN is proposed in Chapter 5 for nonlinear system model identification. The objective is to provide a more accurate system model from measured input-output data for adaptive control. The performance of the RIN is updated by the suggested hybrid trained based on the RTRL method and LSE method, so as to optimize the control RIN model and to accommodate time-varying system dynamics. The effectiveness of the suggested techniques is verified by experimental tests.

Finally, some concluding remarks as well as the future works are summarized in Chapter 6.

Chapter 2 Description of Neuro-Fuzzy Controllers

2.1 The Structure used in this Research

This structure workstation used in this research work for controller development and test is as illustrated in Figure 2.1. This experimental setup consists of a flexible beam clamped at one end while the other end is equipped with a servo motor. The servo motor drives an eccentric load which plays a role of the actuator. The purpose of this structure workstation is to actively dampen out the vibrations in the flexible beam by the developed controllers through the drive motor.

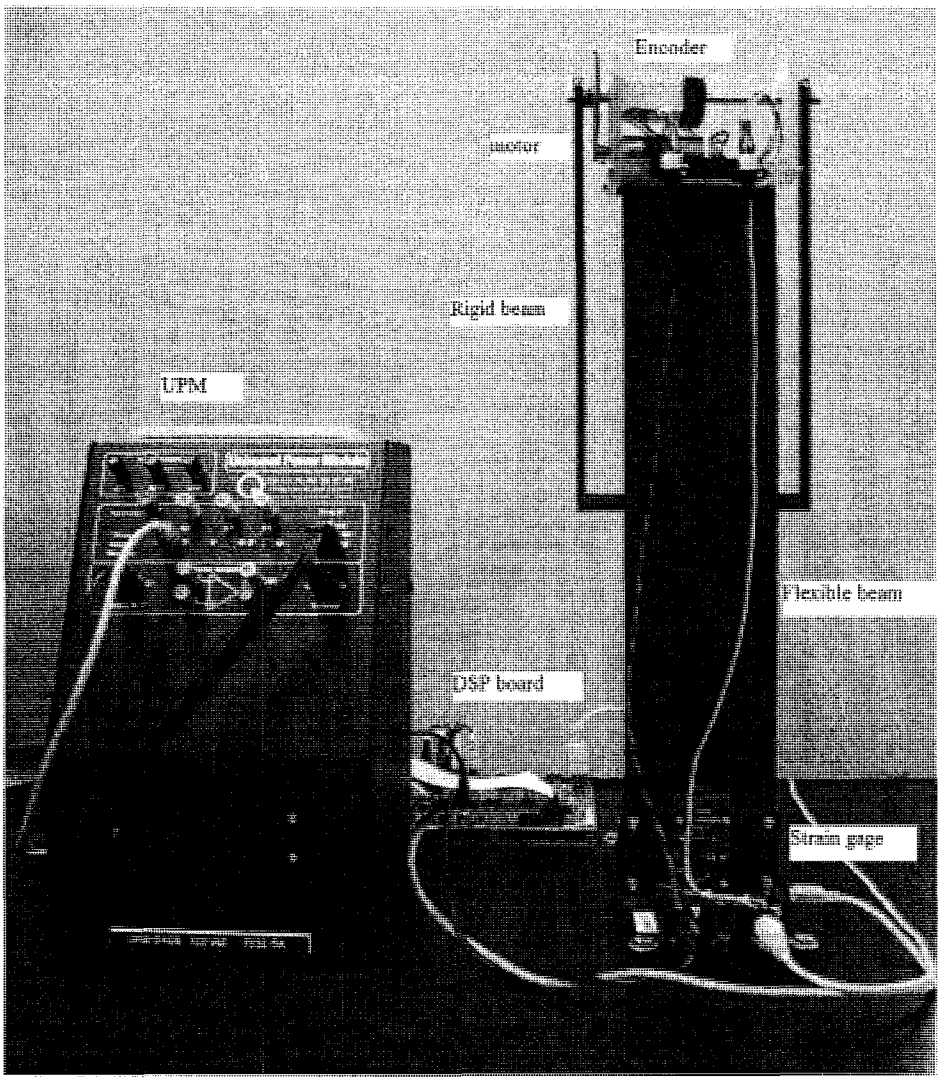


Figure 2.1. The structure system used for vibration control in the flexible beam.

This system consists of the following components:

- Base plate: The structure and some related devices are fixed onto the base plate which is rigid and anchored (by bolts) onto a desk.
- Flexible beam: The beam is instrumented with a strain gage calibrated to give 1 *volt* per 2.54 *cm*, to indirectly measure vibration. The tested beam can be in different dimension, material, and orientation.

- Motor (Series 2338S006): The motor drives the load through a gear ratio of 70:1.
- The encoder (SRV02): The encoder measures the rotation angle by using a 1024 count disc which in quadrature results in 4096 *counts/rev*.
- Strain gage (A9-232-0): The strain gage senses the beam deflection and the signal is sent back to the universal power module.
- Inertia load is applied by two rigid beams and a round cross beam.
- Universal power module (UPM-2405): The power module is a power amplifier that is required to drive each associated actuator. The power module consists of one power supply (±12 Volt), four analog sensor inputs, and one power amplified analog output.
- DSP board (A11-368-3): It is a Quanser Q4 terminal board which contains the general A/D and D/A converters. All of the connectors from the universal power module are connected to the DSP board by wires.

The tested beam can be in different structure, material and orientation. In this work, a (0.35% carbon content) steel beam with dimension (44×10.8×1.5mm) is used. In order to reduce the twisting in the member, the beam is placed in the vertical configuration. Extra magnetic blocks will be used to simulate time-varying system properties.

This system can be modeled as shown in Figure 2.2.

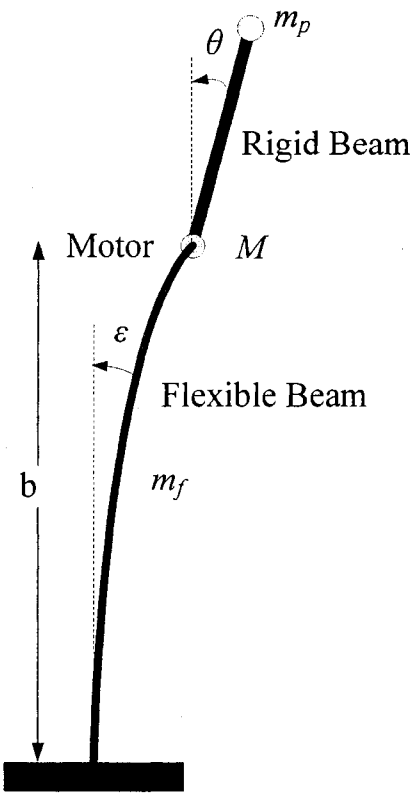


Figure 2.2. Schematic representation of the experimental setup.

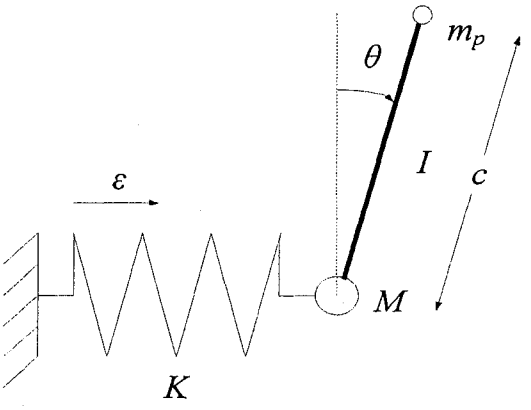


Figure 2.3. A simplified model.

The model as illustrated in Figure 2.2 can be further simplified and shown in Figure 2.3. For small deflections of the flexible beam, the motion of the motor can be considered to be linear

along the specified direction. The related parameters and their associated values used in this work are summarized in Table 2.1.

Table 2.1: Parametric values for the smart structure

Physical parameters	Symbol	Value/Units
Deflection of the flexible beam	ε	---
Rotation angle of the rigid beam from zero position	θ	---
Mass of the motor and its fixture	M	0.6 Kg
Cross beam mass	m_p	0.05 Kg
Rigid beam inertia	I	0.0039 Kg m^2
Rigid beam length	c	0.285 m
Rigid beam mass	m_b	0.072 Kg
Effective stiffness of the flexible beam along x direction	K	30 N/m
Encoder resolution		4096 Counts/rev
Flexible beam length	b	0.44 m
Flexible beam mass	m_f	0.22 Kg
Natural frequency with M attached	f_{nat}	1.25 Hz
Flexible beam effective stiffness along x (first mode only)	k	30 N/m
Strain gage sensitivity	Gs	0.4 V/cm
Strain gage gain	As	2.54 cm/volt

In the following subsections, a brief introduction is given to the background theories related to the developed intelligent controllers so as to have readers easier to follow the subsequent descriptions in the following chapters.

2.2 The fuzzy logic model structures

Fuzzy logic, FL, provides a very useful frame for representing lexically imprecise propositions, in a natural language based reasoning format [80]. Classical sets and variables have crisp boundaries. Although these classical sets are suitable tool for mathematics and computer operations, they do not reflect the nature of human concepts and thoughts, which tend to be abstract and imprecise. In contrast to a classical set, a fuzzy set, as the name implies, is a set without a crisp boundary. That is, the transition from "belong to a set" to "not belong to a set" is gradual; this smooth transition can be characterized by membership functions (MFs) that give fuzzy sets more flexibility in modeling based on IF-THEN linguistic expressions which mimic human reasoning in comparable circumstances.

As an example, if X is a collection of objects denoted generically by x, then a fuzzy set A in X is defined as a set of ordered pairs:

$$A = \{(x, \mu_A(x)) \mid x \in X\}, \tag{2-1}$$

where $\mu_A(x)$ is called the MF for the fuzzy set A. The MF maps each element of X to a membership grade (or membership value) between 0 and 1. Fuzzy sets play an important role in the areas such as nonlinear system modeling, pattern recognition, and communication of information, etc.

The fuzzy inference system is a specific computing framework based on the concepts of fuzzy set theory, fuzzy rules, and fuzzy reasoning [81], which has found many efficient applications in a wide variety of fields such as a single-link flexible manipulator [48]. The basic structure of a fuzzy inference system consists of three conceptual components: 1) a *rule base*, which contains a selection of fuzzy rules; 2) a *database*, which defines the MFs used in the fuzzy rules, and 3) a *reasoning mechanism*, which performs reasoning inference operations, based on the fuzzy rules (and some other given facts) to derive a corresponding output or conclusion.

A typical fuzzy inference system (within the dashed lines) is illustrated in Figure 2.4, where X is a collection of inputs, Y is the output. A and B are fuzzy sets. r is the rule number in the fuzzy inference system, and w_r is the weight in each rule. The basic fuzzy inference system can take either fuzzy inputs or crisp inputs, but the generated outputs are usually fuzzy sets. If a crisp output is required, such as in control applications, an appropriate defuzzification method should be applied to transform an output fuzzy set into a crisp single value.

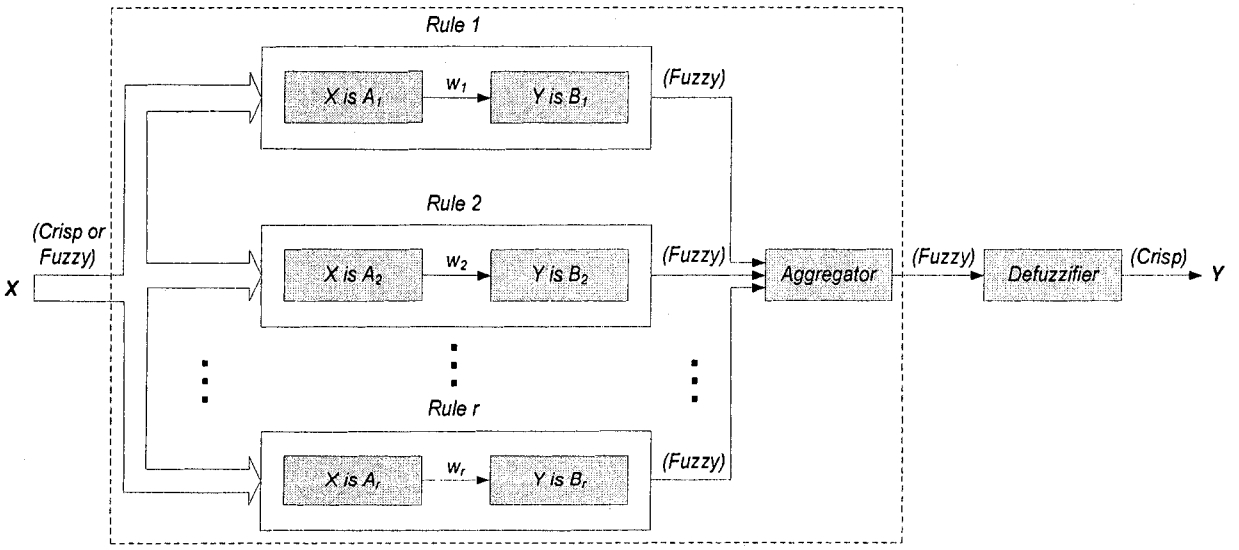


Figure 2.4. Block diagram for a fuzzy inference system.

A number of fuzzy inference systems have been suggested in the literature, such as Mandani fuzzy models (also known as linguistic fuzzy systems) [82] and the Takagi-Sugeno (TS) fuzzy systems [83, 84]. A typical TS fuzzy fuzzy model has the form of

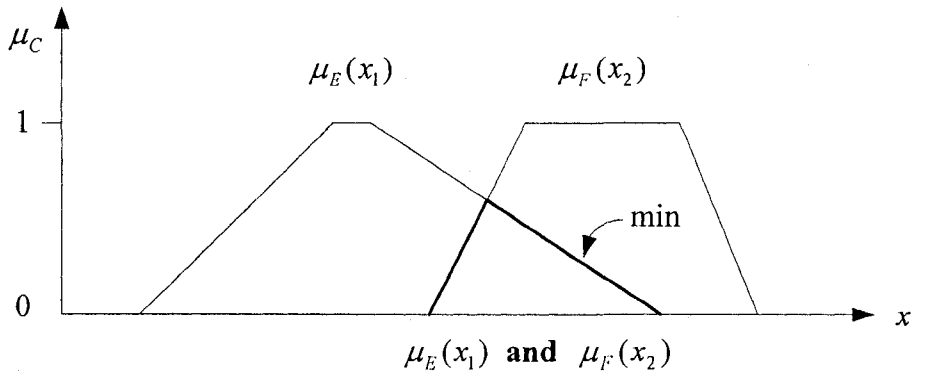
$$\text{IF } (x_1 \text{ is } A) \text{ AND } (x_2 \text{ is } B), \text{ THEN } y = f(x_1, x_2), \tag{2-2}$$

where A and B are input fuzzy sets (antecedent), and $y = f(x_1, x_2)$ is an output function (consequent). If $f(x_1, x_2)$ is a polynomial of the input variables x_1 and x_2, equation (2-2) is called the first order TS model, or TS1. Otherwise, if $f(x_1, x_2)$ is a constant, equation (2-2) becomes the zeroth order TS fuzzy model, or TS0. TS0 can be viewed as a special case of the Mamdani fuzzy inference system, in which each rule's consequent is a fuzzy singleton [82].

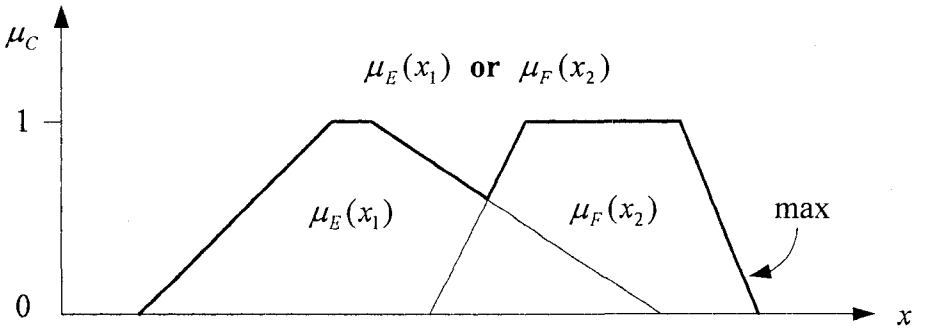
Fuzzy operations require appropriate logical connective operators such as AND (conjunction), OR (disjunction) or NOT (complement). The generalization of conjunction (intersection) to fuzzy sets is called t-norms, such as the minimum and the product operators. Given $C = E \cap F$, the rule firing strength will be

$$\mu_C(x) = \min(\mu_E(x_1), \mu_F(x_2)) \tag{2-3}$$

where $\mu_E(x_1)$ and $\mu_F(x_2)$ are the MFs for inputs x_1 and x_2, respectively. The minimum operator can also be denoted by $\mu_C(x) = \mu_E(x_1) \wedge \mu_F(x_2)$, as shown in Figure 2.5(a)



(a) conjunction



(b) disjunction

Figure 2.5. (a) Conjunction of two fuzzy sets; (b) Disjunction of two fuzzy sets.

The disjunction (union) is generalized by t-conorms, such as the maximum operators. Given $C = E \cup F$, of the corresponding membership degree will be,

$$\mu_C(x) = \max\left(\mu_E(x_1), \mu_F(x_2)\right) \tag{2-4}$$

The above maximum operation can also be represented by $\mu_C(x) = \mu_E(x_1) \vee \mu_F(x_2)$, as illustrated in Figure 2.5(b).

The complement operator is represented by $\mu_{\bar{E}}(x_1) = 1 - \mu_E(x_1)$.

After fuzzy operations, the final output is the weighted average of all rule outputs

$$z = \frac{\sum_{i=1}^{N} \mu_{Ci} f_i}{\sum_{i=1}^{N} \mu_{Ci}} \tag{2-5}$$

where $i = 1, 2 \ldots N$, is the total number of rules.

Compared with a Mamdani system, the TS model can use adaptive techniques for fuzzy model constructions, which can customize the MFs to achieve optimal input/output data mapping. Furthermore, because of the linear dependence of each rule on the input variables, the TS model is ideal to interpolate MIMO controllers to different operating conditions.

2.3 Introduction to neural network models

A NN is a parallel distributed processing model. It resembles the human brain reasoning in two respects:

1. Knowledge is acquired by the network through a learning process.
2. Interneuron connection strengths (e.g., synaptic weights) are used to store the knowledge.

A *neuron* is an information-processing unit that is fundamental to the operation of a NN. Figure 2.6 shows the model for a neuron, which has three basic elements:

1. A set of *synapses* or *connecting links*, each of which is characterized by a *weight* or *strength*. Specifically, a signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}.
2. An *adder* for summing the input signals, weighted by the respective synapses of the neuron. These operations constitute a *linear combiner*.

3. An *activation function* for the output of a neuron. The activation function squashes (limits) the permissible amplitude range of the output signal. Typically, the normalized amplitude range of the output of a neuron is used, such as the intervals [0, 1] and [-1, 1].

Furthermore, an externally applied *threshold* θ_k in Figure 2.6 has the effect of lowering the net input of the activation function. On the other hand, the net input of the activation function may be increased by employing a *bias* term rather than a threshold.

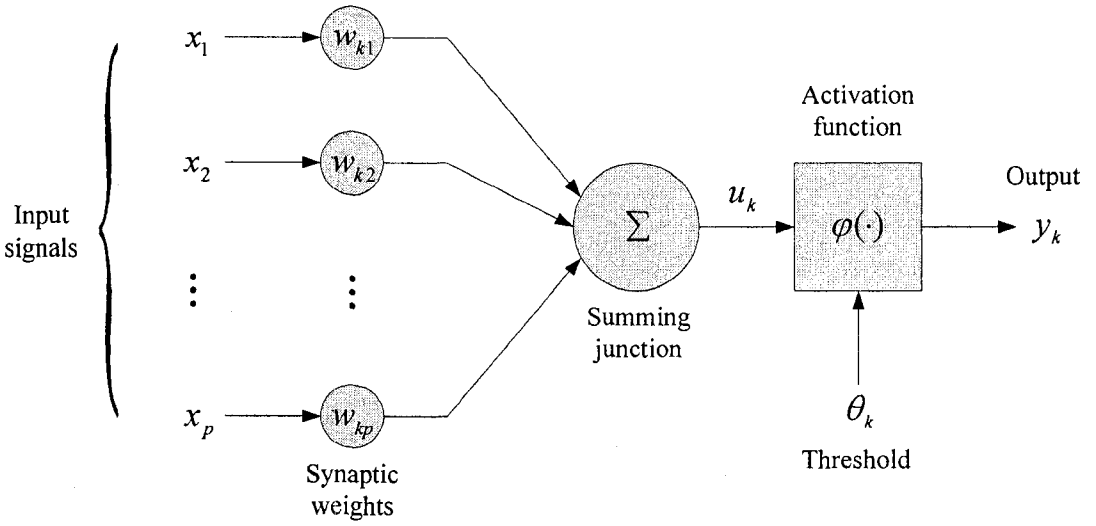


Figure 2.6. Nonlinear model of a neuron.

Mathematically, a neuron k can be described by the following equations:

$$\mu_k = \sum_{j=1}^{p} w_{kj} x_j \tag{2-6}$$

and $$y_k = \varphi(\mu_k - \theta_k) \tag{2-7}$$

where $x_1, x_2, \ldots, x_p$ are the input signals; $w_{k1}, w_{k2}, \ldots, w_{kp}$ are the synaptic weights of neuron k; μ_k is the *linear combiner output*; θ_k is the *threshold*; $\varphi(\cdot)$ is the *activation function*; and y_k is the output of the neuron.

Different squashing functions can be used in a neuron. For example, the commonly used Sigmoidal function is in the form of

$$y_k = \varphi(\mu_k - \theta_k) = \frac{1}{1+e^{-(\mu_k-\theta_k)}} \tag{2-8}$$

Consider the network as shown in Figure 2.7, which contains three inputs, two outputs, and three hidden neurons. Each neuron in this network has the same activation function. For instance, the activation function of neuron 7 is

$$x_7 = \frac{1}{1+e^{-(w_{4,7}x_4+w_{5,7}x_5+w_{6,7}x_6-t_7)}} \tag{2-9}$$

where x_4, x_5, and x_6 are outputs from neurons 4, 5, and 6, respectively, and the parameter set of neuron 7 is denoted by $\{w_{4,7}, w_{5,7}, w_{6,7}, t_7\}$. $w_{i,j}$ is the weight associated with the link connecting neuron i and j, and t_j is the threshold associated with neuron j. (note that this weight-link association is only valid in this type of network.)

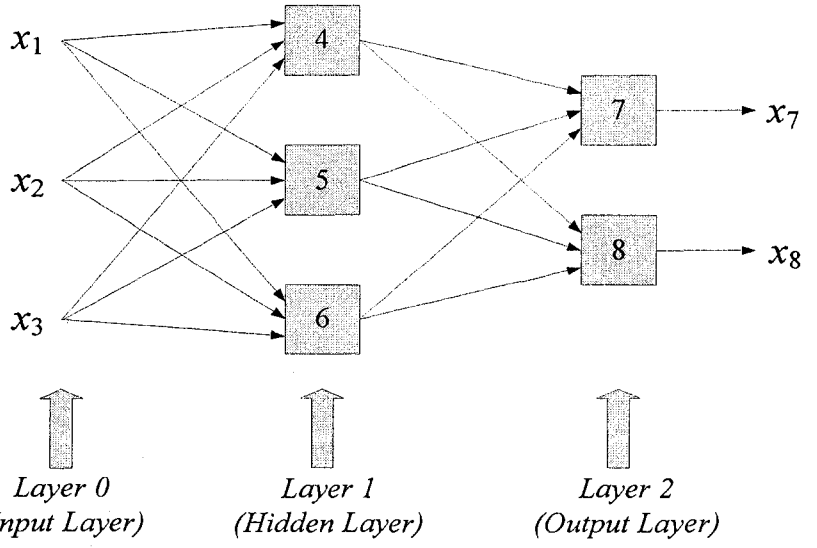


Figure 2.7. One 3-3-2 NN.

NNs should be adjusted (or trained) so as to achieve an optimal input-output mapping, as illustrated in Figure 2.8.The type of learning is determined by the manner in which the parameter changes take place. One of the main properties of a NN is its ability of *learning* from its environment so as to improve its performance. After the network is properly trained, it will become more knowledgeable about its environment.

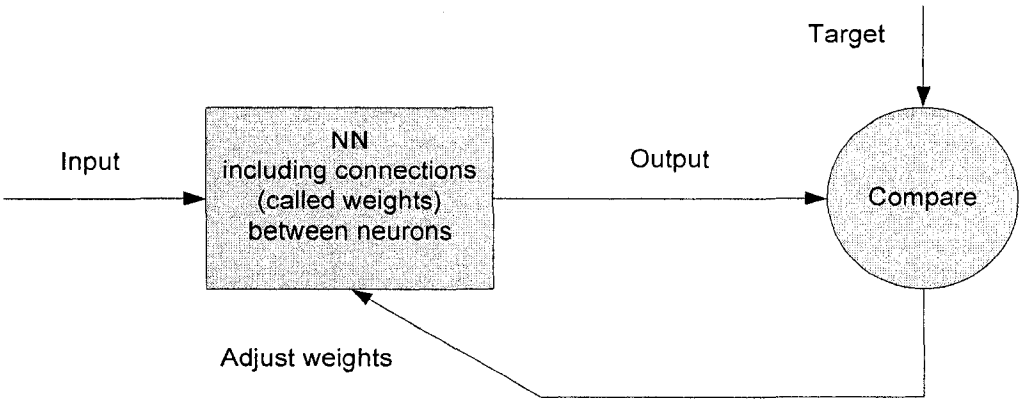


Figure 2.8. Representation of NN training.

Several learning algorithms have been suggested in the literature, and are summarized in Figure 2.9. Supervised learning is performed under the supervision of an external "*target*". Unsupervised learning is performed in a self-organized manner in that no external target or critic is required to instruct synaptic changes in the network. The target stands for the knowledge of the environment represented by a set of input-output data pairs. The network parameters are adjusted under the combined influence of the training data and the error signal between the actual response of the network and the desired response. This training adjustment is carried out iteratively with the aim of eventually making the NN *emulate* the target; the emulation is presumed to be optimum in some statistical sense.

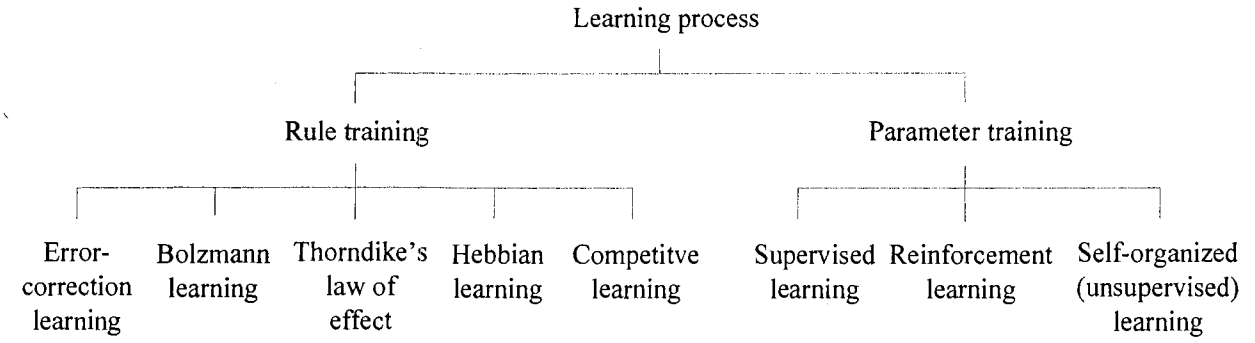


Figure 2.9. Taxonomy of the learning process.

In this thesis, supervised learning is applied in experiments. Gradient (or steepest gradient) method is perhaps the most commonly used training algorithm [85]. Suppose that a given feed forward NN with L layers and each layer contains $N(l)$ neurons ($l = 0, 1 \ldots L\text{-}1$). Assuming that the given training data set has P entries and $O_{l,i}$ is the output of neuron i ($i = 1, 2 \ldots N(l)$) in layer l , then the objective error function for the pth ($1 \leq p \leq P$) entry of the training data can be defined as:

$$E_p = \sum_{k=1}^{N(L)} (d_k - O_{L,k})^2 \tag{2-10}$$

where d_k is the kth component of the pth desired output vector. Therefore the objective is to minimize an overall error measure $E_p = \sum_{p=1}^{P} E_p$.

To use the steepest gradient algorithm to minimize the error measure, the first step is to obtain the gradient vector, which can be illustrated in Figure 2.10 where the arrows → indicate causal relationships. That is, the basic concept in calculating the gradient vector is to pass the derivative information starting from the output layer and going backward, layer by layer, until the input layer is reached.

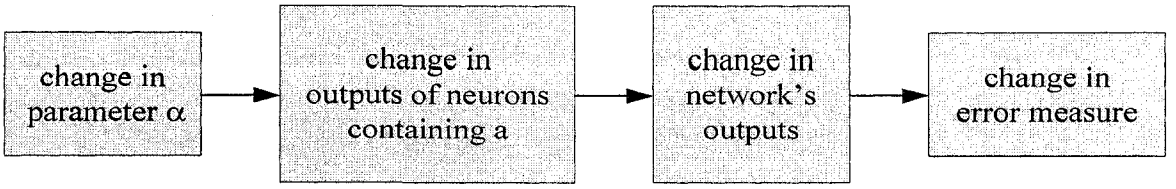


Figure 2.10. The causal relationships to obtain the gradient vector.

To facilitate the discussion, we define the *error signal* $\varepsilon_{l,i}$ as the derivatives of the error measure E_p with respect to the output of node i in layer l, taking both direct and indirect paths into consideration. In symbols,

$$\varepsilon_{l,i} = \frac{\partial^+ E_p}{\partial O_{l,i}} \tag{2-11}$$

This expression has called the ordered derivative by Werbos [86]. The difference between the ordered derivative and the ordinary partial derivative lies in the way we view the function to be differentiated. For an internal node output $O_{l,i}$ (where $l \neq L$), the partial derivative $\frac{\partial E_p}{\partial O_{l,i}}$ is equal to zero, since E_p does not depend on $O_{l,i}$ directly. However, it is obvious that E_p does depend on $O_{l,i}$ indirectly, since a change in $O_{l,i}$ will propagate through indirect paths to the output layer and thus produce a corresponding change in the value of E_p. Therefore, $\varepsilon_{l,i}$ can be viewed as the ratio of these two changes when they are made infinitesimal.

We can improve the performance of the steepest gradient algorithm if we allow the *learning rate* to change during the training process. An adaptive learning rate attempts to keep the *learning step size* as large as possible while keeping learning stable. The step size is the length of each transition along the gradient direction in the parameter space. Usually, we can change the step size to vary the speed of convergence.

2.4 Introduction of NF model structure

By the aforementioned brief introduction to NN and FL schemes, FL and neural systems have contrasting control application requirements. On the one hand, fuzzy control systems are appropriate if sufficient expert knowledge about the process is available, while on the other hand, neural systems are useful if sufficient process data are available or are measurable. Despite this, there exists a lot of similarity and a synergetic relationship between NNs and FL systems [87, 88]. Both the approaches build nonlinear systems based on bounded continuous variables; the difference lies that neural systems are treated in a numeric (quantitative) manner whereas fuzzy systems are treated in a symbolic (qualitative) manner.

Generally, both NNs and FL are powerful control design techniques, but each having its own strengths and weaknesses. NNs can learn from data sets, while FL solutions are easy in knowledge implementation and verification. From the summary in Table 2.2, it becomes obvious that an efficient combination of the two technologies would deliver better solutions.

Table 2.2: Strengths and Weaknesses of NN and FL

	NN	FL
Knowledge Representation	Implicit, the system cannot be easily interpreted or modified	Explicit, verification and optimization is easy and eiffient
Train ability	Trains itself by learning data sets	None, we have to define everything explicitly

As stated in Chapter 1, NF synergetic schemes can be formed by synthesizing the NN and FL systems, which integrate the properties of the explicit knowledge representation of FL with the learning power of NNs. Figure 2.11 illustrates a mathematical framework that maps FL model to neurons in a NN.

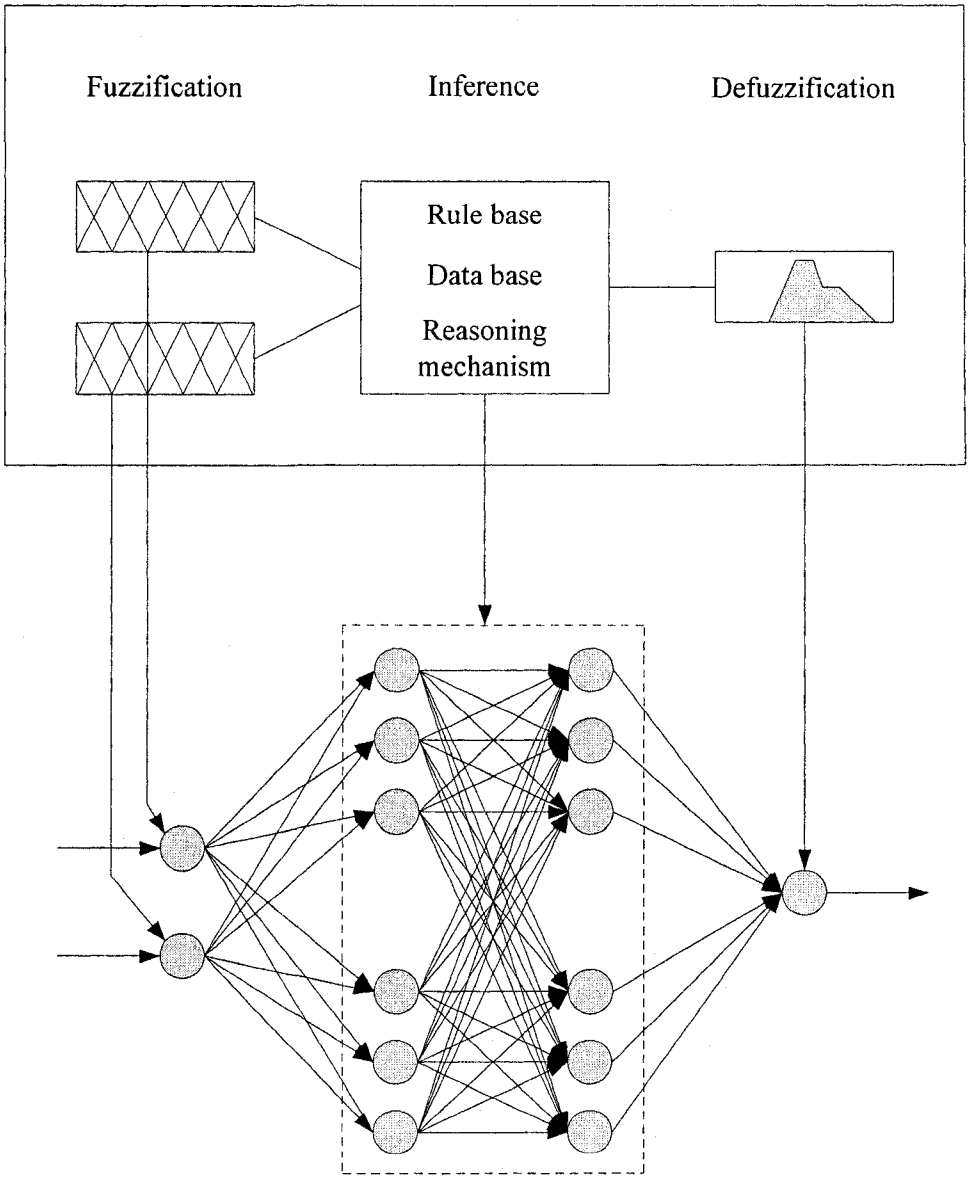


Figure 2.11. A mathematical framework mapping FL model to neurons in an NN.

A NF system should follow the following four primary steps:

Step 1: obtaining training data

The first step is to obtain the data sets that represent the desired system behaviour. Each data set gives sample output values for a combination of input variables.

Step 2: creating a primary FL system

The NF training process starts with an initial FL system, for example, the rules and the MFs.

Step 3: learning

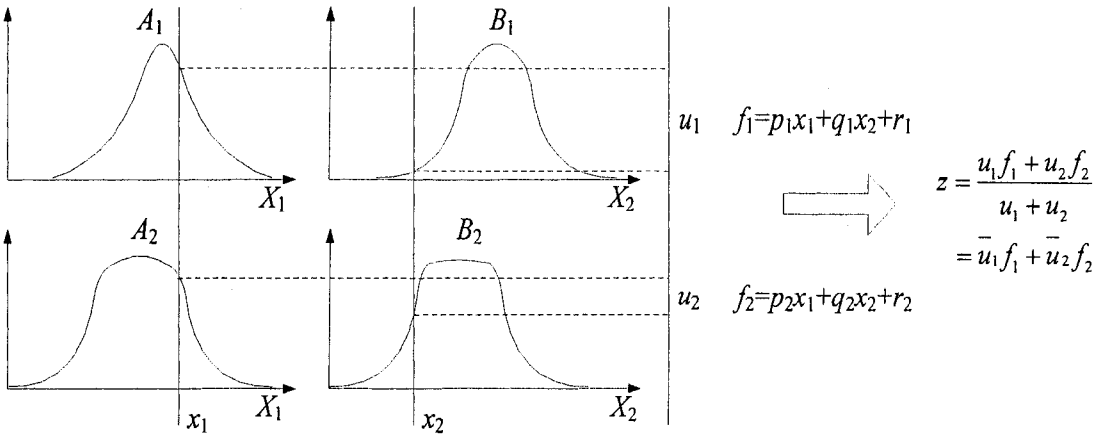
In this step, we must select the parts of the NF system to be trained, such as parameters and/or fuzzy structure, and if the training is in online or offline pattern.

Step 4: verification

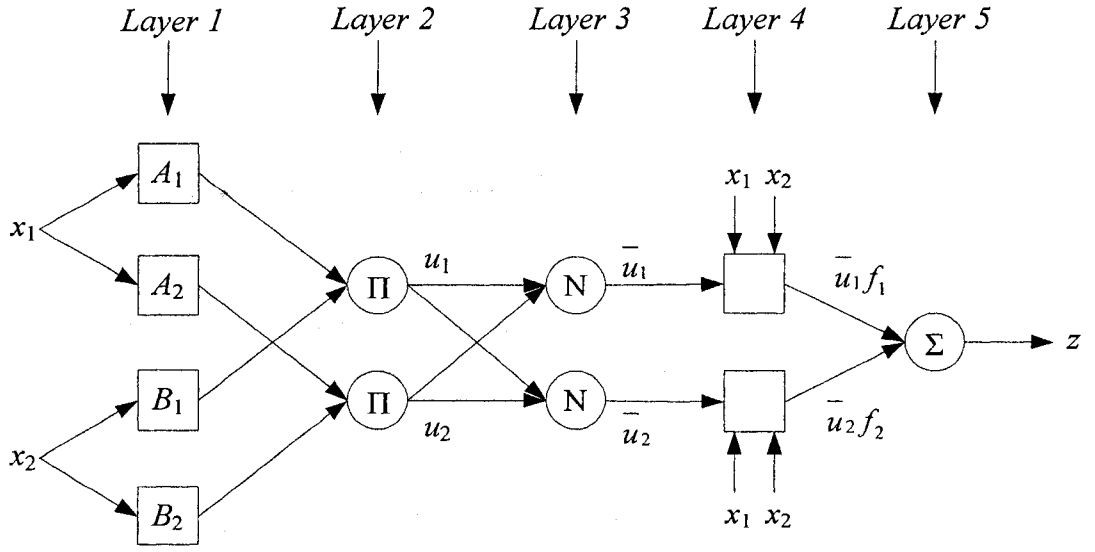
The result of neural fuzzy training is a normal FL system. After system verification, we may implement the solution to the experiments.

Consider a widely used NF example, Adaptive network-based fuzzy inference system (ANFIS) suggested by Jang [89]. ANFIS is a feed forward network as illustrated in Figure 2.12. Although we can apply a gradient algorithm to identify the parameters in this adaptive network, this training process usually takes a long time which is not suitable for control applications. We may observe, however, that an adaptive network's output (assuming there is only one) is linear in some of the network's parameters. As a result, a hybrid method can be used in this case: these consequent linear parameters can be identified by the linear least-squares (LSE) method, whereas nonlinear MF parameters are optimized by gradient algorithm [89, 90] The training

operations takes two procedures: 1) in the forward pass, the consequent parameters are trained by LSE, while the antecedent parameters are fixed; 2) in the backward pass the antecedent parameters are trained by the gradient algorithm, while the consequent parameters remain fixed. This procedure is then iterated until the training goal in terms of cycles or error tolerance is achieved.



(a)



(b)

Figure 2.12. (a) A two-input first-order Takagi-Sugeno fuzzy model with two rules; (b) equivalent ANFIS architecture.

Consider a simple reasoning mechanism in Figure 2.12(a) with a TS1 model having two inputs x_1 and x_2 and one output z [83, 84, 91]. The corresponding equivalent ANFIS architecture is as shown in Figure 2.12(b), which contains five layers: input, fuzzification, normalization, rules, and defuzzification. Consider two fizzy if-then rules with the following format:

$$\text{Rule 1: IF } (x_1 \text{ is } A_1) \text{ AND } (x_2 \text{ is } B_1), \text{ THEN } (f_1 = p_1x_1+q_1x_2+r_1) \tag{2-12}$$

$$\text{Rule 2: IF } (x_1 \text{ is } A_2) \text{ AND } (x_2 \text{ is } B_2), \text{ THEN } (f_2 = p_2x_1+q_2x_2+r_2) \tag{2-13}$$

where A_i and B_i are linguistic fuzzy sets (i=1, 2). μ_i in figure 2.12 represents the rule firing strength. $\overline{\mu}_i$ is a normalized firing strength from layer 3; and $\{p_i, q_i, r_i\}$ are consequent parameters. z is the overall output.

Chapter 3 Designing of Neural Fuzzy Controllers

In order to properly setup the flexible structure system and obtain necessary system information for the development and implementation of the advanced NF controllers, system properties are studied first based on a PD controller.

3.1 System property investigation based on a PD controller

To properly investigate the important system properties for advanced control applications, a PD controller is designed and implemented first to the flexible structure for system control testing. Figure 3.1 illustrates the SIMULINK model for the tested system based on the PD controller.

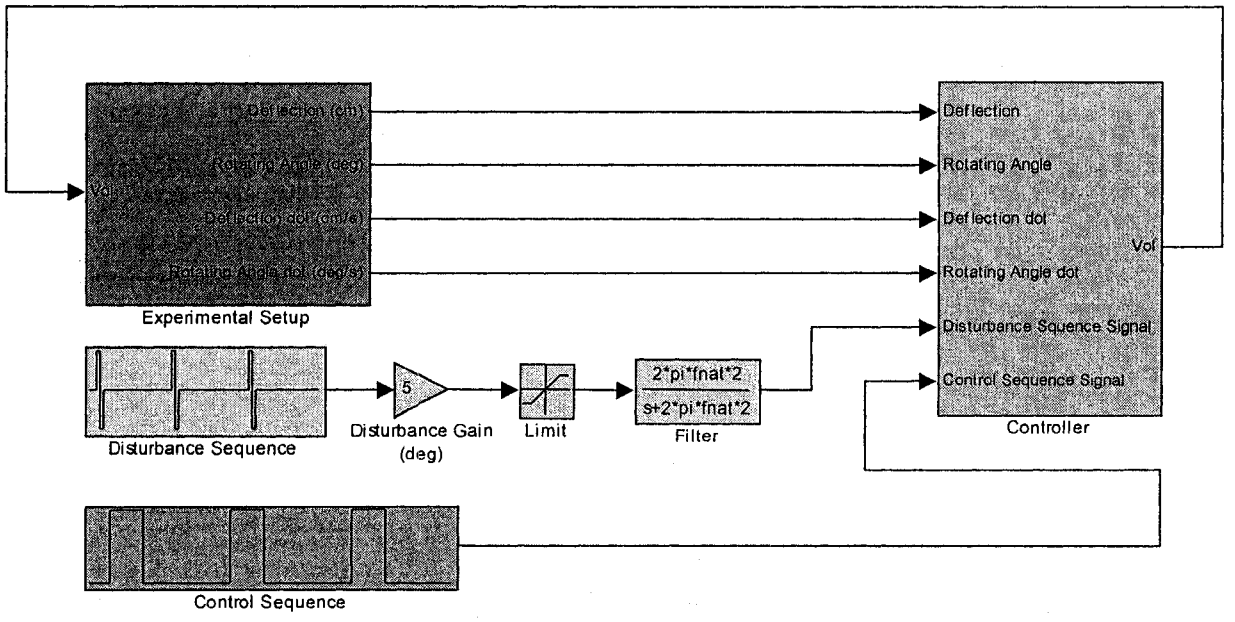


Figure 3.1. The SIMULINK model with the PD controller.

This flexible structure model has been given in Figure 2.2 in Chapter 2. Its input variable is the control output to drive the motor of this structure, and its four output variables are $(\theta, \dot{\theta}, \varepsilon, \dot{\varepsilon})$, where θ and $\dot{\theta}$ represent the rotating angle of the rigid beam and its derivative; ε and $\dot{\varepsilon}$ are the deflection of the flexible beam, and its derivative, respectively. These four elements will be used as the input variables to the two PD controllers.

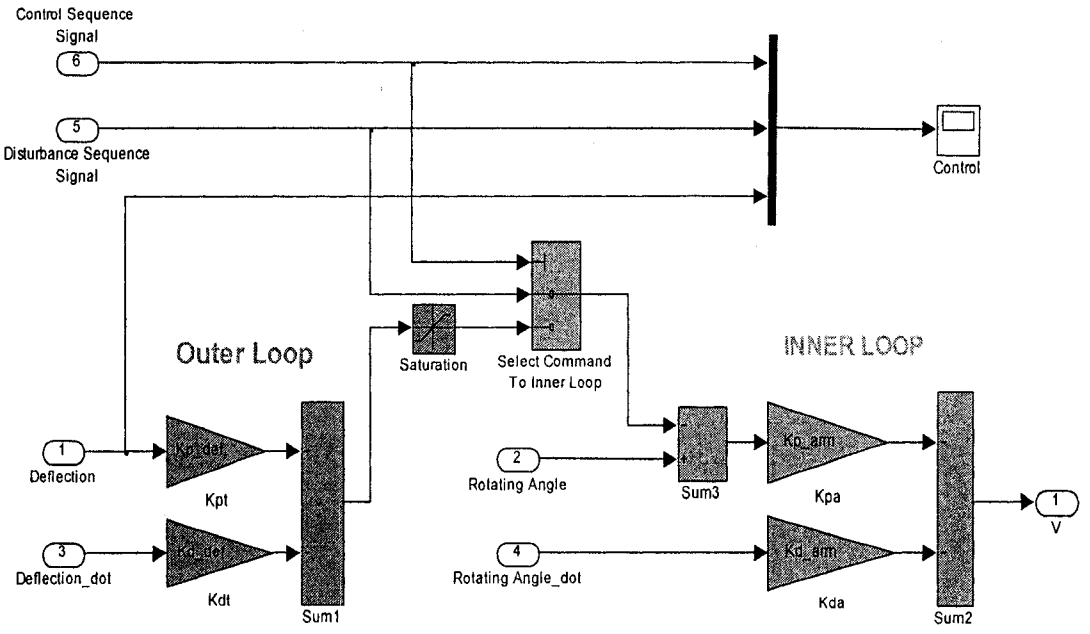


Figure 3.2. The SIMULINK model of the PD controller.

In this case, the control system consists of two loops, an outer PD (proportional gain k_{pt}, derivative gain k_{dt}) and an inner PD loop (proportional gain k_{pa}, derivative gain k_{da}). From Figure 3.2, control inputs 1 to 6 are, respectively, the Deflection variable, the Rotating Angle variable, the Deflection dot variable, the Rotating Angle dot variable, the Disturbance Sequence Signal variable, and the Control Sequence Signal variable. Variables 1 to 4 are from the flexible structure outputs as shown in Figure 3.1.

The disturbance added to the system is defined as an external angle value in degrees, which is illustrated in Figure 3.3. The force is automatically added to the apparatus from t = 1 *s*, and is repeated every 10 *s*. The amplitude of the rotation is 5 *deg*, and the duration for each force is over 0.88 *s*. A control sequence signal is applied with a command value of 1 (i.e., control off), which makes the beam track the disturbance sequence signal command. At $t = 3.5$ *s*, the controller is switched to the command value of 2 (i.e., control on); the *outer loop* commands the *inner loop*, as shown in Figure 3.3, and the full controller is operational. The outer loop remains active until $t = 9$ *s* at which the controller is switched off. The disturbance sequence is repeated every 10 *s* while the control sequence is repeated every 20 *s*, as illustrated in Figure 3.3. The control results of the PD controller are presented by Figure 3.4. In figure 3.4, the unit of time is *s*, the Deflection's unit is *cm*, and the Rotating Angle's unit is *deg*.

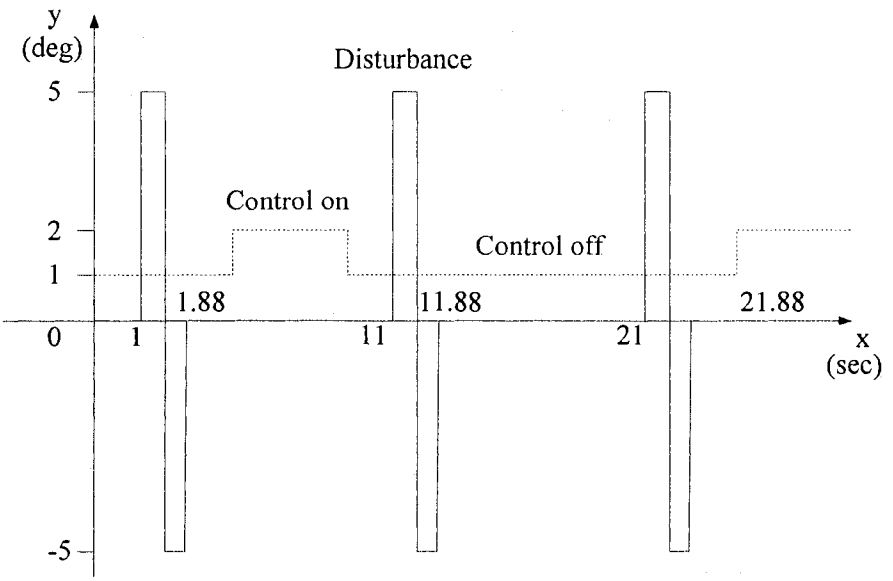
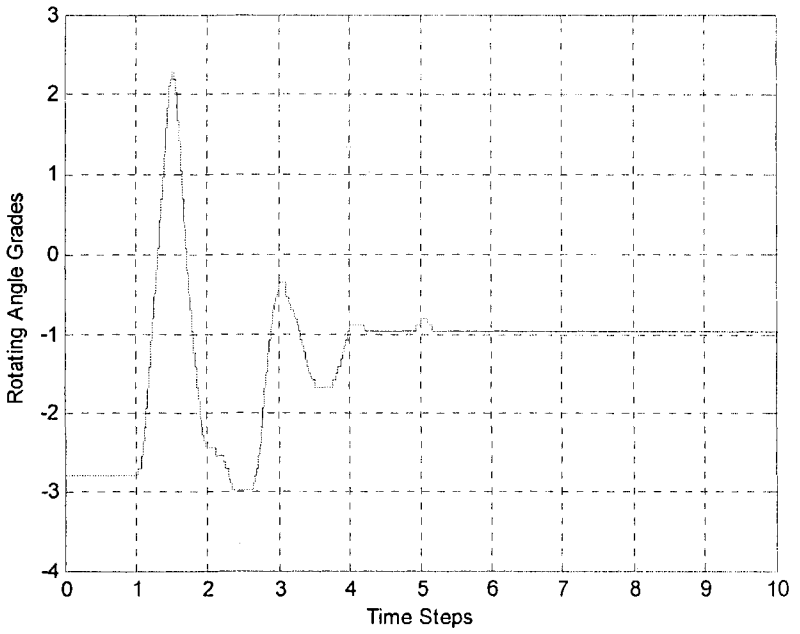
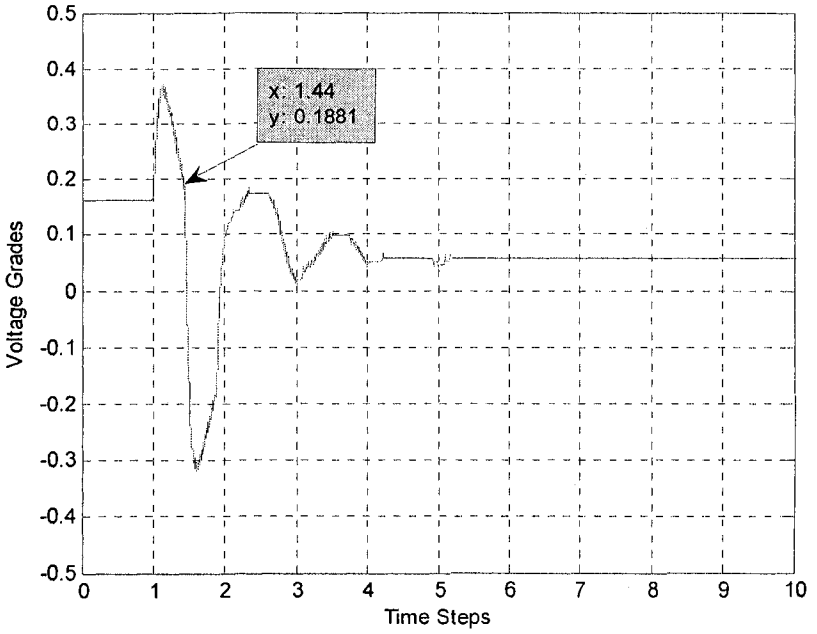


Figure 3.3. The disturbance and control sequence signals.



(a)



(b)

Figure 3.4. (a) The result of Rotating Angle without control signal; (b) The control output (voltage) without control signal.

Figures 3.4(a) and 3.4(b) represent the states of the rotating angle and the control output voltage without control sequence signal, respectively. From the t = 1 to 1.44 s, there exists a peak in Figure 3.4(b). It is because the disturbance signal is added to this system from t = 1 s in the positive direction, with the duration of 0.44 s as shown in Figure 3.2. Figures 3.4(a) and 3.4(b) are similar in shape except the part between t = 1 s and 1.44 s. The reason of choosing these two sets of results without control sequence signal is that the control output voltage value is reflected by the rotating angle directly.

The operation of this PD controller is performed by switching the command to the inner loop, and then making the outer loop to command the inner loop. If the rotating angle and the control output voltage are chosen as the references when the control sequence signal is on, the results are not accurate to represent their relationship, which will be proved as follows.

Using Lagrangian dynamics, if the effects of gravity are neglected, the nonlinear dynamic equations of the system can be obtained as:

$$(M + m_p)\ddot{\varepsilon} - m_p \cdot c \cdot \sin(\theta(t)) \cdot \dot{\theta}^2(t) + m_p \cdot c \cdot \cos(\theta(t)) \cdot \ddot{\theta} + K \cdot \varepsilon(t) = 0 \tag{3-1}$$

$$m_p \cdot c \cdot \cos(\theta(t)) \cdot \ddot{\varepsilon} + (m_p \cdot c^2 + I) \cdot \ddot{\theta} = T \tag{3-2}$$

It should be stated that this model can also be applied to the horizontal or other configurations of the flexible beam.

Consider the motor equation:

$$T = \frac{K_m \cdot K_g}{R_m}(V - K_m \cdot K_g \cdot \dot{\theta}(t)) \tag{3-3}$$

Linearizing (3-1) to (3-3) yields the state space equations:

$$\begin{bmatrix} \dot{\theta} \\ \dot{\varepsilon} \\ \ddot{\theta} \\ \ddot{\varepsilon} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\dfrac{(m_p \cdot c^2 + I) \cdot K}{(m_p \cdot c^2 \cdot M + I \cdot m_p + I \cdot M)} & 0 & 0 & \dfrac{m_p \cdot c \cdot K_m^2 \cdot K_g^2}{(m_p \cdot c^2 \cdot M + I \cdot m_p + I \cdot M) \cdot R_m} \\ \dfrac{m_p \cdot c \cdot K}{(m_p \cdot c^2 \cdot M + I \cdot m_p + I \cdot M)} & 0 & 0 & \dfrac{-(m_p + M) \cdot K_m^2 \cdot K_g^2}{(m_p \cdot c^2 \cdot M + I \cdot m_p + I \cdot M) \cdot R_m} \end{bmatrix}$$

$$\cdot \begin{bmatrix} \theta \\ \varepsilon \\ \dot{\theta} \\ \dot{\varepsilon} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dfrac{-m_p \cdot c \cdot K_m \cdot K_g}{(m_p \cdot c^2 \cdot M + I \cdot m_p + I \cdot M) \cdot R_m} \\ \dfrac{(m_p + M) \cdot K_m \cdot K_g}{(m_p \cdot c^2 \cdot M + I \cdot m_p + I \cdot M) \cdot R_m} \end{bmatrix} \cdot V \qquad (3\text{-}4)$$

Table 3.1: Parametric values for the motor

Physical parameters	Symbol	Value/Units
Motor Torque constant	K_m	0.0767 Nm/Amp
Total gear ratio	K_g	70
Motor Armature resistance	R_m	2.6 Ohm
Motor voltage	V	V
Motor torque	T	Nm

Table 3.1 summarizes the related physical parameters of the tested system. Considering Table 2.1 and Table 3.1, the A and B matrices will be:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -48.0 & 0 & 0 & 31.8 \\ 86.0 & 0 & 0 & -1449.7 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ -5.9 \\ 270 \end{bmatrix}.$$

The Q and R matrices are chosen as: $Q = diag([250000 \quad 1000 \quad 50 \quad 0])$ and $R = 10$, respectively; the resulting gains will be:

$$k = [-192.6 \quad 25.8 \quad 18.9 \quad 0.4405]$$

The feedback controller is in the form of:

$$V = -(k_1 \cdot \varepsilon + k_2 \cdot \theta + k_3 \cdot \dot{\varepsilon} + k_4 \cdot \dot{\theta}) \tag{3-5}$$

If this feedback is implemented, it cannot determine how far the rigid beam would rotate. If the gains or errors are too large, the rotation angle of the the beam will be so large that the cross beam could collide with the flexible beam, which is undesirable.

We can re-write the feedback equation as:

$$V = -k_2(\theta - (-\frac{k_1}{k_2} \cdot \varepsilon - \frac{k_3}{k_2} \cdot \dot{\varepsilon})) - k_4 \cdot \dot{\theta} \tag{3-6}$$

$$V = -k_2(\theta - \theta_d) - k_4 \cdot \dot{\theta} \tag{3-7}$$

$$V = -k_{pa}(\theta - \theta_d) - k_{da} \cdot \dot{\theta} \tag{3-8}$$

Equation (3-8) is a feedback loop that makes the motor track a desired position θ_d which is derived by:

$$\theta_d = -\frac{k_1}{k_2} \cdot \varepsilon - \frac{k_3}{k_2} \cdot \dot{\varepsilon} \tag{3-9}$$

$$\theta_d = -k_{pt}\varepsilon - k_{dt} \cdot \dot{\varepsilon} \tag{3-10}$$

Then resulting gains will be:

$$k_{pa} = 25.8\,V/rad = 0.45\,V/deg$$

$$k_{da} = 0.44\,V/(rad/\sec) = 0.0077\,V/(deg/sec)$$

$$k_{pt} = -116.8/26.4 = -7.46\,rad/m = -4.2742\,deg/cm$$

$$k_{dt} = 12/26.4 = 0.735\,rad/(m/sec) = 0.4211\,deg/(cm/sec)$$

It should be noted that the gain = 3 for the power amplifier when using the available cables in the tests. Correspondingly, the actual implemented values of k_{pa} and k_{da} are divided by 3.

Because k_{da} is very small ($k_{da} = 0.0077\,V/(\text{deg}/\sec)$ in this case), equation (3-8) can be approximated by:

$$V \approx -k_{pa}(\theta - \theta_d) \tag{3-11}$$

which means when the control signal is applied by the value of 1 (control off), and $\theta_d = 0$, then

$$V = -k_{pa}\theta \tag{3-12}$$

In fact, the rotating angle θ is inversely proportional to the voltage V, with an inverse ratio k_{pa}. On the contrary, when the control sequence signal is 2 (i.e., control on), $\theta_d \neq 0$, then V and θ do not follow a linear relationship. That is the reason why we choose the rotating angle and the control output voltage without the control sequence signal.

The control output is voltage. If the disturbance sequence signal is selected as a voltage added to the system, we have to find the relationship between the rotating angle and the voltage.

Based on equations (3-5) - (3-10) and the values of $\{k_{pa}\ k_{da}\ k_{pt}\ k_{dt}\}$, the ratio of V and θ should be k_2 or k_{pa}, which equals 25.8.

Next is to define the disturbance and control sequence signals. As illustrated in Figure 3.5, the period of the disturbance signal is still repeated every 10 *s*, and the amplitude is 5 *deg*. However, there is only one positive value for each sequence signal, and the duration for each disturbance becomes 0.44 *s*. The controller is switched on (i.e., control = 2) at $t = 3.5$ *s*, which remains active until $t = 9$ *s* when the controller is switched off (i.e., control = 1) again. The disturbance sequence repeats over every 10 *s* while the control sequence repeats every 10 *s*.

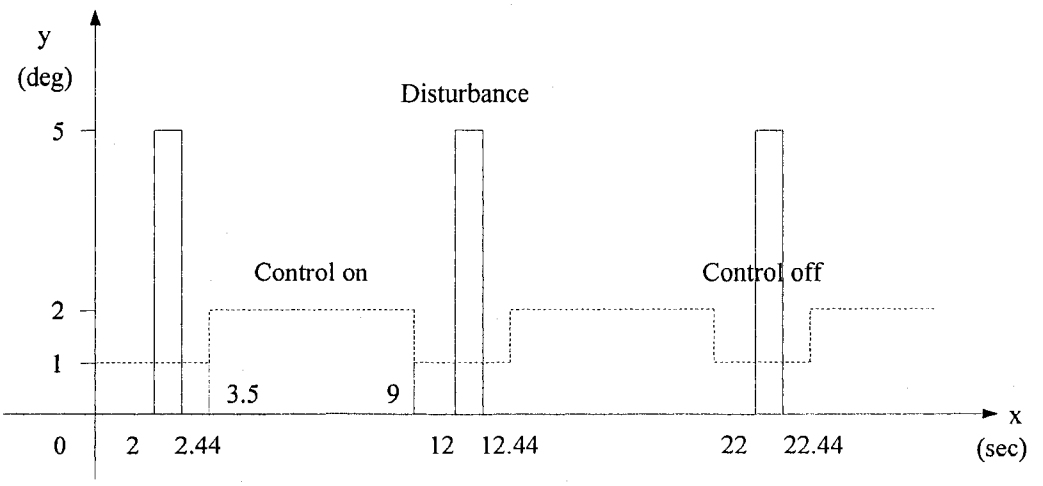


Figure 3.5. The re-defined disturbance and control sequence signals.

First we change the value of Disturbance gain in Figure 3.1 to zero, and give the rigid beam several arbitrary distrubances; then we can examine the values of the rotating angle and the voltage from the scopes, and record the peak values of the angle and the voltage at the time when external forces are being applied to the system. The average ratio of the rotating angle and voltage will be:

$$R_{average} = -\frac{\left(\frac{\theta_1}{V_1} + \frac{\theta_2}{V_2} + \cdots + \frac{\theta_m}{V_m}\right)}{m} \approx 20 \tag{3-13}$$

where m is the number in which the arbitrary force has been applied.

3.2 The TS0-NF controller

3.2.1 Control input and output variables

The purpose of the developed controller is to dampen out the vibrations in the flexible beam by driving the actuating system. Correspondingly, the deflection ε at the top of the flexible beam, in Figure 2.2, is considered as one of the control input variables. A servo motor is equipped at the top of this flexible beam, and drives an eccentric load. This motor plays a role in damping, thus the rotating angle θ of the rigid beam is considered as the second control input variable. The feedback voltage value that comes out of the controller and gets into the structure can be still assumed as the control output variable. Moreover, in order to minimize the computation burden of the classical NF controllers, only two control input variables will be utilized in this case.

The deflection value (ε) in *cm* can be estimated as the multiplication of the output voltage value and a *gain* which is selected as 2.54, with the unit of *cm/volt*. Since the nature frequency of this flexible structure is about 1.13 *HZ*, the deflection can be observed after the multiplied value is low-pass filtered (with a cut off frequency of 2.26 Hz) to remove the higher frequency components.

The disturbance signal is driven by the motor drive through a gear ratio of 70:1. The encoder measures the angle with a 1024 count disc which in quadrature results in 4096 *counts/rev*. Thus, the value of the rotating angle can be determined by the encoder when the voltage is transformed to the count value and is multiplied by a gain of 360/4096 *deg/count*.

3.2.2 The membership functions (MFs)

As discussed in Chapter 2, a fuzzy set is characterized by its MF. Since most fuzzy sets in this work have a universe of discourse X, a convenient and concise method to define an MF is to express it as a mathematical formula. Many parameterized functions can be used as MFs in one and two dimensions, whereas MFs of higher dimensions can also be defined accordingly.

In this work, only one-dimensional MFs (i.e., MFs with a single input) are applied, which are summarized as follows:

1) Triangular MFs

A triangular MF is specified by three parameters $\{a, b, c\}$ such as:

$$\text{triangle}(x; a, b, c) = \begin{cases} 0, & x \le a. \\ \dfrac{x-a}{b-a}, & a \le x \le b. \\ \dfrac{c-x}{c-b}, & b \le x \le c. \\ 0, & c \le x. \end{cases} \tag{3-14}$$

By using minimum (min) and maximum (max) functions, some alternative functions could be formulated:

$$\text{triangle}(x; a, b, c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \tag{3-15}$$

The parameters $\{a, b, c\}$, where $a < b < c$, determine the shape of the triangle function, as illustrated in Figure 3.6 as an example.

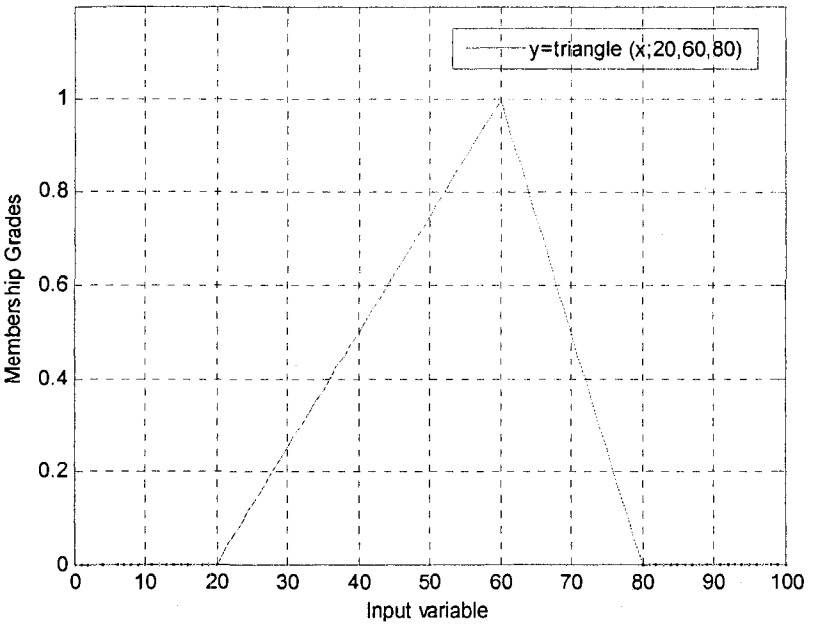


Figure 3.6. Triangular MF defined by triangle (x; 20, 60, 80).

Due to its simple formulation and computational efficiency, a triangular MF is especially useful in real-time implementations. However, since the MFs are composed of straight line segments, they are not continuous at the corners. Some continuous MFs are also applied in this work, which are described as follows.

2) Gaussian MFs

A Gaussian MF is specified by two parameters $\{a, c\}$:

$$\text{Gaussian}(x; a, c) = e^{-\frac{1}{2}(\frac{x-a}{c})^2} \tag{3-16}$$

where a and c represent the center and the width of the function, as illustrated in an example in Figure 3.7

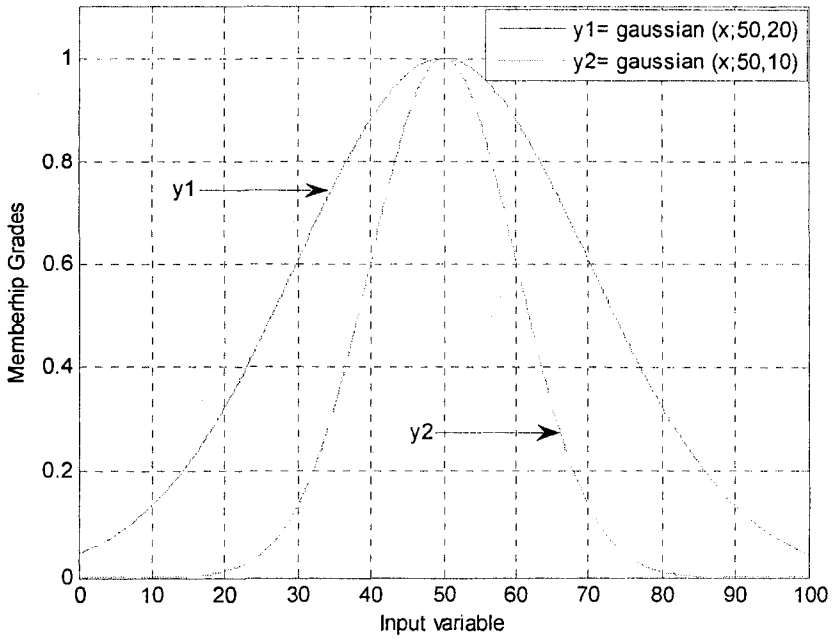


Figure 3.7. Gaussian MF defined by Gaussian y1= (x; 50, 20) and y2= (x; 50, 10).

3) Sigmoidal MFs

A Sigmoidal MF is defined by

$$\text{Sig}(x; a, c) = \frac{1}{1+e^{(-a(x-c))}} \tag{3-17}$$

where a determines the slope at the crossover point $x = c$. Depending on the sign of the parameter a, a Sigmoidal MF is inherently open right or left and thus is appropriate for representing sets such as "very large" or "very negative". Figure 3.8 shows two examples defined by sig (x; 1, -5) and sig (x; -2, 5), respectively.

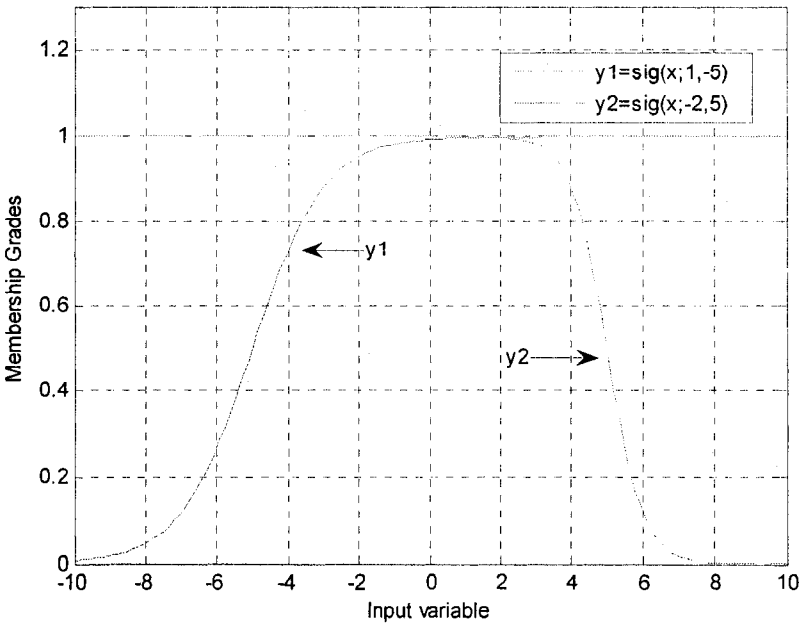


Figure 3.8. Two Sigmoidal functions defined by $y_1 = \text{sig}(x; 1, -5)$ and $y_2 = \text{sig}(x; 2, 5)$.

All these aforementioned three MFs are utilized in the NF controllers. In this TS0-NF controller, three MFs are assigned to each input variable: one Sigmoidal MF open left, one Gaussian MF, and one Sigmoidal MF open right, respectively. The initial states of the MFs to each input variable are shown in Figure 3.9, which are subjected to updating during training processes as discussed in the following sections.

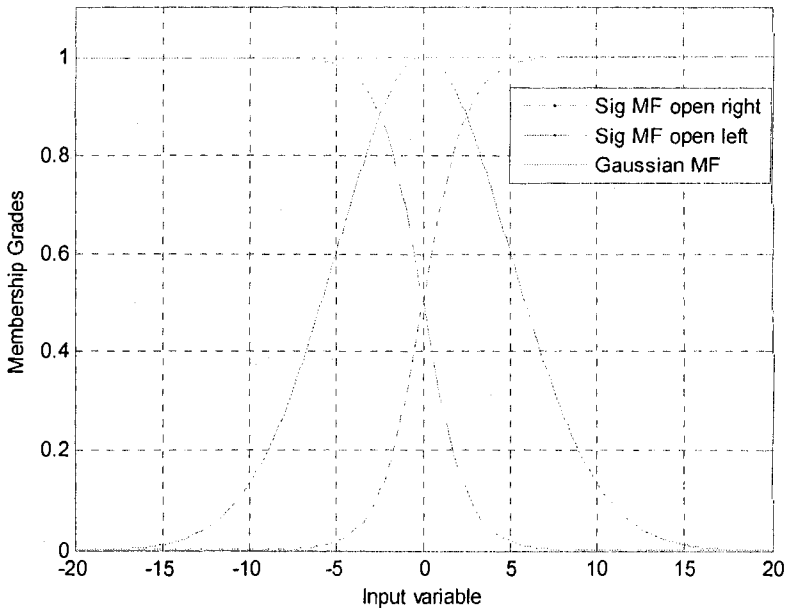


Figure 3.9. The initial MFs for each input variable.

3.2.3 The reasoning rules in the NF controller

As stated earlier, the fuzzy reasoning in the NF controllers are performed by fuzzy logic. If n input variables are considered, x_i, $i = 1, 2...n$, a typical fuzzy rule base is represented in the following form

$$R^j\text{: IF } (x_1 \text{ is } A_1^{\upsilon}) \text{ AND } (x_2 \text{ is } A_2^{\upsilon}) \text{ AND } \ldots \text{ AND } (x_i \text{ is } A_i^{\upsilon}), \text{ THEN } (f_j \text{ is } w_j) \tag{3-18}$$

where A_i^{υ} is a linguistic term; w_j is a singleton, $j = 1, 2...M$, and M is the total number of rules; $\upsilon = 1, 2...L$, L is the number of MFs.

For this TS0-NF controller model, a common rule set with nine fuzzy if-then rules is based on equation (3-18), which is shown in Table 3.2. These rules are presented by the links between the second layer and the third layer.

R^1: IF (x_1 is A_1^1) AND (x_2 is A_2^1), THEN (f_1 is w_1), (3-19)

R^2: IF (x_1 is A_1^1) AND (x_2 is A_2^2), THEN (f_2 is w_2), (3-20)

$\vdots$

R^9: IF (x_1 is A_1^3) AND (x_2 is A_2^3), THEN (f_9 is w_9), (3-21)

Table 3.2: Fuzzy logic rule base

		Deflection ε (x_2)		
		Negative A_2^1	Zero A_2^2	Positive A_2^3
Position θ (x_1)	Negative A_1^1	w_1	w_2	w_3
	Zero A_1^2	w_4	w_5	w_6
	Positive A_1^3	w_7	w_8	w_9

These nine rules are derived from the motion of this structure. The conditions of the rotating angle of the rigid actuating beam and the deflection of the flexible beam are illustrated in Figure 3.10.

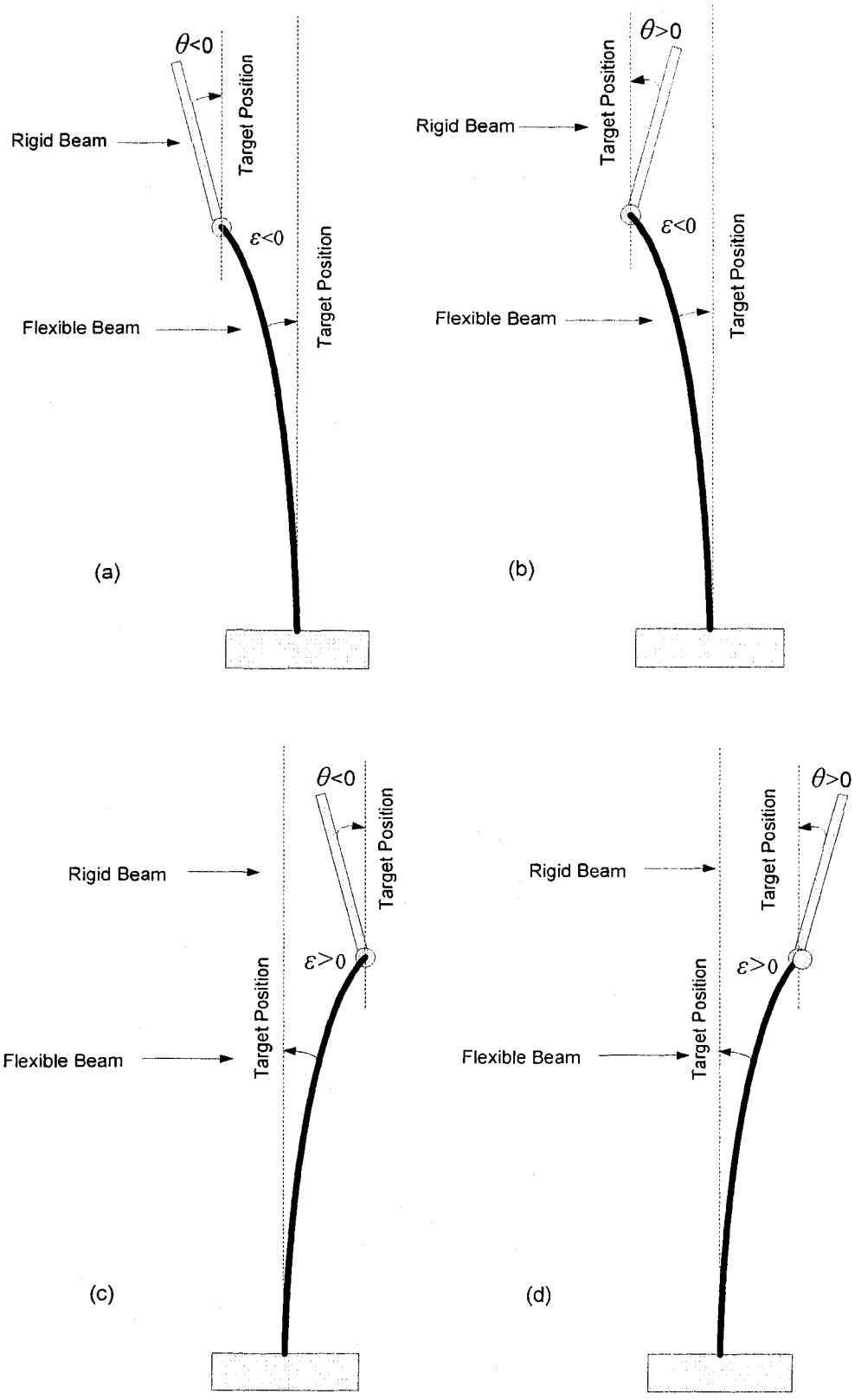


Figure 3.10. Sign convention used to set-up the rule base.

The network architecture of the developed TS0-NF controller is schematically shown in Figure 3.11, in which $M = 9$, $L = 3$, and $n = 2$.

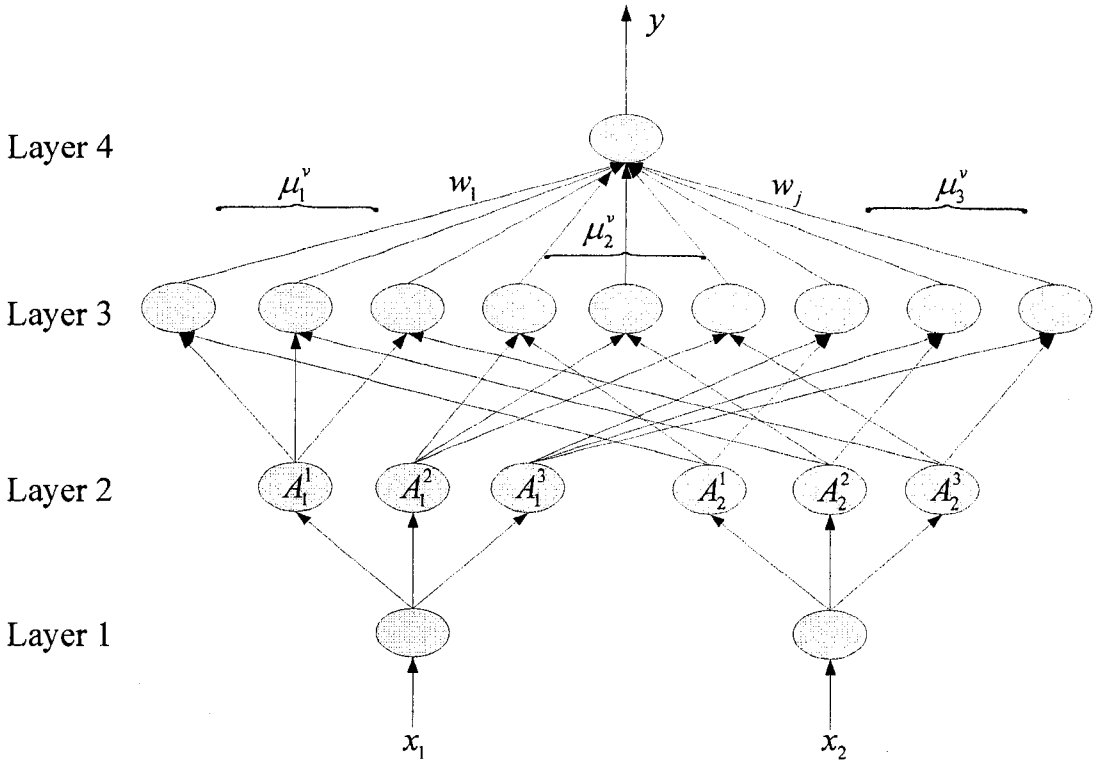


Figure 3.11. Network architecture of the TS0-NF controller.

- ***Layer 1***: The inputs x_1 and x_2 are fed to the network; the neurons in this layer just transmit input values to the next layer.

$$O_i(1) = x_i \tag{3-22}$$

- ***Layer 2***: Fuzzification: The input variables are fuzzified in this layer. If Gaussian and Sigmoid MFs are applied in this case, the respective node outputs will be

$$O_i^1(2) = \mu_{A_i^1}(x_i) = S(x_i, a_i^1, c_i^1) = \frac{1}{1 + e^{-a_i^1 \cdot (x_i - c_i^1)}};\ i = 1, 2 \tag{3-23}$$

$$O_i^2(2) = \mu_{A_i^2}(x_i) = G(x_i, a_i^2, c_i^2) = e^{-\frac{1}{2}\cdot(\frac{x_i - a_i^2}{c_i^2})^2}; \quad (3\text{-}24)$$

$$O_i^3(2) = \mu_{A_i^3}(x_i) = S(x_i, a_i^3, c_i^3) = \frac{1}{1 + e^{-a_i^3 \cdot (-x_i - c_i^3)}}; \quad (3\text{-}25)$$

where a_i^2 and c_i^2 represent the center and spread of the Gaussian MF in equation (3-24), respectively. And a_i^1 and a_i^3 are the slope as $x_i = c_i^1$ and $x_i = -c_i^3$ in equation (3-23) and (3-25). The upper subscripts of $O_i^1(2)$, $O_i^2(2)$, or $O_i^3(2)$ denote the number of MF corresponding to each input variable.

- ***Layer 3***: The links in this layer perform precondition matching of fuzzy rules. Every neuron represents for a specific fuzzy operation, whose output is the product of all the incoming signals:

$$O_1^v(3) = O_1^1(2) \cdot O_2^v(2) = \mu_1^v = \mu_{A_1^1}(x_1) \cdot \mu_{A_2^v}(x_2), \; v = 1, 2, \text{and } 3 \quad (3\text{-}26)$$

$$O_2^v(3) = O_1^2(2) \cdot O_2^v(2) = \mu_2^v = \mu_{A_1^2}(x_1) \cdot \mu_{A_2^v}(x_2), \; v = 1, 2, \text{and } 3 \quad (3\text{-}27)$$

$$O_3^v(3) = O_1^3(2) \cdot O_2^v(2) = \mu_3^v = \mu_{A_1^3}(x_1) \cdot \mu_{A_2^v}(x_2), \; v = 1, 2, \text{and } 3 \quad (3\text{-}28)$$

The upper subscript v of $O_1^v(3)$, $O_2^v(3)$, or $O_3^v(3)$ is the corresponding number of the MFs for the second input variable. The lower subscripts represent the first three outputs, the middle three outputs, and the last three outputs in the third layer, respectively. These letters have been shown in Figure 3.11.

- ***Layer 4***: The single neuron in this layer is a fixed neuron which computes the overall output as the summation of all incoming signals, that is,

$$y = \frac{\sum_{j=1}^{M} \mu_j \cdot w_j}{\sum_{j=1}^{M} \mu_j}, \tag{3-29}$$

where y is the output variable, $\mu_j = \mu_{A_1^v}(x_1) \cdot \mu_{A_2^v}(x_2) \cdots \mu_{A_i^v}(x_i)$ is a firing strength of rule j. Equation (29) can be rewritten:

$$O(4) = y = \frac{\sum_{\beta=1}^{3} (\sum_{v=1}^{3} O_\beta^v(3) \cdot w_\beta^v)}{\sum_{\beta=1}^{3} (\sum_{v=1}^{3} O_\beta^v(3))}$$

$$= \frac{\sum_{\beta=1}^{3} (\sum_{v=1}^{3} \mu_\beta^v \cdot w_\beta^v)}{\sum_{\beta=1}^{3} (\sum_{v=1}^{3} \mu_\beta^v)} \; ; \; \beta = 1, 2, \text{ and } 3 \tag{3-30}$$

The purpose of using equation (3-30) to replace equation (3-29) is to clarify the following fact. Actually, the first input x_1 should be the rotating angle error between the desired and the real positions of the rigid beam, and the second input x_2 should be the deflection error of the flexible beam. Because both the desired input values of this structure are zero, the rotating angle error can be realized as the actual rotating angle, and the deflection error can be also realized as the actual deflection value.

3.3 The TS1-NF controller

As illustrated in section 2.2, IF (x_1 is A) and (x_2 is B), THEN ($y = f(x_1, x_2)$), it is a TS0 model if $f(x_1, x_2)$ is a constant, as described in section 3.2. On the other hand, it becomes a TS1 NF model if $f(x_1, x_2)$ is a first-order polynomial [80, 87].

$$R^j\text{: IF } (x_1 \text{ is } A_1^v) \text{ AND } (x_2 \text{ is } A_2^v), \text{ THEN } (f_j = p_j \cdot x_1 + q_j \cdot x_2 + r_j), \; j=1, 2 \ldots 9 \tag{3-31}$$

For instance:

$$R^1\text{: IF } (x_1 \text{ is } A_1^1) \text{ AND } (x_2 \text{ is } A_2^1), \text{ THEN } (f_1 = p_1 \cdot x_1 + q_1 \cdot x_2 + r_1), \tag{3-32}$$

$$R^2\text{: IF } (x_1 \text{ is } A_1^1) \text{ AND } (x_2 \text{ is } A_2^2), \text{ THEN } (f_2 = p_2 \cdot x_1 + q_2 \cdot x_2 + r_2), \tag{3-33}$$

$$\vdots$$

$$R^9\text{: IF } (x_1 \text{ is } A_1^3) \text{ AND } (x_2 \text{ is } A_2^3), \text{ THEN } (f_9 = p_9 \cdot x_1 + q_9 \cdot x_2 + r_9), \tag{3-34}$$

The network architecture of this TS1-NF controller is schematically shown in Figure 3.12.

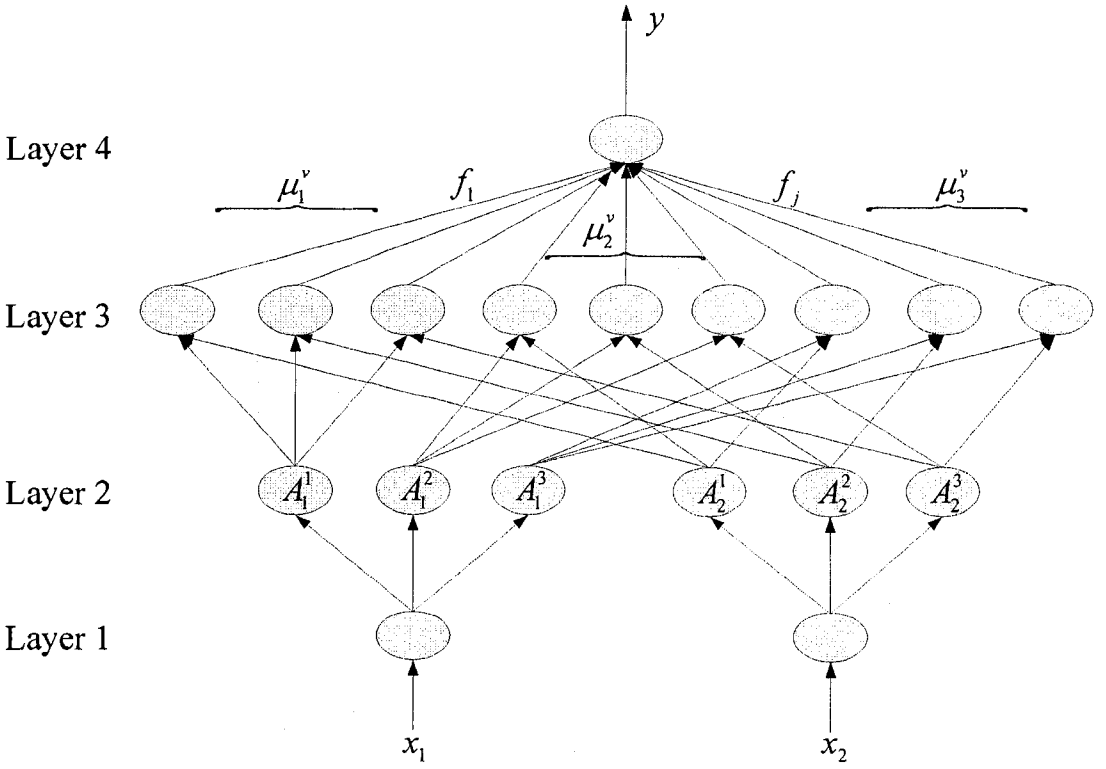


Figure 3.12. Network architecture of the TS1-NF controller.

- ***Layer 1***: The same inputs x_1 and x_2 are still fed to the network. They can be described by equation (3-22). $O_i(1), i = 1, 2$ is the output of the first layer.

- ***Layer 2***: Fuzzification: The input variables are fuzzified in this layer. The same equations (3-23) to (3-25) can be also applied in this case. In order to achieve real-time application by using TS1-NF controller, the parameters in MFs $\{a_i^v, c_i^v, \upsilon = 1, 2, 3$ and $i = 1, 2\}$ have to be fixed and applied as constant to the implementation. $O_i^1(2)$, $O_i^2(2)$, and $O_i^3(2)$ are the outputs of the second layer.
- ***Layer 3***: The links in this layer perform precondition matching of fuzzy rules. Every neuron represents for a specific fuzzy operation, whose output is the product of all the incoming signals, which can be represented by equations (3-26) to (3-28). They have been illustrated in the above section. $O_1^v(3), O_2^v(3)$, and $O_3^v(3)$ are the outputs of the third layer.
- ***Layer 4***: The single neuron in this layer is a fixed neuron which computes the overall output as the summation of all incoming signals, that is,

$$y = \frac{\sum_{j=1}^{M} \mu_j \cdot f_j}{\sum_{j=1}^{M} \mu_j}, \; j = 1, 2 \ldots 9 \tag{3-35}$$

where y is the output variable of TS1-NF controller, equation (35) can be rewritten:

$$O(4) = y = \frac{\sum_{\beta=1}^{3} (\sum_{v=1}^{3} O_\beta^v(3) \cdot f_\beta^v)}{\sum_{\beta=1}^{3} (\sum_{v=1}^{3} O_\beta^v(3))}$$

$$= \frac{\sum_{\beta=1}^{3} (\sum_{v=1}^{3} \mu_\beta^v \cdot f_\beta^v)}{\sum_{\beta=1}^{3} (\sum_{v=1}^{3} \mu_\beta^v)} \; ; \; \beta = 1, 2, \text{and } 3 \tag{3-36}$$

Chapter 4 Testing of Neural Fuzzy Controllers

Once the NF controllers have developed as discussed in Chapter 3, the controller parameters should be trained properly so as to achieve better control performance. In this chapter, the related NF classifiers are firstly trained, then they are implemented and tested for performance evaluation.

4.1 Training of NF controllers

4.1.1 Parameter training for the TS0-NF controller

This TS0-NF controller has both linear and nonlinear system parameters. A hybrid training method is adopted in this work to train the controllers as discussed in Section 2.3. The nonlinear parameters in the MFs of the TS0-NF controller are trained by the use of the steepest gradient method, whereas the linear link weight parameters are fine-tuned by LSE. The training process requires a set of representative data pairs between the network inputs and corresponding target outputs. The training goal is to iteratively minimize the network objective (or error) function defined as

$$E = \frac{1}{2} \cdot (y^d - y)^2 \tag{4-1}$$

where y is the actual output and y^d is the desired output for input vector $\boldsymbol{x} = (x_1, x_2)^T$.

Substituting equations (3-26) - (3-30) into (4-1) yields

$$E = \frac{1}{2} \cdot \left[y^d - \frac{\sum_{j=1}^{M} \mu_j(x) \cdot w_j}{\sum_{j=1}^{M} \mu_j(x)} \right]^2$$

$$= \frac{1}{2} \cdot \left[y^d - \frac{\sum_{\beta=1}^{3} (\sum_{v=1}^{3} \mu_\beta^v \cdot w_\beta^v)}{\sum_{\beta=1}^{3} (\sum_{v=1}^{3} \mu_\beta^v)} \right]^2, \ \beta = 1, 2 \text{ and } 3 \tag{4-2}$$

Since the shape of the MF $\mu_{A_i^\upsilon}(\cdot)$ is defined by a_i^v and c_i^v, $\upsilon = 1, 2, 3$ and $i = 1, 2$. Hence, the learning rules based on gradient algorithm for the MFs can be derived as follows:

$$a_i^v(k+1) = a_i^v(k) - \tau_{a_i^v} \cdot \Delta a_i^v = a_i^v(k) - \tau_{a_i^v} \cdot \frac{\partial E}{\partial a_i^v} \tag{4-3}$$

$$c_i^v(k+1) = c_i^v(k) - \tau_{c_i^v} \cdot \Delta c_i^v = c_i^v(k) - \tau_{c_i^v} \cdot \frac{\partial E}{\partial c_i^v} \tag{4-4}$$

where

$$\frac{\partial E}{\partial a_i^v} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial \mu_{A_i^v}} \cdot \frac{\partial \mu_{A_i^v}}{\partial a_i^v} \tag{4-5}$$

$$\frac{\partial E}{\partial c_i^v} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial \mu_{A_i^v}} \cdot \frac{\partial \mu_{A_i^v}}{\partial c_i^v} \tag{4-6}$$

$a_i^v(k)$ and $c_i^v(k)$ are the values of the variables at the kth time step; Δa_i^v and Δc_i^v are the current gradient; and $\tau_{a_i^v}$ and $\tau_{c_i^v}$ are the learning rates that affect the training convergence and stability of the parameters. A larger learning rate could speed up the convergence but might result in overshooting, while a smaller learning rate has an opposite effect. $\tau_{a_i^v}$ and $\tau_{c_i^v}$ ranging from 10^{-3} to 10 have been tested in this work and are used in the experiments.

For instance, if a_1 (or a_i^v when $\upsilon = 1$ and $i = 1$) is needed to be trained on-line, equation (4-3) and equation (4-5) will be used by the Embedded MATLAB Function in the Figure 4.1.

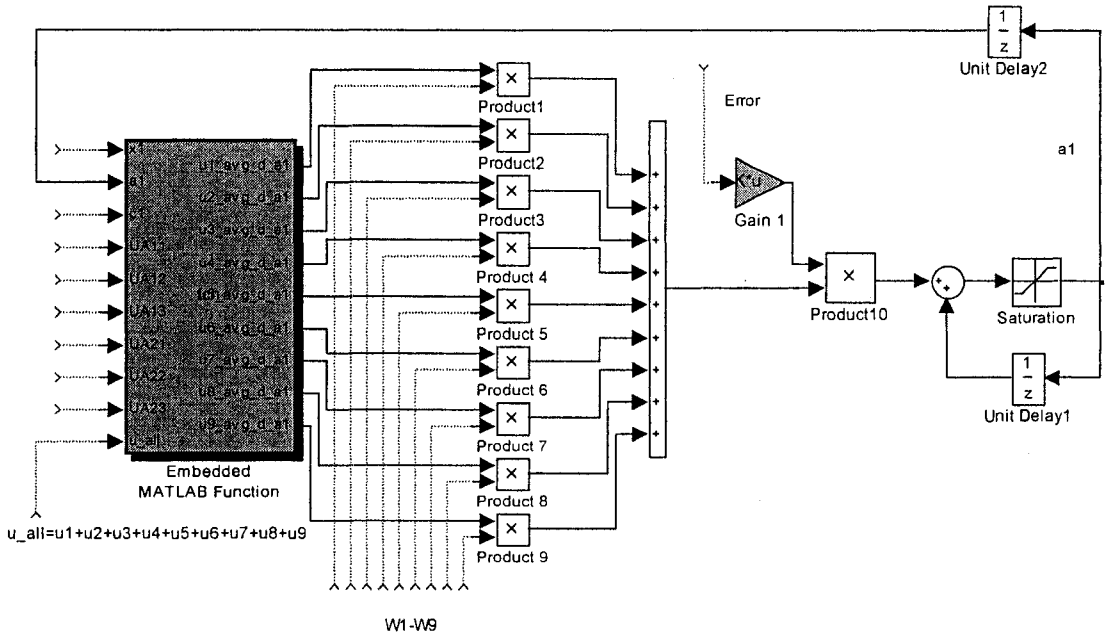


Figure 4.1. Training Block for the updated parameter a_1.

Figure 4.1 was built by using MATLAB blocks. 1) The Embedded MATLAB Function block includes ten inputs: a_1 and c_1,which are needed to be trained by using the gradient method, are the parameters in MF $\mu_{A_1^1}$; UA11, UA12...UA23 present the outputs of the six MFs in the TS0-NF controller's network, respectively; μ_all is the summation of all the μ_j, $j = 1, 2...9$ (or μ_β^v, $\beta = 1, 2, 3$; $v = 1, 2, 3$). The outputs of the Embedded MATLAB Function block are nine derivatives of $\mu_j\_avg\_d$ with respect to the parameter a_1. Each value of $\mu_j\_avg\_d$ is calculated by the equation $\mu_j\_avg\_d = \mu_j / \sum_{j=1}^{9} \mu_j$. 2) The derivative equation can be described by $\partial E / \partial a_1 = \sum_{j=1}^{9} (\mu_j\_avg\_d\_a_1) \cdot w_j$ in figure 4.1. w_j is the consequent parameter,

which will be trained by using the LSE method in the following part. 3) The value of the tuning rate $\tau_{a_i^v}$ can be represented by Gain 1 in Figure 4.1.

The gradient training method forms the basis for many direct methods used in optimizing both constrained and unconstrained problems. Despite its slow convergence, the gradient method may be the most commonly used nonlinear optimization technique due to its simplicity. The gradient method also enables to train a network without the use of all available input/output pairs.

The LES algorithm is used to train the consequent parameters (or the weights) w_β^v. From equation (3-30), it is observed that the final output y is a linear function of the consequent parameters w_j. Hence, given the values of the membership parameters (centers and widths) and g training data $(\boldsymbol{x}^{(k)}, \boldsymbol{y}^{d(k)})$, $k=1, 2\ldots g$, we can form g linear equations in terms of the consequent parameters as follows:

$$\begin{bmatrix} y^{d(1)} \\ y^{d(2)} \\ \cdot \\ \cdot \\ \cdot \\ y^{d(g)} \end{bmatrix} = \begin{bmatrix} \bar{\mu}_1^{(1)} \cdot w_1 + \bar{\mu}_2^{(1)} \cdot w_2 + \ldots + \bar{\mu}_M^{(1)} \cdot w_M \\ \bar{\mu}_1^{(2)} \cdot w_1 + \bar{\mu}_2^{(2)} \cdot w_2 + \ldots + \bar{\mu}_M^{(2)} \cdot w_M \\ \cdot \\ \cdot \\ \cdot \\ \bar{\mu}_1^{(g)} \cdot w_1 + \bar{\mu}_2^{(g)} \cdot w_2 + \ldots + \bar{\mu}_M^{(g)} \cdot w_M \end{bmatrix} = \begin{bmatrix} \bar{\mu}_1^{(1)} \bar{\mu}_2^{(1)} \ldots \bar{\mu}_M^{(1)} \\ \bar{\mu}_1^{(2)} \bar{\mu}_2^{(2)} \ldots \bar{\mu}_M^{(2)} \\ \cdot \\ \cdot \\ \cdot \\ \bar{\mu}_1^{(g)} \bar{\mu}_2^{(g)} \ldots \bar{\mu}_M^{(g)} \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ \cdot \\ \cdot \\ \cdot \\ w_M \end{bmatrix} \tag{4-7}$$

where $\bar{\mu}_j^{(k)} = \mu_j^{(k)} / \sum_{j=1}^{M} \mu_j^{(k)} = \mu_\beta^{v(k)} / \sum_{\beta=1}^{3} (\sum_{v=1}^{3} \mu_\beta^{v(k)})$ and $\mu_j^{(k)}$ can be replaced by $\mu_\beta^{v(k)}$ ($j=1,2,\ldots,M$; $M=9$, $\beta=1,2,3$; $v=1,2,3$) in equation (4-2). The values of $\sum_{\beta=1}^{3} (\sum_{v=1}^{3} \mu_\beta^{v(k)})$ are calculated by equations (3-26) - (3-28) when the input is $\boldsymbol{x}^{(k)}$.

Define the following vectors:

$$\varphi_{TS0}^{T}(k) = [\overline{\mu}_1^{(k)} \quad \overline{\mu}_2^{(k)} \quad \ldots \quad \overline{\mu}_M^{(k)}] \tag{4-8}$$

$$\boldsymbol{\Phi}_{TS0} = [\varphi_{TS0}^{T}(1) \quad \varphi_{TS0}^{T}(2) \quad \ldots \quad \varphi_{TS0}^{T}(g)]^T \tag{4-9}$$

$$\boldsymbol{w}_{TS0} = [w_1 \quad w_2 \quad \ldots \quad w_M]^T \tag{4-10}$$

$$\boldsymbol{Y} = [y^{d(1)} \quad y^{d(2)} \quad \ldots \quad y^{d(g)}]^T \tag{4-11}$$

To simplify notation, equation (4-7) can be expressed in a matrix-vector form:

$$\boldsymbol{Y} = \boldsymbol{\Phi}_{TS0} \cdot \boldsymbol{w}_{TS0} \tag{4-12}$$

According to linear equation (4-12) the tuning of w_j can be viewed as a problem of parameters estimation. The LSE method is a useful technique for parameter estimation, which is very simple if the model is linear in terms of the parameters such as those in equation (4-12). A concise approach to solve equation (4-12) is to use the pseudo inverse technique,

$$\boldsymbol{w}_{TS0}^{*} = (\boldsymbol{\Phi}_{TS0}^{T} \cdot \boldsymbol{\Phi}_{TS0})^{-1} \cdot \boldsymbol{\Phi}_{TS0}^{T} \cdot \boldsymbol{Y} \tag{4-13}$$

where $(\boldsymbol{\Phi}_{TS0}^{T} \cdot \boldsymbol{\Phi}_{TS0})^{-1} \cdot \boldsymbol{\Phi}_{TS0}^{T}$ is the pseudo inverse of $\boldsymbol{\Phi}_{TS0}$ if $\boldsymbol{\Phi}_{TS0}^{T} \cdot \boldsymbol{\Phi}_{TS0}$ is nonsingular. Then the LES loss function can be minimized.

In many cases, the row vectors of matrix $\boldsymbol{\Phi}_{TS0}$ (and the corresponding elements in $\boldsymbol{Y}$) are obtained sequentially in real time; hence the LSE of $\boldsymbol{w}_{TS0}^{T}$ in equation (4-12) can be computed recursively. Let the lth row vector of matrix $\boldsymbol{\Phi}_{TS0}^{T}$ in (4-12) be $\varphi_{TS0}^{T}(l)$ and the lth element of $\boldsymbol{Y}$ be $y^{d(l)}$ then $\boldsymbol{w}_{TS0}^{*}$ can be calculated recursively using the following formula:

$$\boldsymbol{w}_{TS0}^{*}(l+1) = \boldsymbol{w}_{TS0}^{*}(l) + H_{TS0}(l+1) \cdot (y^{d(l)} - \varphi_{TS0}^{T}(l+1) \cdot \boldsymbol{w}_{TS0}^{*}(l)), \tag{4-14}$$

$$H_{TS0}(l+1) = S_{TS0}(l+1) \cdot \varphi_{TS0}(l+1)$$

$$= S_{TS0}(l) \cdot \varphi_{TS0}(l+1) \cdot (I + \varphi_{TS0}^{T}(l+1) \cdot S_{TS0}(l) \cdot \varphi_{TS0}(l+1))^{-1} \quad (4\text{-}15)$$

$$S_{TS0}(l+1) = S_{TS0}(l) - S_{TS0}(l) \cdot \varphi_{TS0}(l+1)$$

$$\cdot (I + \varphi_{TS0}^{T}(l+1) \cdot S_{TS0}(l) \cdot \varphi_{TS0}(l+1))^{-1} \cdot \varphi_{TS0}^{T}(l+1) \cdot S_{TS0}(l)$$

$$= (I - H_{TS0}(l+1) \cdot \varphi_{TS0}^{T}(l+1)) \cdot S_{TS0}(l) \quad (4\text{-}16)$$

The initial conditions are $w_0 = 0$ and $S(0) = \gamma \cdot I$, where γ is a positive large number and I is the identity matrix of dimension $M \times M$.

Since the error $e_{vol} = y^d - y$ at the output *vol* of the NF controller is not directly available, and only the system error $e_\theta = \theta^d - \theta$ and $e_\varepsilon = \varepsilon^d - \varepsilon$ can be measured at the outputs (θ, ε) of plant due to $\theta^d = 0$, and $\varepsilon^d = 0$, e_θ will be the actual rotating angle θ. Furthermore, e_ε will be the actual deflection ε; therefore, we chose θ^d as the desired control output (target output) instead of using the desired controller's output y^d. Moreover, by both the analytical analysis and the PD control experiment, it has been proved that one of the plant's output θ and the control output y follow a reverse relationship. Thus, equation (4-1) can be replaced by:

$$E = \frac{1}{2} \cdot (y^d - y)^2 = \frac{1}{2} \cdot C_r \cdot (\theta^d - \theta)^2 \quad (4\text{-}17)$$

where C_r is the negative ratio of e_{vol} and e_θ. The derivative for parameter a_i^v in equation (4-3) can be calculated by

$$\frac{\partial E}{\partial a_i^v} = \frac{\partial E}{\partial \theta} \cdot \frac{\partial \theta}{\partial y} \cdot \frac{\partial y}{\partial \mu_{A_i^v}} \cdot \frac{\partial \mu_{A_i^v}}{\partial a_i^v} \tag{4-18}$$

The common parts of equation (4-3) will become:

$$\frac{\partial E}{\partial \theta} = C_r \cdot (\theta^d - \theta) \cdot (-1) \tag{4-19}$$

$$\frac{\partial \theta}{\partial y} = \sqrt{C_r} \tag{4-20}$$

$$\frac{\partial y}{\partial \mu_{A_i^v}} = \Gamma_i^v \qquad \text{(by equations (3-26) - (3-30))} \tag{4-21}$$

$$\frac{\partial \mu_{A_i^v}}{\partial a_i^v} = \Lambda_{a_i^v} \qquad \text{(by equations (3-23) - (3-25))} \tag{4-22}$$

Based on the LSE method, $\boldsymbol{w} = [w_1 \quad w_2 \quad \ldots \quad w_M]^T$ will be tuned by equations (4-14) - (4-16).

In equation (4-14), y^d can be replaced by θ^d as illustrated in equation (4-17). Then equation (4-14) can be re-written as following:

$$\hat{\boldsymbol{w}}_{TS0}(l+1) = \hat{\boldsymbol{w}}_{TS0}(l) + H_{TS0}(l+1) \cdot C_{w\_TS0}(\theta^{d(l)} - \varphi_{TS0}^T(l+1) \cdot \hat{\boldsymbol{w}}_{TS0}(l)) \tag{4-23}$$

where C_{w_TS0} is the ratio of e_{vol} and e_θ when the consequent parameters $\boldsymbol{w}$ are trained by LSE. Correspondingly, the tuning rate of the weights is determined by equation (4-24) in this case.

$$\tau_{w\_TS0} = H_{TS0}(l+1) \cdot C_{w\_TS0} \tag{4-24}$$

4.1.2 Parameter training for the TS1-NF controller

The nonlinear MF parameters in the TS1-NF controller are trained by the use of the gradient algorithm similar to that in the TS0-NF controller as discussed in section 4.1.1. The linear consequent parameters of the TS1-NF controller are optimized by the LSE method.

Equation (4-7) can be re-written as:

$$\begin{bmatrix} y^{d(1)} \\ y^{d(2)} \\ \cdot \\ \cdot \\ \cdot \\ y^{d(g)} \end{bmatrix} = \begin{bmatrix} \bar{\mu}_1^{(1)} \cdot f_1 + \bar{\mu}_2^{(1)} \cdot f_2 + \ldots + \bar{\mu}_M^{(1)} \cdot f_M \\ \bar{\mu}_1^{(2)} \cdot f_1 + \bar{\mu}_2^{(2)} \cdot f_2 + \ldots + \bar{\mu}_M^{(2)} \cdot f_M \\ \cdot \\ \cdot \\ \cdot \\ \bar{\mu}_1^{(g)} \cdot f_1 + \bar{\mu}_2^{(g)} \cdot f_2 + \ldots + \bar{\mu}_M^{(p)} \cdot f_M \end{bmatrix} = \begin{bmatrix} \bar{\mu}_1^{(1)} \bar{\mu}_2^{(1)} \ldots \bar{\mu}_M^{(1)} \\ \bar{\mu}_1^{(2)} \bar{\mu}_2^{(2)} \ldots \bar{\mu}_M^{(2)} \\ \cdot \\ \cdot \\ \cdot \\ \bar{\mu}_1^{(g)} \bar{\mu}_2^{(g)} \ldots \bar{\mu}_M^{(g)} \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ \cdot \\ f_M \end{bmatrix}$$

$$= \begin{bmatrix} \bar{\mu}_1^{(1)} \bar{\mu}_2^{(1)} \ldots \bar{\mu}_M^{(1)} \\ \bar{\mu}_1^{(2)} \bar{\mu}_2^{(2)} \ldots \bar{\mu}_M^{(2)} \\ \cdot \\ \cdot \\ \cdot \\ \bar{\mu}_1^{(g)} \bar{\mu}_2^{(g)} \ldots \bar{\mu}_M^{(g)} \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & x_2 & 1 & \cdots & 0 & 0 & 0 \\ & \vdots & & & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & x_1 & x_2 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ q_1 \\ r_1 \\ \vdots \\ p_9 \\ q_9 \\ r_9 \end{bmatrix}$$

$$= \begin{bmatrix} \bar{\mu}_1^{(1)} \cdot x_1 & \bar{\mu}_1^{(1)} \cdot x_2 & \bar{\mu}_1^{(1)} & \bar{\mu}_2^{(1)} \cdot x_1 & \bar{\mu}_2^{(1)} \cdot x_2 & \bar{\mu}_2^{(1)} & \cdots & \bar{\mu}_M^{(1)} \cdot x_1 & \bar{\mu}_M^{(1)} \cdot x_2 & \bar{\mu}_M^{(1)} \\ \bar{\mu}_1^{(2)} \cdot x_1 & \bar{\mu}_1^{(2)} \cdot x_2 & \bar{\mu}_1^{(2)} & \bar{\mu}_2^{(2)} \cdot x_1 & \bar{\mu}_2^{(2)} \cdot x_2 & \bar{\mu}_2^{(2)} & \cdots & \bar{\mu}_M^{(2)} \cdot x_1 & \bar{\mu}_M^{(2)} \cdot x_2 & \bar{\mu}_M^{(2)} \\ & \vdots & & & \vdots & & \ddots & & \vdots & \\ \bar{\mu}_1^{(g)} \cdot x_1 & \bar{\mu}_1^{(g)} \cdot x_2 & \bar{\mu}_1^{(g)} & \bar{\mu}_2^{(g)} \cdot x_1 & \bar{\mu}_2^{(g)} \cdot x_2 & \bar{\mu}_2^{(g)} & \cdots & \bar{\mu}_M^{(g)} \cdot x_1 & \bar{\mu}_M^{(g)} \cdot x_2 & \bar{\mu}_M^{(g)} \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ q_1 \\ r_1 \\ \vdots \\ p_9 \\ q_9 \\ r_9 \end{bmatrix} \quad (4\text{-}25)$$

where g is the given training data $(\boldsymbol{x}^{(k)}, y^{d(k)})$, $k=$ 1, 2... g, and $\bar{\mu}_j^{(k)} = \mu_j^{(k)} / \sum_{j=1}^{M} \mu_j^{(k)} = \mu_\beta^{v(k)} / \sum_{\beta=1}^{3} (\sum_{v=1}^{3} \mu_\beta^{v(k)})$. The values of $\sum_{\beta=1}^{3} (\sum_{v=1}^{3} \mu_\beta^{v(k)})$ are calculated by equations (3-26) - (3-28) when the input is $\boldsymbol{x}^{(k)} = (x_1^{(k)}, x_2^{(k)})^T$.

$$\varphi_{TS1}^T(k) = [\bar{\mu}_1^{(k)} \cdot x_1 \quad \bar{\mu}_1^{(k)} \cdot x_2 \quad \bar{\mu}_1^{(k)} \quad \bar{\mu}_2^{(k)} \cdot x_1 \quad \bar{\mu}_2^{(k)} \cdot x_2 \quad \bar{\mu}_2^{(k)} \quad \cdots \quad \bar{\mu}_M^{(k)} \cdot x_1 \quad \bar{\mu}_M^{(k)} \cdot x_2 \quad \bar{\mu}_M^{(k)}] \tag{4-26}$$

$$\boldsymbol{\Phi}_{TS1} = [\varphi_{TS1}^T(1) \quad \varphi_{TS1}^T(2) \quad \ldots \quad \varphi_{TS1}^T(g)]^T \tag{4-27}$$

$$\boldsymbol{w}_{TS1} = [p_1 \quad q_1 \quad r_1 \quad \cdots \quad p_9 \quad q_9 \quad r_9]^T \tag{4-28}$$

$$\boldsymbol{Y} = [y^{d(1)} \quad y^{d(2)} \quad \ldots \quad y^{d(g)}]^T \tag{4-29}$$

For the LSE method used in TS1-NF controller, y^d can be replaced by θ^d as illustrated in equation (4-17). Thus, the matrix of trained parameters will be calculated by:

$$\hat{\boldsymbol{w}}_{TS1}(l+1) = \hat{\boldsymbol{w}}_{TS1}(l) + H_{TS1}(l+1) \cdot C_{w\_TS1}(\theta^{d(l)} - \varphi_{TS1}^T(l+1) \cdot \hat{\boldsymbol{w}}_{TS1}(l)) \tag{4-30}$$

where C_{w_TS1} is the ratio of e_{vol} and e_θ in tuning the consequent parameters $\boldsymbol{w}_{TS1}$, and the tuning rate of the weights is determined by equation (4-31).

$$\tau_{w\_TS1} = H_{TS1}(l+1) \cdot C_{w\_TS1} \tag{4-31}$$

For the recursive LSE formula to account for the time-varying characteristics of the system, the effects of old data pairs must delay as new data pairs become available. This problem has been well studied in the adaptive control and system identification literature. One typical method is to formulate the squared error measure as a weight and to give higher weighting factors to

more recent data pairs, by applying a forgetting factor λ to the original recursive formula (4-14) - (4-16).

The weights (consequent parameters) are trained by equation (4-30), by the computational procedures in (4-32) and (4-33):

$$H_{TS1}(l+1) = S_{TS1}(l+1) \cdot \varphi_{TS1}(l+1)$$

$$= S_{TS1}(l) \cdot \varphi_{TS1}(l+1) \cdot (\lambda + \varphi_{TS1}^{T}(l+1) \cdot S_{TS1}(l) \cdot \varphi_{TS1}(l+1))^{-1} \tag{4-32}$$

$$S_{TS1}(l+1) = \frac{1}{\lambda}[S_{TS1}(l) - S_{TS1}(l) \cdot \varphi_{TS1}(l+1)$$

$$\cdot (\lambda + \varphi_{TS1}^{T}(l+1) \cdot S_{TS1}(l) \cdot \varphi_{TS1}(l+1))^{-1} \cdot \varphi_{TS1}^{T}(l+1) \cdot S_{TS1}(l)]$$

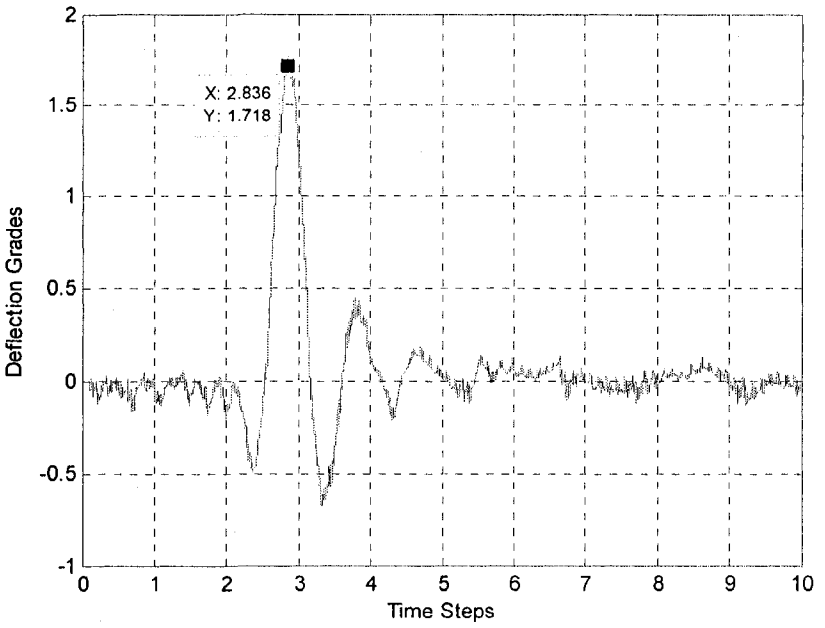
$$= \frac{1}{\lambda}[(I - H_{TS1}(l+1) \cdot \varphi_{TS1}^{T}(l+1)) \cdot S_{TS1}(l)] \tag{4-33}$$

The typical value of λ in practice is between 0.9 and 1. The smaller λ is, the faster the effects of old data decay, and more capable the algorithm is in tracking time-varying parameters. A small λ sometimes causes numerical instability, therefore, the value of λ is often task-dependent and has to be determined experimentally.

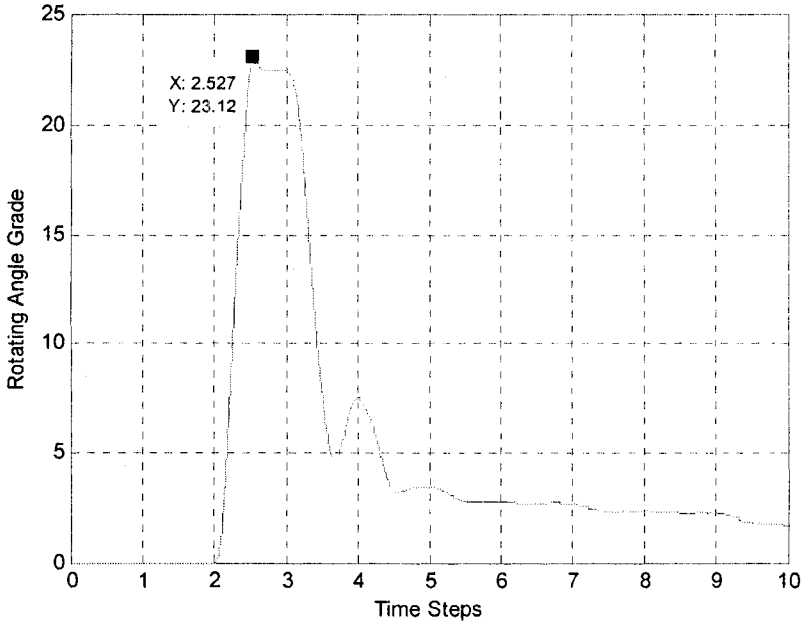
4.2 Testing the NF controllers

4.2.1 The PD controller

For the purpose of comparison, the control test results from a PD control are also included in this section. Since the required disturbance signal in the following experiments is in voltage with a value of 1 *v*, we have to use the angle of 20 *deg* for the Disturbance Gain to replace the angle of 5 *deg* as illustrated in Figure 3.5. The experiment results by using the PD controller with new Disturbance and Control sequence signals are shown in Figure 4.2. In this chapter, the unit of time is *s*, the Deflection's unit is *cm*, and the Rotating Angle's unit is *deg*.



(a)



(b)

Figure 4.2. (a) The control result of the Deflection; (b) The control result of the Rotating Angle.

The *overshoot* of deflection in Figure 4.2(a) is 1.718 *cm*, and the *settling time* is about 3 *s* from $t = 2$ *s* to $t = 5$ s. The measurement of overshooting is applied to describe a situation where the initial reaction of a variable to a shock is greater than its long-run response. The settling time is a measurement to describe how fast the response converges within ± settling time percentage (*Sp*) of the steady state or the final value of the response. *Sp* depends on specific application requirements, and $Sp = 2\%$ is used in this work.

Figure 4.2(b) describes the control output in terms of the Rotating Angle, and its maximum magnitude value is 23.12 *deg*. This peak magnitude can also reflect the control request indirectly. The larger value of the rotating angle is, the more control request of the motor is consumed.

4.2.2 The NN controller

Usually, an NN consists of many interconnected nonlinear processing elements called neurons. The conventional NN is that individual neurons sum their weighted inputs and yield an output through a nonlinear activation function, which is called a static model. As the extension of the static NN, dynamic recurrent NN is constructed by the static neurons through adding state feedback. In this thesis, a three-layer dynamic recurrent NN is developed, which includes one hidden layer with 2 nonlinear neurons and an output layer with one nonlinear neuron. The nonlinear neurons are used to capture the nonlinearities of the approximated system. Each neuron has two incoming signal, and its output is computed by passing the summation of all incoming signals through the nonlinear Sigmoid function.

As a comparison, the experiment results of the NN controller with the same Disturbance sequence signal (Figure 4.3) is as shown in Figure 4.4.

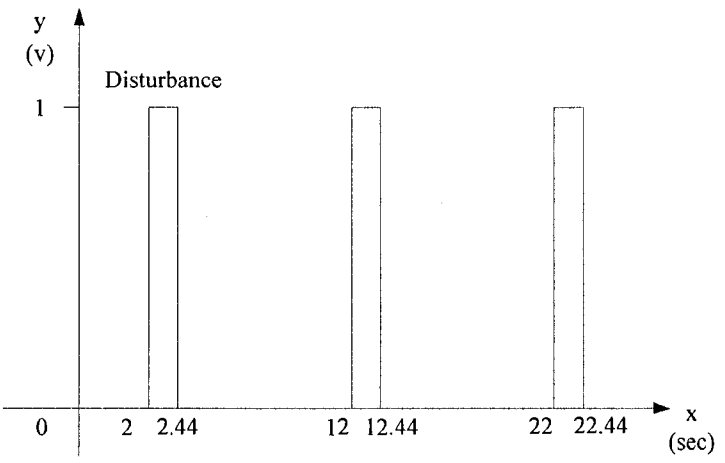
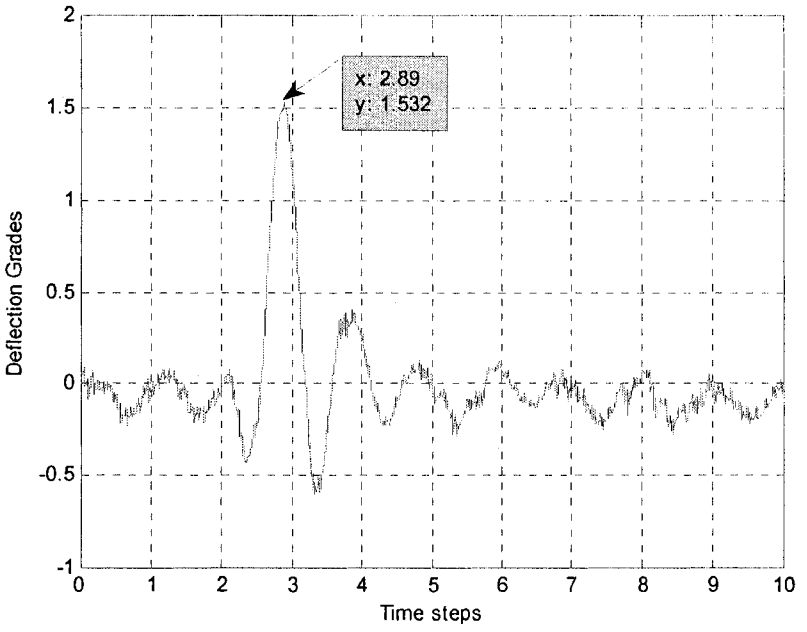


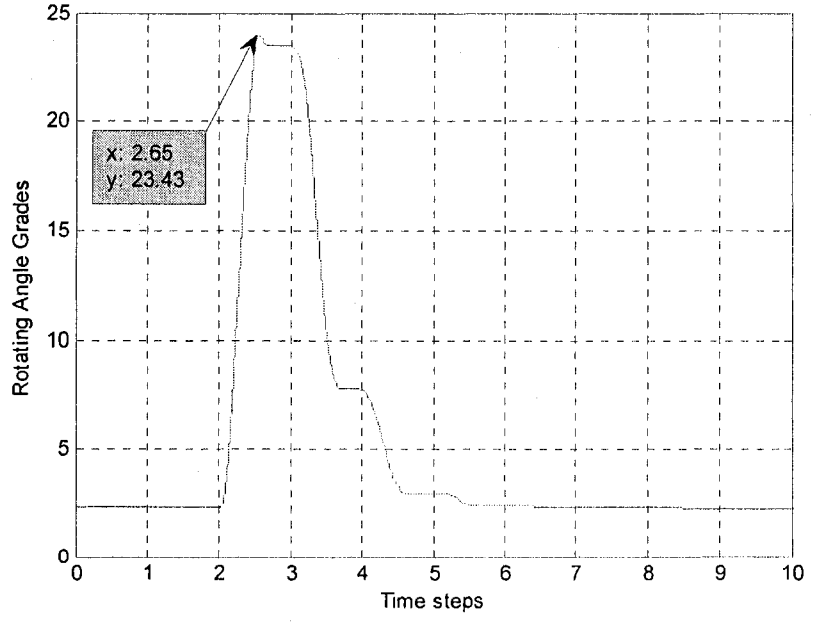
Figure 4.3. The disturbance signal using in TS0-NF controller.

The difference of disturbance signals between the PD controller and the TS0-NF controller is related to the unit of sequence signal: voltage with the amplitude of 1*v* in the NF controller, and degree with the amplitude of 20 *deg* in the PD controller. There exists no switching control

sequence signal. The disturbance sequence still repeats every 10 *s*, and each pulse's duration is over 0.44 *s*.



(a)



(b)

Figure 4.4. (a) the control result of Deflection; (b) the control result of Rotating Angle.

Figure 4.4(a) and 4.4(b) show the control results of the Deflection and the Rotating Angle by using the NN controller, respectively. The overshoot of deflection is 1.532 *cm*. The largest magnitude of negative value is about 0.5 *cm*, and the settling time is around 2 *s*. In Figure 4.4(b) the overshoot of Rotating Angle is 23.43 *deg*, and the steady state value is near 2.5 *deg*.

In terms of the overshoot, the NN controller has a relatively smaller overshoot than in the PD. The better performance of the NN controller is due to its training effect to fine-tune system parameters for better modeling. However, the experimental structure is simple in this case; any external environmental influence can make a significant change in system dynamics. Therefore, it is difficult to recognize NN controller's other advantages over the PD controller. Although the NN results are very limited to a couple of data sets, NN can be used for time-varying systems but PD cannot.

4.2.3 The TS0-NF controller

The structure of the TS0-NF controller is shown in Figure 4.5. Two inputs (θ, ε) are chosen in this controller in the first layer, which is different from the inputs $(\theta, \dot{\theta}, \varepsilon, \dot{\varepsilon})$ in the PD controller.

Six MFs (subsystem 1, subsystem 2 ... subsystem 6) based on Sigmoidal and Gaussian functions are used for both input variables in the second layer. The implemented SIMULINK models of the MFs are shown in Figure 4.6.

In this experiment, the sampling time of the system is chosen $0.001s$. The learning rate in equation (4-24) is selected as $\tau_{w\_TS0} = 0.6$ to improve training convergence and instability. Due

to the limitation of tolerance of the updated parameters when the system is stable, by simulation tests, the learning rate in equation (4-3) and (4-4) is selected as $\tau_{a_i^v} = 0.001$, or $\tau_{c_i^v} = 0.001$.

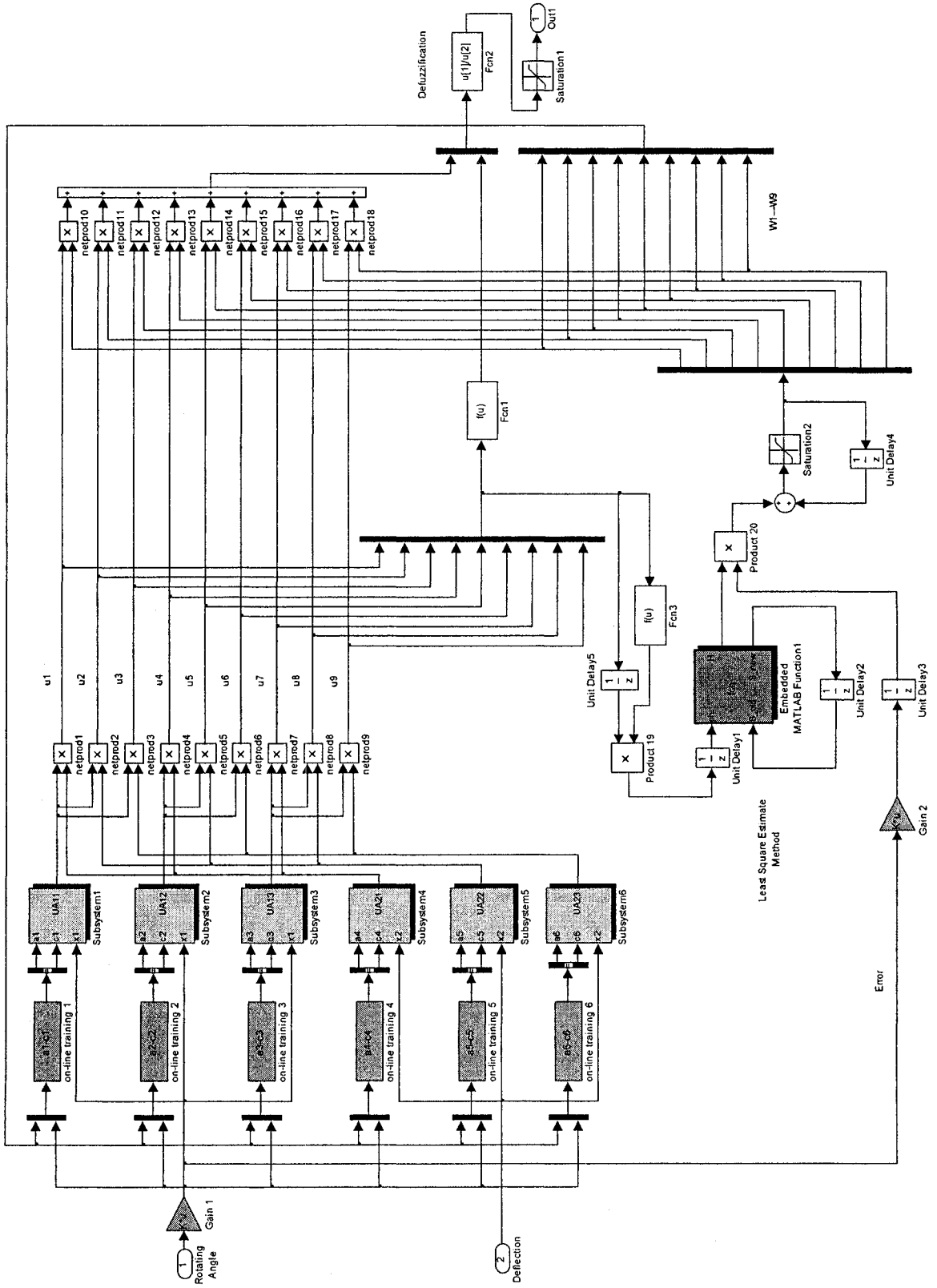
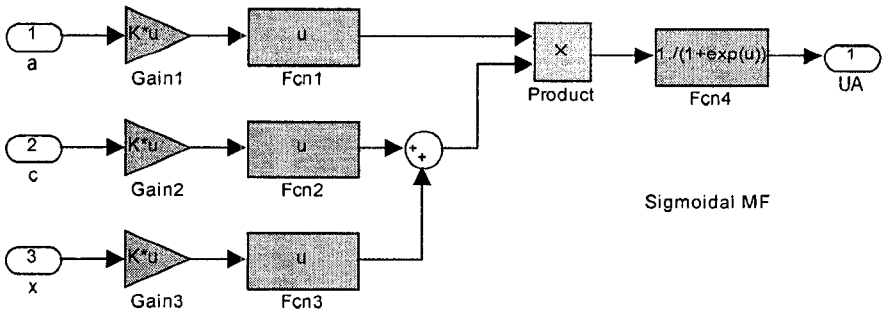
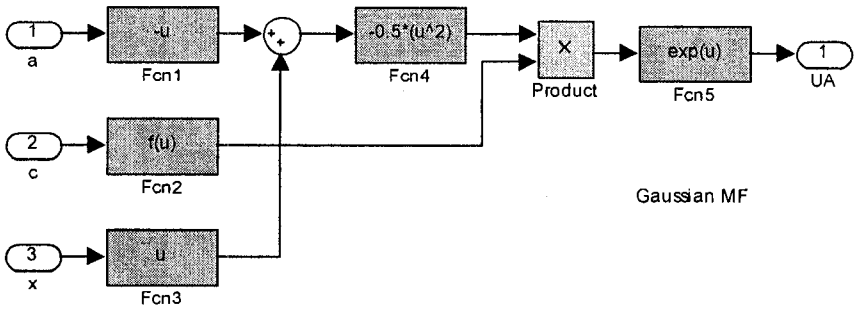


Figure 4.5. The SIMULINK model of the TS0-NF controller.



(a)

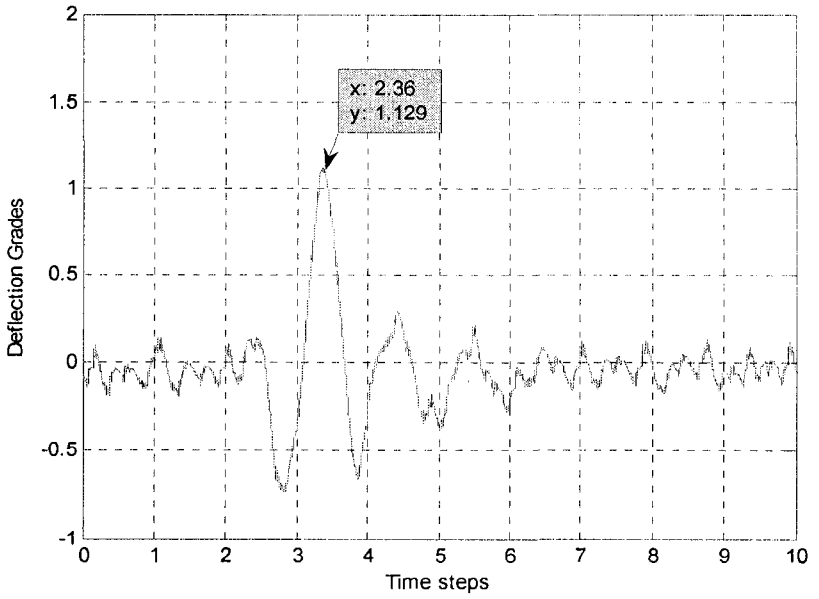


(b)

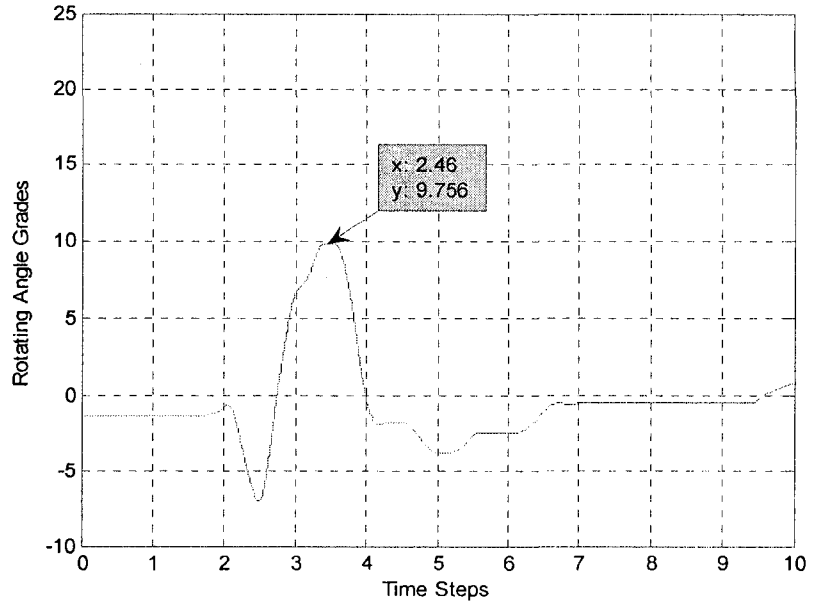
Figure 4.6. (a) Sigmoidal MF; (b) Gaussian MF built by the Matlab blocks.

The difference between the subsystem 1 (or 4) and the subsystem 3 (or 6) is depending on the sign (negative or positive) of x_1 (or x_2) to make the Sigmoidal MF open left or open right.

The experiment results with the developed TS0-NF controller based on the Disturbance sequence signal (Figure 4.3) is as shown in Figure 4.7.



(a)



(b)

Figure 4.7. (a) The control result of Deflection; (b) the control result of Rotating Angle.

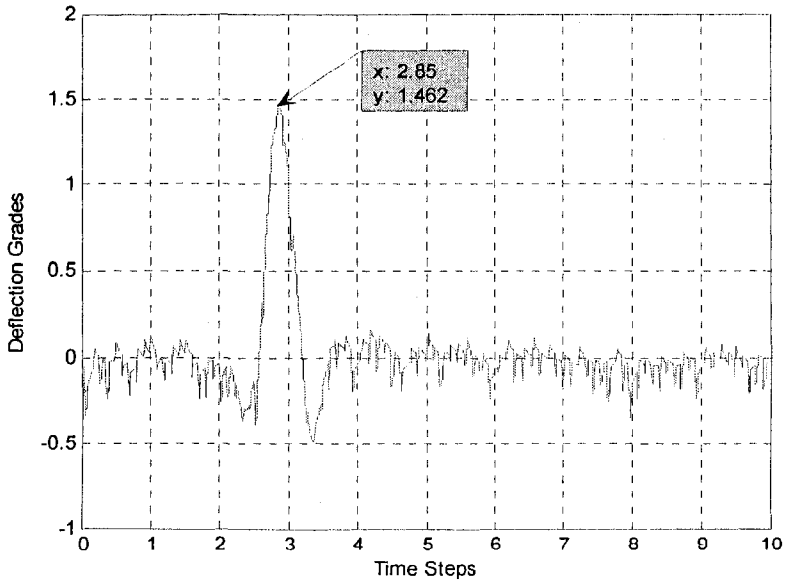
Figures 4.7(a) and 4.7(b) illustrate the control results of the Deflection and the Rotating Angle by using the TS0-NF controller, respectively. The overshoot of deflection in Figure 4.7(a) is 1.129 *cm*. The largest negative value is -0.7 *cm*, and the settling time is about 3.5 *s*. In Figure

4.7(b) the overshoot of the Rotating Angle is 9.756 *deg*, and the largest negative value is near -7 *deg*.

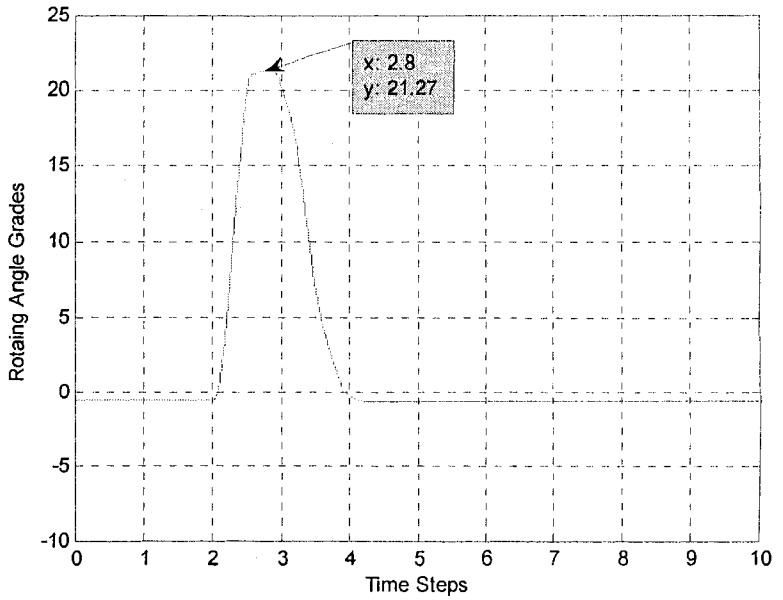
4.2.4 The TS1-NF controller

In this experiment, since the TS1-NF controller involves more linear parameters (twenty seven) to be tuned by the LSE method than those in the TS0-NF controller (27 vs. 9), more time will be required in training in this experiment. Correspondingly, the sampling time is chosen 0.002 *s* (instead of 0.001 *s* in the TS0-NF controller). The learning rate for linear consequent parameters in equation (4-31) is $\tau_{w\_TS1} = 0.7$. Because it is a recursive LSE training process, the forgetting factor $\lambda = 0.9998$ is selected in equation (4-32) and equation (4-33) in this experiment.

The experiment results with the developed TS1-NF controller based on the same Disturbance sequence signal (Figure 4.3) are as shown in Figure 4.8.



(a)



(b)

Figure 4.8. (a) the control result of Deflection; (b) the control result of Rotating Angle.

Figure 4.8 shows the control result of Deflection and the Rotating Angle by using the TS1-NF controller, with the overshoot of deflection 1.462 *cm*, the largest negative value -0.5 *cm*, the settling time 2 *s,* the overshoot of the Rotating Angle 21.27 *deg*, and the largest negative value 0 *deg*.

Based on the aforementioned test results from different controllers, the following conclusions can be reached.

1) Both the proposed two NF controllers (TS0 and TS1) can be real-time implemented efficiently (i.e., over a short sampling time).

2) The adopted hybrid training method can effectively optimize the controller parameters. The training errors can be directly used for self updating of the controllers.

3) In terms of the Deflection among the four controllers, TS0-NF controller has the best performance (with the amplitude of overshoot 1.129 *cm*). Although TS1-NF controller has a little

larger overshoot than in the TS0-NF controller, it has a smaller undershoot with the amplitude less than 0.5 *cm*. That is, both NF controllers have their own advantages and disadvantages.

4) In terms of the Rotating Angle among these controllers, although the TS0-NF controller has a relatively large undershoot, it has a much smaller overshoot (9.756 *deg*) than other controllers. That is, TS0-NF controller requires less control request of motor consumed. On the other hand, although the TS1-NF controller has a relatively larger Rotating Angle than in the PD controller, it can reach a desired steady state (zero degree) after controlling; its control request is needed on one side only.

5) Compare the control results with the NN controller and NF controllers, both the NF controllers outperform the NN controller in terms of the overshoot, system convergence and control request (e.g. Rotating Angle). From figure 4.4, it is seen that the training convergence of the NN controller is slow. The reason is that the NN controller is a black box processor; the resulting distributed knowledge after training representation in a NN is usually difficult to understand. Expert knowledge cannot be implemented in a NN effectively to further improve control reasoning and performance.

In order to overcome the need for precise process representation, the FL provides the controller a high-level IF-THEN control reasoning framework, moreover, the controller structure and parameters are optimized by NN based training. Therefore, both TS0 and TS1 NF controllers are used in this work.

6) In terms of the settling time, the TS1-NF controller has the shortest settling time compared with other controllers; it takes only 2 *s* to make the structure to the steady state.

Based upon the above comprehensive investigation, it is clear that the developed TS0-NF and TS1-NF controllers outperform the classical PD and NN controllers. The reasons are summarized as follows:

- Intelligent controllers possess the ability to perform in noisy environments and can tolerate faults and missing data; they can represent the imprecise models by elastic (non-crisp) or fuzzy constraints on variables.
- NF controllers can approximate arbitrary continuous functions by appropriate training processes. Properly changing the learning rates can significantly improve training convergence and accuracy.
- Based on on-line training, more accurate system parameters enable the NF controllers to achieve optimal or near optimal input-output mappings.

Chapter 5 Vibration Control of Time-Varying Systems

5.1 Overview

As discussed in Chapter 1 (section 1.2), system identification is necessary in modeling system whose dynamic properties cannot be easily represented in terms of first principles or known physical laws. If the dynamics of the system to be controlled varied with time, the key step is to accurately identify the new system in real-time. In general, the purposes of system identification are as follows:

- To predict a system's behavior such as in time series prediction and weather forecasting.
- To explain the interactions and relationships between inputs and outputs of a system.
- To design a controller based on the model of a system, and to do computer simulation of the system under control.

System identification usually involves two steps:

1. *Structure identification:* In this step, a priori knowledge about the target system is applied to determine a class of models within which the most suitable model is to be selected. Usually this class of models is denoted by a parameterized function $y = f(\boldsymbol{u}; \boldsymbol{w})$, where y is the model's output, $\boldsymbol{u}$ is the input vector, and $\boldsymbol{w}$ is the parameter vector. The determination of the function f is problem dependent.
2. *Parameter identification:* In this step, the selected structure of the model is optimized to determine the parameter vector $\boldsymbol{w} = \tilde{\boldsymbol{w}}$ such that the resulting model $\tilde{y} = f(\boldsymbol{u}; \tilde{\boldsymbol{w}})$ can describe the system appropriately.

In general, system identification is not a one-pass process, but a repetitive identification procedure for both the structure and the parameters until a satisfactory model is derived. The general procedures are as follows [81]:

1. Specify and parameterize a class of mathematical models representing the system to be identified.
2. Perform parameter identification to choose the parameters that best fit the training data set.
3. Conduct validation tests to see if the model identified responds correctly to an unseen data set.
4. Terminate the procedure as soon as the results of the validation test are satisfied. Otherwise, another class of models is selected and steps 2 through 4 are repeated.

For the estimation of model parameters from measured data, the procedure can be undertaken in three distinct steps: data collection, system identification and determination of modal parameters from the identified system description. These steps are illustrated shown in Figure 5.1.

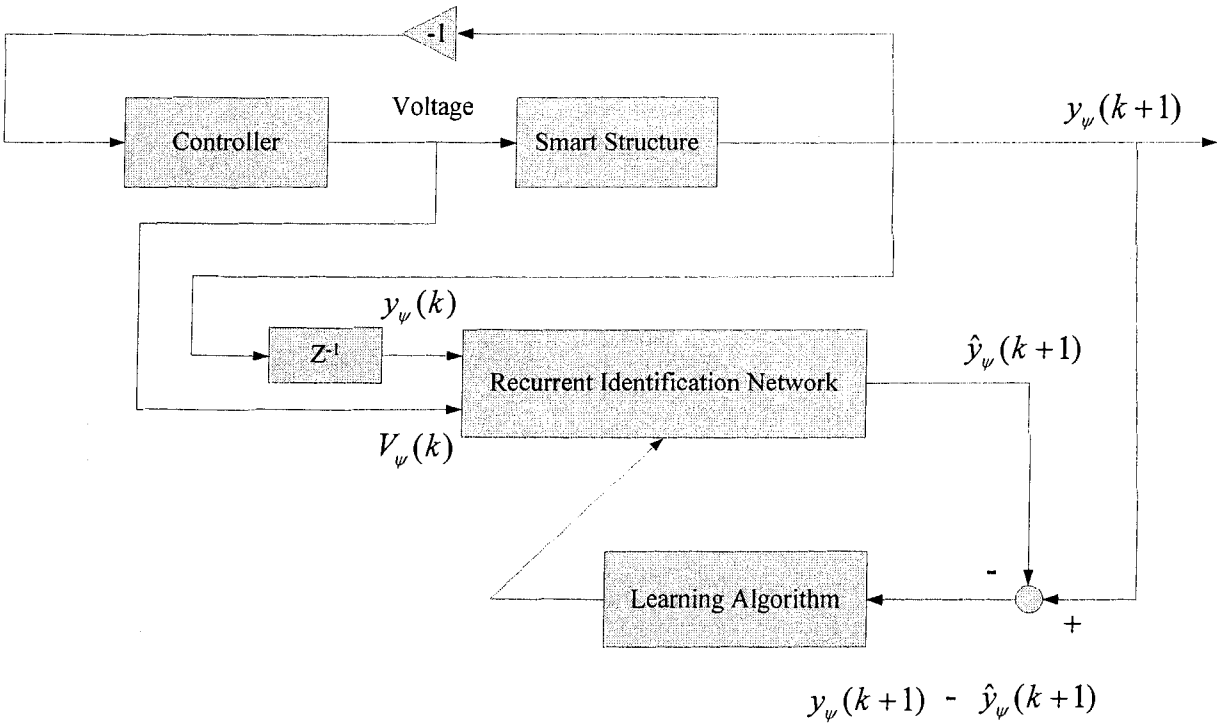


Figure 5.1. Block diagram for parameter identification.

As shown in Figure 5.1, an input after a negative feedback of $y_\psi(k+1)$, $k=1, 2 \ldots m$, as time steps, is applied to the controller; the difference between the target system's output $y_\psi(k+1)$ and the model's output $\hat{y}_\psi(k+1)$ is used, in an appropriate manner, to update all network parameters so as to reduce this difference. Actually, both desired and predict outputs $y_\psi(k+1)$ and $\hat{y}_\psi(k+1)$ involve two parts: one is the rotating angle of the rigid beam $\theta_\psi(k+1)$, and another is the deflection of the flexible beam $\varepsilon_\psi(k+1)$, such as $y_\psi(k+1) = \{\theta_\psi(k+1); \varepsilon_\psi(k+1)\}$. Both the output of controller $V_\psi(k)$ and the outputs of smart structure after a unit delay $y_\psi(k)$ will be the inputs to this identification network. These m desired input-output data pairs, $\psi = 1, 2 \ldots m$, are the training data sets (or sampled data sets).

In this chapter, a novel recurrent identification network, RIN, is developed to adaptively recognize the system dynamics. A new hybrid online training technique, based on RTRL and LSE, is adopted to optimize RIN parameters; its training efficiency is compared with a classical training method based on the gradient algorithm.

5.2 The recurrent identification network (RIN)

The network architecture of the developed RIN is schematically shown in Figure 5.2. This RIN has three layers, each hidden layer neuron having a dynamic recurrent feedback link. The mapping relationship of this RIN network is described as follows.

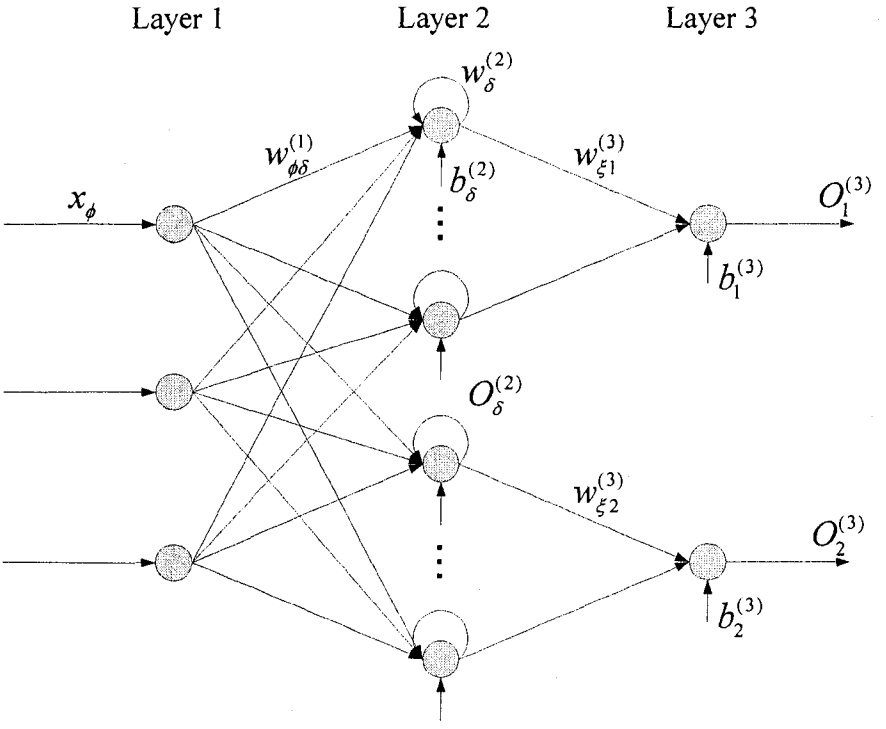


Figure 5.2. Structure of RIN by using gradient method.

The input layer contains three inputs in this case. They transmit the input information to the second (or hidden) layer:

$$I_\phi(k) = x_\phi(k) \ (\phi = 1, 2, 3) \tag{5-1}$$

Hidden layer: a feedback link is connected to each neuron in layer 2 in order to improve the controlled accuracy and convergence property (stability). The signal in each feedback link represents the neuron output in the previous time step (i.e., at k-1), and at each time step it is stored in the training database.

$$O_\delta^{(2)}(k) = f(net_\delta(k)) \quad (\delta = 1, 2...10) \tag{5-2}$$

$$net_\delta(k) = w_\delta^{(2)} \cdot O_\delta^{(2)}(k-1) + \sum_{\phi=1}^{3} w_{\phi\delta}^{(1)} \cdot x_\phi(k) + b_\delta^{(2)} \tag{5-3}$$

where $net_\delta(k)$ is the input to the δ th neuron in the hidden layer; $O_\delta^{(2)}(k)$ is the output of the δ th neuron in the hidden layer; $f(*)$ is the activation function, which is chosen as a continuous nonlinear sigmoid function, $f(x) = \dfrac{1}{1+e^{-x}}$; $w_{\phi\delta}^{(1)}, w_\delta^{(2)}$ are the weights of the inputs to the hidden layer, and the weights of the hidden neurons, respectively; $b_\delta^{(2)}$ are the bias of the second layer.

Output layer: the neurons in this layer are fixed neurons labelled, which compute the outputs as the summation of incoming signals as shown in equations (5-4) and (5-5).

$$O_1^{(3)}(k) = \sum_{\delta=1}^{5} w_{\xi 1}^{(3)} \cdot O_\delta^{(2)}(k) + b_\delta^{(3)} \ (\xi = 1, 2...5) \tag{5-4}$$

$$O_2^{(3)}(k) = \sum_{\delta=6}^{10} w_{\xi 2}^{(3)} \cdot O_\delta^{(2)}(k) + b_\delta^{(3)} \tag{5-5}$$

where $O_1^{(3)}(k)$ and $O_2^{(3)}(k)$ are the neural network outputs in the third layer; $w_{\xi 1}^{(3)}$, $w_{\xi 2}^{(3)}$,are the weights of the hidden neurons in the output layer. $b_1^{(3)}, b_2^{(3)}$ are the bias of the third layer.

The error function is as defined in equation (4-1)

$$E_1 = \frac{1}{2} \cdot (y_1^d(k) - y_1(k))^2 \tag{5-6}$$

$$E_2 = \frac{1}{2} \cdot (y_2^d(k) - y_2(k))^2 \tag{5-7}$$

where $y_1^d(k)$ and $y_2^d(k)$ are the desired outputs of the system; and $y_1(k) = O_1^{(3)}(k)$ and $y_2(k) = O_2^{(3)}(k)$ are the output of the RIN.

45000 data are chosen as the training data for each input variable $x_\phi(k)$ and each output variable $y_l(k)$, $l = 1, 2$. After 100 training *epochs*, all the related weights can achieve the desired outputs. Each epoch represents one cycle when all 45000 data sets are input to the scheme for processing.

The weights of the dynamic recurrent network will be updated by using the gradient method, as follows

$$\frac{\partial E_1}{\partial w_{\xi 1}^{(3)}} = -e_1(k) \cdot \frac{\partial y_1(k)}{\partial w_{\xi 1}^{(3)}} = -e_1(k) \cdot \frac{\partial O_1^{(3)}(k)}{\partial w_{\xi 1}^{(3)}},$$

$$= -e_1(k) \cdot O_\delta^{(2)}(k), (\delta = 1, 2 \ldots 5) \tag{5-8}$$

$$\frac{\partial E_2}{\partial w_{\xi 2}^{(3)}} = -e_2(k) \cdot \frac{\partial y_2(k)}{\partial w_{\xi 2}^{(3)}} = -e_2(k) \cdot \frac{\partial O_2^{(3)}(k)}{\partial w_{\xi 2}^{(3)}},$$

$$= -e_2(k) \cdot O_\delta^{(2)}(k), (\delta = 6, 7 \ldots 10) \tag{5-9}$$

$$\frac{\partial E_1}{\partial w_\delta^{(2)}} = -e_1(k) \cdot \frac{\partial O_1^{(3)}(k)}{\partial O_\delta^{(2)}(k)} \cdot \frac{\partial O_\delta^{(2)}(k)}{\partial w_\delta^{(2)}},$$

$$= -e_1(k) \cdot w_{\xi 1}^{(3)} \cdot \varphi_\delta^{(\phi)}(k), (\delta = 1, 2 \ldots 5) \tag{5-10}$$

$$\frac{\partial E_2}{\partial w_\delta^{(2)}} = -e_2(k) \cdot \frac{\partial O_2^{(3)}(k)}{\partial O_\delta^{(2)}(k)} \cdot \frac{\partial O_\delta^{(2)}(k)}{\partial w_\delta^{(2)}},$$

$$= -e_2(k) \cdot w_{\xi 2}^{(3)} \cdot \varphi_\delta^{(\phi)}(k), (\delta = 6, 7 \ldots 10) \tag{5-11}$$

$$\frac{\partial E_1}{\partial w_{\phi\delta}^{(1)}} = -\sum_{\delta=1}^{5} e_1(k) \cdot \frac{\partial O_1^{(3)}(k)}{\partial O_\delta^{(2)}(k)} \cdot \frac{\partial O_\delta^{(2)}(k)}{\partial w_{\phi\delta}^{(1)}}$$

$$= -\sum_{\delta=1}^{5} e_1(k) \cdot w_{\xi 1}^{(3)} \cdot \varpi_{\phi\delta}^{(\phi)}(k), (\xi = 1, 2 \ldots 5) \tag{5-12}$$

$$\frac{\partial E_2}{\partial w_{\phi\delta}^{(1)}} = -\sum_{\delta=6}^{10} e_2(k) \cdot \frac{\partial O_2^{(3)}(k)}{\partial O_\delta^{(2)}(k)} \cdot \frac{\partial O_\delta^{(2)}(k)}{\partial w_{\phi\delta}^{(1)}}$$

$$= -\sum_{\delta=6}^{10} e_2(k) \cdot w_{\xi 2}^{(3)} \cdot \varpi_{\phi\delta}^{(\phi)}(k), (\xi = 1, 2 \ldots 5) \tag{5-13}$$

$$\frac{\partial E_1}{\partial b_\delta^{(2)}} = -e_1(k) \cdot \frac{\partial O_1^{(3)}(k)}{\partial O_\delta^{(2)}(k)} \cdot \frac{\partial O_\delta^{(2)}(k)}{\partial b_\delta^{(2)}},$$

$$= -e_1(k) \cdot w_{\xi 1}^{(3)} \cdot \vartheta_\delta^{(\phi)}(k), (\delta = 1, 2 \ldots 5) \tag{5-14}$$

$$\frac{\partial E_2}{\partial b_\delta^{(2)}} = -e_2(k) \cdot \frac{\partial O_2^{(3)}(k)}{\partial O_\delta^{(2)}(k)} \cdot \frac{\partial O_\delta^{(2)}(k)}{\partial b_\delta^{(2)}},$$

$$= -e_2(k) \cdot w_{\xi 2}^{(3)} \cdot \vartheta_\delta^{(\phi)}(k), (\delta = 6, 7 \ldots 10) \tag{5-15}$$

$$\frac{\partial E_\delta}{\partial b_1^{(3)}} = -e_1(k)\cdot\frac{\partial y_1(k)}{\partial b_1^{(3)}} = -e_1(k)\cdot\frac{\partial O_1^{(3)}(k)}{\partial b_1^{(3)}},$$

$$= -e_1(k) \tag{5-16}$$

$$\frac{\partial E_\delta}{\partial b_2^{(3)}} = -e_2(k)\cdot\frac{\partial y_2(k)}{\partial b_2^{(3)}} = -e_2(k)\cdot\frac{\partial O_2^{(3)}(k)}{\partial b_2^{(3)}},$$

$$= -e_2(k) \tag{5-17}$$

$$\varphi_\delta^{(\phi)}(k) = \frac{\partial O_\delta^{(2)}(k)}{\partial w_\delta^{(2)}} = f'(net_\delta(k))\cdot(O_\delta^{(2)}(k-1) + w_\delta^{(2)}\cdot\frac{\partial O_\delta^{(2)}(k-1)}{\partial w_\delta^{(2)}})$$

$$= f'(net_\delta(k))\cdot(O_\delta^{(2)}(k-1) + w_\delta^{(2)}\cdot\varphi_\delta^{(\phi)}(k-1)) \tag{5-18}$$

$$\varpi_{\phi\delta}^{(\phi)}(k) = \frac{\partial O_\delta^{(2)}(k)}{\partial w_{\phi\delta}^{(1)}} = f'(net_\delta(k))\cdot(x_\phi(k) + w_\delta^{(2)}\cdot\frac{\partial O_\delta^{(2)}(k-1)}{\partial w_{\phi\delta}^{(1)}})$$

$$= f'(net_\delta(k))\cdot(x_\phi(k) + w_\delta^{(2)}\cdot\varpi_{\phi\delta}^{(\phi)}(k-1)) \tag{5-19}$$

$$\vartheta_\delta^{(\phi)}(k) = \frac{\partial O_\delta^{(2)}(k)}{\partial b_\delta^{(2)}} = f'(net_\delta(k)) \tag{5-20}$$

where $e_1(k) = y_1^d(k) - y_1(k)$, $e_2(k) = y_2^d(k) - y_2(k)$; $f'(net_\delta(k)) = O_\delta^{(2)}\cdot(1 - O_\delta^{(2)})$. Note that equations (5-18) to (5-20) are nonlinear dynamic recurrent equations which can be obtained recurrently using the initial conditions $\varphi_\delta^{(2)}(0) = 0$ and $\varpi_{\phi\delta}^{(\phi)}(0) = 0$.

Thus, the weights of the RIN can be updated by

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \eta_{BP}\cdot(-\frac{\partial E}{\partial \boldsymbol{w}}) \tag{5-21}$$

where $\boldsymbol{w}$ are the RIN weights including $w_{\phi\delta}^{(1)}$, $w_{\delta}^{(2)}$, $w_{\xi 1}^{(3)}$, $w_{\xi 2}^{(3)}$; η_{BP} is the adaptive learning rate.

5.3 The real-time recurrent learning (RTRL) with the LSE method

Consider a network consisting of a total of N neurons with M external input connections. Let $\boldsymbol{x}(k)$ denote the M-by-1 external input vector applied to the network at the discrete time instant k, and let $\boldsymbol{O}(k+1)$ denote the corresponding N-by-1 vector of individual neuron outputs generated one step later at time $k+1$. The input vector $\boldsymbol{x}(k)$ and one-step delayed output vector $\boldsymbol{O}(k)$ are concatenated to form the $(M+N)$-by-1 vector. Let A denote the set of indices i, $i = 1, 2...M$, for which $x_i(k)$ is an external input, and let B denote the set of indices j, $j = 1, 2...N$, for which $O_j(k)$ is the output of a neuron.

We may distinguish two distinct layers in the network, namely, a *concatenated input-output layer* and a *processing layer*, which is illustrated in Figure 5.3 for $M = 3$ and $N = 6$. The three input variables will be assumed the same as those as illustrated in Figure 5.2, that is, $\boldsymbol{x}(k)$ $\{\theta_d(k), \varepsilon_d(k), vol(k)\}$. N of the feedback connections are in actual *self-feedback* links. Because the number of the output neurons $\boldsymbol{O}(k+1)$ equals the number of one-step delayed output vector $\boldsymbol{O}(k)$, we denote the indices also by r, $r = 1, 2...N$. Moreover, the indices l, $l = 1, 2$, are denoted for which $y_l(k+1)$ are the two outputs of the developed recurrent network (namely, the predicted data sets $\{\theta(k+1), \varepsilon(k+1)\}$.

Three kinds of weights are included in this network, w_{ij}, w_{rj}, and w_{jl}. They represent the weights of the connections between these M external inputs and N hidden layer neurons, the

feedback links between the hidden layer neurons, and the connections between the hidden layer neurons and the two output neurons, respectively.

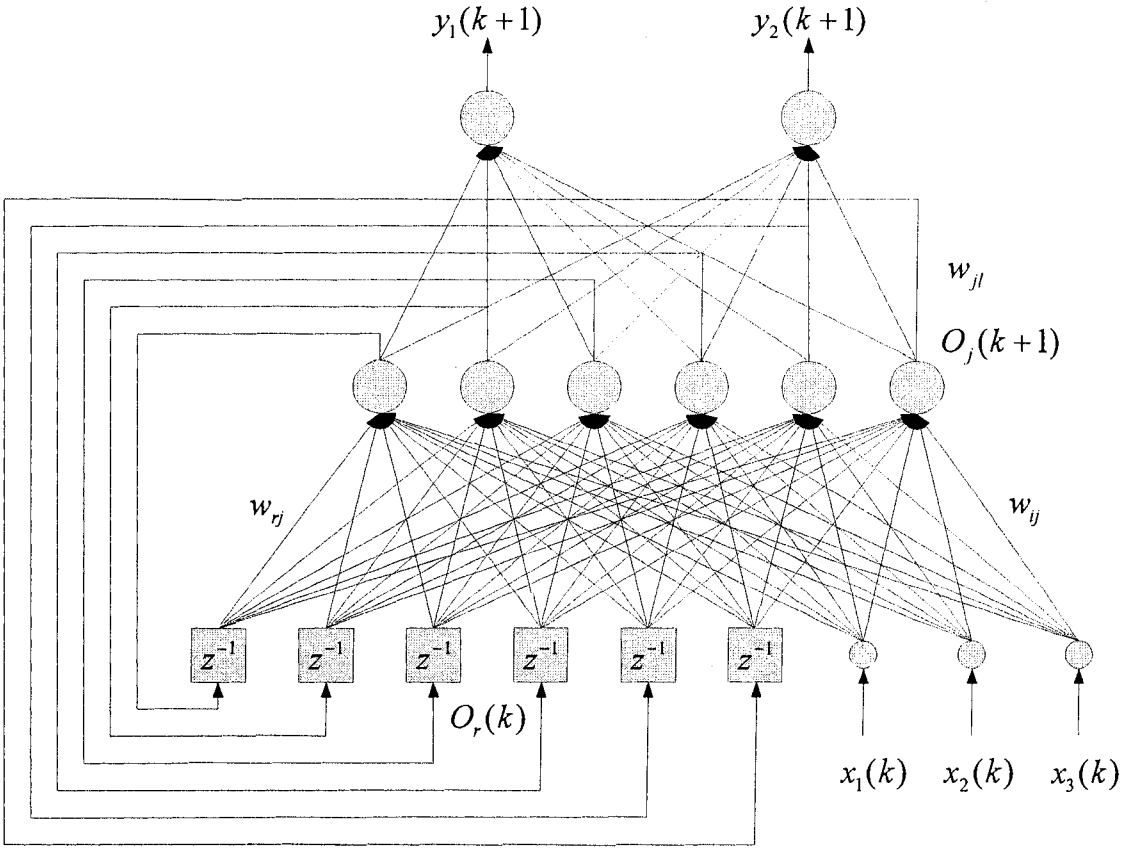


Figure 5.3. The architecture of a real-time recurrent network.

The internal activation of neuron j at time k is represented by

$$p_j(k) = \sum_{i=1}^{3} w_{ij}(k) \cdot x_i(k) + \sum_{r=1}^{6} w_{rj}(k) \cdot O_r(k) \tag{5-22}$$

At the following time step k+1, the output of neuron j is computed by passing $p_j(k)$ through the nonlinearity $\sigma(\cdot)$, or

$$O_j(k+1) = \sigma(p_j(k)) \tag{5-23}$$

It should be noted that as indicated in equation (5-23), the external input vector $\boldsymbol{x}(k)$ $\{\theta_d(k), \varepsilon_d(k), vol(k)\}$ at time k does not influence the output of any neuron in the network until time k+1.

With these obtained dynamic equations, the next task is to derive the RTRL technique for system training. RTRL is a modified steepest gradient algorithm, which aims to match the outputs of certain neurons in the processing layer to desired values at specific time instants.

Due to the desired (target) data sets being $\{\theta_d(k), \varepsilon_d(k)\}$, we define the time-varying error $e_\theta(k)$ and $e_\varepsilon(k)$:

$$e_\theta(k) = \theta_d(k) - \theta(k) \tag{5-24}$$

$$e_\varepsilon(k) = \varepsilon_d(k) - \varepsilon(k) \tag{5-25}$$

The *instantaneous* sum of squared error at time k will be defined as

$$E(k) = \frac{1}{2}\left[(\theta_d(k) - \theta(k))^2 + (\varepsilon_d(k) - \varepsilon(k))^2\right] \tag{5-26}$$

The objective is to minimize the overall cost function $E_{total}(k)$ over all time instants; that is,

$$E_{total}(k) = \sum_k E(k) \tag{5-27}$$

To accomplish this objective we compute the steepest gradients of the cost function,

$$\nabla_{w_{ij}} E_{total} = \frac{\partial E_{total}}{\partial w_{ij}} = \sum_k \frac{\partial E(k)}{\partial w_{ij}} = \sum_k \nabla_{w_{ij}} E(k) \tag{5-28}$$

$$\nabla_{w_{rj}} E_{total} = \frac{\partial E_{total}}{\partial w_{rj}} = \sum_k \frac{\partial E(k)}{\partial w_{rj}} = \sum_k \nabla_{w_{rj}} E(k) \tag{5-29}$$

where $\nabla_{w_{ij}} E(k)$ and $\nabla_{w_{rj}} E(k)$ are the gradient of $E(k)$ with respect to the weights w_{ij} and w_{rj}, respectively.

In order to develop the RTRL algorithm that can be used to train the recurrent network in *real time*, we will use an instantaneous *estimate* of the gradient, $\nabla_{w_{ij}} E(k)$ and $\nabla_{w_{rj}} E(k)$ at each time step k, which will result in an approximation to the steepest gradient method. Figure 5.4 illustrates the error propogation within the network.

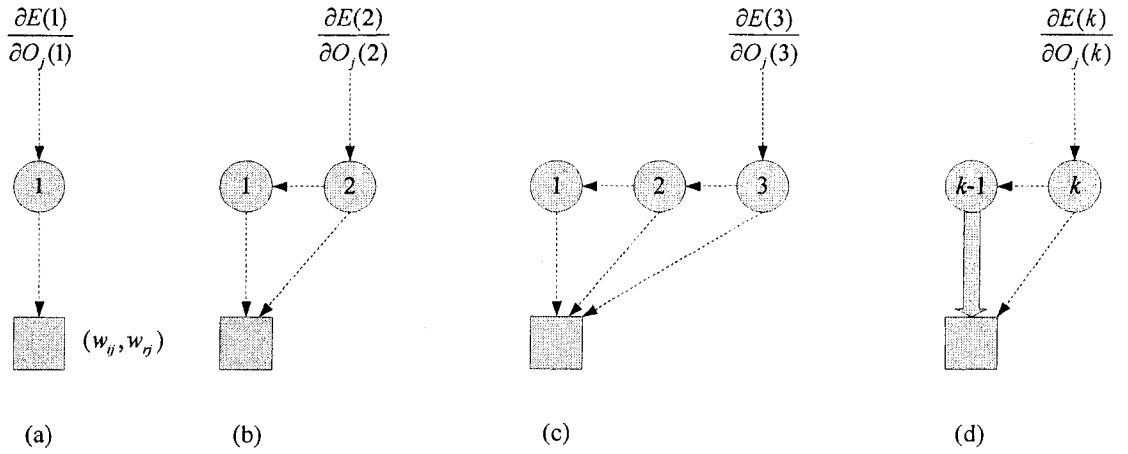


Figure 5.4. Error-propagation networks at different time steps: (a) $i = 1$; (b) $i = 2$; (c) $i = 3$; (d) a general situation where the thick arrow represents $\partial^+ O_j(k-1)/\partial w_{ij}$ and $\partial^+ O_j(k-1)/\partial w_{rj}$.

If $k = 1$, the error-propagation is as shown in Figure 5.4(a) and we have:

$$\frac{\partial^+ O_j(1)}{\partial w_{ij}} = \frac{\partial O_j(1)}{\partial w_{ij}} \text{ and } \frac{\partial^+ E(1)}{\partial w_{ij}} = \frac{\partial E(1)}{\partial O_j(1)} \cdot \frac{\partial^+ O_j(1)}{\partial w_{ij}} \tag{5-30}$$

$$\frac{\partial^+ O_j(1)}{\partial w_{rj}} = \frac{\partial O_j(1)}{\partial w_{rj}} \text{ and } \frac{\partial^+ E(1)}{\partial w_{rj}} = \frac{\partial E(1)}{\partial O_j(1)} \cdot \frac{\partial^+ O_j(1)}{\partial w_{rj}} \tag{5-31}$$

If $k = 2$, the error-propagation network is as illustrated in Figure 5.4(b) and we have:

$$\frac{\partial^+ O_j(2)}{\partial w_{ij}} = \frac{\partial O_j(2)}{\partial w_{ij}} + \frac{\partial O_j(2)}{\partial O_j(1)} \cdot \frac{\partial O^+{}_j(1)}{\partial w_{ij}} \quad \text{and}$$

$$\frac{\partial^+ E(2)}{\partial w_{ij}} = \frac{\partial E(2)}{\partial O_j(2)} \cdot \frac{\partial^+ O_j(2)}{\partial w_{ij}} \tag{5-32}$$

$$\frac{\partial^+ O_j(2)}{\partial w_{rj}} = \frac{\partial O_j(2)}{\partial w_{rj}} + \frac{\partial O_j(2)}{\partial O_j(1)} \cdot \frac{\partial O^+{}_j(1)}{\partial w_{rj}} \quad \text{and}$$

$$\frac{\partial^+ E(2)}{\partial w_{rj}} = \frac{\partial E(2)}{\partial O_j(2)} \cdot \frac{\partial^+ O_j(2)}{\partial w_{rj}} \tag{5-33}$$

If $k = 3$, the error-propagation network is as shown in Figure 5.4(c) and we have:

$$\frac{\partial^+ O_j(3)}{\partial w_{ij}} = \frac{\partial O_j(3)}{\partial w_{ij}} + \frac{\partial O_j(3)}{\partial O_j(2)} \cdot \frac{\partial O^+{}_j(2)}{\partial w_{ij}} \quad \text{and}$$

$$\frac{\partial^+ E(3)}{\partial w_{ij}} = \frac{\partial E(3)}{\partial O_j(3)} \cdot \frac{\partial^+ O_j(3)}{\partial w_{ij}} \tag{5-34}$$

$$\frac{\partial^+ O_j(3)}{\partial w_{rj}} = \frac{\partial O_j(3)}{\partial w_{rj}} + \frac{\partial O_j(3)}{\partial O_j(2)} \cdot \frac{\partial O^+{}_j(2)}{\partial w_{rj}} \quad \text{and}$$

$$\frac{\partial^+ E(3)}{\partial w_{rj}} = \frac{\partial E(3)}{\partial O_j(3)} \cdot \frac{\partial^+ O_j(3)}{\partial w_{rj}} \tag{5-35}$$

In general, for the error-propagation at time instant k, we have:

$$\frac{\partial^+ O_j(k)}{\partial w_{ij}} = \frac{\partial O_j(k)}{\partial w_{ij}} + \frac{\partial O_j(k)}{\partial O_j(k-1)} \cdot \frac{\partial O^+{}_j(k-1)}{\partial w_{ij}} \quad \text{and}$$

$$\frac{\partial^+ E(k)}{\partial w_{ij}} = \frac{\partial E(k)}{\partial O_j(k)} \cdot \frac{\partial^+ O_j(k)}{\partial w_{ij}} \tag{5-36}$$

$$\frac{\partial^+ O_j(k)}{\partial w_{rj}} = \frac{\partial O_j(k)}{\partial w_{rj}} + \frac{\partial O_j(k)}{\partial O_j(k-1)} \cdot \frac{\partial O^+{}_j(k-1)}{\partial w_{rj}} \quad \text{and}$$

$$\frac{\partial^+ E(k)}{\partial w_{rj}} = \frac{\partial E(k)}{\partial O_j(k)} \cdot \frac{\partial^+ O_j(k)}{\partial w_{rj}} \tag{5-37}$$

Where $\frac{\partial O^+{}_j(k-1)}{\partial w_{ij}}$ and $\frac{\partial O^+{}_j(k-1)}{\partial w_{rj}}$ are already available from the calculation at the previous time instant. Figure 5.4(d) shows this general situation, where the thick arrow represents $\frac{\partial O^+{}_j(k-1)}{\partial w_{ij}}$ and $\frac{\partial O^+{}_j(k-1)}{\partial w_{rj}}$, which are already available at the time instant k-1.

Therefore, by trying to minimize each individual $E(k)$, we can recursively find the gradients $\frac{\partial E^+(k)}{\partial w_{ij}}$ and $\frac{\partial E^+(k)}{\partial w_{rj}}$ at each time instant; there is no need to wait until the end of the presented sequence. Since this is an approximation of the original gradient formula:

$$\Delta w_{ij} = -\eta_{ij} \frac{\partial E^+(k)}{\partial w_{ij}} \tag{5-38}$$

$$\Delta w_{rj} = -\eta_{rj} \frac{\partial E^+(k)}{\partial w_{rj}} \tag{5-39}$$

the learning rate of η_{ij} and η_{rj} in equations (5-38) and (5-39) should be kept small.

The weights w_{ij} and w_{rj} are updated in accordance with

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k) \tag{5-40}$$

$$w_{rj}(k+1) = w_{rj}(k) + \Delta w_{rj}(k) \tag{5-41}$$

The developed RIN is trained by a hybrid technique based on the RTRL and LSE. The parameters of the recurrent links and the connections between these M external inputs and N hidden layer neurons are trained by the use of the RTRL as discussed in the last section. The weights w_{jl} connected between the hidden layer neurons and the two output neurons in this network are fine-tuned by using the LSE method when the antecedent parameters (parameters in MFs) remain fixed.

Given g training data, equation (4-7) can be re-written as:

$$\begin{bmatrix} y_l^{d(1)} \\ y_l^{d(2)} \\ . \\ . \\ . \\ y_l^{d(g)} \end{bmatrix} = \begin{bmatrix} O_1^{(1)} \cdot w_{11} + O_2^{(1)} \cdot w_{21} + ... + O_j^{(1)} \cdot w_{jl} \\ O_1^{(2)} \cdot w_{11} + O_2^{(2)} \cdot w_{21} + ... + O_j^{(2)} \cdot w_{jl} \\ . \\ . \\ . \\ O_1^{(g)} \cdot w_{11} + O_2^{(g)} \cdot w_{21} + ... + O_j^{(g)} \cdot w_{jl} \end{bmatrix}$$

$$= \begin{bmatrix} O_1^{(1)} & O_2^{(1)} & \cdots & O_j^{(1)} \\ O_1^{(2)} & O_2^{(2)} & \cdots & O_j^{(2)} \\ & \vdots & \vdots & \\ O_1^{(g)} & O_2^{(g)} & \cdots & O_j^{(g)} \end{bmatrix} \cdot \begin{bmatrix} w_{11} \\ w_{21} \\ \vdots \\ w_{jl} \end{bmatrix} \tag{5-42}$$

where $O_j^{(g)}$ ($j = 1, 2, ..., N$; $N = 6$) are the outputs of neurons in the second layer.

$$\varphi^T(k) = [O_1^{(k)} \quad O_2^{(k)} \quad ... \quad O_j^{(k)}] \tag{5-43}$$

$$\boldsymbol{\Phi} = [\varphi^T(1) \quad \varphi^T(2) \quad ... \quad \varphi^T(g)]^T \tag{5-44}$$

$$\boldsymbol{w}_l = [w_{11} \quad w_{21} \quad \ldots \quad w_{jl}]^T \tag{5-45}$$

$$\boldsymbol{Y}_l = [y_l^{d(1)} \quad y_l^{d(2)} \quad \ldots \quad y_l^{d(g)}]^T \tag{5-46}$$

where $k = 1, 2 \ldots g$.

To simplify notations, equation (5-42) will be expressed in a matrix-vector form:

$$\boldsymbol{Y}_l = \boldsymbol{\Phi} \cdot \boldsymbol{w}_l \tag{5-47}$$

Then equations (4-13) can be represented by

$$\boldsymbol{w}_l^* = (\boldsymbol{\Phi}^T \cdot \boldsymbol{\Phi})^{-1} \cdot \boldsymbol{\Phi}^T \cdot \boldsymbol{Y}_l, \tag{5-48}$$

where $(\boldsymbol{\Phi}^T \cdot \boldsymbol{\Phi})^{-1} \cdot \boldsymbol{\Phi}^T$ is the pseudo inverse of $\boldsymbol{\Phi}$ if $\boldsymbol{\Phi}^T \cdot \boldsymbol{\Phi}$ is nonsingular.

Let the zth row vector of matrix $\boldsymbol{\Phi}^T$ defined in equation (5-47) be $\varphi^T(z)$ and the zth element of $\boldsymbol{Y}_l$ be $y_l^{d(z)}$; adding a forgetting factor λ ($\lambda = 0.9998$ in this case), then $\boldsymbol{w}_l^*$ can be calculated recursively using the following formulas:

$$\boldsymbol{w}_l^*(z+1) = \boldsymbol{w}_l^*(z) + H_{RTRL}(z+1) \cdot (y_l^{d(z)} - \varphi^T(z+1) \cdot \boldsymbol{w}_l^*(z)) \tag{5-49}$$

$$H_{RTRL}(z+1) = S_{RTRL}(z+1) \cdot \varphi(z+1)$$

$$= S_{RTRL}(z) \cdot \varphi(z+1) \cdot (\lambda + \varphi^T(z+1) \cdot S_{RTRL}(z) \cdot \varphi(z+1))^{-1} \tag{5-50}$$

$$S_{RTRL}(z+1) = \frac{1}{\lambda}[S_{RTRL}(z) - S_{RTRL}(z) \cdot \varphi(z+1)$$

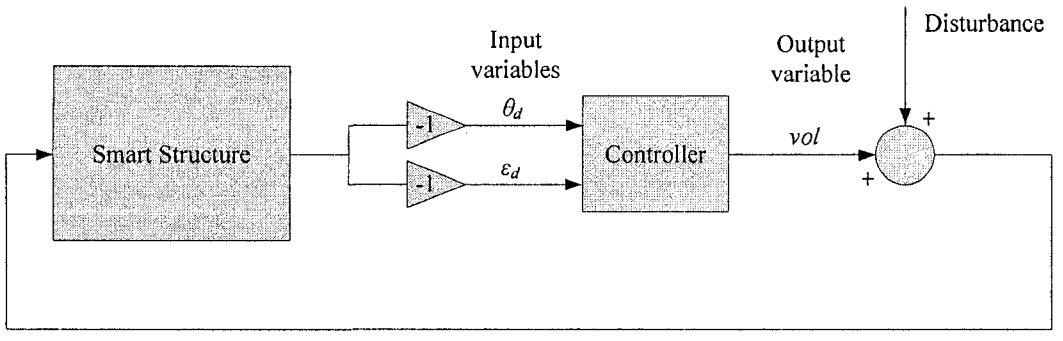
$$\cdot(\lambda+\varphi^T(z+1)\cdot S_{RTRL}(z)\cdot\varphi(z+1))^{-1}\cdot\varphi^T(z+1)\cdot S_{RTRL}(z)]$$

$$=\frac{1}{\lambda}[(I-H_{RTRL}(z+1)\cdot\varphi^T(z+1))\cdot S_{RTRL}(z)] \tag{5-51}$$

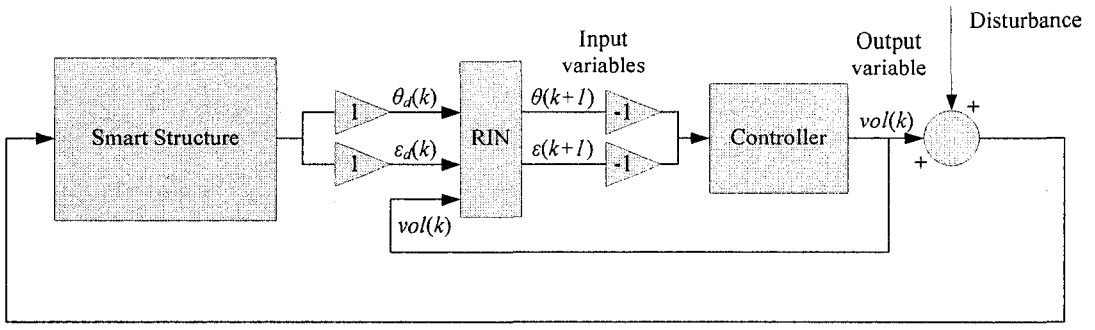
5.4 Control performance comparison based on training techniques

5.4.1 Implementation of the RIN in the experiment

Figure 5.5(a) shows the block diagram of the flexible structure control system without RIN. θ and ε are the control input variables, and *vol* is the control output variable. The block 'controller' is the NF controller as developed in Chapter 3. Figure 5.5(b) illustrates the block diagram of the experiment when the RIN is applied. The difference between the systems in Figures 5.5(a) and 5.5(b) is related to the RIN which is used to automatically estimate the model of the tested flexible structure. In training the RIN, the desired data sets are those of the control results $\{\theta_d(k),\varepsilon_d(k)\}$ without RIN. Then when the actual outputs of RIN $\{\theta(k+1),\varepsilon(k+1)\}$ are similar to those of $\{\theta_d(k),\varepsilon_d(k)\}$, it is assumed that this RIN has identified the model of tested flexible structure.



(a)



(b)

Figure 5.5. (a) The flexible structure control system without RIN; (b) The flexible structure control system with RIN.

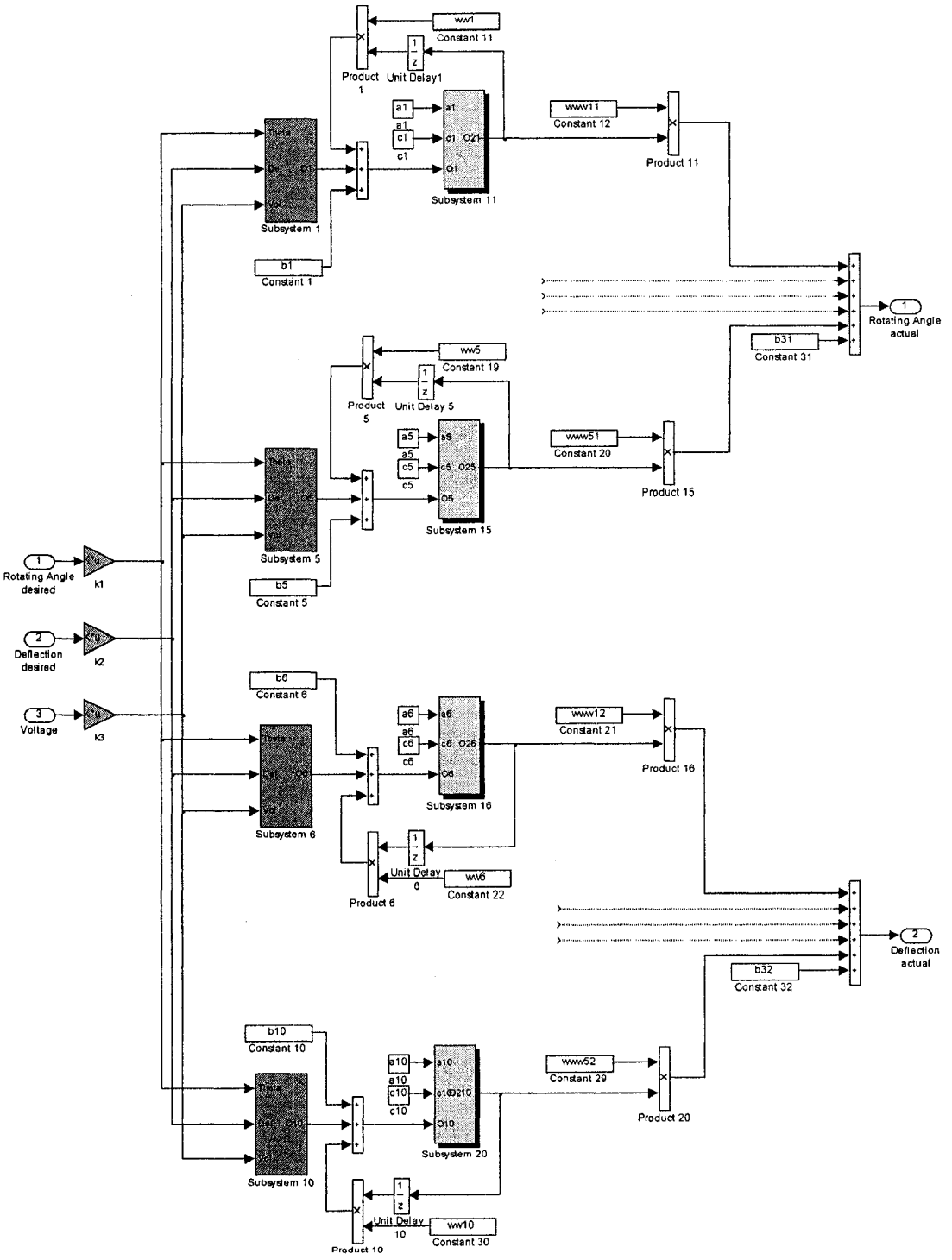


Figure 5.6. The RIN structure by using MATLAB blocks.

Figure 5.6 shows the implemented RIN in SIMULINK in experiment. The connections between the first layer and the second layer of this network are described by the first ten subsystems describe as shown in Figure 5.7. The unit continuous nonlinear sigmoid functions, $f(x) = \frac{1}{1+e^{-a_\delta(x_\delta - c_\delta)}}$, $\delta = 1, 2...10$, are represented by and the last ten subsystems as shown in Figure 4.6(a). Furthermore, it is assumed that $a_\delta = 1$, and $c_\delta = 0$ in this experiment. In Figure 5.6, $w_{\phi\delta}^{(1)}$ are replaced by the variables $w(\phi q)$, $\phi = 1, 2, 3$, $\delta = q = 1, 2...10$; $w_\delta^{(2)}$ are replaced by the variables $ww(\delta)$; $w_{\xi 1}^{(3)}$ and $w_{\xi 1}^{(3)}$ are replaced by $www(\xi 1)$ and $www(\xi 2)$, $\xi = 1, 2...5$. In the following tests, the sampling time is 0.001 s.

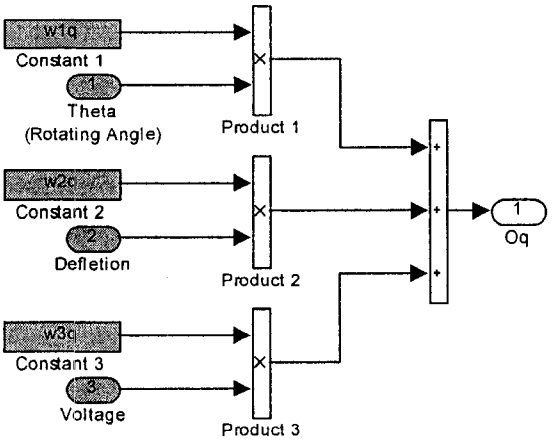


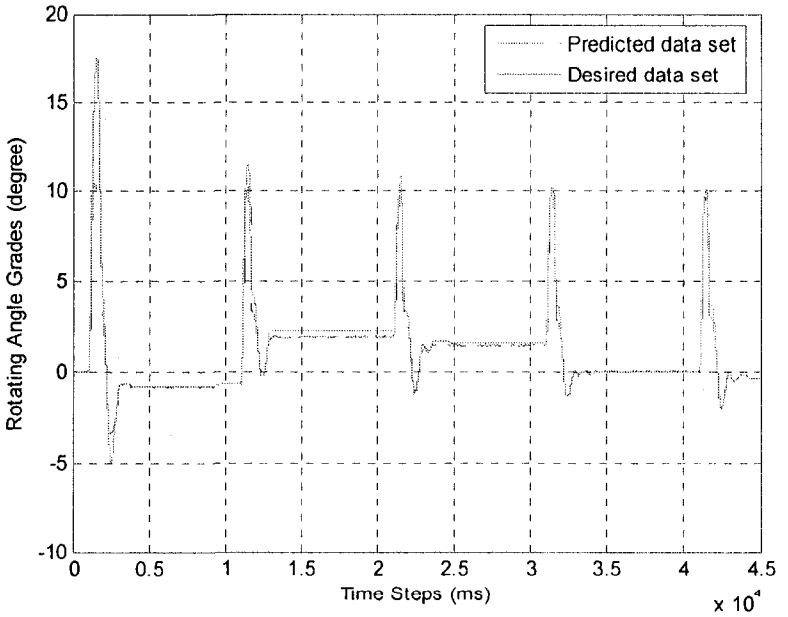
Figure 5.7. The connections between the first layer and the second layer.

5.4.2 The predicted results based on the gradient training algorithm

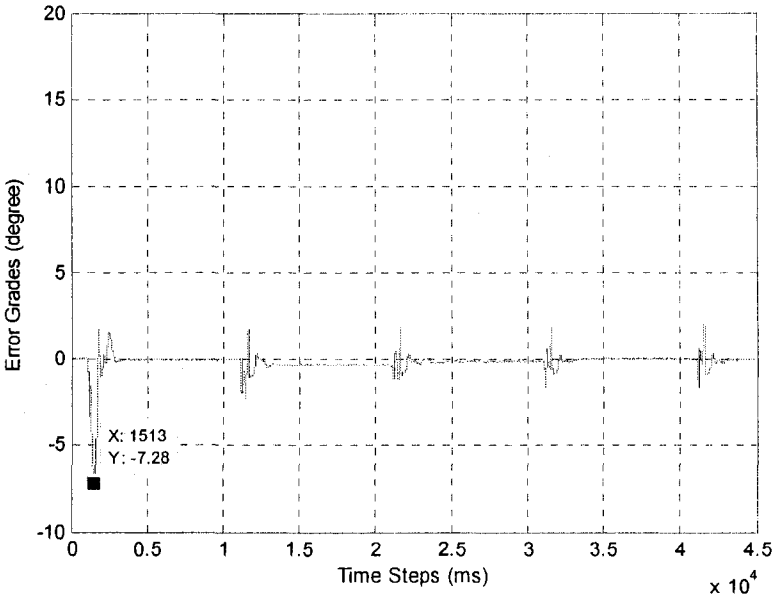
In this test, based on limitations of the DSP board, the sampling frequency is selected as 1000 Hz, or the sampling time is 0.001 s. The RIN is trained based on the classical hybrid training method of the gradient algorithm. As stated in section 5.4.1, if the actual outputs of RIN

$\{\theta(k+1), \varepsilon(k+1)\}$ are similar with the desired control results $\{\theta_d(k), \varepsilon_d(k)\}$, it is believed that the RIN has identified the model of the flexible structure. To improve training efficiency, the *TS0-NF controller* as developed in Chapter 3 is applied in this RIN experiment. Z-shaped MFs, triangle MFs, and S-shaped MFs are applied in the second layer of the NF controller for each input variable; all of the antecedent parameters remain constant in training to improve learning efficiency.

The desired data set $\theta_d(k)$ and the predicted data set $\theta(k+1)$ of the Rotating Angle are shown in Figure 5.8(a), and the predicted error $\theta(k+1) - \theta_d(k)$ of Rotating Angle is shown in Figure 5.8(b). It is seen from Figure 5.8(b) that the largest absolute value of the Rotating Angle error is 7.28 *deg.*



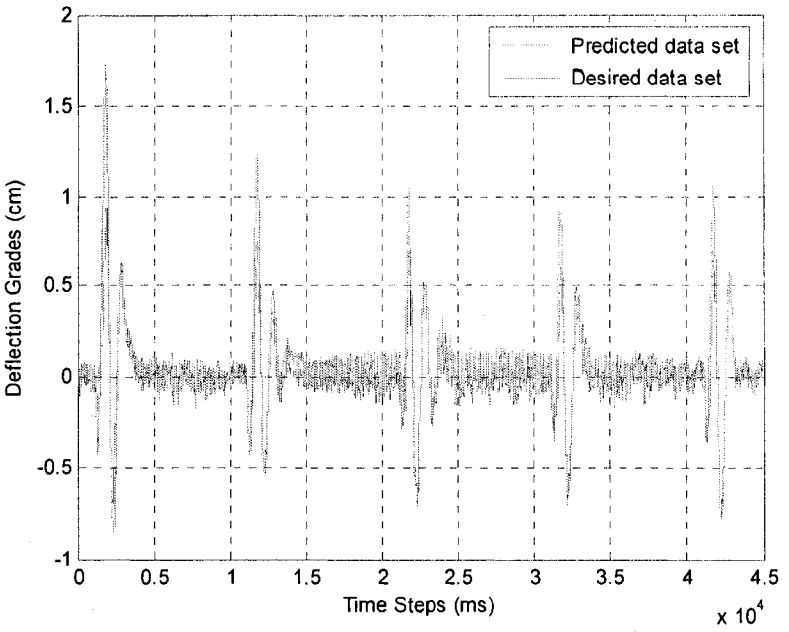
(a)



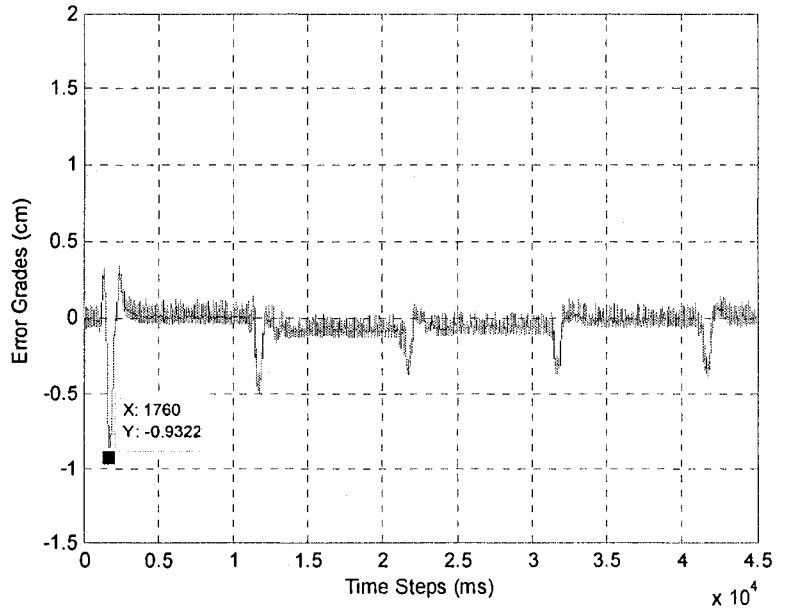
(b)

Figure 5.8. Test results based on gradient training algorithm. (a) The desired data set (solid line) and the predicted data set (dashed line) of the Rotating Angle; (b) the predicted error of Rotating Angle.

The desired data set $\varepsilon_d(k)$ and the predicted data set $\varepsilon(k+1)$ of the Deflection and the predicted error $\varepsilon(k+1)-\varepsilon_d(k)$ are as shown in Figure 5.9(a) and 5.9(b), respectively. It is clear that the largest Deflection error is 0.932 *cm* in Figure 5.9(b). There are several small undershoots in Figure5.9 (b) when each disturbance is applied. The average undershoot value is around -0.3 *cm*.



(a)

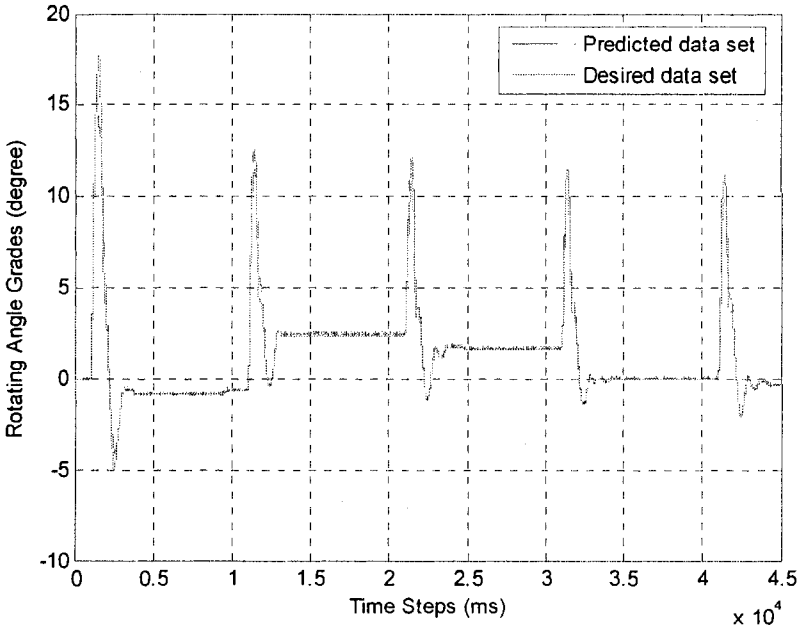


(b)

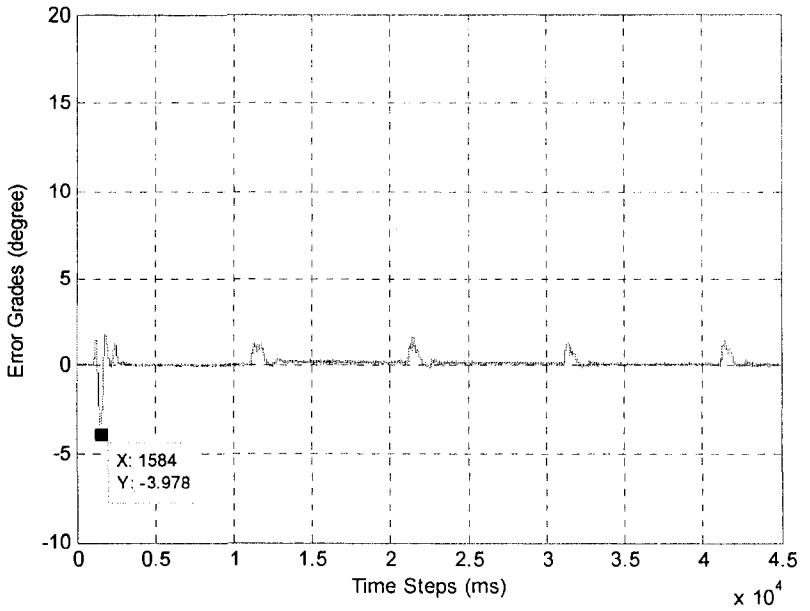
Figure 5.9. Test results based on gradient training algorithm. (a) The desired data set (solid line) and the predicted data set (dashed line) of the Deflection; (b) the predicted error of Deflection.

5.4.3 The predicted results based on RTRL with the LSE training method

In this test, the RIN is trained based on the adopted hybrid training method of RTRL and LSE. The desired data set and the predicted data set of the Rotating Angle are shown in Figure 5.10(a). The predicted error of Rotating Angle is shown in Figure 5.10(b), where the largest Rotating Angle error is 3.978 *deg.*



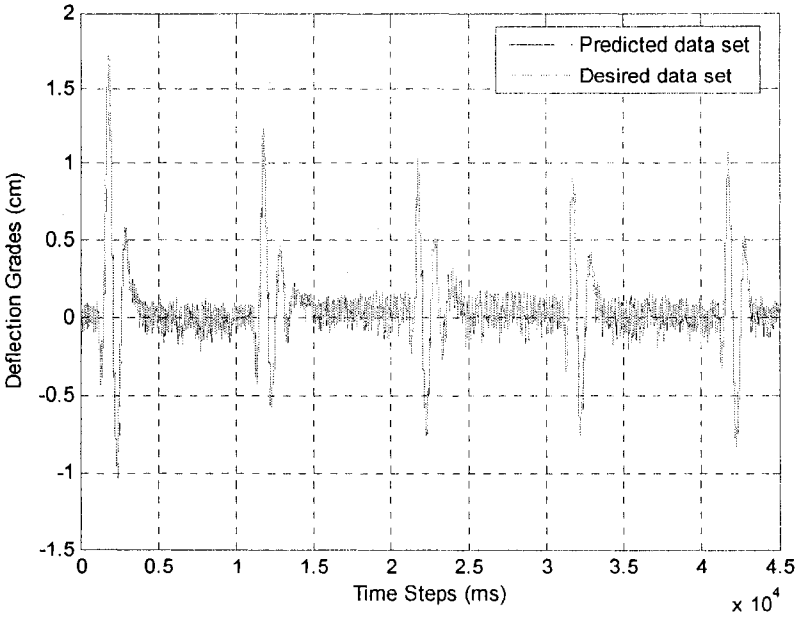
(a)



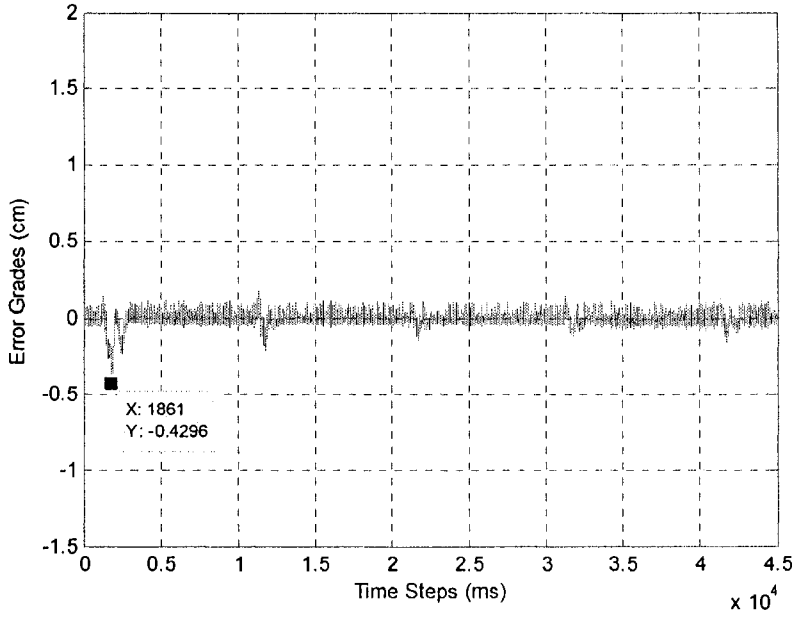
(b)

Figure 5.10. (a) The desired data set and the predicted data set of the Rotating Angle; (b) the predicted error of Rotating Angle by using RTRL with LSE.

The desired data set and the predicted data set of the Deflection are shown in Figure 5.11(a). The predicted error of Deflection is shown in the Figure 5.11(b), and the largest Deflection error is 0.429 *cm*. It is seen from Figure 5.11(b) that there are several small undershoots with the average value is around -0.1 *cm*, when each disturbance is introduced,



(a)



(b)

Figure 5.11. (a) The desired data set and the predicted data set of the Deflection; (b) the predicted error of Deflection by using RTRL with LSE.

From figure 5.8 to figure 5.11, it is seen that both RINs can effectively capture the system's dynamic behaviors. Compared to the predicted results by using the gradient method (Figure 5.8

and Figure 5.9) and RTRL with LSE method (Figure 5.10 and Figure 5.11), RTRL with LSE method has a better prediction on both the Rotating Angle of the rigid beam and the Deflection of the flexible beam because of its adaptive network architecture in which more dynamic recurrent feedback links are connected to each neuron in the second layer; and because of its effective training process which simply minimizes the instantaneous error squared, thereby reducing the storage requirement to the minimum possible.

Figure 5.12 shows the root mean squared error (RMSE) curves for the NN controller introduced in Chapter 4 and two designed RINs with the proposed NF controller. 45000 data are chosen as the training data for each input variable and each output variable.

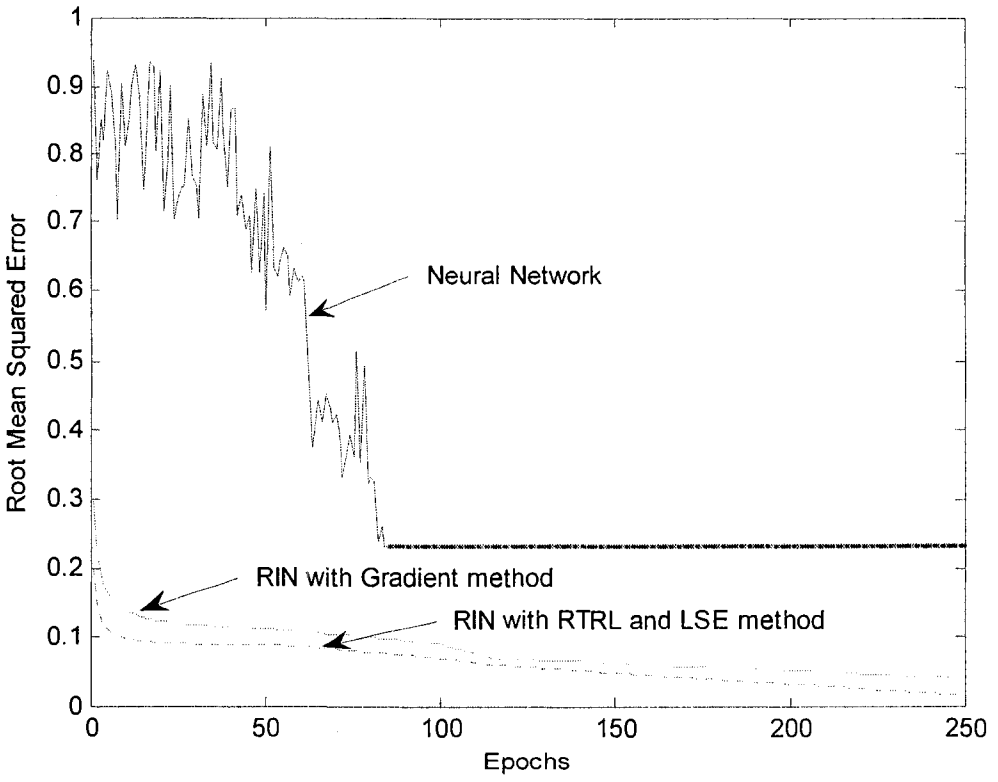


Figure 5.12. RMSE curves for the NN and RINs.

In this figure, it is found that the training convergence of NN controller is slower, and the convergence of the RIN with the novel hybrid training algorithm (RTRL and LSE) is the fastest in performance.

5.4.4 The control results based on the gradient training algorithm

The experiment result by using the designed RIN trained by the gradient method, with the Disturbance sequence signal (as shown in Figure 4.3) is illustrated in Figure 5.13. The overshoot is 0.902 *cm*, the largest negative value is -0.35 *cm*, and the settling time is about 2 *s*.

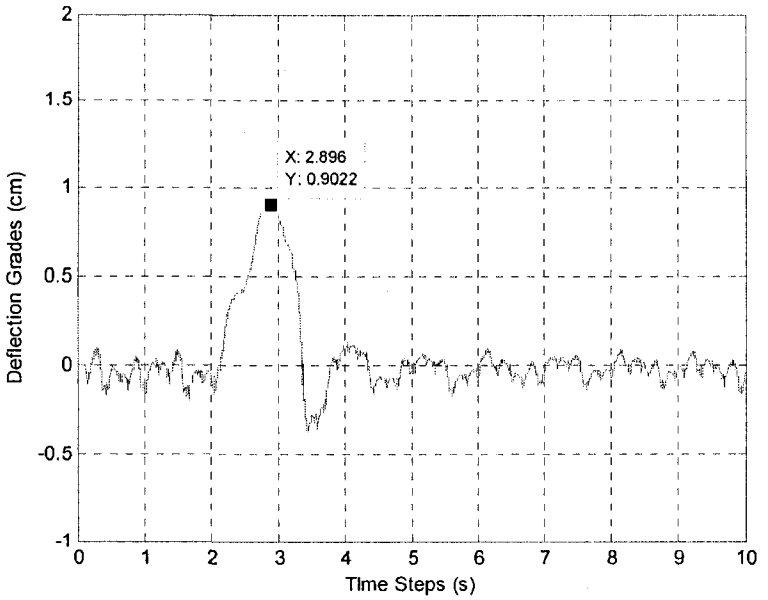


Figure 5.13. The control result of Deflection.

5.4.5 The control results based on RTRL with the LSE training method

The experiment result by using RTRL with LSE method with the Disturbance sequence signal (as shown in Figure 4.3) is in the Figure 5.14. The overshoot is 0.331 *cm*, and the largest negative value is still near -0.35 *cm*. The settling time is less than 2 *s*.

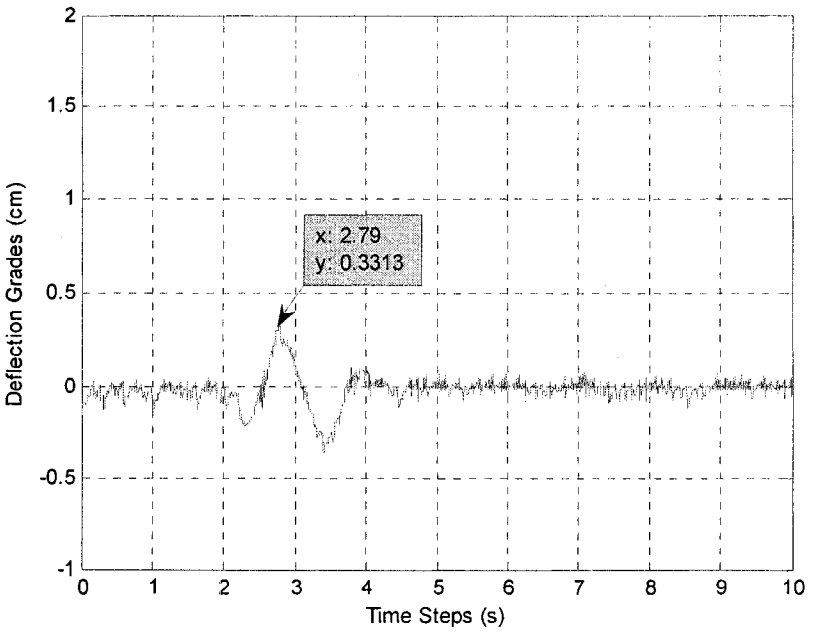


Figure 5.14. The control result of Deflection.

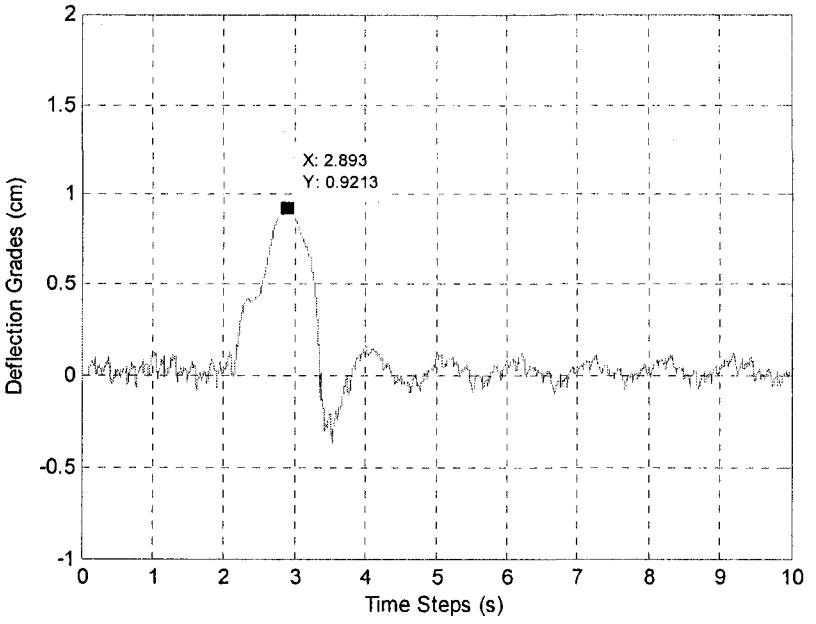
Compare the control results of Deflection. Firstly, the control results with RIN trained by either gradient or RTRL with LSE methods are better than those without RIN. The reason is that the nonlinear dynamical equations of the tested flexible structure are derived by using Lagrangian method. Errors are introduced due to linearization. Secondly, the Deflection control by using RTRL with LSE outperforms the control based on gradient training. The error-propagation E_i in the RLRT with LSE approach is calculated at each time step instead of trying to minimize the overall error function $E_{total}(k) = \sum_k E(k)$ at the end of each sequence as in gradient algorithm; therefore, the control based on RTRL with LSE is much better in performance and faster in convergence.

5.5 Control of time-varying systems

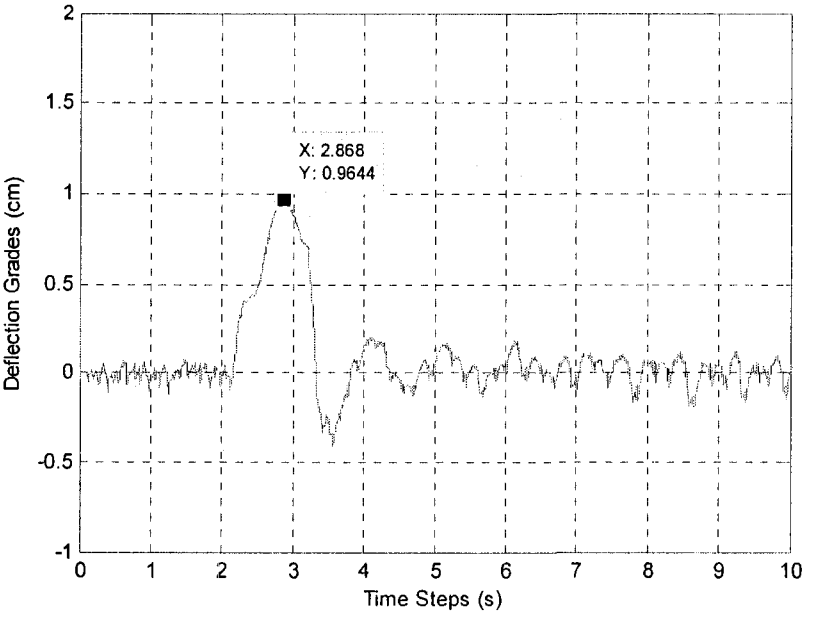
In this experiment, a couple of magnetic blocks (each one is 100g) are used to change system dynamics of the flexible beam to simulate the time-varying conditions. If the magnetic blocks are attached on the flexible beam and/or their positions are changed continuously, the natural frequency of the system will be changed and the overall dynamic equations of the tested system will be modified correspondingly. Different magnetic block position corresponds to a different system dynamics. This test is to verify the effectiveness of the proposed RIN in automatically identifying new system model and transferring it for control applications. It should be stated that the PD controller cannot work properly in this test because of the lack of accurate estimate of the dynamic characteristics of the tested system.

5.5.1 Control test based on the gradient training

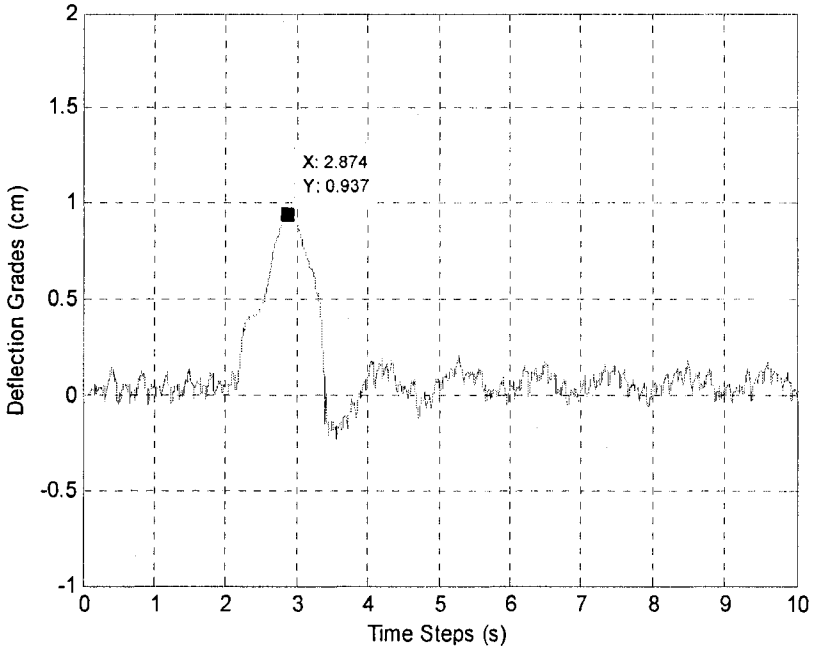
The experiment results by using the designed RIN based on the classical gradient training with the Disturbance sequence signal (as shown in Figure 4.3) are illustrated in Figure 5.15. Figure 5.15(a) shows the control result of Deflection when the magnetic blocks are attached at a bottom position of the flexible beam, with the overshoot of 0.921 *cm*. Figure 5.15(b) demonstrates the control result of Deflection when the magnetic blocks are attached at the middle position of the flexible beam, with an overshoot about 0.964 *cm*. Figure 5.15(c) shows the control result when the blocks are at a top position of the flexible beam, where the overshoot is 0.937 *cm*. From these results, it is seen that their undershooting and the settling time are similar as in Figure 5.13 when the system has no extra blocks attached.


X: 2.893
Y: 0.9213
Deflection Grades (cm)
Time Steps (s)
2
1.5
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7
8
9
10

(a)


X: 2.868
Y: 0.9644
Deflection Grades (cm)
Time Steps (s)
2
1.5
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7
8
9
10

(b)



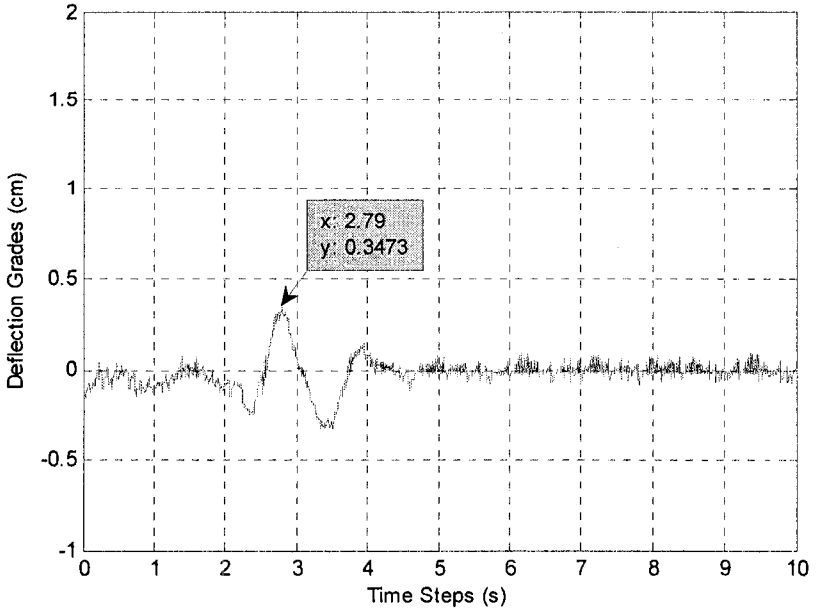
(c)

Figure 5.15. (a) using RIN with BP method when the magnetic material at lower position; (b) the magnetic material at middle position; (c) the magnetic material at upper position.

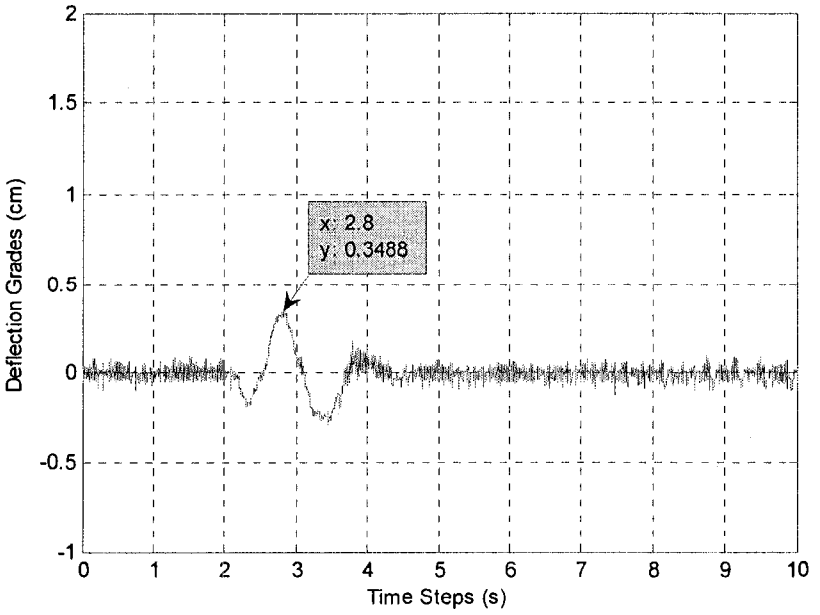
5.5.2 Control test based on RTRL with LSE training method

When the RIN is trained by using the hybrid technique based on RTRL and LSE, the experiment results are as shown in Figure 5.16. The control result of Deflection when the magnetic blocks are attached at a bottom position of the flexible beam is illustrated in Figure 5.16(a), where the overshoot is 0.347 *cm*. Figure 5.16 (b) shows the control result of Deflection when extra blocks are positioned at the middle area of the flexible beam, with the overshoot of 0.349 *cm*. Figure 5.16 (c) demonstrates the corresponding control result when the blocks are attached at a top position of the flexible beam; The overshoot is 0.302 *cm*. Compared with the

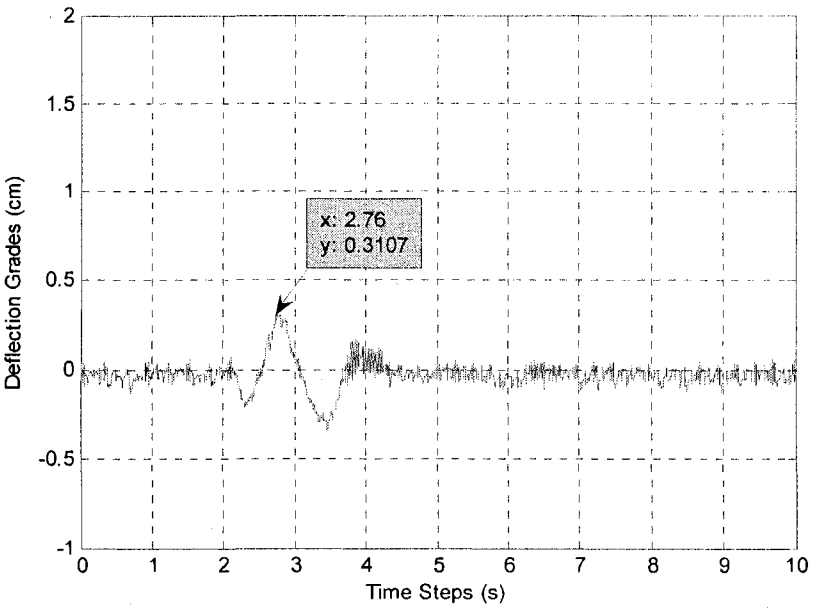
results shown in Figure 5.14, the corresponding undershooting and the settling time are similar as those tests when the system has no external materials attached.



(a)



(b)



(c)

Figure 5.16. (a) using RIN with RTRL method when the magnetic material at lower position; (b) the magnetic material at middle position; (c) the magnetic material at upper position.

Compare the control results of Deflection when the dynamics is changed by attaching external materials, the RIN can quickly and effectively identify the system new dynamics and implement it to the control applications accurately.

5.6 Result discussion

1) In terms of the control requests of the system with RIN, the control results of Rotating Angle, based on both training methods, are about +15 *deg*. Specifically, when the system dynamics is changed by attaching the extra magnetic block, the Rotating Angle is still around 15 *deg*. That is, the control requests of the motor remain similar when the system dynamics varies.

2) In terms of the selection of appropriate learning rate, by simulation tests, the learning rate of weights is chosen as 0.01 in gradient algorithm, and 0.001 in the RTRL with LSE. Gradient training is usually performed off-line in the sense that the parameters are not updated until the sequence ($k=1, 2 \ldots m$) is completely input to the system. However, RTRL with LSE is an on-line training technique. The parameters can be updated immediately after each input-output pair is presented to the system. Thus, the gradient algorithm requires more time than RTRL with LSE method to train and minimize the error functions between the actual and the desired control outputs.

3) When the dynamics is changed by attaching an external materials to the system, PD controller cannot adapt it to the system's new dynamics by properly modifying its gains, it cannot be for the vibration control of time-varying systems. During this experiment with the PD controller, the control motor generates a lot of noise as the flexible beam with the block is vibrating.

Chapter 6 Conclusions and Future Work

6.1 Conclusions

In this thesis, new intelligent schemes have been developed for active vibration control mainly for flexible engineering structures. The control effects are achieved based on two sequence signals: disturbance signal and the control signal. The principle of adding the disturbance signal to the system is to change the degree of the rigid beam to generate the disturbance effects.

Novel NF controllers have been developed for active vibration control, in which the control reasoning is performed by FL whereas fuzzy system parameters are updated by NN based training algorithms. NF controllers are intended to integrate the merits from both FL and NN based controllers while overcome their respective limitations. Because of the more complex architecture, the NF controller requires higher computational effect, which will result in lower sampling frequency. Consequently, high computational burden may make it difficult to implement an NF controller for real-time applications. To solve this problem, the developed NF controllers employ simplified network architectures so as to improve control process computationally efficiency. The developed NF controllers are trained by using a hybrid algorithm for fast identification of parameters: The antecedent parameters are updated by gradient method whereas the consequent parameters are tuned by LSE.

For time-varying systems, the dynamics of the system will change; correspondingly, the new dynamic equations have to be re-determined accordingly. It is a hard task especially for nonlinear systems. In this work, a novel recurrent network, RIN, has been developed to real-time

system dynamics identification. A novel training technique is adopted to improve the convergence of the RIN scheme and to accommodate different system conditions. This training technique is hybrid method: the recurrent network parameters are optimized by the RTRL algorithm whereas the consequent linear system parameters fine-tuned by LSE.

The effectiveness of the developed intelligent controllers, RIN, and the related training techniques has been verified by online experimental tests on a flexible beam structure. The time-varying effects are undertaken by moving the position of magnetic blocks attached onto the flexible beam. From the comprehensive investigation, it is demonstrated that the developed NF control technique a good candidate for real-time active vibration control for time-varying systems. Based upon the experiment results, firstly it is shown that the developed NF controller has the capability of mapping the input-output data sets to achieve the target (or desired) output of the time-varying systems. It outperforms the classical as well as other related intelligent controllers. The novel NF controller has better performance in terms of overshoot, settling time, and the Rotating Angle (control requests). The online learning algorithm can also improve the controller's training convergence.

Secondly, the control results with the RIN are better than those without the RIN. It is because the modeling errors are unavoidable when the system dynamics are transferred from nonlinear equations to linear equations. Furthermore, if system dynamics varies, the RIN can effectively identify the system's new dynamics and implement it for real-time control applications.

Thirdly, the Deflection control results by the hybrid training technique based on the RTRL and LSE are better than those by the classical training algorithm based on gradient. The control effect by using RTRL with LSE method is the most obvious and fastest.

6.2 Future works

1) In Chapter 3, two novel NF controllers (TS0-NF and TS1-NF) have been developed. In the TS1-NF controller, only the linear consequent parameters are trained on-line during the experiment. In future work, a more efficient training algorithm will be suggested to online train both the liner and nonlinear fuzzy parameters in the controller.

2) Improve the architecture of the proposed RIN, by adding more neurons in each layer or re-design the network connections, so as to enhance the mapping accuracy between the input-output spaces.

3) Propose more efficient online training algorithms to further improve the training convergence of the controller and RIN, and to recognize and accommodate time-varying system dynamics.

Reference

[1]. D. Younesian, E. Esmailzadeh and R. Sedaghati. Passive Vibration Control of Beams Subjected to Random Excitations with Peaked PSD. Journal of Vibration and Control, 12 (2006)941–953.

[2]. C. R. Fuller, G. P. Gibbs, and R. J. Silcox. Simultaneous active control of flexural and extensional waves in beams. Journal of Material. System and Structure, 1 (1990) 235-247.

[3]. M. J. Brennan, S. J. Elliott, and R. J. Pinnington. Active control of vibrations transmitted through struts. In International Conf. on Motion and Vibration Control, Yokohama, September (1992).

[4]. S. O. R. Moheimani, D. Halim, and A. J. Fleming. Spatial Control of Vibration: Theory and Experiments. World Scientific Publishing Co. Pte. Ltd., Singapore, (2003).

[5]. C. R. Fuller, S. J. Elliott, and P. A. Nelson. Active Control of Vibration. Academic Press, London, (1996).

[6]. A. Preumont. Vibration Control of Active Structures: An Introduction. Kluwer Academic Publisher, Dordrecht, (2002).

[7]. M. T. Ho and Y. W. Tu. PID controller design for a flexible-link manipulator. Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, (2005) 6841 -6846.

[8]. C. Murrugarra, J. Grieco, G. Fernandez and O. D. Castro. Design of a PD position control based on the Lyapunov theory for a robot manipulator flexible-link. IEEE International Conference on Robotics and Biomimetics, (2006) 890-895.

[9]. B. Siciliano, B. S. Yuan, and W. J. Book. Model Reference Adaptive Control of a One Link Flexible Arm. 25th IEEE Conf Decision and Contr., Athens, Dec. (1986).

[10]. J. Yuh. Application of Discrete-Time Model Reference Adaptive Control to a Flexible Single-Link Robot. J. Robotic Syst., vol. 4, (1987).

[11]. J. Fei, G. Song. Adaptive vibration suppression of a smart flexible beam using direct model reference adaptive control (MRAC). Proceedings of SPIE Vol. 5383 (2004) 406-416.

[12]. F. Harahima and T. Ueshiba. Adaptive Control of Flexible Arm Using the End-Point Position Sensing. Proc. Japan-USA Symp. Flexible Automat., Osaka (Japan), July (1986).

[13]. D. M. Rovner and R. H. Cannon. Experiments Towards On-Line Identification and

Control of a Very Flexible One-Link Manipulator. Intl. J. Robotics Res., vol. 6, Winter (1987).

[14]. I. D. Landau. Trends in Adaptive Control of Robot Manipulator. 27th IEEE Conf Decision and Cont., Austin, TX, Dec (1988) 1604-1606.

[15]. A. J. Koivo and K. S. Lee, "Self-Tuning Control of Planar Two-Link Manipulator with Non-Rigid Arm," IEEE Intl. Con5 Robotics and Automat., Scottsdale, AZ, May (1989) 1030-1035.

[16]. S. J. Elliott and P.A. Nelson. Active noise control. IEEE Signal Processing Magazine, vol. 8 (1993) 12-35.

[17]. R. L. Clark and W. R. Saunders. Adaptive Structure: dynamics and control. New York: Wiley (1 998).

[18]. W. T. Baumann. An adaptive feedback approach to structural vibration suppression. Journal of Sound and Vibration, vol. 205 (1997) 121-133.

[19]. H. W. Park, H. S. Yang, Y. P. Park, and S. H. Kim. Position and vibration control of a flexible robot manipulator using hybrid controller. Robotics and Autonomous Systems, 28 (1999) 1–41.

[20]. M. Karkoub, and K. Tamma. Modelling and μ-synthesis control of flexible manipulators. Computers and Structures, 79 (2001) 543–551.

[21]. B. Yao, and L. Xu. Output Feedback Adaptive Robust Control of Uncertain Linear Systems With Disturbances. Transactions of the ASME. Vol. 128 (2006) 938-945.

[22]. R. Lozano-Leal. On the design of dissipative LQG-type controllers. In Proc. of the 27th Conf. on Decision and Control, Austin, USA, December (1988).

[23]. R. Lozano-Leal and S. M. Joshi. Strictly positive real transfer functions revisited. IEEE Trans. on Automatic Control, 35 (1990) 1243–1245.

[24]. R. Lozano, B. Brogliato, O. Egeland, and B. Maschke. Dissipative Systems Analysis and Control. Theory and Applications. Springer, Heidelberg, (2000).

[25]. David Vinke and M. Vidyasagar. New Techniques for H_2 optimal control of a flexible beam. Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, California April (1991) 2592-2597.

[26]. R. Toscano. H_2/H_∞Robust Static Output Feedback Control Design Without Solving Linear Matrix Inequalities. Transactions of the ASME. Vol. 129 (2007) 860-866.

[27]. C. Trautman, D. Wang, Experimental H_∞ control of a single flexible link with a shoulder joint, in: IEEE International Conference on Robotics and Automation, vol. 1, 1995, pp.

1235-1241.

[28]. W. Chen, K. Yeh, L. Chiang, Y. Chen, J. Wu, Modeling, Hinf Control and Stability Analysis for Structural Systems Using Takagi-Sugeno Fuzzy Model, Journal of Vibration and Control, 13 (2007) 1519–1534.

[29]. C. Lin, Q. G. Wang, and T. H. Lee. Observer-Based H_∞ Control for T-S Fuzzy Systems with Time Delay: Delay-Dependent Design Method. IEEE Transactions on fuzzy systems, MAN, and CYBERNETICS-Part B: CYBERNETICS, Vol. 37 (2007) 1030-1038.

[30]. K.D. Young, U. Ozguner, Frequency shaping compensator design for sliding mode, International Journal of Control 57 (1993) 1005-1019.

[31]. Y. Kitamura, K. Iwabuchi, K. Nonami, H. Nishimura, Positioning control of flexible arm using frequency-shaped sliding mode control, in: Third International Conference on Motion and Vibration Control, (1996) 178-183.

[32]. K. Nonami, H. Nishimura, H. Tian, H_∞/μcontrol-based frequency shaped sliding mode control for flexible structures, JSME International Journal, (1996).

[33]. C. Pai, and A. Sinha, Sliding Mode Output Feedback Control of Vibration in a Flexible Structure, Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control, 129 (2007) 851- 855.

[34]. G. Song, and H. Gu, Active Vibration Suppression of a Smart Flexible Beam Using a Sliding Mode Based Controller, Journal of Vibration and Control, 13 (2007) 1095–1107.

[35]. M. Itik, M. U. Salamci, F. D. Ulker, and Y. Yaman. Active Vibration Suppression of a Flexible Beam via Sliding Mode and Hinf Control. Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference,Seville, Spain, December (2005) 12-15.

[36]. H. C. Gu and G. B. Song. Adaptive robust sliding-mode control of a flexible beam using PZT sensor and actuator. Proceedings of the IEEE International Symposium on Intelligent Control Taipei, Taiwan, September (2004) 78-83.

[37]. M. C. Pai, and A. Sinha. Sliding Mode Output Feedback Control of Vibration in a Flexible Structure. Journal of Dynamic Systems, Measurement, and Control. Vol. 129 (2007) 851-855.

[38]. S.B. Choi, H.C. Shin, A hybrid actuator scheme for robust position control of a flexible single-link manipulator, Journal of Robotic Systems 13 (1996) 359-370.

[39]. D. Halim and S. O. R. Moheimani. Spatial resonant control of flexible structures - application to a piezoelectric laminate beam. IEEE Trans. on Control Systems Technology, 9 (2001) 37–53.

[40]. M. R. Bai and G. M. Lin. The development of a DSP-based active small amplitude vibration control system for flexible beams by using the LQG algorithms and intelligent materials. Journal of Sound and Vibration. Vol 198 (1996)411-427.

[41]. Ian R. Petersen, Himanshu R. Pota. Minimax LQG optimal control of a flexible beam. Control Engineering Practice. 11 (2003) 1273–1287.

[42]. L. Gaudiller, and F. Matichard, A Nonlinear Method for Improving the Active Control Efficiency of Smart Structures Subjected to Rigid Body Motions, IEEE/ASME Transactions on Mechatronics, 12 (2007).

[43]. J. Jiang, and Q. Gao, An Application of Nonlinear PID Control to a Class of Truck ABS Problems.

[44]. R. Kelly, and R. Carelli, A Class of Nonlinear PD-Type Controllers for Robot Manipulators, Journal of Robotic Systems, 13 (1996) 793-802.

[45]. L. Tian, and C. Collins. A dynamic recurrent neural network-based controller for a rigid-flexible manipulator system. Mechatronics, 14 (2004) 471–490.

[46]. S. Bruckner, and S. Rudolph. Neural Networks Applied to Smart Structure Control. Proceedings SPIE Conference on Applications and Science of Computational Intelligence III, Orlando, Florida, April (2000).

[47]. J. Harris. Comparative aspects of neural networks and fuzzy logic for real-time control. Neural networks for control and systems, London, (1992) 72-93.

[48]. R. JAMIL, Inverse Dynamics Based Tuning of a Fuzzy Logic Controller for a Single-Link Flexible Manipulator, Journal of Vibration and Control, 13 (2007) 1741–1759.

[49]. G. Yan, and Lily L. Zhou. Fuzzy logic and genetic algorithms for intelligent control of structures using MR dampers. Proc. of SPIE conference on Smart Structures and Materials: Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems, Vol. 5391 (2004) 555-565.

[50]. A. Jnifene, and W. Andrews. Fuzzy Logic Control of the End-Point Vibration in an Experimental Flexible Beam. Journal of Vibration and Control, 10 (2004) 493-506.

[51]. II. Kim, J. H. Lee, and E. O. Bang. A New Approach to Adaptive Membership Function for Fuzzy Inference System. Third International Conference on Knowledge-Based Intelligent Information Engineering Systems, Adelaide, Australia (1999)112-116.

[52]. T. H. S. Li, and S. H. Tsai. T-S Fuzzy Bilinear Model and Fuzzy Controller Design for a Class of Nonlinear Systems. IEEE Transactions on fuzzy systems, vol. 15 (2007) 494-506.

[53]. C. S. Ting. An observer-based approach to controlling time-delay chaotic systems via Takagi-Sugeno fuzzy model. Information Sciences 177 (2007) 4314–4328.

[54]. M. Caswara, and H. Unbehauen, A Neurofuzzy Approach to the Control of a Flexible-Link Manipulator, IEEE Transaction on Robotics and Automation, 18 (2002).

[55]. F. Tian, and C. Collins, Adaptive neuro-fuzzy control of a flexible manipulator, Mechatronics 15 (2005) 1305–1320.

[56]. J. S. Roger Jang. ANFIS: Adaptive-Network-Based Fuzzy Inference System. IEEE Transactions on fuzzy systems, MAN, and CYBERNETICS, Vol. 23 (1993) 665-685.

[57]. Z. L. Sun, K. F. Au, and T. M. Choi. A Neuro-Fuzzy Inference System Through Integration of Fuzzy Logic and Extreme Learning Machines. IEEE Transactions on systems, MAN, and CYBERNETICS-Part B: CYBERNETICS, Vol. 37 (2007) 1321-1331.

[58]. S. Pletl. Neuro-Fuzzy Control of Rigid and Flexible-Joint Robotic Manipulator. Proceedings of Teaching Fuzzy Systems Joint Tempus Workshop, Budapest, Hungary, (1995) 93-97.

[59]. K. S. Jeong, Y. S. Hong, and C. K. Park. Neuro-Fuzzy Control of a Robot Manipulator for a Trajectory Design. IEEE International Workshop on Robot and Human Communication, (1997) 260-265.

[60]. F. C. Sun, L. Li, H. X. Li, and H. P. Liu. Neuro-Fuzzy Dynamic-Inversion-Based Adaptive Control for Robotic Manipulators-Discrete Time Case. IEEE Transactions on industrial electronics, Vol. 54 (2007) 1342-1351.

[61]. P. Ibanez, Identification of dynamic parameters of linear and non-linear structural models from experimental data, Nuclear Engineering and Design 25 (1973) 30-41.

[62]. S. F. Masri, T. K. Caughey, A nonparametric identification technique for nonlinear dynamic problems, Journal of Applied Mechanics 46 (1979) 433-447.

[63]. K. Worden, G.R. Tomlinson, Nonlinearity in Structural Dynamics: Detection, Identification and Modelling, Institute of Physics Publishing, Bristol and Philadelphia, 2001.

[64]. D. E. Adams, R.J. Allemang, Survey of nonlinear detection and identification techniques

for experimental vibrations structural dynamic model through feedback, in: Proceedings of the International Seminar on Modal Analysis (ISMA), Leuven, (1998) 269-281.

[65]. F. M. Hemez, S.W. Doebling, Inversion of structural dynamics simulations: state-of-the-art and orientations of the research, in: Proceedings of the International Seminar on Modal Analysis (ISMA), Leuven, 2000.

[66]. K. Worden, Nonlinearity in structural dynamics: the last ten years, in: Proceedings of the European COST F3 Conference on System Identification and Structural Health Monitoring, Madrid, (2000) 29-52.

[67]. F. M. Hemez, S.W. Doebling, Review and assessment of model updating for non-linear, transient dynamics, Mechanical Systems and Signal Processing 15 (2001) 45-74.

[68]. J. C. Golinval, G. Kerschen, V. Lenaerts, F. Thouverez, P. Argoul, European COST action F3 on structural dynamics. Working group 3: identification of non-linear systems; Introduction and conclusions, Mechanical Systems and Signal Processing 17 (2003) 177-178, 251-254.

[69]. M. Juntunen, J. Linjama, Presentation of the VTT benchmark, Mechanical Systems and Signal Processing 17 (2003) 179-182.

[70]. F. Thouverez, Presentation of the ECL benchmark, Mechanical Systems and Signal Processing 17 (2003) 195-202.

[71]. E. H. Dowell, B.I. Epureanu, Preface, Nonlinear Dynamics, 39 (2005).

[72]. G. Kerschen, K. Worden, A. F. Vakakis and J. C. Golinval, Past, present and furure of nonlinear system identification in structural dynamics, Journal of Mechanical System and Signal Processing, 20 (2006) 505-592.

[73]. M. Khalil, A. Sarkar, and S. Adhikari, Linear system identification using proper orthogonal decomposition, Journal of Mechanical System and Signal Processing, 21 (2007) 3123-3145.

[74]. K. Christodoulou, and C. Papadimitriou, Structural identification based on optimally weighted modal residuals, Journal of Mechanical System and Signal Processing, 21 (2007) 4-23.

[75]. P. Mastorocostas, and J. Theocharis, A Recurrent Fuzzy-Neural Model for Dynamic System Identification, IEEE Transactions on system, man and cybernetics-part B: Cybernetics, 32 (2002).

[76]. L. Ljung, System Identification—Theory for the User, second ed., Prentice-Hall PTR, Upper Saddle River, NJ, (1999).

[77]. L. Jin, N. Nikiforuk, and M. Gupta, Approximation of discrete-time state-space trajectories using dynamic recurrent neural networks, IEEE Trans. Automat. Contr., 40 (1995) 1266–1270.

[78]. C. Ku and Y. Lee, Diagonal recurrent neural networks for dynamic systems control, IEEE Trans. Neural Networks, 6 (1995) 144-156.

[79]. S. Narendra and K. Parthasarathy, Identification and control of dynamical system using neural networks, IEEE Trans. Neural Networks, 1 (1990) 4-27.

[80]. H. B. Verbruggen, and R. Babuška. Fuzzy logic control: advances in applications. Singapore; River Edge, NJ: World Scientific (1999).

[81]. J. S. R. Jang, C. T. Sun, E. Mizutani. Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence (1997).

[82]. H. B. Verbruggen and R. Babuska. Fuzzy Logic Control Advances in applications. World Scientific Publishing Co. Pte. Ltd, Farrer Road, Singapore, 1999.

[83]. M. Sugeno and G. T. Kang, Structure identification of fuzzy model, Fuzzy Sets and Systems, 28 (1988) 15-33.

[84]. T. Takagi and M. Sugeno, Fuzzy identification of system and its applications to modeling and control, IEEE Transactions on Systems, Man, and Cybernetics 15 (1985) 116-132.

[85]. D. E. Rummelhart, G. E. Hinton, and R. J. Williams. Learning Representations by Back-Propagating Errors. Nature 323 (1986) 533-536.

[86]. P. Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. PhD thesis, Harvard University, 1974.

[87]. G. Kong, and B. Kosko, Adaptive fuzzy systems for backing up a truck-and-trailer, IEEE Trans. Neural Networks, 3 (1992) 211-221.

[88]. H. Takagi, and I. Hayashi, NN-driven fuzzy reasoning, Int. J. Approx. Reason. 5 (1991) 191-212.

[89]. J. S. Roger Jang. ANFIS: Adaptive-network-based fuzzy inference system. IEEE Trans. Systems, Man & Cybernetics, 23 (1993) 665-685.

[90]. J. S. Roger Jang. Fuzzy modeling using generalized neural networks and Kalman filter algorithm. In Proceedings of the Ninth National Conference on Artificial Intelligence

(AAAI-91), (1991) 762-767.

[91]. T. Takagi and M. Sugeno. Derivation of fuzzy control rules from human operator's control actions. Proceedings of the IFAC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis, July (1983) 55-60.