

國立交通大學

電機與控制工程學系

博士論文

以函數鏈結為基礎之類神經模糊網路及其應用

**A Functional-Link-Based Neuro-Fuzzy Network and Its Applications**

研究生：陳政宏

指導教授：林進燈

中華民國九十七年七月

以函數鏈結為基礎之類神經模糊網路及其應用  
A Functional-Link-Based Neuro-Fuzzy Network and Its  
Applications

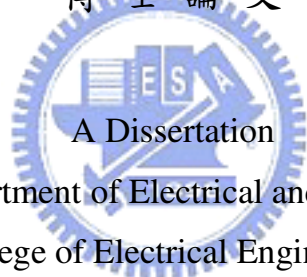
研究生：陳政宏

Student : Cheng-Hung Chen

指導教授：林進燈 博士

Advisor : Dr. Chin-Teng Lin

國立交通大學  
電機與控制工程學系  
博士論文



Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Electrical and Control Engineering

July 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年七月

# 國立交通大學

## 博碩士論文全文電子檔著作權授權書

(提供授權人裝訂於紙本論文書名頁之次頁用)

本授權書所授權之學位論文，為本人於國立交通大學 電機與控制工程 系所  
                                 組， 九十六 學年度第 二 學期取得博士學位之論文。

論文題目：以函數鏈結為基礎之類神經模糊網路及其應用

指導教授：林進燈


同意     不同意

本人茲將本著作，以非專屬、無償授權國立交通大學與台灣聯合大學系統圖書館：基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會與學術研究之目的，國立交通大學及台灣聯合大學系統圖書館得不限地域、時間與次數，以紙本、光碟或數位化等各種方法收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行線上檢索、閱覽、下載或列印。

論文全文上載網路公開之範圍及時間：

本校及台灣聯合大學系統區域網路	■ 中華民國 99 年 2 月 23 日公開
校外網際網路	■ 中華民國 99 年 2 月 23 日公開

授權人：陳政宏

親筆簽名：

中華民國 97 年 7 月 30 日

# 國立交通大學

## 博碩士紙本論文著作權授權書

(提供授權人裝訂於全文電子檔授權書之次頁用)

本授權書所授權之學位論文，為本人於國立交通大學 電機與控制工程 系所  
   組，九十六 學年度第 二 學期取得博士學位之論文。


論文題目：以函數鏈結為基礎之類神經模糊網路及其應用

指導教授：林進燈

### ■ 同意

本人茲將本著作，以非專屬、無償授權國立交通大學，基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會與學術研究之目的，國立交通大學圖書館得以紙本收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行閱覽或列印。

本論文為本人向經濟部智慧局申請專利(未申請者本條款請不予理會)的附件之一，申請文號為：  ，請將論文延至      年      月      日再公開。



授權人：陳政宏

親筆簽名：陳政宏

中華民國 97 年 7 月 30 日

# 國家圖書館博碩士論文電子檔案上網授權書

ID:GT009312816

本授權書所授權之論文為授權人在國立交通大學電機學院電機與控制工程系所  
\_\_\_\_\_ 組 九十六 學年度第 二 學期取得博士學位之論文。

論文題目：以函數鏈結為基礎之類神經模糊網路及其應用

指導教授：林進燈

茲同意將授權人擁有著作權之上列論文全文（含摘要），非專屬、無償授權國家圖書館，不限地域、時間與次數，以微縮、光碟或其他各種數位化方式將上列論文重製，並得將數位化之上列論文及論文電子檔以上載網路方式，提供讀者基於個人非營利性質之線上檢索、閱覽、下載或列印。

※ 讀者基於非營利性質之線上檢索、閱覽、下載或列印上列論文，應依著作權法相關規定辦理。

授權人：陳政宏

親筆簽名：陳政宏



中華民國 97 年 7 月 30 日

1. 本授權書請以黑筆撰寫，並列印二份，其中一份影印裝訂於附錄三之二(博碩士紙本論文著作權授權書)之次頁；另一份於辦理離校時繳交給系所助理，由圖書館彙總寄交國家圖書館。

## 推薦函

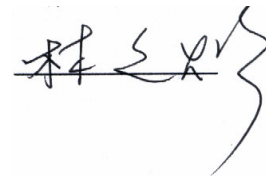
- 一、事由：推薦電機與控制工程系博士班研究生陳政宏提出論文以參加國立交通大學博士論文口試。
- 二、說明：本校電機與控制工程系博士班研究生陳政宏已完成博士班規定之學科及論文研究訓練。  
有關學科部分，陳君以修必應修學分（請查學籍資料），通過資格考試；有關論文方面，陳君已完成“以函數鏈結為基礎之類神經模糊網路及其應用”初稿。其論文“A Functional-Link-Based Neuro-Fuzzy Network for Nonlinear System Control”和“Using an Efficient Immune Symbiotic Evolution Learning for Compensatory Neuro-Fuzzy Controller”等兩篇已經被 *IEEE Trans. on Fuzzy Systems* 期刊所接受。另有論文“A Hybrid of Cooperative Particle Swarm Optimization and Cultural Algorithm for Neural Fuzzy Networks and Its Prediction Applications”也已經被 *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 期刊所接受（請參閱博士論文著作目錄）。
- 三、總言之，陳君已具備國立交通大學電機與控制工程系博士班研究生應有之教育及訓練水準，因此推薦陳君參加國立交通大學電機與控制工程系博士論文口試。

此致

國立交通大學電機與控制工程學系

電機與控制工程學系教授

林進燈



中華民國 97 年 5 月

# 國立交通大學

## 論文口試委員會審定書

本校電機與控制工程學系博士班陳政宏君

所提論文：以函數鏈結為基礎之類神經模糊網路及其應用

合於博士資格水準，業經本委員會評審認可。

口試委員：蔡文祥 李祖聖  
傅立許 楊介澤  
洪卓良 林正堅  
林進燈 楊金榮

指導教授：林進燈

系主任：邵偉斌

中華民國 97 年 7 月 23 日

Department of Electrical and Control Engineering  
National Chiao Tung University  
Hsinchu, Taiwan R.O.C.

Date: July 23, 2008

We have carefully read the dissertation entitled A Functional-Link-Based Neuro-Fuzzy Network and Its Applications submitted by Cheng-Hung Chen in partial fulfillment of the requirements of the degree of **DOCTOR OF PHILOSOPHY** and recommend its acceptance.

Wen-hsiung Tsai

Juan-Hong S. L.

Li-Chen Chen

Yung-Yung Chen

Tzung-Pei Hong

Cheng-Gia L.

Chi-Jeng L.

Jyh-Hong Chen

\_\_\_\_\_

\_\_\_\_\_

Thesis Advisor: Chi-Jeng L.

Chairman: J. C. Chien



# 以函數鏈結為基礎之類神經模糊網路及其應用

研究生：陳政宏 指導教授：林進燈 博士

國立交通大學電機與控制工程學系（研究所）博士班

## 摘 要

本篇論文提出一以函數鏈結為基礎之類神經模糊網路及其相關學習演算法。此類神經模糊網路採用函數鏈結類神經網路當作模糊法則的後件部。此後件部是輸入變數的非線性組合，它是利用函數展開的方式，能在高維度的輸入空間中提供良好的非線性決策能力，因此，可使網路輸出更具體且更逼近目標輸出。本論文主要為三大部分。第一部份將詳細介紹以函數鏈結為基礎之類神經模糊網路及其線上學習演算法。此演算法包含架構學習及參數學習，架構學習是藉由熵的量測決定是否要增長一個新的法則，參數學習是使用倒傳遞演算法調整網路上的所有參數。由於倒傳遞演算法常常會得到局部最佳解。因此，在第二部份中，我們提出一改良式差分進化演算法，所提出的演算法與傳統差分進化演算法是不同的，在於我們使用一有效的搜尋機制使得每條個體能更新在目前最佳解和亂數搜尋解之間，並採用以群為基底的突變方式以提高個體間彼此的差異性。但以上的進化演算法，無法決定該使用多少法則數。因此，在第三部份中，我們提出一以法則為基礎的共生差分進化演算法。此演算法是利用多個子族群進行進化，每個子族群的個體代表每條模糊法則，且每個子族群能各自進化。此外，這演算法也能自動決定子族群數，並最佳化網路上的所有參數。最後，我們將與其他方法比較，以證實所提出的網路架構及其相關演算法之有效性。

# **A Functional-Link-Based Neuro-Fuzzy Network and Its Applications**

Student : Cheng-Hung Chen

Advisor : Dr. Chin-Teng Lin

Department of Electrical and Control Engineering

National Chiao-Tung University

## **Abstract**

This dissertation proposes a functional-link-based neuro-fuzzy network (FLNFN) and its related learning algorithms. The proposed FLNFN model uses a functional link neural network to the consequent part of the fuzzy rules. The consequent part uses a nonlinear functional expansion to form arbitrarily complex decision boundaries. Thus, the local properties of the consequent part in the FLNFN model enable a nonlinear combination of input variables to be approximated more effectively. This dissertation consists of three major parts. In the first part, the FLNFN model and an online learning are presented. The online learning algorithm consists of structure learning and parameter learning. The structure learning depends on the entropy measure to determine the number of fuzzy rules. The parameter learning, based on back-propagation, can adjust the shape of the membership function and the corresponding weights of the consequent part. Unfortunately, the back-propagation learning algorithm may reach the local minima very quickly. Therefore, a modified differential evolution (MODE) is presented to optimize the FLNFN parameters in the second part. The proposed MODE learning algorithm differs from the traditional differential evolution. The MODE adopts a method to effectively search between the best individual and randomly chosen individuals, and the MODE also provides a cluster-based mutation scheme, which maintains useful diversity in the population to increase the search

capability. But, the aforementioned algorithm cannot determine how many rules to be used. Therefore, a rule-based symbiotic modified differential evolution (RSMODE) is proposed for the FLNFN model in the third part. The RSMODE adopts a multi-subpopulation scheme that uses each individual represents a single fuzzy rule and each individual in each subpopulation evolves separately. Furthermore, the proposed RSMODE learning algorithm can also determine the number of rule-based subpopulation and adjust the FLNFN parameters. Finally, the proposed FLNFN model and its related learning algorithms are applied in various control problems. Results of this dissertation demonstrate the effectiveness of the proposed methods.



# Acknowledgment

在這研究期間裡，首先要感謝我的博士班指導教授 林進燈博士和碩士班指導教授 林正堅博士，在二位教授豐富的學識、殷勤的教導及嚴謹的督促下，使我學習到許多的寶貴知識及在面對事情中應有的處理態度、方法，並且在研究與投稿論文的過程中，二位教授有許多深入的見解及看法且對於斟酌字句、思慮周延，更是我該學習的目標。師恩好蕩，指導提攜，銘感於心。

由衷感謝口試委員傅立成教授、周至宏教授、洪宗貝教授、李祖聖教授、蔡文祥教授及楊谷洋教授給予許多寶貴的建議與指正，使得這篇論文更加完整。同時要感謝交大多媒體實驗室的學長 劉得正博士及蒲鶴章博士還有全體同學及學弟妹，在研究的過程中不斷的互相砥礪及討論。感謝過去 91 級朝陽資工系的所有同學，在這過程中，一起分享不同的經驗。感謝一路走來陪伴身邊的所有朋友，使得我的研究生涯變得多彩多姿。

特別要感謝我的父親、母親、妹妹，在這段日子中不斷的給予支持及鼓勵，讓我能夠專心於研究的工作並完成博士學位。最後誠摯地以本論文研究成果獻給我的師長、父母、家人及所有的朋友們。

陳政宏

九十七年七月二十七日

# Contents

Abstract in Chinese.....	i
Abstract in English .....	ii
Acknowledgment.....	iv
Contents.....	v
List of Tables .....	vii
List of Figures.....	viii
1 Introduction .....	1
1.1 Motivation .....	1
1.2 Literature Survey .....	2
1.3 Organization of Dissertation.....	6
2 A Functional-Link-Based Neuro-Fuzzy Network (FLNFN).....	8
2.1 Structure of Functional-Link-Based Neuro-Fuzzy Network.....	8
2.1.1 Functional Link Neural Networks .....	9
2.1.2 Structure of the FLNFN Model.....	10
2.2 Learning Algorithms of the FLNFN Model .....	13
2.2.1 Structure Learning Phase.....	14
2.2.2 Parameter Learning Phase.....	17
2.2.3 Convergence Analysis.....	19
2.3 Experimental Results.....	19
2.4 Summary.....	35
3 A Modified Differential Evolution for the FLNFN Model.....	37
3.1 A Brief Introduction of Differential Evolution.....	37
3.2 A Modified Differential Evolution .....	40
3.2.1 Initialization Phase .....	41
3.2.2 Evaluation Phase.....	41
3.2.3 Reproduction Phase .....	42
3.2.4 Cluster-Based Mutation Phase.....	43
3.3 Experimental Results.....	44
3.4 Summary.....	70
4 A Rule-Based Symbiotic Modified Differential Evolution for the FLNFN Model .....	71
4.1 A Basic Concept of Symbiotic Evolution.....	72
4.2 A Rule-Based Symbiotic Modified Differential Evolution .....	72
4.2.1 Initialization Phase .....	74
4.2.2 Parameter Learning Phase .....	76
4.3 Experimental Results.....	81

4.4 Summary.....	93
5 Conclusion and Future Works .....	94
Appendix .....	97
A. Proof of the Universal Approximator Theorem.....	97
B. Proof of Convergence Theorem.....	100
Bibliography .....	105
Vita.....	113
Publication List.....	114



# List of Tables

Table 2.1: Comparison of performance of various controllers to control of water bath temperature system.....	26
Table 2.2: Comparison of performance of various controllers to control of BIBO nonlinear plant.....	31
Table 2.3: Comparison of performance of various controllers to control of ball and beam system.....	33
Table 2.4: Comparison of performance of various controllers to control of MIMO plant.....	34
Table 3.1: Parameter settings before training.....	45
Table 3.2: Comparison of performance of various controllers to control of water bath temperature system.....	49
Table 3.3: Comparison of performance of various controllers to control of the planetary train type inverted pendulum system with a 0.1s sampling rate.....	58
Table 3.4: Comparison of performance of various controllers to control of the magnetic levitation system with a 0.1s sampling rate.....	69
Table 4.1: Parameter settings before training.....	81
Table 4.2: Comparison of performance of various controllers to control of water bath temperature system.....	86
Table 4.3: Comparison of performance of various controllers to control of ball and beam system.....	88
Table 4.4: Comparison of performance of various controllers to control of backing up the truck.....	93

# List of Figures

Figure 2.1: Structure of FLNN. ....	9
Figure 2.2: Structure of proposed FLNFN model. ....	11
Figure 2.3: Flow diagram of the structure/parameter learning for the FLNFN model.....	14
Figure 2.4: Conventional online training scheme.....	21
Figure 2.5: (a) Final regulation performance of FLNFN controller in water bath system. (b) Error curves of the FLNFN controller, TSK-type NFN controller and FLNN controller between $k=81$ and $k=100$ . ....	24
Figure 2.6: (a) Behavior of FLNFN controller under impulse noise in water bath system. (b) Error curves of FLNFN controller, TSK-type NFN controller and FLNN controller.....	25
Figure 2.7: (a) Behavior of FLNFN controller when a change occurs in the water bath system. (b) Error curves of FLNFN controller, TSK-type NFN controller and FLNN controller.....	25
Figure 2.8: (a) Tracking of FLNFN controller when a change occurs in the water bath system. (b) Error curves of FLNFN controller, TSK-type NFN controller and FLNN controller.....	26
Figure 2.9: Block diagram of FLNFN controller-based control system.....	27
Figure 2.10: Final system response in first case of example 2. (a) The dashed line represents plant output and the solid line represents the reference model. (b) Error curves of FLNFN controller and TSK-type NFN controller.....	29
Figure 2.11: Final system response in second case of example 2. (a) The dashed line represents plant output and the solid line represents the reference model. (b) Error curves of FLNFN controller and TSK-type NFN controller.....	29
Figure 2.12: Final system response in third case of example 2. (a) The dashed line represents plant output and the solid line represents the reference model. (b) Error curves of FLNFN controller and TSK-type NFN controller.....	30
Figure 2.13: Final system response in fourth case of example 2. The dashed line represents plant output and the solid line represents the reference model.....	30
Figure 2.14: Ball and beam system. ....	31
Figure 2.15: Responses of ball and beam system controlled by FLNFN model (solid curves) and TSK-type NFN model (dotted curves) under four initial conditions.....	33
Figure 2.16: Responses of four states of ball and beam system under the control of the trained FLNFN controller. ....	33
Figure 2.17: Desired output (solid line) and model output using FLNFN controller (dotted line) of (a) Output 1. (b) Output 2 in Example 4. Error curves of FLNFN controller (solid line) and TSK-type NFN controller (dotted line) for (c) output 1 and (d)	



output 2.....	35
Figure 3.1: An example of a two-dimensional cost function showing its contour lines and the process for generating $v_{i,G+1}$ .....	39
Figure 3.2: Illustration of the crossover process for N=7 parameters.....	40
Figure 3.3: Coding FLNFN into an individual in the proposed MODE method.....	41
Figure 3.4: A mutation operation in the modified differential evolution.....	44
Figure 3.5: (a) Final regulation performance of FLNFN-MODE controller in water bath system. (b) Error curves of the FLNFN-MODE controller, FLNFN-DE controller and FLNFN-GA controller.....	46
Figure 3.6: (a) Behavior of FLNFN-MODE controller under impulse noise in water bath system. (b) Error curves of FLNFN-MODE controller, FLNFN-DE controller and FLNFN-GA controller.....	47
Figure 3.7: (a) Behavior of FLNFN-MODE controller when a change occurs in the water bath system. (b) Error curves of FLNFN-MODE controller, FLNFN-DE controller and FLNFN-GA controller.....	48
Figure 3.8: (a) Tracking of FLNFN-MODE controller when a change occurs in the water bath system. (b) Error curves of FLNFN-MODE controller, FLNFN-DE controller and FLNFN-GA controller.....	49
Figure 3.9: A physical model geometry of the planetary train type inverted pendulum.....	50
Figure 3.10: Control block diagram for the planetary train type inverted pendulum system.....	53
Figure 3.11: The experimental planetary train type inverted pendulum system.....	54
Figure 3.12: (a)-(d) Final regulation performance of the FLNFN-MODE controller, PID controller, FLNFN-DE controller and FLNFN-GA controller. (e) Scaling curves of the FLNFN-MODE controller and PID controller between the 1.5 <sup>th</sup> second and the 3.5 <sup>th</sup> second.....	56
Figure 3.13: (a)-(d) Tracking of the FLNFN-MODE controller, PID controller, FLNFN-DE controller and FLNFN-GA controller, respectively, for a square wave with amplitude $\pm 0.02$ and frequency 0.5Hz. (e) Tracking curves of the FLNFN-MODE controller and PID controller between the 4 <sup>th</sup> second and the 8 <sup>th</sup> second.....	58
Figure 3.14: Sphere and coil arrangement of the magnetic levitation system.....	59
Figure 3.15: Control block diagram for the magnetic levitation system.....	61
Figure 3.16: Experimental magnetic levitation system.....	62
Figure 3.17: (a)-(d) Experimental results of FLNFN-MODE controller, PID controller, FLNFN-DE controller and FLNFN-GA controller due to periodic sinusoidal command for reference position and actual position, tracking error and control effort.....	64
Figure 3.18: (a)-(d) Experimental results of FLNFN-MODE controller, PID controller,	

FLNFN-DE controller and FLNFN-GA controller due to periodic square command for reference position and actual position, tracking error and control effort. ....	65
Figure 3.19: (a)-(d) Behavior of the FLNFN-MODE controller, PID controller, FLNFN-DE controller and FLNFN-GA controller under impulse noise in a magnetic levitation system for reference and actual positions, tracking error, and control effort. ....	67
Figure 3.20: (a)-(d) Behavior of the FLNFN-MODE controller, PID controller, FLNFN-DE controller, and FLNFN-GA controller when a change occurs in the magnetic levitation system for reference and actual positions, tracking error, and control effort. ....	69
Figure 4.1: Flowchart of the RSMODE learning algorithm. ....	73
Figure 4.2: Coding a fuzzy rule into an individual in the RSMODE learning algorithm. ....	74
Figure 4.3: Structure of the individual in the RSMODE learning algorithm. ....	77
Figure 4.4: Illustration of the MODE process for 8-dimensional vector. ....	79
Figure 4.5: A mutation operation in the rule-based symbiotic modified differential evolution. ....	81
Figure 4.6: Learning curves of best performance of the FLNFN-RSMODE, FLNFN-RSDE, FLNFN-DE and FLNFN-GA in Example 1. ....	83
Figure 4.7: (a) Final regulation performance of FLNFN-RSMODE controller in water bath system. (b) Error curves of the FLNFN-RSMODE controller, FLNFN-RSDE controller, the FLNFN-DE controller, and FLNFN-GA controller. ....	83
Figure 4.8: (a) Behavior of FLNFN-RSMODE controller under impulse noise in water bath system. (b) Error curves of FLNFN-RSMODE controller, FLNFN-RSDE controller, the FLNFN-DE controller and FLNFN-GA controller. ....	84
Figure 4.9: (a) Behavior of FLNFN-RSMODE controller when a change occurs in the water bath system. (b) Error curves of FLNFN-RSMODE controller, FLNFN-RSDE controller, the FLNFN-DE controller, and FLNFN-GA controller. ....	85
Figure 4.10: (a) Tracking of FLNFN-RSMODE controller when a change occurs in the water bath system. (b) Error curves of FLNFN-RSMODE controller, FLNFN-RSDE controller, the FLNFN-DE controller, and FLNFN-GA controller. ....	86
Figure 4.11: Learning curves of best performance of the FLNFN-RSMODE, FLNFN-RSDE, FLNFN-DE, and FLNFN-GA in Example 2. ....	87
Figure 4.12: Responses of ball and beam system controlled by FLNFN-RSMODE controller (solid curves) and FLNFN-RSDE controller (dotted curves) under four initial conditions. ....	88
Figure 4.13: Responses of four states of the ball and beam system under the control of the FLNFN-RSMODE controller. ....	88
Figure 4.14: Diagram of simulated truck and loading zone. ....	89

Figure 4.15: Learning curves of best performance of the FLNFN-RSMODE, FLNFN-RSDE, FLNFN-DE and FLNFN-GA in Example 3. ....91

Figure 4.16: The moving trajectories of the truck where the solid curves represent the six sets of training trajectories and the dotted curves represent the moving trajectories of the truck under the FLNFN-RSMODE controller. ....91

Figure 4.17: Trajectories of truck, starting at four initial positions under the control of the FLNFN-RSMODE after learning using training trajectories. ....92



# Chapter 1

## Introduction

### 1.1 Motivation

In the field of artificial intelligence, neural networks are essentially low-level computational structures and algorithms that offer good performance when they deal with sensory data. However, it is difficult to understand the meaning of each neuron and each weight in the networks. Generally, fuzzy systems are easy to appreciate because they use linguistic terms and if-then rules. However, they lack the learning capacity to fine-tune fuzzy rules and membership functions. Therefore, neuro-fuzzy networks combine the benefits of neural networks and fuzzy systems to solve many engineering problems. Neuro-fuzzy networks bring the low-level learning and computational power of neural networks into fuzzy systems and give the high-level human-like thinking and reasoning of fuzzy systems to neural networks.

Recently, neuro-fuzzy networks have become popular topics of research, and are applied in many areas, such as prediction, control, identification, recognition, decision-making, etc. Neuro-fuzzy networks have some significant issues including how to design an adaptive neuro-fuzzy network and how to design an effective learning algorithm. Therefore, we propose a functional-link-based neuro-fuzzy network (FLNFN) and its related learning algorithms in this dissertation. The proposed FLNFN model, which combines a neuro-fuzzy

network with a functional link neural network, is designed to improve the accuracy of functional approximation. Each fuzzy rule that corresponds to a functional link neural network consists of a functional expansion of input variables. The consequent part of the proposed model is a nonlinear combination of input variables. Hence, the local properties of the consequent part in the FLNFN model enable a nonlinear combination of input variables to be approximated more effectively.

Training of the parameters is the main problem in designing a neuro-fuzzy network. Backpropagation (BP) training is commonly adopted to solve this problem. It is a powerful training technique that can be applied to networks with a forward structure. Since the steepest descent approach is used in BP training to minimize the error function, the algorithms may reach the local minima very quickly and never find the global solution. The aforementioned disadvantages lead to suboptimal performance, even for a favorable neuro-fuzzy network topology. Therefore, technologies, that can be used to train the system parameters and find the global solution while optimizing the overall structure, are required. Next, we propose a rule-based symbiotic modified differential evolution (RSMODE) for the proposed FLNFN model. The RSMODE can automatically determine the number of fuzzy rules and generate initial subpopulation. Furthermore, each individual in each subpopulation evolves separately using a modified differential evolution (MODE). The proposed MODE adopts a method to effectively search between the best individual and randomly chosen individuals. Finally, the proposed FLNFN model is applied in various control problems and practical applications. Results of this dissertation demonstrate the effectiveness of the proposed method.

## 1.2 Literature Survey

Recently, neuro-fuzzy networks [1]-[20] provide the advantages of both neural networks and fuzzy systems, unlike pure neural networks or fuzzy systems alone. Neuro-fuzzy networks

(NFN) bring the low-level learning and computational power of neural networks into fuzzy systems and give the high-level human-like thinking and reasoning of fuzzy systems to neural networks.

Two typical types of neuro-fuzzy networks are the Mamdani-type and the Takagi-Sugeno-Kang (TSK)-type. For Mamdani-type neuro-fuzzy networks [4]-[6], the minimum fuzzy implication is adopted in fuzzy reasoning. For TSK-type neuro-fuzzy networks (TSK-type NFN) [7]-[10], the consequence part of each rule is a linear combination of input variables. Many researchers [9]-[10] have shown that TSK-type neuro-fuzzy networks offer better network size and learning accuracy than Mamdani-type neuro-fuzzy networks. In the typical TSK-type neuro-fuzzy network, which is a linear polynomial of input variables, the model output is approximated locally by the rule hyper-planes. Nevertheless, the traditional TSK-type neuro-fuzzy network does not take full advantage of the mapping capabilities that may be offered by the consequent part.

Introducing a nonlinear function, especially a neural structure, to the consequent part of the fuzzy rules has yielded the NARA [21] and the CANFIS [22] models. These models [21]-[22] apply multilayer neural networks to the consequent part of the fuzzy rules. Although the interpretability of the model is reduced, the representational capability of the model is markedly improved. However, the multilayer neural network has such disadvantages as slower convergence and greater computational complexity. Therefore, this dissertation uses the functional link neural network (FLNN) [23]-[25] to the consequent part of the fuzzy rules, called a functional-link-based neuro-fuzzy network (FLNFN). The consequent part of the proposed FLNFN model is a nonlinear combination of input variables, which differs from the other existing models [5], [9]-[10]. The FLNN is a single layer neural structure capable of forming arbitrarily complex decision regions by generating nonlinear decision boundaries with nonlinear functional expansion. The FLNN [26] was conveniently used for function approximation and pattern classification with faster convergence rate and less computational

loading than a multilayer neural network. Moreover, using the functional expansion can effectively increase the dimensionality of the input vector, so the hyper-planes generated by the FLNN will provide a good discrimination capability in input data space.

In addition, training of the parameters is the main problem in designing a neuro-fuzzy network. Backpropagation (BP) training is commonly adopted to solve this problem. It is a powerful training technique that can be applied to networks with a forward structure. Since the steepest descent approach is used in BP training to minimize the error function, the algorithms may reach the local minima very quickly and never find the global solution. The aforementioned disadvantages lead to suboptimal performance, even for a favorable neuro-fuzzy network topology. Therefore, technologies, that can be used to train the system parameters and find the global solution while optimizing the overall structure, are required.

Recent development in genetic algorithms (GAs) has provided a method for neuro-fuzzy system design. Genetic fuzzy systems (GFSSs) [27]-[31] hybridize the approximate reasoning of fuzzy systems with the learning capability of genetic algorithms. GAs represent highly effective techniques for evaluating system parameters and finding global solutions while optimizing the overall structure. Thus, many researchers have developed GAs to implement fuzzy systems and neuro-fuzzy systems in order to automate the determination of structures and parameters [32]-[52].

Carse *et al.* [32] presented a GA-based approach to employ variable length rule sets and simultaneously evolves fuzzy membership functions and relations called Pittsburgh-style fuzzy classifier system. Herrera *et al.* [33] proposed a genetic algorithm-based tuning approach for the parameters of membership functions used to define fuzzy rules. This approach relied on a set of input-output training data and minimized a squared-error function defined in terms of the training data. Homaifar and McCormick [34] presented a method that simultaneously found the consequents of fuzzy rules and the center points of triangular membership functions in the antecedent using genetic algorithms. Velasco [35] described a

Michigan approach which generates a special place where rules can be tested to avoid the use of bad rules for online genetic learning. Ishibuchi *et al.* [36] applied a Michigan-style genetic fuzzy system to automatically generate fuzzy IF-THEN rules for designing compact fuzzy rule-based classification systems. The genetic learning process proposed is based on the iterative rule learning approach and it can automatically design fuzzy rule-based systems by Cordon *et al.* [37]. A GA-based learning algorithm called structural learning algorithm in a vague environment (SLAVE) was proposed in [38]. SLAVE used an iterative approach to include more information in the process of learning one individual rule. Furthermore, a very interesting algorithm was proposed by Russo in [39] which attempted to combine all good features of fuzzy systems, neural networks and genetic algorithm for fuzzy model derivation from input-output data. Chung *et al.* [40] adopted both neural networks and GAs to automatically determine the parameters of fuzzy logic systems. They utilized a feedforward neural network for realizing the basic elements and functions of a fuzzy controller. In [41], a hybrid of evolution strategies and simulated annealing algorithms is employed to optimize membership function parameters and rule numbers which are combined with genetic parameters.

Three main strategies, including Pittsburgh-type, Michigan-type, and the iterative rule learning genetic fuzzy systems, focus on generating and learning fuzzy rules in genetic fuzzy systems. First, the Pittsburgh-type genetic fuzzy system [42] was characterized by using a fuzzy system as an individual in genetic operators. Second, the Michigan-type genetic fuzzy system was used for generating fuzzy rules in [43], where each fuzzy rule was treated as an individual. Thus, the rule generation methods in [43] were referred to as fuzzy classifier systems. Third, the iterative rule learning genetic fuzzy system [44] was adopted to search one adequate rule set for each iteration of the learning process. Moreover, Ishibuchi *et al.* [45]-[48] proposed genetic algorithms for constructing a fuzzy system consisting of a small number of linguistic rules. Mitra *et al.* [49]-[52] presented some approaches that exploit the benefits of



soft computation tools for rule generation.

In the aforementioned literatures, it has been fully demonstrated that GAs are very powerful in searching for the true profile. However, the search is extremely time-consuming, which is one of the basic disadvantages of all GAs. Although the convergence in some special cases can be improved by hybridizing GAs with some local search algorithms, it is achieved at the expense of the versatility and simplicity of the algorithm. Similar to GAs, DE [53]-[55] also belongs to the broad class of evolutionary algorithms, but DE has many advantages such as the strong search ability and the fast convergence ability over GAs or any other traditional optimization approach, especially for real valued problems [55]. Therefore, we propose a rule-based symbiotic modified differential evolution (RSMODE) for the proposed FLNFN model. The RSMODE is to adjust the system parameters and find the global solution while optimizing the overall structure.

### **1.3 Organization of Dissertation**

The overall objective of this dissertation is to develop a novel neuro-fuzzy network and its related learning algorithm. Organization and objectives of each chapter in this dissertation are as follows.

In Chapter 2, we propose a functional-link-based neuro-fuzzy network (FLNFN) structure for nonlinear system control. The proposed FLNFN model uses a functional link neural network (FLNN) to the consequent part of the fuzzy rules. This dissertation uses orthogonal polynomials and linearly independent functions in a functional expansion of the FLNN. Thus, the consequent part of the proposed FLNFN model is a nonlinear combination of input variables. An online learning algorithm, which consists of structure learning and parameter learning, is also presented. The structure learning depends on the entropy measure to determine the number of fuzzy rules. The parameter learning, based on the gradient descent

method, can adjust the shape of the membership function and the corresponding weights of the FLNN.

In Chapter 3, we present a modified differential evolution (MODE) for the proposed FLNFN model. The proposed MODE learning algorithm adopts an evolutionary learning method to optimize the FLNFN parameters. The MODE algorithm uses a method to effectively search toward the current best individual. Furthermore, the MODE algorithm also provides a cluster-based mutation scheme, which maintains useful diversity in the population to increase the search capability.

In Chapter 4, we propose a rule-based symbiotic modified differential evolution (RSMODE) for the proposed FLNFN model. The proposed RSMODE learning algorithm consists of initialization phase and parameter learning phase. The initialization phase can determine the number of subpopulation which satisfies the fuzzy partition of input variables using the entropy measure. The parameter learning phase combines two strategies including a subpopulation symbiotic evolution and a modified differential evolution. The RSMODE can automatically generate initial subpopulation and each individual in each subpopulation evolves separately using a modified differential evolution. We also compare our method with other methods in the literature early. Finally, conclusions and future works are summarized in the last section.

## Chapter 2

### A Functional-Link-Based Neuro-Fuzzy Network

In this chapter, a functional-link-based neuro-fuzzy network (FLNFN) model is presented for nonlinear system control. The FLNFN model, which combines a neuro-fuzzy network with a functional link neural network (FLNN), is designed to improve the accuracy of functional approximation. Each fuzzy rule that corresponds to a FLNN consists of a functional expansion of input variables. The orthogonal polynomials and linearly independent functions are adopted as functional link neural network bases. An online learning algorithm, consisting of structure learning and parameter learning, is proposed to construct the FLNFN model automatically. The structure learning algorithm determines whether or not to add a new node which satisfies the fuzzy partition of input variables. Initially, the FLNFN model has no rules. The rules are automatically generated from training data by entropy measure. The parameter learning algorithm is based on back-propagation to tune the free parameters in the FLNFN model simultaneously to minimize an output error function.

#### 2.1 Structure of Functional-Link-Based Neuro-Fuzzy Network

This section describes the structure of functional link neural networks and the structure of the FLNFN model. In functional link neural networks, the input data usually incorporate high order effects and thus artificially increase the dimensions of the input space using a functional

expansion. Accordingly, the input representation is enhanced and linear separability is achieved in the extended space. The FLNFN model adopted the functional link neural network generating complex nonlinear combination of input variables to the consequent part of the fuzzy rules. The rest of this section details these structures.

### 2.1.1 Functional Link Neural Networks

The functional link neural network is a single layer network in which the need for hidden layers is removed. While the input variables generated by the linear links of neural networks are linearly weighted, the functional link acts on an element of input variables by generating a set of linearly independent functions (i.e., the use of suitable orthogonal polynomials for a functional expansion), and then evaluating these functions with the variables as the arguments. Therefore, the FLNN structure considers trigonometric functions. For example, for a two-dimensional input  $\mathbf{X}=[x_1, x_2]^T$ , the enhanced input is obtained using trigonometric functions in  $\Phi=[x_1, \sin(\pi x_1), \cos(\pi x_1), \dots, x_2, \sin(\pi x_2), \cos(\pi x_2), \dots]^T$ . Thus, the input variables can be separated in the enhanced space [23]. In the FLNN structure with reference to Fig. 2.1, a set of basis functions  $\Phi$  and a fixed number of weight parameters  $\mathbf{W}$  represent  $f_{\mathbf{w}}(x)$ . The theory behind the FLNN for multidimensional function approximation has been discussed elsewhere [24] and is analyzed below.

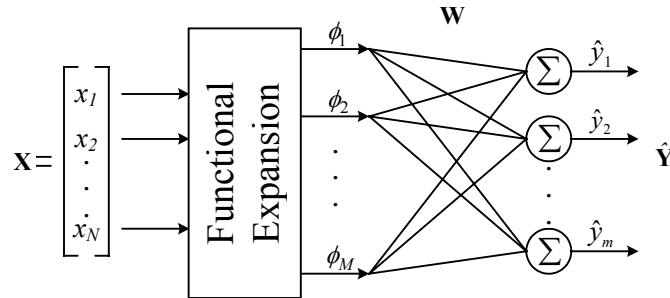


Figure 2.1: Structure of FLNN.

Consider a set of basis functions  $\mathbf{B} = \{\phi_k \in \Phi(A)\}_{k \in \mathbf{K}}$ ,  $\mathbf{K} = \{1, 2, \dots\}$  with the following

properties; 1)  $\phi_1 = 1$ , 2) the subset  $B_j = \{\phi_k \in \mathbf{B}\}_{k=1}^M$  is a linearly independent set, meaning that if  $\sum_{k=1}^M w_k \phi_k = 0$ , then  $w_k = 0$  for all  $k = 1, 2, \dots, M$ , and 3)  $\sup_j \left[ \sum_{k=1}^j \|\phi_k\|_A^2 \right]^{1/2} < \infty$ .

Let  $\mathbf{B} = \{\phi_k\}_{k=1}^M$  be a set of basis functions to be considered, as shown in Fig. 2.1. The FLNN comprises  $M$  basis functions  $\{\phi_1, \phi_2, \dots, \phi_M\} \in \mathbf{B}_M$ . The linear sum of the  $j$ th node is given by

$$\hat{y}_j = \sum_{k=1}^M w_{kj} \phi_k(\mathbf{X}) \quad (2.1)$$

where  $\mathbf{X} \in A \subset \mathfrak{R}^N$ ,  $\mathbf{X} = [x_1, x_2, \dots, x_N]^T$  is the input vector and  $\mathbf{W}_j = [w_{j1}, w_{j2}, \dots, w_{jM}]^T$  is the weight vector associated with the  $j$ th output of the FLNN.  $\hat{y}_j$  denotes the local output of the FLNN structure and the consequent part of the  $j$ th fuzzy rule in the FLNFN model. Thus, Eq.(2.1) can be expressed in matrix form as  $\hat{y}_j = \mathbf{W}_j \Phi$ , where  $\Phi = [\phi_1(x), \phi_2(x), \dots, \phi_N(x)]^T$  is the basis function vector, which is the output of the functional expansion block. The  $m$ -dimensional linear output may be given by  $\hat{\mathbf{Y}} = \mathbf{W} \Phi$ , where  $\hat{\mathbf{Y}} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_m]^T$ ,  $m$  denotes the number of functional link bases, which equals the number of fuzzy rules in the FLNFN model, and  $\mathbf{W}$  is a  $(m \times M)$ -dimensional weight matrix of the FLNN given by  $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_M]^T$ . In the FLNFN model, the corresponding weights of functional link bases do not exist in the initial state, and the amount of the corresponding weights of functional link bases generated by the online learning algorithm is consistent with the number of fuzzy rules. Section 3 details the online learning algorithm.

### 2.1.2 Structure of the FLNFN Model

This subsection describes the FLNFN model, which uses a nonlinear combination of input variables (FLNN). Each fuzzy rule corresponds to a sub-FLNN, comprising a functional link. Figure 2.2 presents the structure of the proposed FLNFN model.

The FLNFN model realizes a fuzzy if-then rule in the following form.

*Rule-j:* IF  $x_1$  is  $A_{1j}$  and  $x_2$  is  $A_{2j}$  ... and  $x_i$  is  $A_{ij}$  ... and  $x_N$  is  $A_{Nj}$

$$\begin{aligned} \text{THEN } \hat{y}_j &= \sum_{k=1}^M w_{kj} \phi_k \\ &= w_{1j} \phi_1 + w_{2j} \phi_2 + \dots + w_{Mj} \phi_M \end{aligned} \quad (2.2)$$

where  $x_i$  and  $\hat{y}_j$  are the input and local output variables, respectively;  $A_{ij}$  is the linguistic term of the precondition part with Gaussian membership function;  $N$  is the number of input variables;  $w_{kj}$  is the link weight of the local output;  $\phi_k$  is the basis trigonometric function of input variables;  $M$  is the number of basis function, and *Rule-j* is the  $j$ th fuzzy rule.

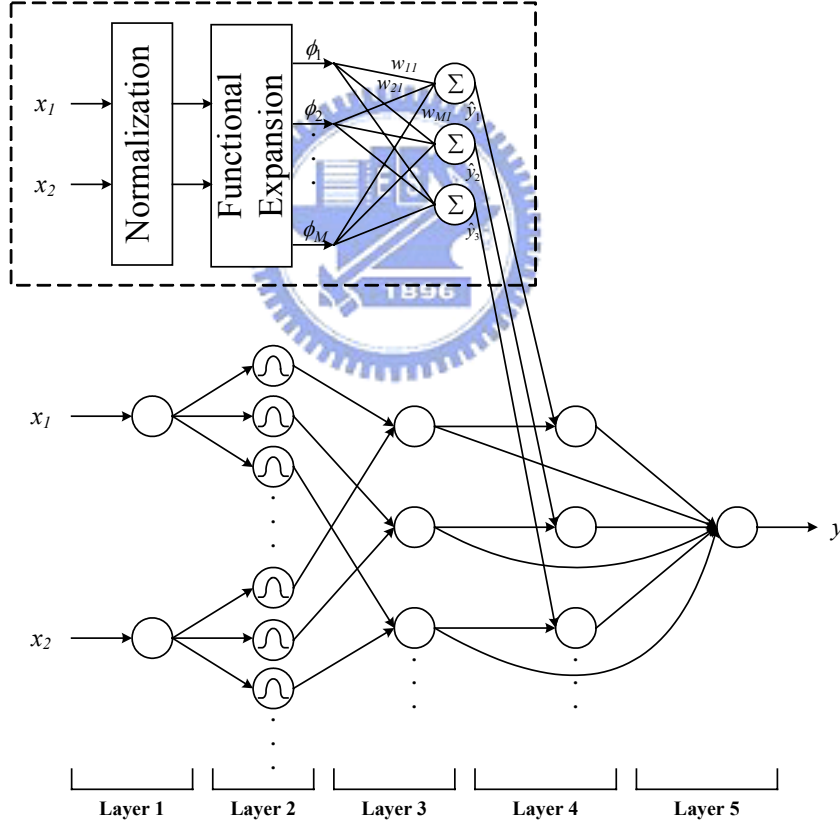


Figure 2.2: Structure of proposed FLNFN model.

The operation functions of the nodes in each layer of the FLNFN model are now described. In the following description,  $u^{(l)}$  denotes the output of a node in the  $l$ th layer.

No computation is performed in layer 1. Each node in this layer only transmits input values to the next layer directly:

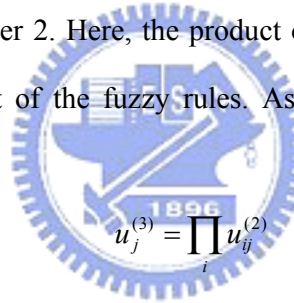
$$u_i^{(1)} = x_i. \quad (2.3)$$

Each fuzzy set  $A_{ij}$  is described here by a Gaussian membership function. Therefore, the calculated membership value in layer 2 is

$$u_{ij}^{(2)} = \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right) \quad (2.4)$$

where  $m_{ij}$  and  $\sigma_{ij}$  are the mean and variance of the Gaussian membership function, respectively, of the  $j$ th term of the  $i$ th input variable  $x_i$ .

Nodes in layer 3 receive one-dimensional membership degrees of the associated rule from the nodes of a set in layer 2. Here, the product operator described above is adopted to perform the precondition part of the fuzzy rules. As a result, the output function of each inference node is



$$u_j^{(3)} = \prod_i u_{ij}^{(2)} \quad (2.5)$$

where the  $\prod_i u_{ij}^{(2)}$  of a rule node represents the firing strength of its corresponding rule.

Nodes in layer 4 are called consequent nodes. The input to a node in layer 4 is the output from layer 3, and the other inputs are calculated from a functional link neural network, as shown in Fig. 2.2. For such a node,

$$u_j^{(4)} = u_j^{(3)} \cdot \sum_{k=1}^M w_{kj} \phi_k \quad (2.6)$$

where  $w_{kj}$  is the corresponding link weight of functional link neural network and  $\phi_k$  is the functional expansion of input variables. The functional expansion uses a trigonometric polynomial basis function, given by  $[\phi_1 \ \phi_2 \ \phi_3 \ \phi_4 \ \phi_5 \ \phi_6] = [x_1 \sin(\pi x_1) \cos(\pi x_1) x_2 \sin(\pi x_2) \cos(\pi x_2)]$  for two-dimensional input variables. Therefore,  $M$  is the number of basis functions,  $M = 3 \times N$ , where  $N$  is the number of input variables.

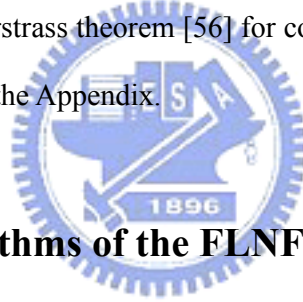
Moreover, the output nodes of functional link neural network depend on the number of fuzzy rules of the FLNFN model.

The output node in layer 5 integrates all of the actions recommended by layers 3 and 4 and acts as a defuzzifier with,

$$y = u^{(5)} = \frac{\sum_{j=1}^R u_j^{(4)}}{\sum_{j=1}^R u_j^{(3)}} = \frac{\sum_{j=1}^R u_j^{(3)} \left( \sum_{k=1}^M w_{kj} \phi_k \right)}{\sum_{j=1}^R u_j^{(3)}} = \frac{\sum_{j=1}^R u_j^{(3)} \hat{y}_j}{\sum_{j=1}^R u_j^{(3)}} \quad (2.7)$$

where  $R$  is the number of fuzzy rules, and  $y$  is the output of the FLNFN model.

As described above, the number of tuning parameters for the FLNFN model is known to be  $(2+3 \times P) \times N \times R$ , where  $N$ ,  $R$  and  $P$  denote the number of inputs, existing rules, and outputs, respectively. The proposed FLNFN model can be demonstrated to be a universal uniform approximation by Stone-Weierstrass theorem [56] for continuous functions over compact sets. The detailed proof is given in the Appendix.



## 2.2 Learning Algorithms of the FLNFN Model

This section presents an online learning algorithm for constructing the FLNFN model. The proposed learning algorithm comprises a structure learning phase and a parameter learning phase. Figure 2.3 presents flow diagram of the learning scheme for the FLNFN model. Structure learning is based on the entropy measure used to determine whether a new rule should be added to satisfy the fuzzy partitioning of input variables. Parameter learning is based on supervised learning algorithms. The back-propagation algorithm minimizes a given cost function by adjusting the link weights in the consequent part and the parameters of the membership functions. Initially, there are no nodes in the network except the input-output nodes, i.e., there are no any nodes in the FLNFN model. The nodes are created automatically as learning proceeds, upon the reception of online incoming training data in the structure and parameter learning processes. The rest of this section details the structure learning phase and



the parameter learning phase. Finally in this section, the stability analysis of the FLNFN model based on the Lyapunov approach is performed the convergence property.

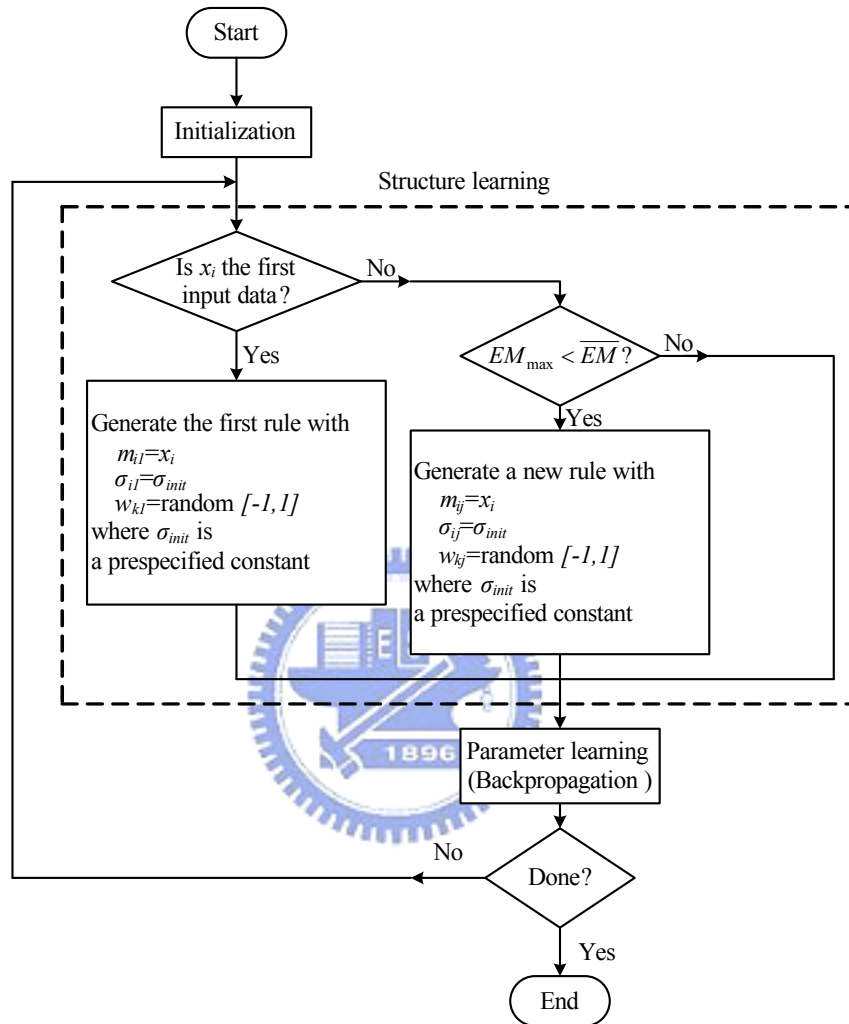


Figure 2.3: Flow diagram of the structure/parameter learning for the FLNFN model.

### 2.2.1 Structure Learning Phase

The first step in structure learning is to determine whether a new rule from should be extracted the training data and to determine the number of fuzzy sets in the universal of discourse of each input variable, since one cluster in the input space corresponds to one potential fuzzy logic rule, in which  $m_{ij}$  and  $\sigma_{ij}$  represent the mean and variance of that cluster, respectively. For each incoming pattern  $x_i$ , the rule firing strength can be regarded as

the degree to which the incoming pattern belongs to the corresponding cluster. Entropy measure between each data point and each membership function is calculated based on a similarity measure. A data point of closed mean will has lower entropy. Therefore, the entropy values between data points and current membership functions are calculated to determine whether or not to add a new rule. For computational efficiency, the entropy measure can be calculated using the firing strength from  $u_{ij}^{(2)}$  as follow;

$$EM_j = -\sum_{i=1}^N D_{ij} \log_2 D_{ij} \quad (2.8)$$

where  $D_{ij} = \exp(u_{ij}^{(2)-1})$  and  $EM_j \in [0,1]$ . According to Eq. (2.8), the measure is used to generate a new fuzzy rule and new functional link bases for new incoming data is described as follows. The maximum entropy measure

$$EM_{\max} = \max_{1 \leq j \leq R(t)} EM_j \quad (2.9)$$

is determined, where  $R(t)$  is the number of existing rules at time  $t$ . If  $EM_{\max} \leq \overline{EM}$ , then a new rule is generated, where  $\overline{EM} \in [0,1]$  is a prespecified threshold that decays during the learning process.

In the structure learning phase, the threshold parameter  $\overline{EM}$  is an important parameter. The threshold is set to between zero and one. A low threshold leads to the learning of coarse clusters (i.e., fewer rules are generated), whereas a high threshold leads to the learning of fine clusters (i.e., more rules are generated). If the threshold value equals zero, then all the training data belong to the same cluster in the input space. Therefore, the selection of the threshold value  $\overline{EM}$  will critically affect the simulation results. As a result of our extensive experiments and by carefully examining the threshold value  $\overline{EM}$ , which uses the range  $[0, 1]$ , we concluded that the relationship between threshold value  $\overline{EM}$  and the number of input variables. Accordingly,  $\overline{EM}$  is defined as 0.26-0.3 times of the number of input variables.

Once a new rule has been generated, the next step is to assign the initial mean and

variance to the new membership function and the corresponding link weight for the consequent part. Since the goal is to minimize an objective function, the mean, variance and weight are all adjustable later in the parameter learning phase. Hence, the mean, variance and weight for the new rule are set as follows;

$$m_{ij}^{(R_{(t+1)})} = x_i \quad (2.10)$$

$$\sigma_{ij}^{(R_{(t+1)})} = \sigma_{init} \quad (2.11)$$

$$w_{kj}^{(R_{(t+1)})} = random[-1, 1] \quad (2.12)$$

where  $x_i$  is the new input and  $\sigma_{init}$  is a prespecified constant. The whole algorithm for the generation of new fuzzy rules and fuzzy sets in each input variable is as follows. No rule is assumed to exist initially exist:

Step 1: IF  $x_i$  is the first incoming pattern THEN do

{Generate a new rule

with mean  $m_{i1}=x_i$ , variance  $\sigma_{i1}=\sigma_{init}$ , weight  $w_{k1}=random[-1, 1]$

where  $\sigma_{init}$  is a prespecified constant.

}

Step 2: ELSE for each newly incoming  $x_i$ , do

{Find  $EM_{max} = \max_{1 \leq j \leq R_{(t)}} EM_j$

IF  $EM_{max} \geq \overline{EM}$

do nothing

ELSE

{ $R_{(t+1)} = R_{(t)} + 1$

generate a new rule

with mean  $m_{iR_{(t+1)}} = x_i$ , variance  $\sigma_{iR_{(t+1)}} = \sigma_{init}$ , weight  $w_{kR_{(t+1)}} = random[-1, 1]$

where  $\sigma_{init}$  is a prespecified constant. }

}

### 2.2.2 Parameter Learning Phase

After the network structure has been adjusted according to the current training data, the network enters the parameter learning phase to adjust the parameters of the network optimally based on the same training data. The learning process involves determining the minimum of a given cost function. The gradient of the cost function is computed and the parameters are adjusted with the negative gradient. The back-propagation algorithm is adopted for this supervised learning method. When the single output case is considered for clarity, the goal to minimize the cost function  $E$  is defined as

$$E(t) = \frac{1}{2} [y(t) - y^d(t)]^2 = \frac{1}{2} e^2(t) \quad (2.13)$$

where  $y^d(t)$  is the desired output and  $y(t)$  is the model output for each discrete time  $t$ . In each training cycle, starting at the input variables, a forward pass is adopted to calculate the activity of the model output  $y(t)$ .

When the back-propagation learning algorithm is adopted, the weighting vector of the FLNFN model is adjusted such that the error defined in Eq. (2.13) is less than the desired threshold value after a given number of training cycles. The well-known back-propagation learning algorithm may be written briefly as

$$W(t+1) = W(t) + \Delta W(t) = W(t) + \left( -\eta \frac{\partial E(t)}{\partial W(t)} \right) \quad (2.14)$$

where, in this case,  $\eta$  and  $W$  represent the learning rate and the tuning parameters of the FLNFN model, respectively. Let  $\mathbf{W} = [m, \sigma, w]^T$  be the weighting vector of the FLNFN model. Then, the gradient of error  $E(\cdot)$  in Eq. (2.13) with respect to an arbitrary weighting vector  $W$  is

$$\frac{\partial E(t)}{\partial W} = e(t) \frac{\partial y(t)}{\partial W}. \quad (2.15)$$

Recursive applications of the chain rule yield the error term for each layer. Then the

parameters in the corresponding layers are adjusted. With the FLNFN model and the cost function as defined in Eq. (2.13), the update rule for  $w_{kj}$  can be derived as follows;

$$w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj}(t) \quad (2.16)$$

where

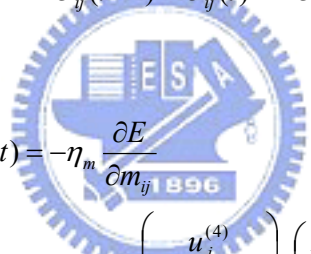
$$\begin{aligned} \Delta w_{kj}(t) &= -\eta_w \frac{\partial E}{\partial w_{kj}} \\ &= -\eta_w \cdot e \cdot \left( \frac{u_j^{(3)} \phi_k}{\sum_{j=1}^R u_j^{(3)}} \right). \end{aligned}$$

Similarly, the update laws for  $m_{ij}$ , and  $\sigma_{ij}$  are

$$m_{ij}(t+1) = m_{ij}(t) + \Delta m_{ij}(t) \quad (2.17)$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \Delta \sigma_{ij}(t) \quad (2.18)$$

where



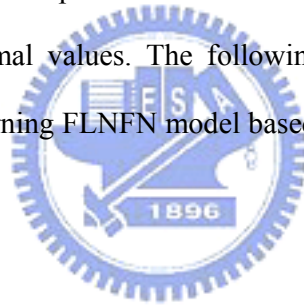
$$\begin{aligned} \Delta m_{ij}(t) &= -\eta_m \frac{\partial E}{\partial m_{ij}} \\ &= -\eta_m \cdot e \cdot \left( \frac{u_j^{(4)}}{\sum_{j=1}^R u_j^{(3)}} \right) \cdot \left( \frac{2(u_i^{(1)} - m_{ij})}{\sigma_{ij}^2} \right) \\ \Delta \sigma_{ij}(t) &= -\eta_\sigma \frac{\partial E}{\partial \sigma_{ij}} \\ &= -\eta_\sigma \cdot e \cdot \left( \frac{u_j^{(4)}}{\sum_{j=1}^R u_j^{(3)}} \right) \cdot \left( \frac{2(u_i^{(1)} - m_{ij})^2}{\sigma_{ij}^3} \right) \end{aligned}$$

where  $\eta_w$ ,  $\eta_m$  and  $\eta_\sigma$  are the learning rate parameters of the weight, the mean, and the variance, respectively. In this dissertation, both the link weights in the consequent part and the parameters of the membership functions in the precondition part are adjusted by using the back-propagation algorithm. Recently, many researchers [10], [57] tuned the consequent parameters using either least mean squares (LMS) or recursive least squares (RLS) algorithms to obtain optimal parameters. However, they still used the back-propagation algorithm to

adjust the precondition parameters.

### **2.2.3 Convergence Analysis**

The selection of suitable learning rates is very important. If the learning rate is small, convergence will be guaranteed. In this case, the speed of convergence may be slow. However, the learning rate is large, and then the system may become unstable. The Appendix derives varied learning rates, which guarantee convergence of the output error based on the analyses of a discrete Lyapunov function, to train the FLNFN model effectively. The convergence analyses in this dissertation are performed to derive specific learning rate parameters for specific network parameters to ensure the convergence of the output error [58]-[59]. Moreover, the guaranteed convergence of output error does not imply the convergence of the learning rate parameters to their optimal values. The following simulation results demonstrate the effectiveness of the online learning FLNFN model based on the proposed delta adaptation law and varied learning rates.



## **2.3 Experimental Results**

This dissertation demonstrated the performance of the FLNFN model for nonlinear system control. This section simulates various control examples and compares the performance of the FLNFN model with that of other models. The FLNFN model is adopted to design controllers in four simulations of nonlinear system control problems - water bath temperature control system [60], control of a bounded input bounded output (BIBO) nonlinear plant [58], control of the ball and beam system [61], and multi-input multi-output (MIMO) plant control [62].

### **Example 1: Control of Water Bath Temperature System**

The goal of this section is to elucidate the control of the temperature of a water bath

system according to,

$$\frac{dy(t)}{dt} = \frac{u(t)}{C} + \frac{Y_0 - y(t)}{T_R C} \quad (2.19)$$

where  $y(t)$  is the output temperature of the system in  $^{\circ}C$  ;  $u(t)$  is the heat flowing into the system;  $Y_0$  is room temperature;  $C$  is the equivalent thermal capacity of the system, and  $T_R$  is the equivalent thermal resistance between the borders of the system and the surroundings.

$T_R$  and  $C$  are assumed to be essentially constant, and the system in Eq.(2.19) is rewritten in discrete-time form to some reasonable approximation. The system

$$y(k+1) = e^{-\alpha T_s} y(k) + \frac{\delta}{1 + e^{0.5y(k)-40}} u(k) + [1 - e^{-\alpha T_s}] y_0 \quad (2.20)$$

is obtained, where  $\alpha$  and  $\delta$  are some constant values of  $T_R$  and  $C$ . The system parameters used in this example are  $\alpha = 1.0015e^{-4}$ ,  $\delta = 8.67973e^{-3}$  and  $Y_0 = 25.0(^{\circ}C)$ , which were obtained from a real water bath plant considered elsewhere [60]. The plant input  $u(k)$  is limited to 0 and 5V, and the sampling period is  $T_s = 30$  second.

The conventional online training scheme is adopted for online training. Figure 2.4 presents a block diagram for the conventional online training scheme. This scheme has two phases - the training phase and the control phase. In the training phase, the switches S1 and S2 are connected to nodes 1 and 2, respectively, to form a training loop. In this loop, we can define a training data with input vector  $I(k) = [y_p(k+1) \ y_p(k)]$  and desired output  $u(k)$ , where the input vector of the FLNFN controller is the same as that used in the general inverse modeling [63] training scheme. In the control phase, the switches S1 and S2 are connected to nodes 3 and 4, respectively, forming a control loop. In this loop, the control signal  $\hat{u}(k)$  is generated according to the input vector  $I'(k) = [y_{ref}(k+1) \ y_p(k)]$ , where  $y_p$  is the plant output and  $y_{ref}$  is the reference model output.

A sequence of random input signals  $u_{rd}(k)$  limited to 0 and 5V is injected directly into the

simulated system described in Eq. (2.20), using the online training scheme for the FLNFN controller. The 120 training patterns are selected based on the input-outputs characteristics to cover the entire reference output. The temperature of the water is initially  $25^{\circ}c$ , and rises progressively when random input signals are injected. After 10000 training iterations, four fuzzy rules are generated. The obtained fuzzy rules are as follows.

*Rule-1:* IF  $x_1$  is  $\mu(32.416,11.615)$  and  $x_2$  is  $\mu(27.234,7.249)$

$$\text{THEN } \hat{y}_1 = 32.095x_1 + 74.849 \sin(\pi x_1) - 34.546 \cos(\pi x_1) \\ - 17.026x_2 - 41.799 \sin(\pi x_2) + 35.204 \cos(\pi x_2)$$

*Rule-2:* IF  $x_1$  is  $\mu(34.96,9.627)$  and  $x_2$  is  $\mu(46.281,13.977)$

$$\text{THEN } \hat{y}_2 = 21.447x_1 + 11.766 \sin(\pi x_1) - 77.705 \cos(\pi x_1) \\ - 52.923x_2 - 61.827 \sin(\pi x_2) + 70.946 \cos(\pi x_2)$$

*Rule-3:* IF  $x_1$  is  $\mu(62.771,6.910)$  and  $x_2$  is  $\mu(62.499,15.864)$

$$\text{THEN } \hat{y}_3 = 25.735x_1 - 10.907 \sin(\pi x_1) - 46.359 \cos(\pi x_1) \\ - 40.322x_2 + 36.752 \sin(\pi x_2) + 103.33 \cos(\pi x_2)$$

*Rule-4:* IF  $x_1$  is  $\mu(79.065,8.769)$  and  $x_2$  is  $\mu(64.654,9.097)$

$$\text{THEN } \hat{y}_4 = 46.055x_1 - 37.223 \sin(\pi x_1) - 57.759 \cos(\pi x_1) \\ - 5.8152x_2 + 61.065 \sin(\pi x_2) + 34.838 \cos(\pi x_2)$$

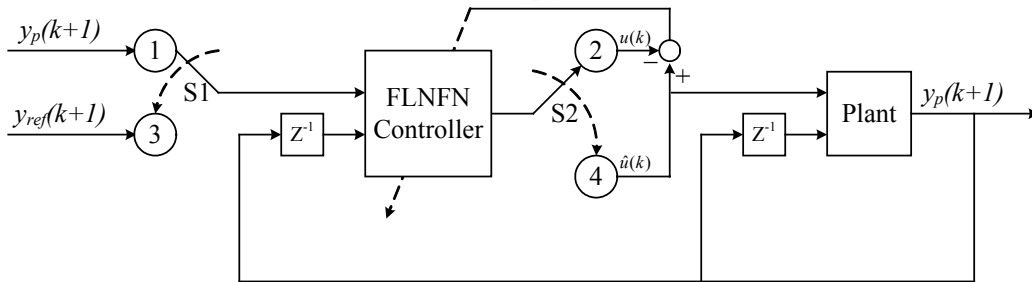


Figure 2.4: Conventional online training scheme.

This dissertation compares the FLNFN controller to the proportional-integral-derivative (PID) controller [64], the manually designed fuzzy controller [1], the functional link neural network [25] and the TSK-type neuro-fuzzy network (TSK-type NFN) [9]. Each of these controllers is applied to the water bath temperature control system. The performance measures include the set-points regulation, the influence of impulse noise, and a large parameter



variation in the system, and the tracking capability of the controllers.

The first task is to control the simulated system to follow three set-points.

$$y_{ref}(k) = \begin{cases} 35^{\circ}c, & \text{for } k \leq 40 \\ 55^{\circ}c, & \text{for } 40 < k \leq 80 \\ 75^{\circ}c, & \text{for } 80 < k \leq 120. \end{cases} \quad (2.21)$$

Figure 2.5(a) presents the regulation performance of the FLNFN controller. The regulation performance was also tested using the FLNN controller and the TSK-type NFN controller. Figure 2.5(b) plots the error curves of the FLNFN controller, the FLNN controller and the TSK-type NFN controller between  $k=81$  and  $k=100$ . In this figure, the FLNFN controller obtains smaller errors than the other two controllers. To test their regulation performance, a performance index, the sum of absolute error (SAE), is defined by

$$SAE = \sum_k |y_{ref}(k) - y(k)| \quad (2.22)$$

where  $y_{ref}(k)$  and  $y(k)$  are the reference output and the actual output of the simulated system, respectively. The SAE values of the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller and the TSK-type NFN controller are 352.8, 418.5, 401.5, 379.2 and 361.9, which values are given in the second row of Table 2.1. The proposed FLNFN controller has a much better SAE value of regulation performance than the other controllers.

The second set of simulations is performed to elucidate the noise-rejection ability of the five controllers when some unknown impulse noise is imposed on the process. One impulse noise value  $-5^{\circ}C$  is added to the plant output at the 60<sup>th</sup> sampling instant. A set-point of  $50^{\circ}C$  is adopted in this set of simulations. For the FLNFN controller, the same training scheme, training data and learning parameters as were used in the first set of simulations. Figure 2.6(a) and (b) present the behaviors of the FLNFN controller under the influence of impulse noise, and the corresponding errors, respectively. The SAE values of the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller and the TSK-type

NFN controller are 270.4, 311.5, 275.8, 324.51 and 274.75, which are shown in the third row of Table 2.1. The FLNFN controller performs quite well. It recovers very quickly and steadily after the occurrence of the impulse noise.

One common characteristic of many industrial-control processes is that their parameters tend to change in an unpredictable way. The value of  $0.7 * u(k-2)$  is added to the plant input after the 60<sup>th</sup> sample in the third set of simulations to test the robustness of the five controllers. A set-point of  $50^{\circ}\text{C}$  is adopted in this set of simulations. Figure 2.7(a) presents the behaviors of the FLNFN controller when in the plant dynamics change. Figure 2.7(b) presents the corresponding errors of the FLNFN controller, the FLNN controller and the TSK-type NFN controllers. The SAE values of the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller and the TSK-type NFN controller are 263.3, 322.2, 273.5, 311.5 and 265.4, which values are shown in the fourth row of Table 2.1. The results present the favorable control and disturbance rejection capabilities of the trained FLNFN controller in the water bath system.

In the final set of simulations, the tracking capability of the FLNFN controller with respect to ramp-reference signals is studied. Define

$$y_{ref}(k) = \begin{cases} 34^{\circ}\text{C} & \text{for } k \leq 30 \\ (34 + 0.5 * (k - 30))^{\circ}\text{C} & \text{for } 30 < k \leq 50 \\ (44 + 0.8 * (k - 50))^{\circ}\text{C} & \text{for } 50 < k \leq 70 \\ (60 + 0.5 * (k - 70))^{\circ}\text{C} & \text{for } 70 < k \leq 90 \\ 70^{\circ}\text{C} & \text{for } 90 < k \leq 120 \end{cases} . \quad (2.23)$$

Figure 2.8(a) presents the tracking performance of the FLNFN controller. Figure 2.8(b) presents the corresponding errors of the FLNFN controller, the FLNN controller, and the TSK-type NFN controller. The SAE values of the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller, and the TSK-type NFN controller are 44.2, 100.6, 88.1, 98.4 and 54.2, which are shown in the fifth row of Table 2.1. The results present the favorable control and disturbance rejection capabilities of the trained FLNFN controller in the water

bath system. The aforementioned simulation results, presented in Table 2.1, demonstrate that the proposed FLNFN controller outperforms other controllers.

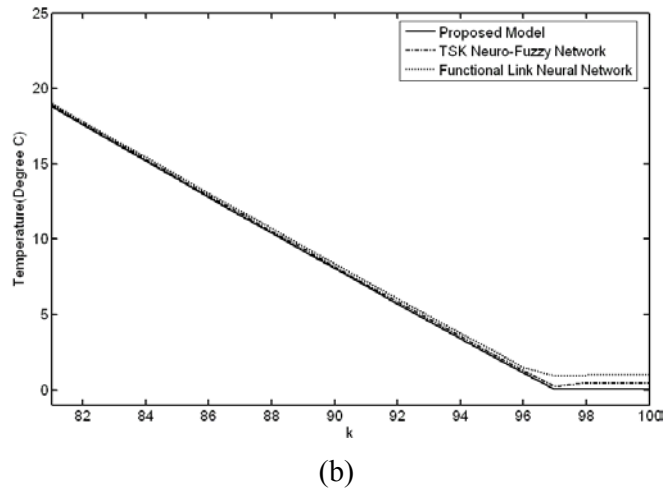
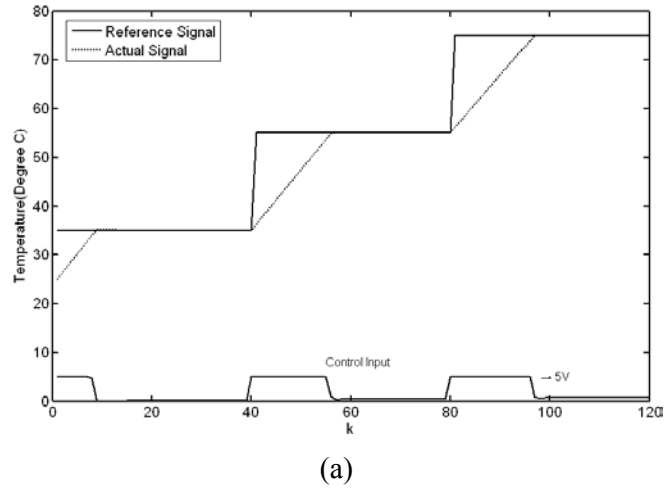
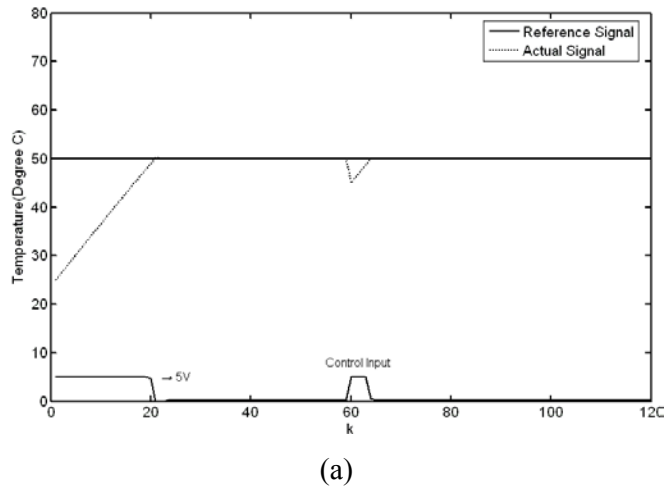
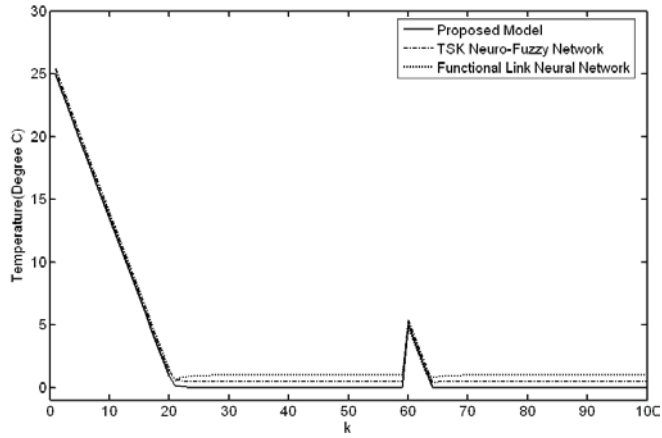


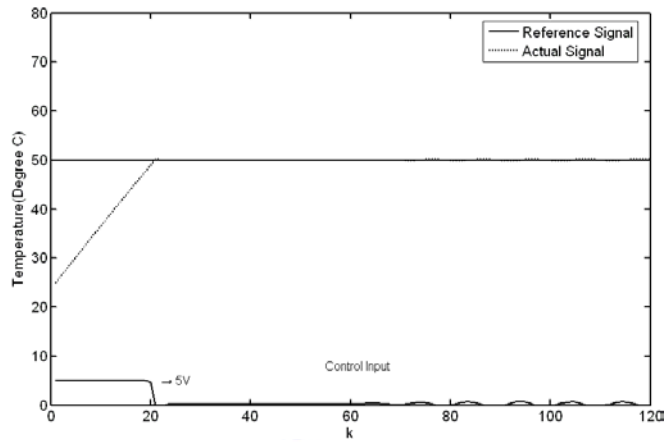
Figure 2.5: (a) Final regulation performance of FLNFN controller in water bath system. (b) Error curves of the FLNFN controller, TSK-type NFN controller and FLNN controller between  $k=81$  and  $k=100$ .



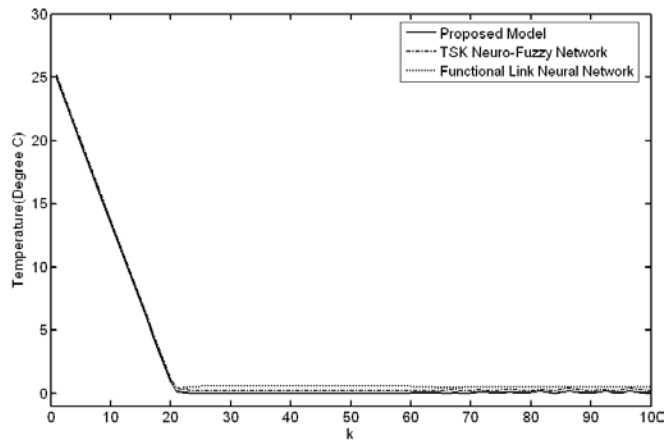


(b)

Figure 2.6: (a) Behavior of FLNFN controller under impulse noise in water bath system. (b) Error curves of FLNFN controller, TSK-type NFN controller and FLNN controller.

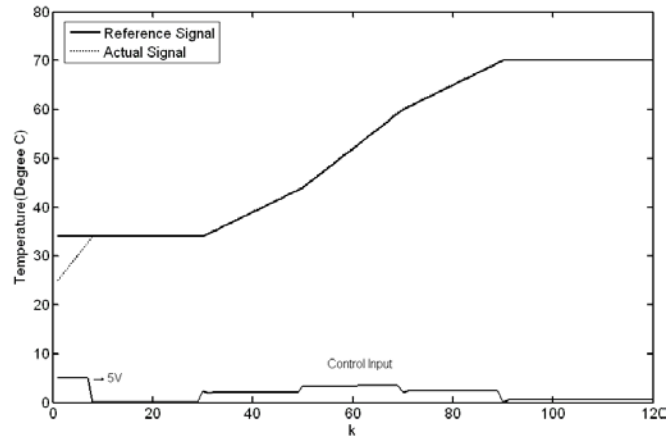


(a)

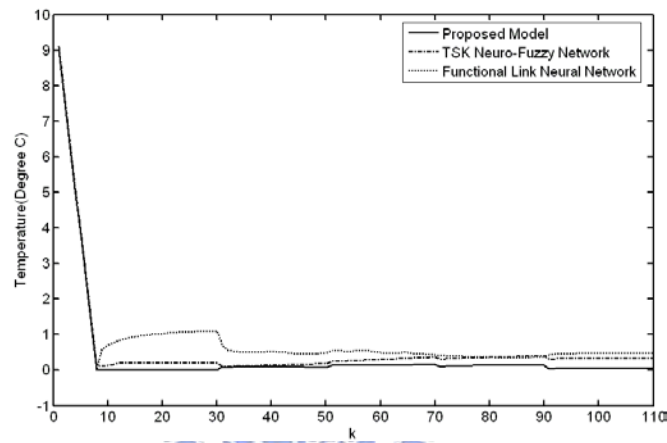


(b)

Figure 2.7: (a) Behavior of FLNFN controller when a change occurs in the water bath system. (b) Error curves of FLNFN controller, TSK-type NFN controller and FLNN controller.



(a)



(b)

Figure 2.8: (a) Tracking of FLNFN controller when a change occurs in the water bath system. (b) Error curves of FLNFN controller, TSK-type NFN controller and FLNN controller.

Table 2.1: Comparison of performance of various controllers to control of water bath temperature system.

$SAE = \sum_{k=1}^{120}  y_{ref}(k) - y(k) $	FLNFN Controller	PID Controller [64]	Fuzzy Controller [65]	FLNN Controller [25]	TSK-type NFN Controller [9]
Regulation Performance	354.84	418.5	401.5	379.22	361.96
Influence of Impulse Noise	272.61	311.5	275.8	324.51	274.75
Effect of Change in Plant Dynamics	264.35	322.2	273.5	311.54	265.48
Tracking Performance	44.28	100.6	88.1	98.43	54.28

## Example 2: Control of Bounded Input Bounded Output Nonlinear Plant

In this case, the plant is described by the difference equation

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k). \quad (2.24)$$

The reference model is described by the difference equation

$$y_r(k+1) = 0.6y_r(k) + r(k) \quad (2.25)$$

where  $r(k) = \sin(2\pi k / 10) + \sin(2\pi k / 25)$ . Figure 2.9 present the block diagram of the FLNFN-based control system. The inputs to the FLNFN controller are the reference input, the previous plant output, and the previous control signal; the output of the FLNFN controller is the control signal to the plant. The online algorithm developed in this dissertation is adopted to adjust the structure and the parameters of the FLNFN controller such that the error between the output of the plant and the desired output from a reference model approaches a small value after some train cycles.

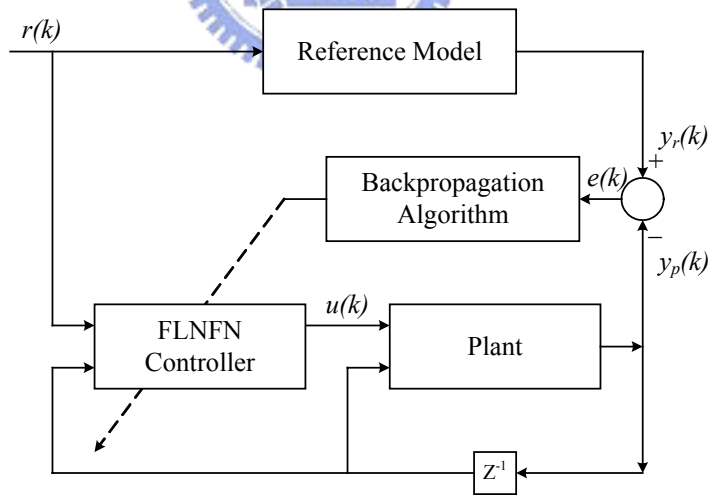
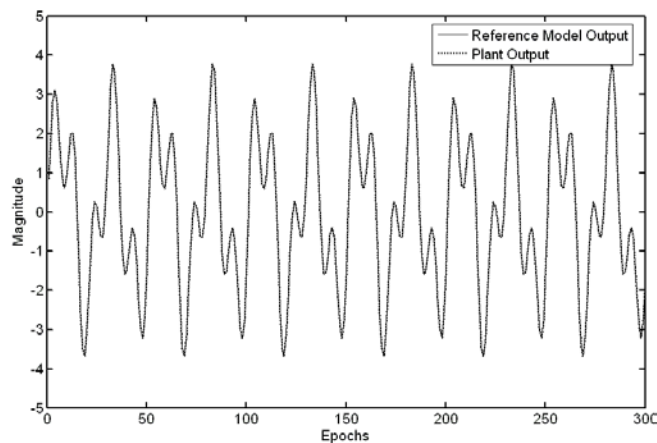


Figure 2.9: Block diagram of FLNFN controller-based control system.

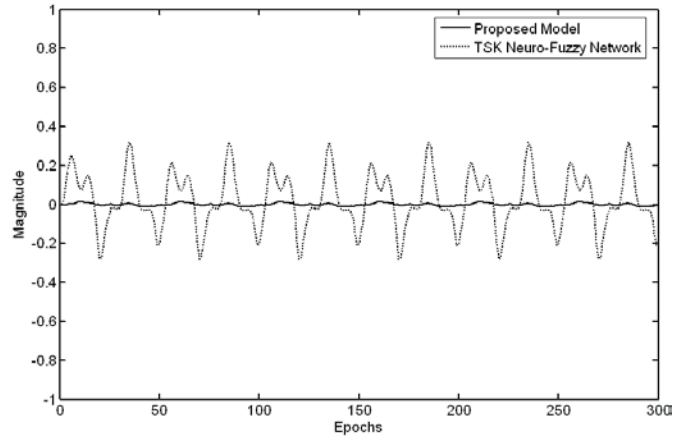
After 500 training iterations, six fuzzy rules are generated. In this example, the proposed FLNFN controller is compared to the FLNN controller [25] and the TSK-type NFN controller [9]. Each of the controllers is applied to control the bounded input bounded output (BIBO)

nonlinear plant. In the following four cases, the FLNFN controller is demonstrated to outperform the other models.

In the first case, the reference input is given by Eq. (2.25) and the final result is shown in Fig. 2.10(a). Figure 2.10(b) presents the error curves of the FLNFN controller and the TSK-type NFN controller. In this figure, the FLNFN controller yields smaller errors than the TSK-type NFN controller. In the second case, after 100 epochs, the reference input is changed to  $r(k) = \sin(2\pi k / 25)$ . Figures 2.11(a)-(b) plot the result of the FLNFN controller and the corresponding errors of the FLNFN controller and the TSK-type NFN controller. In the third case, after 100 epochs, the reference input is changed to an impulse signal. Figure 2.12(a) presents the simulation result. Figure 2.12 (b) present the corresponding errors of the FLNFN controller, the FLNN controller and the TSK-type NFN controllers. In the fourth case, a disturbance of 2.0 is added to the system between the 100th and the 150th epochs. In this case, the FLNFN-based control system can recover from the disturbance quickly, as shown in Fig. 2.13. The RMS (root mean square) error is adopted to evaluate the performance. Table 2.2 presents the RMS errors of the FLNFN controller, the FLNN controller and the TSK-type NFN controller. Table 2.2 shows that, according to the simulation results, the proposed FLNFN controller outperforms the other models.

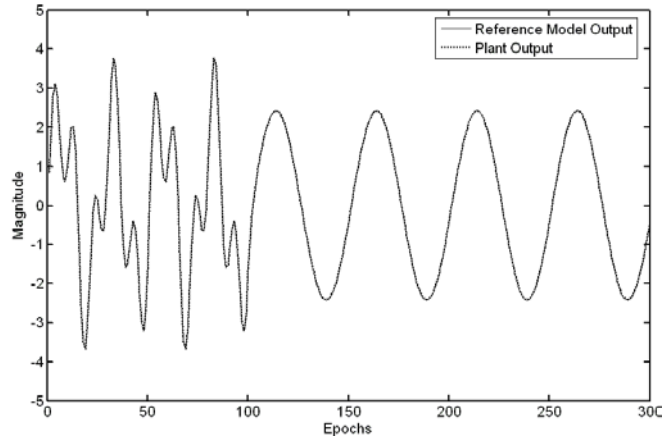


(a)

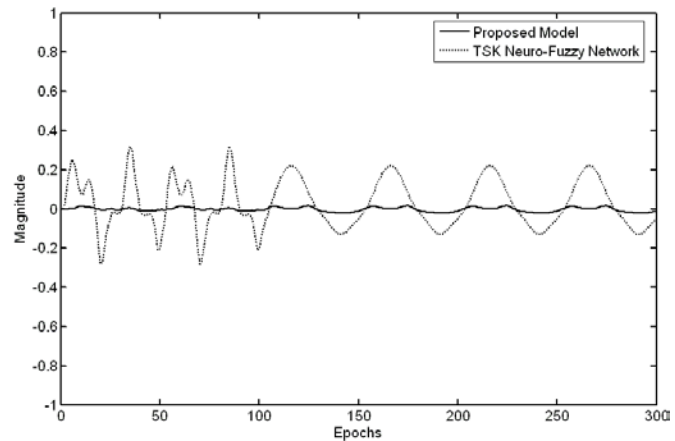


(b)

Figure 2.10: Final system response in first case of example 2. (a) The dashed line represents plant output and the solid line represents the reference model. (b) Error curves of FLNFN controller and TSK-type NFN controller.



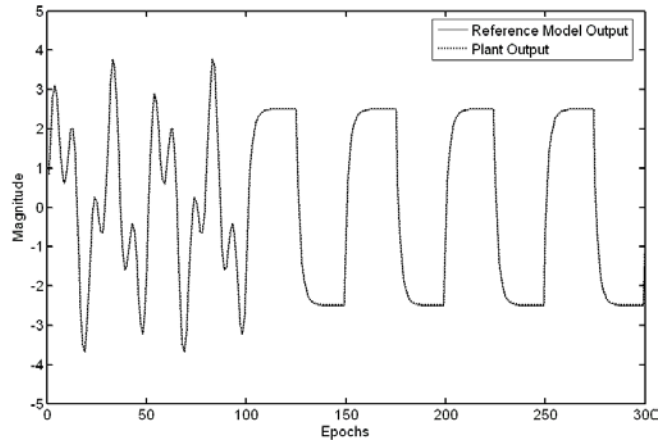
(a)



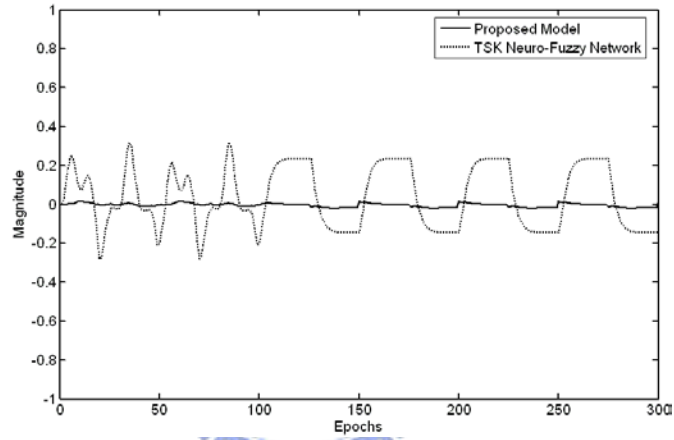
(b)

Figure 2.11: Final system response in second case of example 2. (a) The dashed line represents plant output and the solid line represents the reference model. (b) Error curves of FLNFN controller and TSK-type NFN controller.





(a)



(b)

Figure 2.12: Final system response in third case of example 2. (a) The dashed line represents plant output and the solid line represents the reference model. (b) Error curves of FLNFN controller and TSK-type NFN controller.

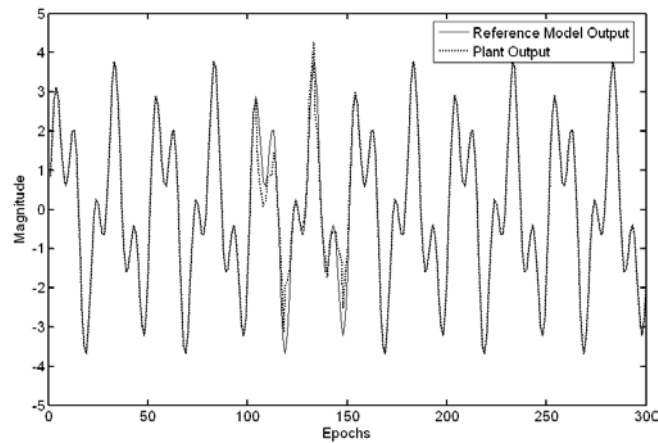


Figure 2.13: Final system response in fourth case of example 2. The dashed line represents plant output and the solid line represents the reference model.

Table 2.2: Comparison of performance of various controllers to control of BIBO nonlinear plant.

Method	FLNFN Controller	FLNN Controller [25]	TSK-type NFN Controller [9]
Training Steps	500	1000	500
Parameter Numbers	6 rules/ 60 parameters	79 parameters	9 rules/ 63 parameters
RMS error of case1	0.0004	0.0211	0.0084
RMS error of case2	0.0006	0.0208	0.0075
RMS error of case3	0.0007	0.0303	0.0095

### Example 3: Control of Ball and Beam System

Figure 2.14 presents the ball and beam system [61]. The beam is made to rotate in the vertical plane by applying a torque at the center of rotation and the ball is free to roll along the beam. The ball must remain in contact with the beam.

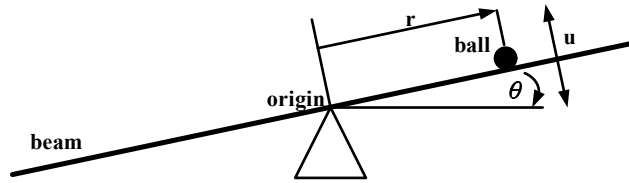


Figure 2.14: Ball and beam system.

The ball and beam system can be written in state space form as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ B(x_1 x_4^2 - G \sin x_3) \\ x_4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u, \quad (2.26)$$

$$y = x_1$$

where  $x = (x_1, x_2, x_3, x_4)^T \equiv (r, \dot{r}, \theta, \dot{\theta})^T$  is the state of the system and  $y = x_1 \equiv r$  is the output of the system. The control  $u$  is the angular acceleration ( $\ddot{\theta}$ ) and the parameters  $B = 0.7143$  and  $G = 9.81$  are set in this system. The purpose of control is to determine  $u(x)$  such that the closed-loop system output  $y$  converges to zero from different initial conditions.

According to the input/output-linearization algorithm [61], the control law  $u(x)$  is determined as follows; for state  $x$ , compute  $v(x) = -\alpha_3\phi_4(x) - \alpha_2\phi_3(x) - \alpha_1\phi_2(x) - \alpha_0\phi_1(x)$ , where  $\phi_1(x) = x_1$ ,  $\phi_2(x) = x_2$ ,  $\phi_3(x) = -BG \sin x_3$ ,  $\phi_4(x) = -BGx_4 \cos x_3$  and  $\alpha_i$  are chosen such that  $s^4 + \alpha_3s^3 + \alpha_2s^2 + \alpha_1s + \alpha_0$  is a Hurwitz polynomial. Compute  $a(x) = -BG \cos x_3$  and  $b(x) = BGx_4^2 \sin x_3$ ; then  $u(x) = [v(x) - b(x)] / a(x)$ .

In this simulation, the differential equations are solved using the second/third-order Runge-Kutta method. The FLNFN model is trained to approximate the aforementioned conventional controller of a ball and beam system.  $u(x) = [v(x) - b(x)] / a(x)$  is adopted to generate the input/output train pair with  $x$  obtained by randomly sampling 200 points in the region  $U = [-5, 5] \times [-3, 3] \times [-1, 1] \times [-2, 2]$ . After online structure-parameter learning, 14 fuzzy rules are generated. The controller after learning was tested under the following four initial conditions;  $x(0) = [2.4, -0.1, 0.6, 0.1]^T$ ,  $[1.6, 0.05, -0.5, -0.05]^T$ ,  $[-1.6, -0.05, 0.5, 0.05]^T$  and  $[-2.4, 0.1, -0.6, -0.1]^T$ . Figure 2.15 plots the output responses of the closed-loop ball and beam system controlled by the FLNFN model and the TSK-type NFN model. These responses approximate those of the original controller under the four initial conditions. In this figure, the curves of the FLNFN model tend quickly to stabilize. Figure 2.16 also presents the behavior of the four states of the ball and beam system, starting at the initial condition  $[-2.4, 0.1, -0.6, -0.1]^T$ . In this figure, the four states of the system decay gradually to zero. The results demonstrate the perfect control capability of the trained FLNFN model. The performance of the FLNFN controller is compared with that of the FALCON controller [5], the FLNN controller [25] and the TSK-type NFN controller [9]. Table 2.3 presents the comparison

results. The results demonstrate that the proposed FLNFN controller outperforms other controllers.

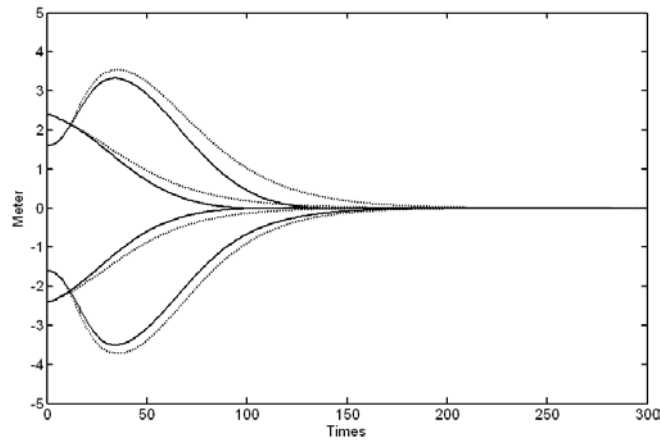


Figure 2.15: Responses of ball and beam system controlled by FLNFN model (solid curves) and TSK-type NFN model (dotted curves) under four initial conditions.

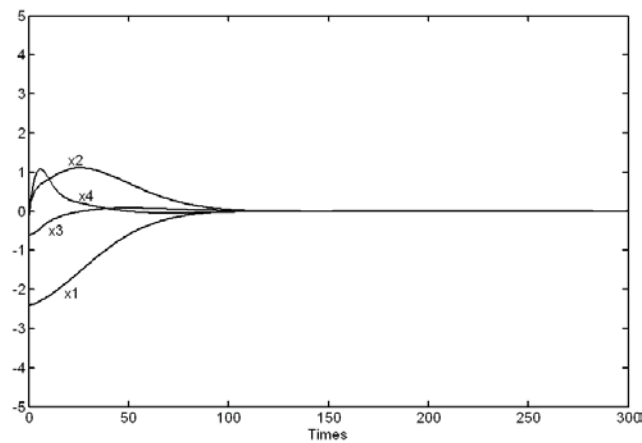


Figure 2.16: Responses of four states of ball and beam system under the control of the trained FLNFN controller.

Table 2.3: Comparison of performance of various controllers to control of ball and beam system.

Method	FLNFN Controller	FALCON Controller [5]	FLNN Controller [25]	TSK-type NFN Controller [9]
Training Steps	500	50000	1000	500
Parameter Numbers	14 rules/ 280 parameters	28 rules/ 280 parameters	317 parameters	22 rules/ 286 parameters
RMS errors	0.056	0.2	0.153	0.079

#### Example 4: Control of Multi-input Multi-output (MIMO) Plant

In this example, the MIMO plants [62] to be controlled are described by the equations

$$\begin{bmatrix} y_{p1}(k+1) \\ y_{p2}(k+1) \end{bmatrix} = \begin{bmatrix} 0.5 \frac{y_{p1}(k)}{1+y_{p2}^2(k)} \\ 0.5 \frac{y_{p1}(k)y_{p2}(k)}{1+y_{p2}^2(k)} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}. \quad (2.27)$$

The controlled outputs should follow the desired output  $y_{r1}$  and  $y_{r2}$  as specified by the following 250 pieces of data;

$$\begin{bmatrix} y_{r1}(k) \\ y_{r2}(k) \end{bmatrix} = \begin{bmatrix} \sin(k\pi/45) \\ \cos(k\pi/45) \end{bmatrix}. \quad (2.28)$$

The inputs of the FLNFN are  $y_{p1}(k)$ ,  $y_{p2}(k)$ ,  $y_{r1}(k)$  and  $y_{r2}(k)$ , and the outputs are  $u_1(k)$  and  $u_2(k)$ . After 500 training iterations, four fuzzy rules are generated. In this example, the proposed FLNFN controller is compared to the FLNN controller [25] and the TSK-type NFN controller [9]. Each of the controllers is applied to control the MIMO plant. To demonstrate the performance of the proposed controller, Figures 2.17(a) and (b) plot the control results of the desired output and the model output using FLNFN controller. Figures 2.17 (c) and (d) show the error curves of the FLNFN controller and the TSK-type NFN controller. Table 2.4 presents the RMS errors of the FLNFN controller, the FLNN controller and the TSK-type NFN controller. Table 2.4 shows that, according to the simulation results, the proposed FLNFN controller is better than the other controllers.

Table 2.4: Comparison of performance of various controllers to control of MIMO plant.

Method	FLNFN Controller	FLNN Controller [25]	TSK-type NFN Controller [9]
Training Steps	500	1000	500
Parameter Numbers	4 rules/ 128 parameters	161 parameters	10 rules/ 140 parameters
RMS errors	0.0002	0.0738	0.0084

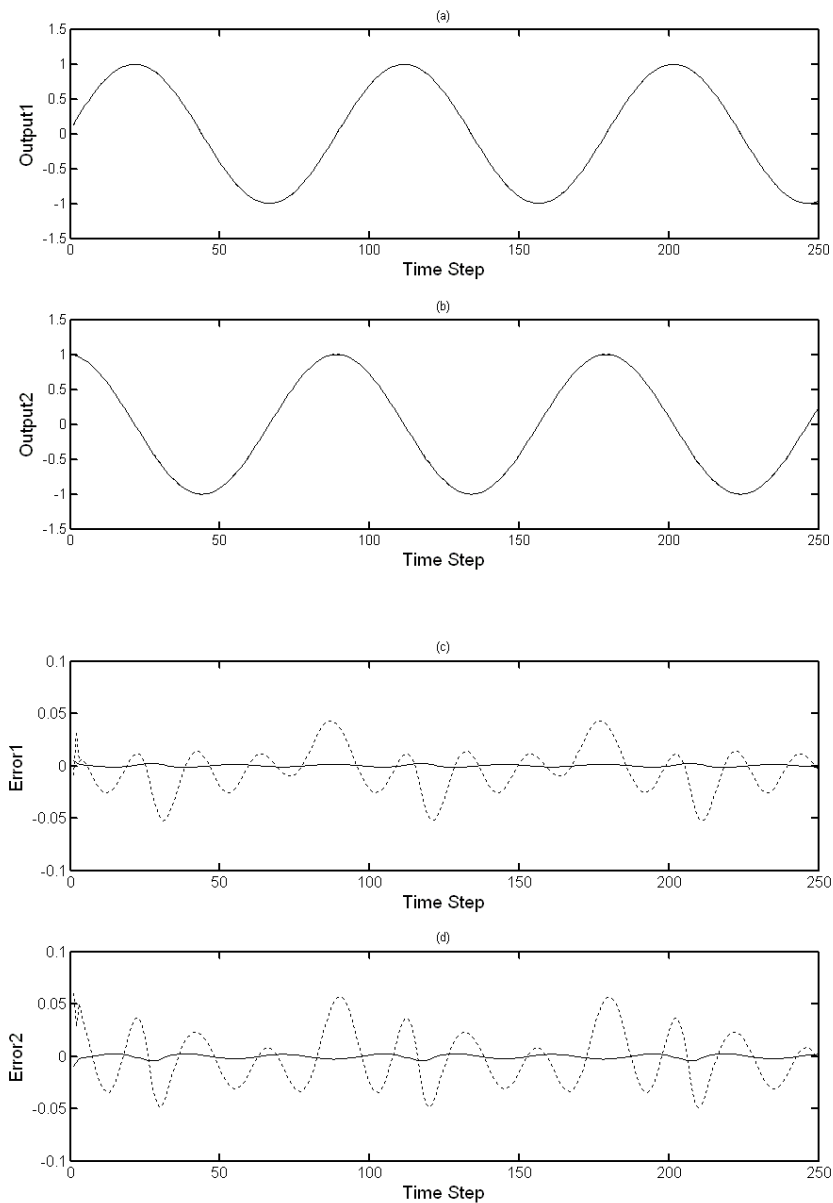


Figure 2.17: Desired output (solid line) and model output using FLNFN controller (dotted line) of (a) Output 1. (b) Output 2 in Example 4. Error curves of FLNFN controller (solid line) and TSK-type NFN controller (dotted line) for (c) output 1 and (d) output 2.

## 2.4 Summary

This dissertation proposes a functional-link-based neuro-fuzzy network (FLNFN) structure for nonlinear system control. The FLNFN model uses a functional link neural network to the

consequent part of the fuzzy rules. The FLNFN model can automatically construct and adjust free parameters by performing online structure/parameter learning schemes concurrently. The FLNFN model was proven to be a universal approximator and to convergence stably. Finally, the proposed FLNFN model yields better simulation results than other existing models under some circumstances.



## Chapter 3

# A Modified Differential Evolution for the FLNFN Model

In Chapter 2, we have developed the functional-link-based neuro-fuzzy network (FLNFN). However, the back-propagation learning algorithm may reach the local minima very quickly. Therefore, a modified differential evolution (MODE) is presented to optimize the FLNFN parameters in this chapter. The proposed MODE learning algorithm has two crucial ideas. First, MODE adopts a method to effectively search between the best-so-far individual and randomly chosen individuals. Therefore, MODE not only explores the search space by randomly chosen individuals but also exploits the search capability of a near global optimal solution by the best-so-far individual. Second, MODE provides a cluster-based mutation scheme, which maintains useful diversity in the population to increase the search capability. The cluster-based mutation scheme prevents the MODE from being trapped in local optima of the search space.

### 3.1 A Brief Introduction of Differential Evolution

This section describes basic concepts concerning differential evolution (DE) [53]. Differential evolution is a parallel direct search method which utilizes  $NP$   $N$ -dimensional parameter



vectors

$$x_{i,G}, \quad i = 1, 2, \dots, NP \quad (3.1)$$

as a population for each generation  $G$ .  $NP$  does not change during the minimization process. The initial vector population is chosen randomly and should cover the entire parameter space. As a rule, we will assume a uniform probability distribution for all random decisions unless otherwise stated. In case a preliminary solution is available, the initial population might be generated by adding normally distributed random deviations to the nominal solution. DE generates new parameter vectors by adding the weighted difference between two population vectors to a third vector. Let this operation be called mutation. The mutated vector's parameters are then mixed with the parameters of another predetermined vector, the target vector, to yield the so-called trial vector. Parameter mixing is often referred to as "crossover" in the ES-community and will be explained later in more detail. If the trial vector yields a lower cost function value than the target vector, the trial vector replaces the target vector in the following generation. This last operation is called selection. Each population vector has to serve once as the target vector so that  $NP$  competitions take place in one generation.

More specifically DE's basic strategy can be described as follows:

**Mutation**—For each target vector  $x_{i,G}$ ,  $i = 1, 2, \dots, NP$ , a mutant vector is generated according to

$$v_{i,G+1} = x_{r_1,G} + F \cdot (x_{r_2,G} - x_{r_3,G}) \quad (3.2)$$

with random indexes  $r_1, r_2, r_3 \in \{1, 2, \dots, NP\}$ , integer, mutually different and  $F > 0$ . The randomly chosen integers  $r_1, r_2$  and  $r_3$  are also chosen to be different from the running index  $i$ , so that  $NP$  must be greater or equal to four to allow for this condition.  $F$  is a real and constant factor  $\in [0, 2]$  which controls the amplification of the differential variation  $(x_{r_2,G} - x_{r_3,G})$ .

Figure 3.1 shows a two-dimensional example that illustrates the different vectors which play a

part in the generation of  $v_{i,G+1}$ .

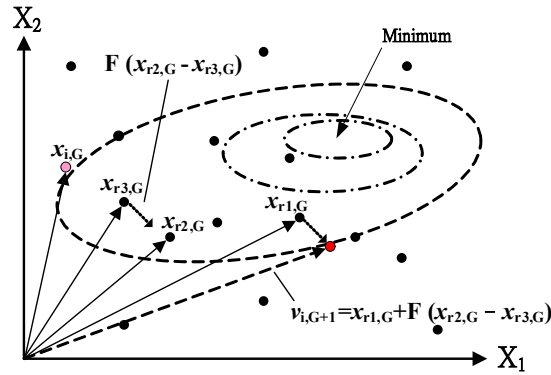
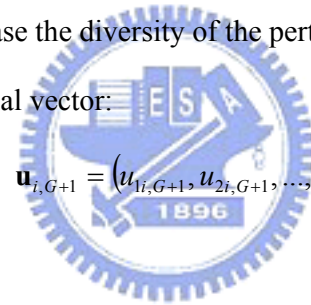


Figure 3.1: An example of a two-dimensional cost function showing its contour lines and the process for generating  $v_{i,G+1}$ .

**Crossover**—In order to increase the diversity of the perturbed parameter vectors, crossover is introduced. To this end, the trial vector:



$$\mathbf{u}_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \dots, u_{Ni,G+1}) \quad (3.3)$$

is formed, where

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } (r(j) \leq CR) \text{ or } j = rn(i) \\ x_{ji,G} & \text{if } (r(j) > CR) \text{ or } j \neq rn(i) \end{cases}, \quad j = 1, \dots, N. \quad (3.4)$$

In Eq.(3.4),  $r(j)$  is the  $j$ th evaluation of a uniform random number generator with outcome  $\in [0, 1]$ .  $CR$  is the crossover constant  $\in [0, 1]$  which has to be determined by the user.

$rn(i)$  is a randomly chosen index  $\in \{1, 2, \dots, N\}$  which ensures that  $u_{i,G+1}$  gets at least one parameter from  $v_{i,G+1}$ . Figure 3.2 gives an example of the crossover mechanism for

7-dimensional vectors.

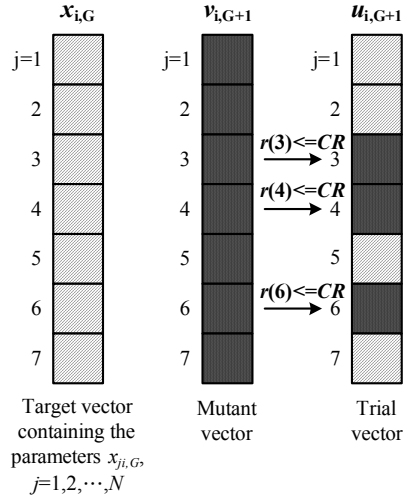
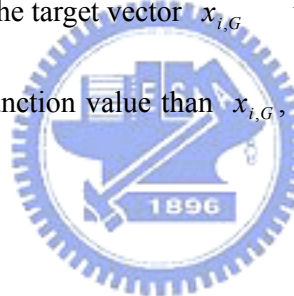


Figure 3.2: Illustration of the crossover process for  $N=7$  parameters.

**Selection**—To decide whether or not it should become a member of generation  $G+1$ , the trial vector  $u_{i,G+1}$  is compared to the target vector  $x_{i,G}$  using the greedy criterion. If vector  $u_{i,G+1}$  yields a smaller cost function value than  $x_{i,G}$ , then  $x_{i,G+1}$  is set to  $u_{i,G+1}$ ; otherwise, the old value  $x_{i,G}$  is retained.



### 3.2 A Modified Differential Evolution

This section describes a modified differential evolution (MODE) for the FLNFN model. The MODE learning algorithm consists of four major phases – the initialization phase, the evaluation phase, the reproduction phase and the cluster-based mutation phase. First, the initialization phase creates an initial population. Second, the evaluation phase evaluates the performance of each individual using an objective function. Third, the reproduction phase generates new individuals and select survivors to the next phase. Fourth, the cluster-based mutation phase ensures diversity and prevents a population from converging to a suboptimal solution. The whole learning process is described step-by-step below.

### 3.2.1 Initialization Phase

#### A. Coding Step

The foremost step in MODE is the coding of the neuro-fuzzy network into an individual. Figure 3.3 shows an example of the coding of parameters of the neuro-fuzzy network into an individual where  $i$  and  $j$  represent the  $i$ th input variable and the  $j$ th rule, respectively. In this dissertation, a Gaussian membership function is adopted with variables that represent the mean and variance of the membership function. Figure 3.3 represents the neuro-fuzzy network given by Eq. (2.3), where  $m_{ij}$  and  $\sigma_{ij}$  are the mean and variance of a Gaussian membership function, respectively, and  $w_{kj}$  represents the corresponding link weight of the consequent part that is connected to the  $j$ th rule node. In this dissertation, a real number represents the position of each individual.

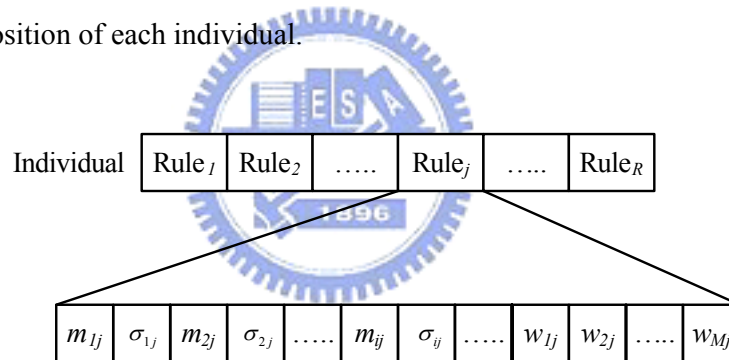


Figure 3.3: Coding FLNFN into an individual in the proposed MODE method.

#### B. Create Initial Population

Before the MODE learning algorithm is applied, every individual  $x_{i,g}$  must be created randomly in the range  $[0, 1]$ , where  $i=1, 2, \dots, PS$  represents the  $i$ th individual for each generation  $g$  and  $PS$  denotes the population size.

### 3.2.2 Evaluation Phase

In this dissertation, we adopt a fitness function (i.e., objective function) to evaluate the performance of each individual. The fitness function is defined as follows:

$$Fitness\ Value = \frac{1}{1 + \sqrt{\frac{1}{N_t} \sum_{k=1}^{N_t} (y_k - \bar{y}_k)^2}} \quad (3.5)$$

where  $y_k$  represents the model output of the  $k$ th data,  $\bar{y}_k$  represents the desired output of the  $k$ th data, and  $N_t$  represents the number of the training data.

### 3.2.3 Reproduction Phase

#### A. Parent Choice

Each individual in the current generation is allowed to breed through mating with other randomly selected individuals from the population. Specifically, for each individual  $x_{i,g}$ ,  $i=1, 2, \dots, PS$ , where  $g$  denotes the current generation, three other random individuals,  $x_{r_1,g}$ ,  $x_{r_2,g}$  and  $x_{r_3,g}$ , are selected from the population such that  $r_1, r_2$ , and  $r_3 \in \{1, 2, \dots, PS\}$  and  $i \neq r_1 \neq r_2 \neq r_3$ . This way, a parent pool of four individuals is formed to produce an offspring.

#### B. Offspring Generation

After choosing the parents, MODE applies a differential operation to generate a mutated individual  $v_{i,g}$ , according to the following equation:

$$v_{i,g} = x_{r_1,g} + (1 - F) \cdot (x_{r_2,g} - x_{r_3,g}) + F \cdot (x_{best} - x_{r_1,g}) \quad (3.6)$$

where  $F$ , commonly known as the scaling factor, is defined as  $\frac{g}{G}$  to control the rate at which the population evolves,  $g$  denotes the current generation,  $G$  is the maximum number of generations, and  $x_{best}$  is the best-so-far individual in each generation. To complement the differential operation search strategy, then MODE uses a crossover operation, often referred to as discrete recombination, in which the mutated individual  $v_{i,g}$  is mated with  $x_{i,g}$  and generates the offspring  $u_{i,g}$ . The elements of an individual  $u_{i,g}$  are inherited from  $x_{i,g}$  and

$v_{i,g}$ , which are determined by a parameter called crossover probability ( $CR \in [0,1]$ ), as follows:

$$u_{id,g} = \begin{cases} v_{id,g}, & \text{if } Rand(d) \leq CR \\ x_{id,g}, & \text{if } Rand(d) > CR \end{cases} \quad (3.7)$$

where  $d = 1, 2, \dots, D$  denotes the  $d$ th element of individual vectors.  $Rand(d) \in [0,1]$  is the  $d$ th evaluation of a random number generator.

### C. Survivor Selection

MODE applies selection pressure only when selecting survivors. A knockout competition is played between each individual  $x_{i,g}$  and its offspring  $u_{i,g}$ , and the winner is selected deterministically based on objective function values and is then promoted to the next phase. Moreover, the best individual also reserves to the next generation.

#### 3.2.4 Cluster-Based Mutation Phase

To prevent the MODE from being trapped in the local optima of the search space (i.e., problems in which there are a number of points that are better than all their neighboring solutions, but do not have as good a fitness as the globally optimal solution), we adopt a cluster-based mutation scheme, which maintains diversity in the population to increase the search capability. We use an easy and fast self-cluster algorithm (SCA) [66] to cluster the population. Each cluster can be viewed as a subspace with similar biological features in the environment that can support different types of life; that is, these similar individuals of each cluster direct the search toward the same local optima. Then, for each cluster, the best individual will be reserved and other individuals will be suitably mutated to the next generation. Mutation is an operator that randomly alters the allele of an element. Figure 3.4 shows the mutation of an individual. The mutation value is generated according to

$$\text{Mean: } Individual[d] = m_{ij} + random[0,1] \times \sigma_{ij} \quad (3.8)$$

$$\text{Variance: } \text{Individual}[d] = 2 \times \text{random}[0,1] \times \sigma_{ij} \quad (3.9)$$

$$\text{Other parameters: } \text{Individual}[d] = \text{random}[-1,1] \quad (3.10)$$

where  $m_{ij}$  and  $\sigma_{ij}$  are the current mean and variance in the current individual, respectively.

Following the mutation step, a new individual can be introduced into the each population.

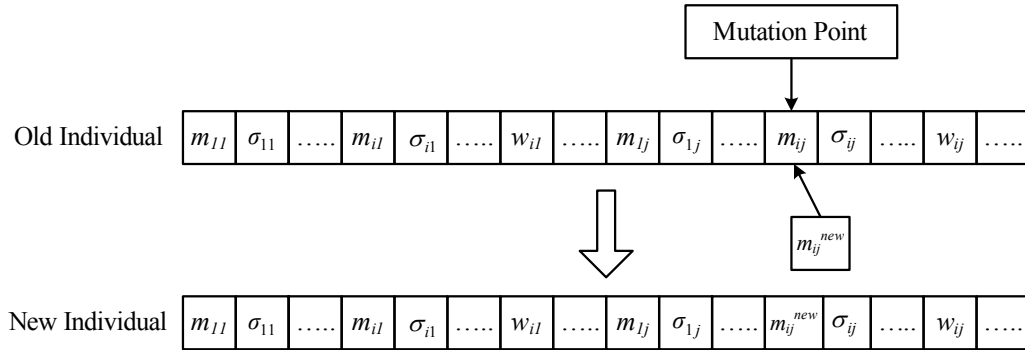


Figure 3.4: A mutation operation in the modified differential evolution.

### 3.3 Experimental Results

This dissertation demonstrated the performance of the FLNFN-MODE controller for nonlinear system control. The FLNFN-MODE controller is applied to control of water bath temperature system. In addition, this experimentation discusses the use of a real-time digital control environment with a hardware-in-the-loop (HIL) control application. We configure a real-time visual simulation (VisSim) environment including a RT-DAC4/PCI motion control card and HIL systems to demonstrate the performance of the FLNFN-MODE controller for practical control applications. VisSim is a Windows-based program for the modeling and simulation of complex nonlinear dynamic systems. VisSim combines an intuitive drag and drop block diagram interface with a powerful simulation engine. We can generate a VisSim diagram using a customizable ANSI C code directly. In this dissertation, we applied the FLNFN-MODE controller to the planetary train type inverted pendulum system and the magnetic levitation system in the VisSim. The experiment compares the performance with

that of the FLNFN-MODE controller, the FLNFN-DE controller, and the FLNFN-GA controller. Table 3.1 presents the parameter settings before training used in the three computer simulations for the MODE. In the DE, the population size is set to 50, the maximum number of generation is set to 2000, and the crossover rate is set to 0.9. In the GA, the population size is set to 50, the maximum number of generation is set to 2000, the crossover rate is set to 0.5, and the mutation rate is set to 0.3.

Table 3.1: Parameter settings before training.

Parameter	Value
Population Size	50
Maximum Number of Generation	2000
Crossover Rate	0.9
Coding Type	Real Number

### Example 1: Control of Water Bath Temperature System

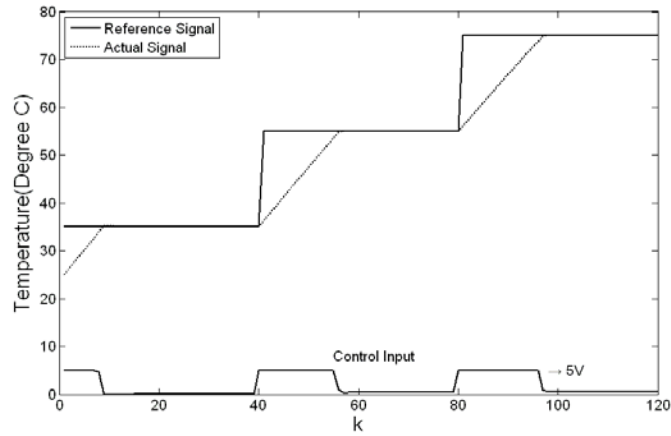
The description of the system is the same as Example 1 of Section 2.3. In this example, four fuzzy rules are adopted and the population size is set to 50. The evolution proceeded for 2000 generations, and was repeated thirty times.

This dissertation compares the FLNFN-MODE controller to the FLNFN-DE controller and the FLNFN-GA controller. Each of these controllers is applied to the water bath temperature control system. The performance measures include the set-points regulation, the influence of impulse noise, and a large parameter variation in the system, and the tracking capability of the controllers.

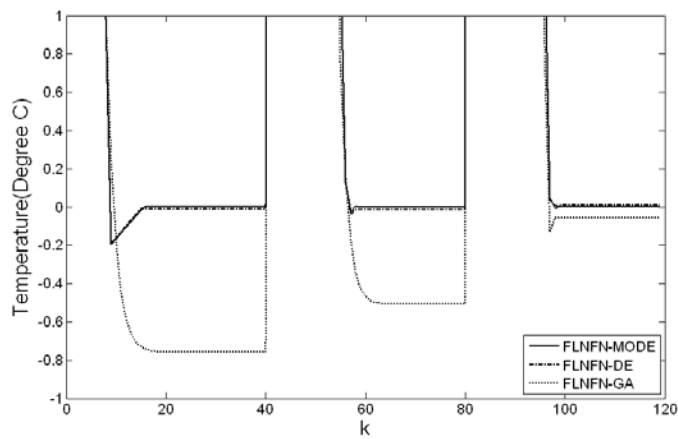
Figure 3.5(a) presents the regulation performance of the FLNFN-MODE controller. The regulation performance was also tested using the FLNFN-DE controller and the FLNFN-GA controller. Figure 3.5(b) plots the error curves of the FLNFN-MODE controller, the FLNFN-DE controller and the FLNFN-GA controller. Figure 3.6(a) and (b) present the



behaviors of the FLNFN-MODE controller under the influence of impulse noise, and the corresponding errors, respectively. Figure 3.7(a) presents the behaviors of the FLNFN-MODE controller when in the plant dynamics change. Figure 3.7(b) presents the corresponding errors of the FLNFN-MODE controller, the FLNFN-DE controller and the FLNFN-GA controllers. Figure 3.8(a) presents the tracking performance of the FLNFN-MODE controller. Figure 3.8(b) presents the corresponding errors of the FLNFN-MODE controller, the FLNFN-DE controller, and the FLNFN-GA controller. The aforementioned simulation results, presented in Table 3.2, demonstrate that the proposed FLNFN-MODE controller outperforms other controllers.



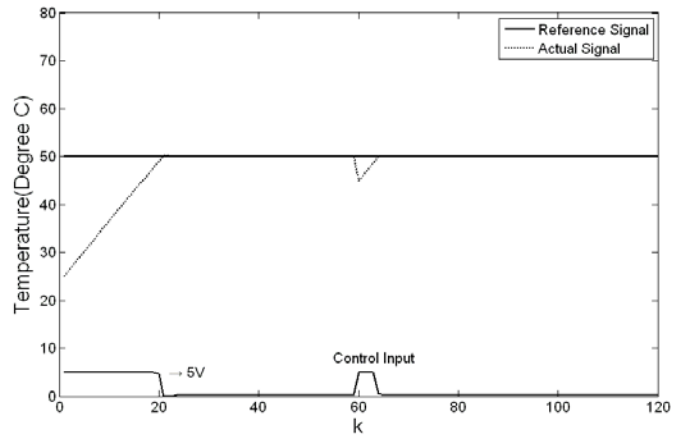
(a)



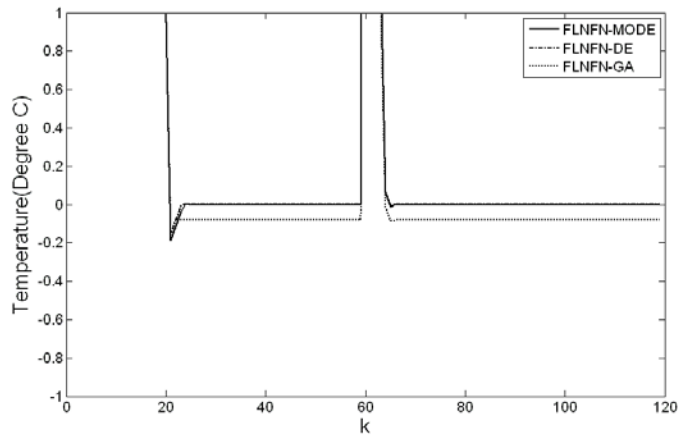
(b)

Figure 3.5: (a) Final regulation performance of FLNFN-MODE controller in water bath

system. (b) Error curves of the FLNFN-MODE controller, FLNFN-DE controller and FLNFN-GA controller.

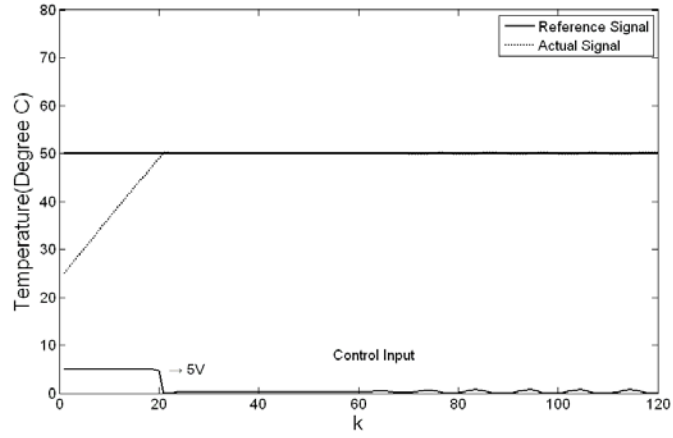


(a)

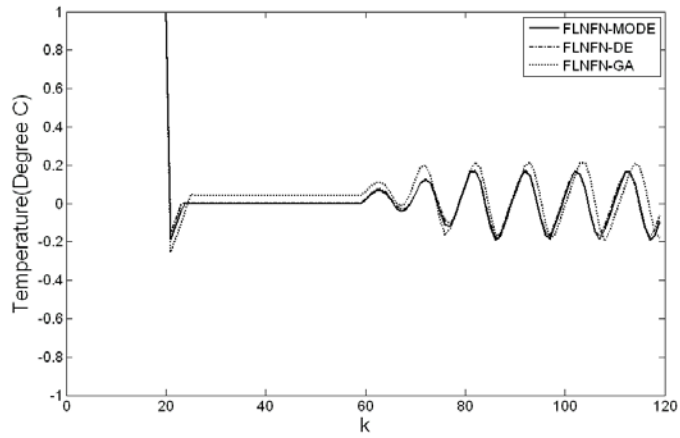


(b)

Figure 3.6: (a) Behavior of FLNFN-MODE controller under impulse noise in water bath system. (b) Error curves of FLNFN-MODE controller, FLNFN-DE controller and FLNFN-GA controller.

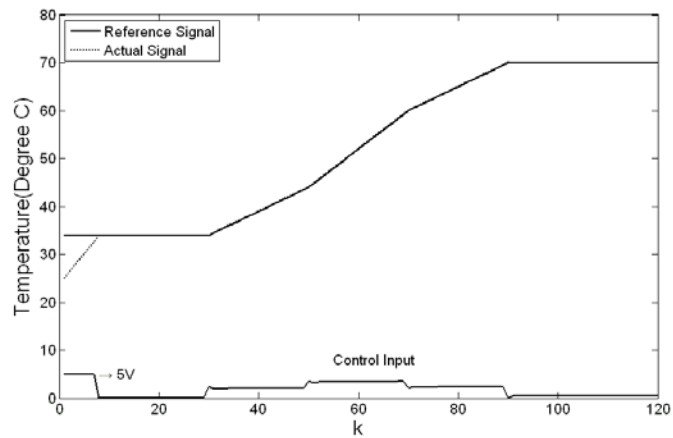


(a)

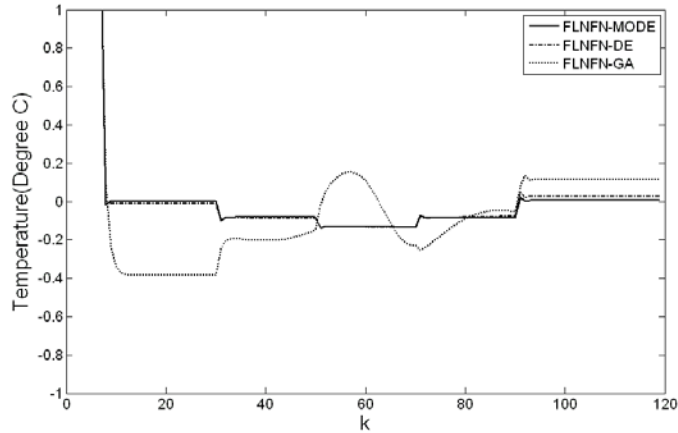


(b)

Figure 3.7: (a) Behavior of FLNFN-MODE controller when a change occurs in the water bath system. (b) Error curves of FLNFN-MODE controller, FLNFN-DE controller and FLNFN-GA controller.



(a)



(b)

Figure 3.8: (a) Tracking of FLNFN-MODE controller when a change occurs in the water bath system. (b) Error curves of FLNFN-MODE controller, FLNFN-DE controller and FLNFN-GA controller.

Table 3.2: Comparison of performance of various controllers to control of water bath temperature system.

$SAE = \sum_{k=1}^{120}  y_{ref}(k) - y(k) $	FLNFN-MODE Controller	FLNFN-DE Controller	FLNFN-GA Controller
Regulation Performance	352.78	352.91	372.85
Influence of Impulse Noise	270.59	270.65	282.21
Effect of Change in Plant Dynamics	263.39	263.25	270.66
Tracking Performance	42.03	42.92	62.02

## Example 2: Control of Planetary Train Type Inverted Pendulum System

In order to predict the dynamic behavior of a system from given input command and initial conditions of the system, it is necessary to make a mathematical model of the planetary train type inverted pendulum system [67]. The dynamic behavior of the system is helpful in sizing the actuator, choosing the amplifier power, designing the details of the mechanisms, and tuning the controller by computer simulation. To clarify the kinematic and dynamic

relations, three major movable parts, the center gear, the planetary gear and the pendulum, are depicted in Fig. 3.9.

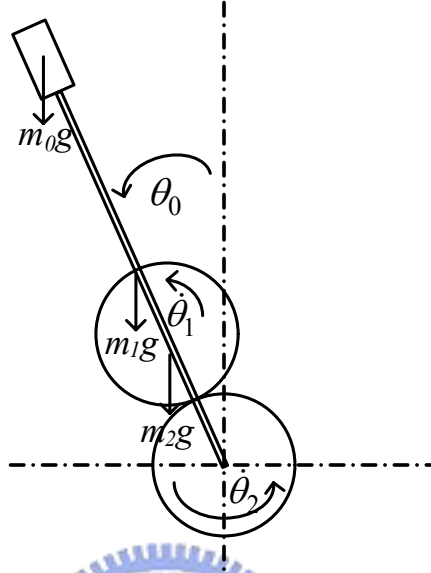


Figure 3.9: A physical model geometry of the planetary train type inverted pendulum.

The kinematic relations of the three components can be allotted to two mutual movements as follows. First, we assume the pendulum to be stationary ( $\dot{\theta}_0 = 0$ ). The ratio of the movements of the planetary gear to the center gear is

$$e = \frac{\dot{\theta}'_1}{\dot{\theta}_2} = -\frac{N_2}{N_1} = -\frac{r_2}{r_1} \quad (3.11)$$

where  $\theta_1$  and  $\theta_2$  are the angle of the center gear and the planetary gear, respectively;  $N_1$  and  $N_2$  are the number of the tooth of the center gear and the planetary gear, respectively;  $r_1$  and  $r_2$  are the radius of the center gear and the planetary gear; the dot denotes the time derivative; and  $\dot{\theta}'_1$  refers to the angular velocity of the planetary gear in this case. Thus,

$$\dot{\theta}'_1 = e\dot{\theta}_2. \quad (3.12)$$

Second, we assume the center gear to be stationary ( $\dot{\theta}_2 = 0$ ) and allow the pendulum and planetary gear to turn. The velocity of the planetary gear center can then be expressed as

$$v = (r_1 + r_2)\dot{\theta}_0. \quad (3.13)$$

This velocity lets the angular velocity of the planetary gear ( $\dot{\theta}_1''$ ) in this case be

$$\dot{\theta}_1'' = \frac{v}{r_1} = \frac{r_1 + r_2}{r_1} \dot{\theta}_0. \quad (3.14)$$

Combining Eqs. (3.13) and (3.14), we obtain

$$\begin{aligned} \dot{\theta}_1 &= \dot{\theta}_1' + \dot{\theta}_1'' \\ &= e\dot{\theta}_2 + \frac{r_1 + r_2}{r_1} \dot{\theta}_0. \end{aligned} \quad (3.15)$$

For the purpose of obtaining the relations between the input motor torque  $\tau_2$ , the output responses of the pendulum  $\theta_0$ , and the center gear  $\theta_2$ , we will use Lagrangian mechanics. Using this method can ensure that we analyze the mechanism in a systematic approach. It starts with the findings of kinetic energy and potential energy of each movable part.

$$\begin{aligned} K_2 &= \frac{1}{2} I_2 \dot{\theta}_2^2 \\ P_2 &= 0 \\ K_1 &= \frac{1}{2} m_1 [(r_1 + r_2) \dot{\theta}_0]^2 + \frac{1}{2} I_1 \dot{\theta}_1^2 \\ P_1 &= m_1 g (r_1 + r_2) \cos \theta_0 \\ K_0 &= \frac{1}{2} \left( \frac{1}{3} m_0 l^2 \right) \dot{\theta}_0^2 \\ P_0 &= m_0 g \frac{1}{2} \cos \theta_0 \end{aligned} \quad (3.16)$$

where  $K_2, K_1$ , and  $K_0$  are the kinetic energy of the center gear, the planetary gear and the pendulum, respectively;  $P_2, P_1$ , and  $P_0$  are the potential energy of the center gear, the planetary gear and the pendulum, respectively;  $I_2$  and  $I_1$  are the moment of inertia of the center gear and the planetary gear, respectively;  $m_1$  and  $m_0$  are the mass of the planetary gear and the pendulum, respectively; and  $l$  is the length of the pendulum.

Substituting Eq. (3.15) into Eq. (3.16), we obtain Lagrangian as follows:

$$\begin{aligned}
L &= \sum K - \sum P \\
&= \left[ \frac{1}{2} m_1 (r_1 + r_2)^2 + \frac{1}{6} m_0 l^2 \right] \dot{\theta}_0^2 \\
&\quad + \frac{1}{2} I_1 \left( e \dot{\theta}_2 + \frac{r_1 + r_2}{r_1} \dot{\theta}_0 \right)^2 + \frac{1}{2} I_2 \dot{\theta}_2^2 \\
&\quad - \left[ m_1 (r_1 + r_2) + \frac{1}{2} m_0 \right] g \cos \theta_0
\end{aligned} \tag{3.17}$$

Because  $\theta_0$  and  $\theta_2$  are two independent variables, we regard them as generalized coordinates. Using Lagrange's equation, two of the dynamic equations are derived as follows:

$$\begin{aligned}
\tau_2 &= \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\theta}_2} \right) - \frac{\partial L}{\partial \theta_2} \\
&= (e^2 I_1 + I_2) \ddot{\theta}_2 + e \left( \frac{r_1 + r_2}{r_1} \right) I_1 \ddot{\theta}_0
\end{aligned} \tag{3.18}$$

$$\begin{aligned}
\tau_0 &= \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\theta}_0} \right) - \frac{\partial L}{\partial \theta_0} \\
&= \left[ \left( \frac{r_1 + r_2}{r_1} \right)^2 I_1 + m_1 (r_1 + r_2)^2 + \frac{1}{3} m_0 l^2 \right] \ddot{\theta}_0 \\
&\quad + e \left( \frac{r_1 + r_2}{r_1} \right) I_1 \ddot{\theta}_2 - \left[ m_1 (r_1 + r_2) + \frac{1}{2} m_0 \right] g \sin \theta_0
\end{aligned} \tag{3.19}$$

In Eq. (3.19), there is no external torque applied in the pendulum, so we assign zero to the variable  $\tau_0$ .

In this dissertation, the proposed control structure is shown in Fig. 3.10. The applied encoder is used to sense the angle of the pendulum and then to translate the signal as a feedback signal. The pendulum angle is controlled by a motor torque until the pendulum is balanceable. To validate the usefulness of the proposed control system under different reference trajectories, two cases, including the set-point command (i.e., the stick angle command is equal to zero) and the periodic square command (i.e., the stick angle command is equal to the square wave) are used in this experimentation.

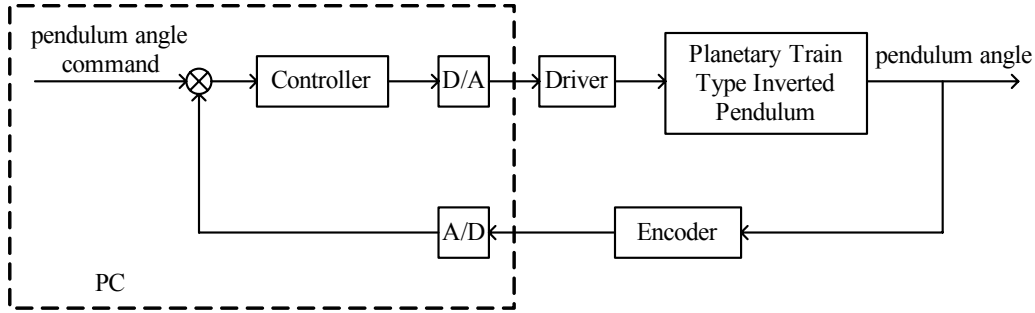


Figure 3.10: Control block diagram for the planetary train type inverted pendulum system.

This dissertation compares the FLNFN-MODE controller to the PID controller, the FLNFN-DE controller, and the FLNFN-GA controller. Each of these controllers is applied to the planetary train type inverted pendulum system. The PID controller is implemented as follows:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de}{dt} \quad (3.20)$$

where  $u(t)$  is the control output and the error,  $e(t)$ , is defined as  $e(t) = \text{desired value} - \text{measured value}$  of quantity being controlled. The control gains  $K_p=830$ ,  $K_i=0$ , and  $K_d=0.3284$  are designed. The training patterns of the FLNFN model are generated using the various PID controllers with different control gains.

Figure 3.11 shows an experimental planetary train type inverted pendulum system test used to validate the experimentation results. The performance measures include the set-points regulation (Case 1) and the square command tracking capability (Case 2) of the controllers. In Case 1, the proposed system is controlled to follow the set-points, which is equal to zero. Figure 3.12(a)-(d) presents the regulation performance of the FLNFN-MODE controller, the PID controller, the FLNFN-DE controller, and the FLNFN-GA controller. Figure 3.12(e) plots the scaling curves of the FLNFN-MODE controller and the PID controller between the 1.5<sup>th</sup> second and the 3.5<sup>th</sup> second. To test their regulation performance, two performance indexes, the sum of absolute error (SAE) for the pendulum angle and the pendulum speed, are defined by



$$SAE_{\theta_0} = \sum |\theta_0^{ref} - \theta_0| \quad \text{and} \quad SAE_{\dot{\theta}_0} = \sum |\dot{\theta}_0^{ref} - \dot{\theta}_0| \quad (3.21)$$

where  $\theta_0^{ref}$  and  $\theta_0$  are the referred pendulum angle and the actual pendulum angle, respectively, and  $\dot{\theta}_0^{ref}$  and  $\dot{\theta}_0$  are the referred pendulum speed and the actual pendulum speed, respectively. The  $SAE_{\theta_0}$  and  $SAE_{\dot{\theta}_0}$  of the experimental results are presented in Table 3.3.

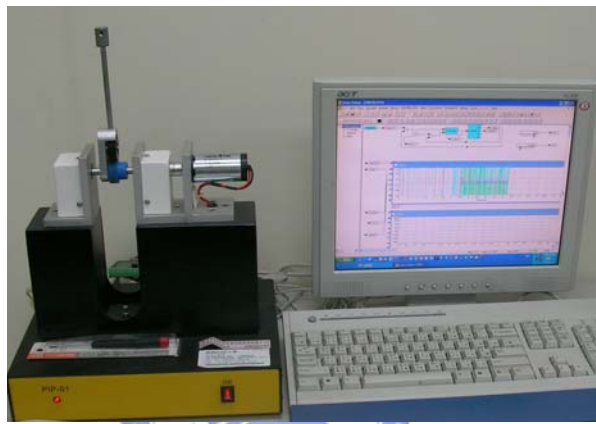
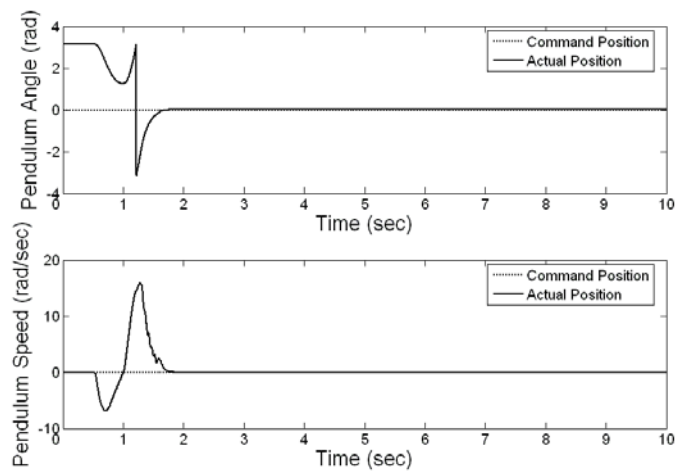
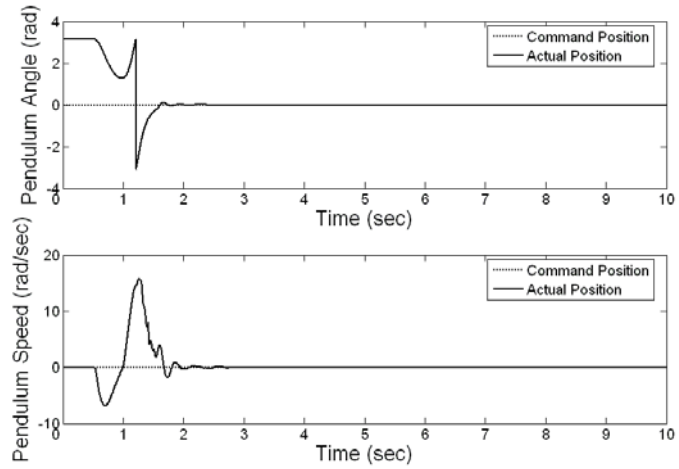


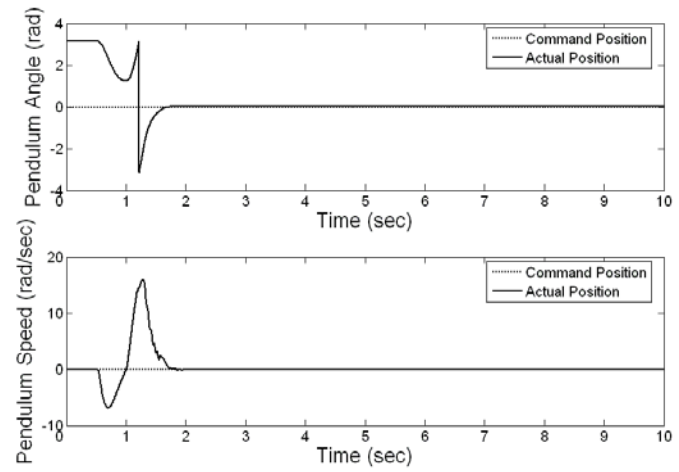
Figure 3.11: The experimental planetary train type inverted pendulum system.



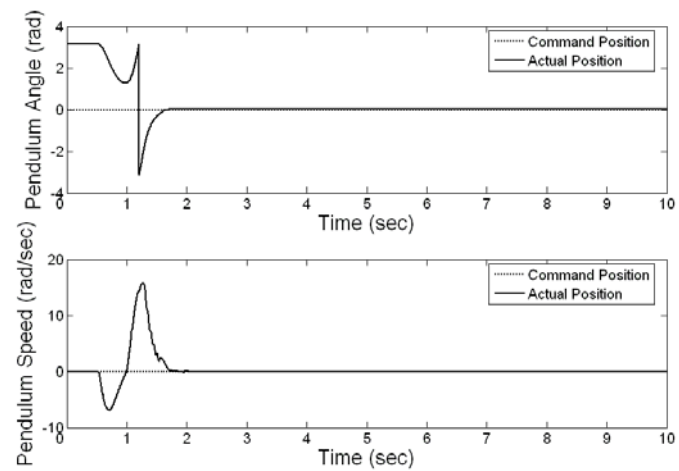
(a)



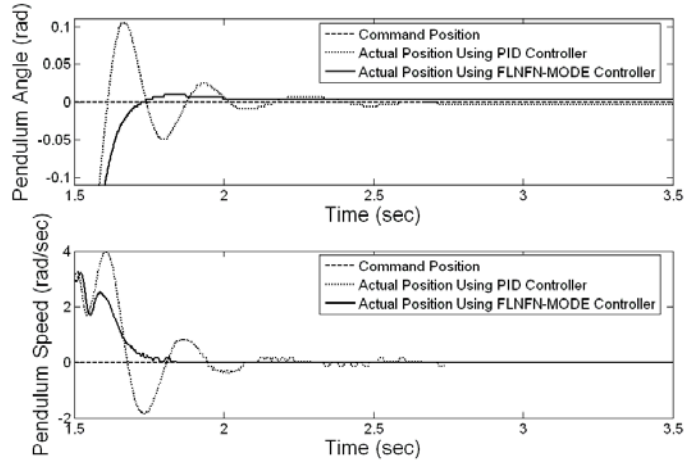
(b)



(c)



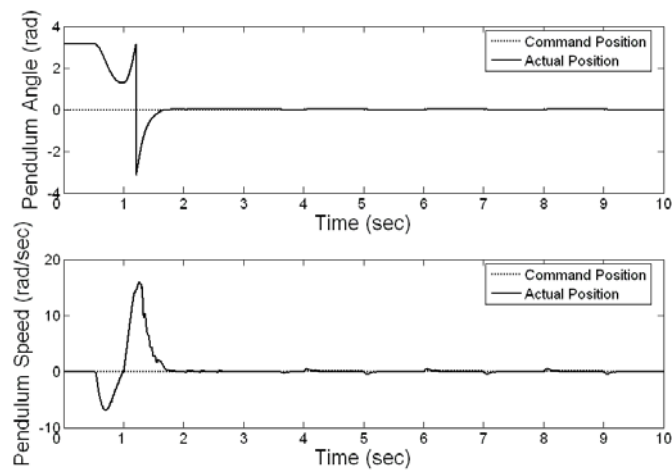
(d)



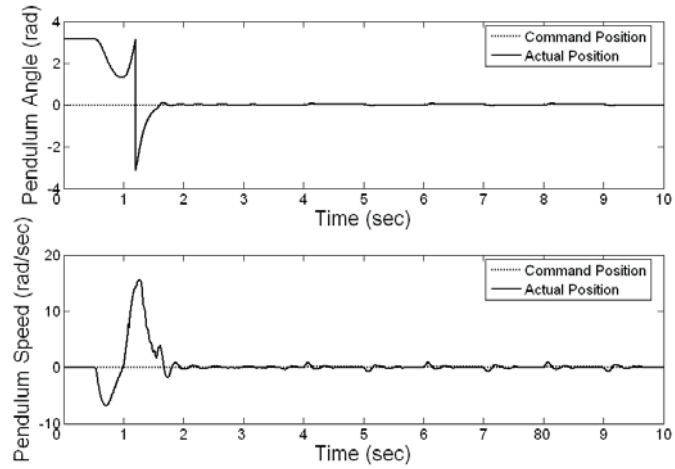
(e)

Figure 3.12: (a)-(d) Final regulation performance of the FLNFN-MODE controller, PID controller, FLNFN-DE controller and FLNFN-GA controller. (e) Scaling curves of the FLNFN-MODE controller and PID controller between the 1.5<sup>th</sup> second and the 3.5<sup>th</sup> second.

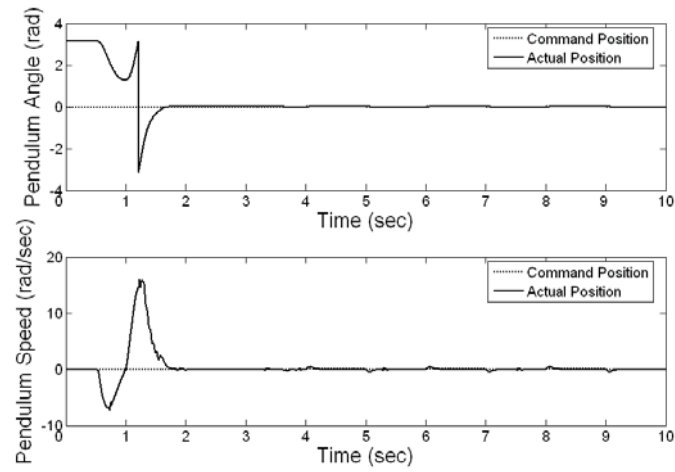
In Case 2, the tracking capability of the proposed system is tested using a square wave with amplitude  $\pm 0.02$  and frequency 0.5Hz. Figure 3.13(a)-(d) presents the regulation performance of the FLNFN-MODE controller, the PID controller, the FLNFN-DE controller, and the FLNFN-GA controller. Figure 3.13(e) plots the scaling curves of the FLNFN-MODE controller and the PID controller between the 4<sup>th</sup> second and the 8<sup>th</sup> second. A summary of the experimental results are presented in Table 3.3. As presented in Table 3.3, the proposed FLNFN-MODE controller outperforms the other controllers.



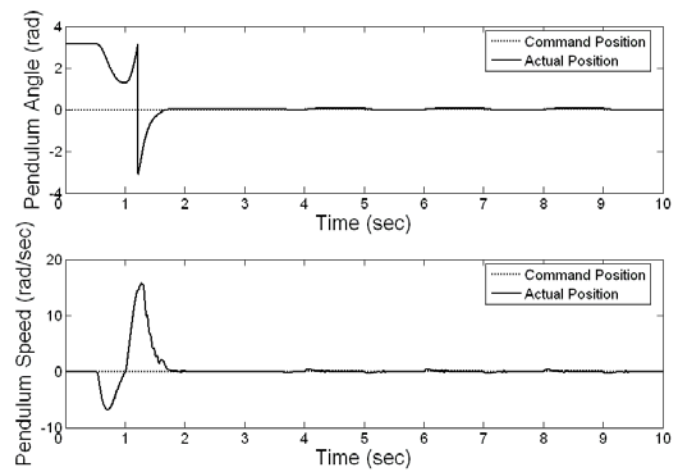
(a)



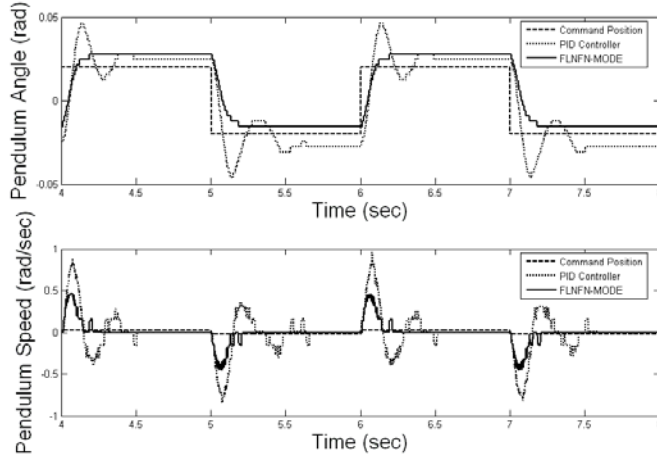
(b)



(c)



(d)



(e)

Figure 3.13: (a)-(d) Tracking of the FLNFN-MODE controller, PID controller, FLNFN-DE controller and FLNFN-GA controller, respectively, for a square wave with amplitude  $\pm 0.02$  and frequency 0.5Hz. (e) Tracking curves of the FLNFN-MODE controller and PID controller between the 4<sup>th</sup> second and the 8<sup>th</sup> second.

Table 3.3: Comparison of performance of various controllers to control of the planetary train type inverted pendulum system with a 0.1s sampling rate.

	FLNFN-MODE Controller		PID Controller		FLNFN-DE Controller		FLNFN-GA Controller	
	$SAE_{\theta_0}$	$SAE_{\dot{\theta}_0}$	$SAE_{\theta_0}$	$SAE_{\dot{\theta}_0}$	$SAE_{\theta_0}$	$SAE_{\dot{\theta}_0}$	$SAE_{\theta_0}$	$SAE_{\dot{\theta}_0}$
Case 1	33.3549	68.5454	34.0881	73.3770	33.4316	69.1548	33.5696	69.7101
Case 2	33.6101	72.3521	34.4442	80.4085	33.7968	72.9001	33.8245	73.3907

### Example 3: Control of Magnetic Levitation System

In order to construct a physical model of the behavior of the magnetic levitation system [68], it is necessary to make some statements about the system and also to make some simplifying assumptions. The physical model of the sphere and coil of the magnetic levitation system is shown in Fig. 3.14. The applied control is voltage, which is converted into a current within the mechanical unit. The current passes through an electromagnet which creates the corresponding magnetic field in its vicinity. The sphere is placed along the vertical axis of the electromagnet.

Using the fundamental principle of dynamics, the behavior of the ferromagnetic ball is given by the following electromechanical equation

$$m \frac{d^2x}{dt^2} = mg - F_B(x, i) \quad (3.22)$$

where  $m$  is the mass of the levitated ball,  $g$  denotes the acceleration due to gravity,  $x$  is the distance of the ball from the electromagnet,  $i$  is the current across the electromagnet, and  $F_B(x, i)$  is the magnetic control force.

The effect of the magnetic field from the electromagnetic is to introduce a magnetic dipole in the sphere which itself becomes magnetized. The force acting on the sphere is then composed of gravity and the magnetic force acting on the induced dipole. The magnetic field at a distance of  $x$  from the end of the coil may be calculated from the *Biot-Savart Law*. This states that the magnetic field produced by a small segment of wire,  $d\mathbf{S}$ , carrying a current  $I$  is given by

$$d\mathbf{B} = \frac{\mu_0 I d\mathbf{S} \times \mathbf{r}}{4\pi r^3} \quad (3.23)$$

where  $\mu_0$  is the permeability of free space and  $d\mathbf{S} \times \mathbf{r}$  is the vector product of  $d\mathbf{S}$  and  $\mathbf{r}$ .

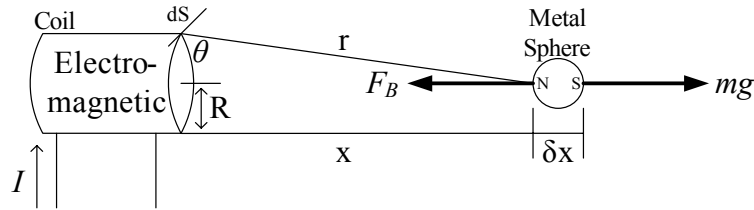


Figure 3.14: Sphere and coil arrangement of the magnetic levitation system.

We are interested in the field along the axis of the coil. Consideration of symmetry shows that the magnetic field perpendicular to the axis is zero. To evaluate the integral in Eq. (3.23), we position the current carrying element  $d\mathbf{S}$  to lie horizontally on the top of the coil and specify it by its unit vector components similarly, we specify the vector  $\mathbf{r}$  by its unit vector components. Then we have

$$d\mathbf{S} = dS[0 \ 1 \ 0] \quad \text{and} \quad \mathbf{r} = r[\sin \theta \ 0 \ \cos \theta]. \quad (3.24)$$

In Eq. (3.23), the vector product of  $d\mathbf{S}$  and  $\mathbf{r}$  from Eq. (3.24) is given by

$$d\mathbf{S} \times \mathbf{r} = dS r [\cos \theta \quad 0 \quad \sin \theta]. \quad (3.25)$$

Now, from considerations of symmetry the field component perpendicular to the coil axis must be zero on the axis. This is the  $\sin \theta$  component in Eq. (3.25). Also

$$\text{the radius of the coil } R = r \cos \theta. \quad (3.26)$$

Hence, from Eqs. (3.23), (3.25) and (3.26), the magnetic field component  $dB_x$  along the axis is given by

$$dB_x = \frac{\mu_0 I dS R}{4\pi r^3} \quad \text{and} \quad r^3 = (R^2 + x^2)^{\frac{3}{2}}. \quad (3.27)$$

Hence integrating Eq. (3.27) round a single loop gives

$$\begin{aligned} B_x &= \frac{\mu_0 I 2\pi R^2}{4\pi (R^2 + x^2)^{\frac{3}{2}}} \\ &= \frac{\mu_0 I R^2}{2 (R^2 + x^2)^{\frac{3}{2}}}. \end{aligned} \quad (3.28)$$

To evaluate the field due to the many turns along the axis of the coil, let  $n$  be the number of turns per meter and let  $L$  (m) be its length. Now, we sum all the contributions from all the turns of the coils, as shown in Fig 3.14. When Eq. (3.28) is used, the total axial field from all the turns of the coil becomes

$$B_{Total} = \frac{\mu_0 n I R^2}{2} \int_X^{X+L} \frac{dx}{(R^2 + x^2)^{\frac{3}{2}}} \quad (3.29)$$

Integrating Eq. (3.29) by parts gives

$$B_{Total} = \frac{\mu_0 n I R^2}{2} \left[ \frac{X+L}{R^2(R^2 + (X+L)^2)^{\frac{1}{2}}} - \frac{X}{R^2(R^2 + X^2)^{\frac{1}{2}}} \right] \quad (3.30)$$

We can rewrite Eq. (3.30) as

$$B(x) = K_1 I G(X) \quad (3.31)$$

The force on the ball due to the field is proportional to the induced dipole strength and the field strength. The induced dipole strength is itself proportional to the field strength and,

hence, the upwards force on the ball due to the field  $B$  is given by

$$F_b = K_1 K_2 I^2 (G(X) - G(X + \delta X)). \quad (3.32)$$

Therefore,

$$F_b \approx -K_3 I \delta X G'(X) \quad (3.33)$$

where  $G'(X)$  denotes the derivation and  $\delta X$  is the dipole separation. On the assumption that the poles are located at the centre of the mass of each hemisphere of the ball,  $\delta X$  is small compared to  $L$  and  $R$  and may be taken as a constant. Therefore, Eq. (3.33) becomes

$$F_b \approx K I^2 G'(X). \quad (3.34)$$

In this dissertation, the proposed control structure is shown in Fig. 3.15. The applied photo detector is used to detect the position of the levitated object and then to translate the signal as feedback signal.

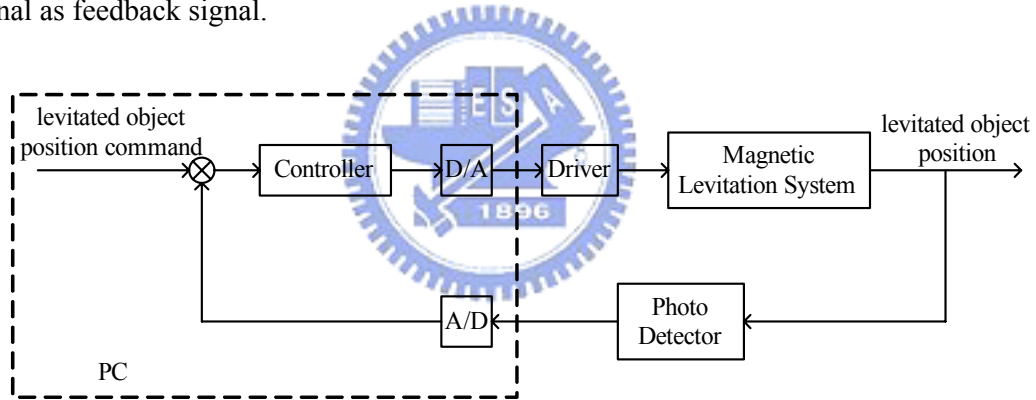


Figure 3.15: Control block diagram for the magnetic levitation system.

In this experiment, the proposed FLNFN-MODE controller is compared to the PID controller, the FLNFN-DE controller, and the FLNFN-GA controller. Each of the controllers is applied to control the magnetic levitation system. As in example 2, the PID controller with  $K_p=1.7$ ,  $K_i=0$ , and  $K_d=0.031$  is designed. The training patterns of the FLNFN model are generated using the various PID controllers with different control gains. Figure 3.16 shows an experimental magnetic levitation system which is tested to validate the experimentation results. In the following four cases, the FLNFN-MODE controller is demonstrated to have outperformed the other controllers.



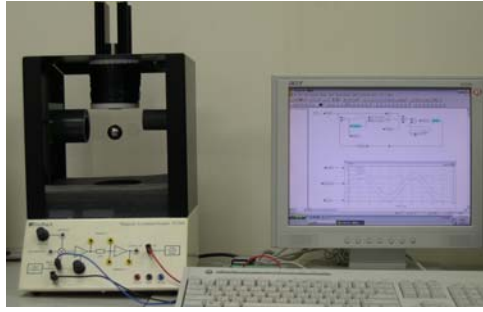
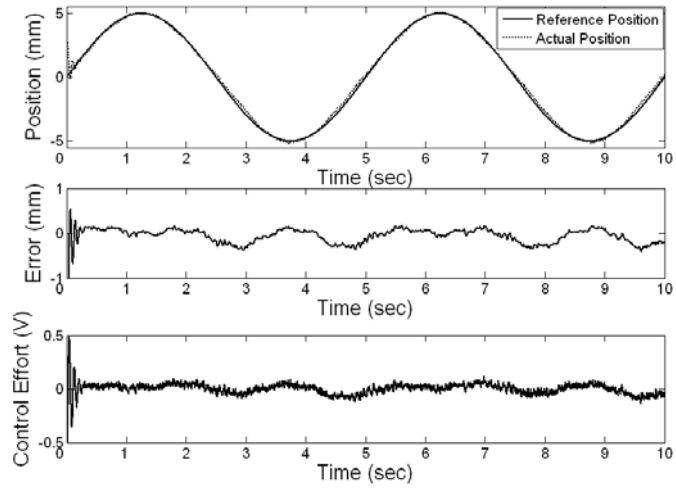


Figure 3.16: Experimental magnetic levitation system.

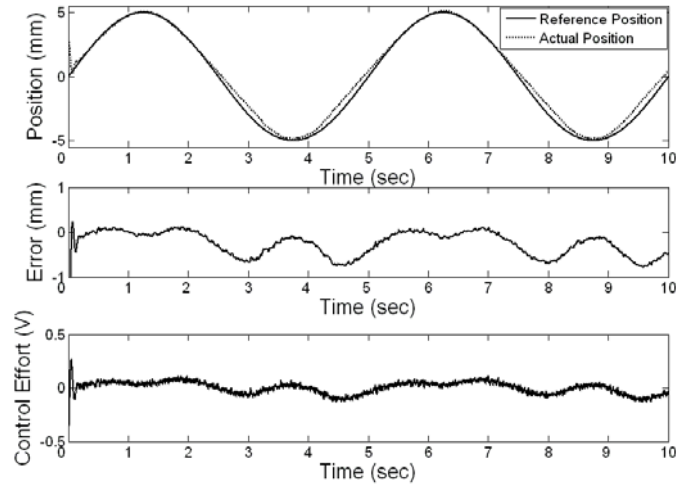
The first case and the second case are used to verify the tracking capability of the controllers. In the first case, the reference signal is given by a sinusoidal wave with amplitude 0.5 and frequency 0.2Hz, and in the second case, the reference signal is presented by a square wave with amplitude 0.5 and frequency 0.2Hz. The final experimental results of the FLNFN-MODE controller, the PID controller, the FLNFN-DE controller and the FLNFN-GA controller are shown in Fig. 3.17(a)-(d) and Fig. 3.18(a)-(d). To evaluate their performance, a performance index, the sum of absolute error (SAE), is defined by

$$SAE_p = \sum |P^{ref} - P| \quad (3.35)$$

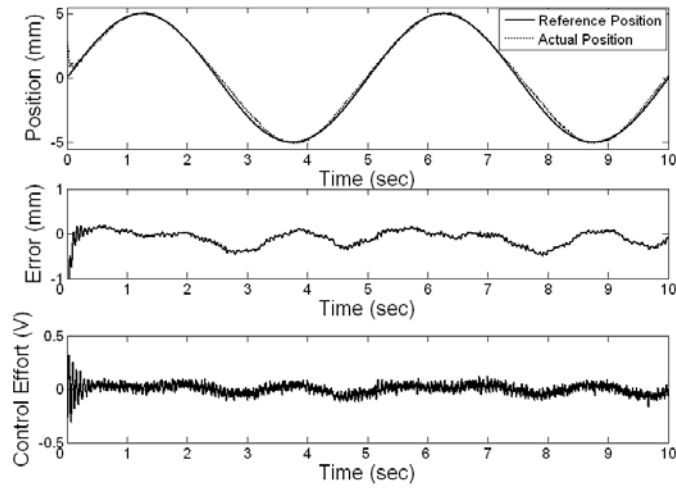
where  $P^{ref}$  and  $P$  are the reference trajectory and the actual position of the simulated system, respectively. In the first case, the  $SAE_p$  values of the FLNFN-MODE controller, the PID controller, the FLNFN-DE controller and the FLNFN-GA controller are, respectively, 12.9002, 27.7017, 13.9169 and 15.1572, which values are given in the second row of Table 3.4. In the second case, the  $SAE_p$  values of the FLNFN-MODE controller, the PID controller, the FLNFN-DE controller and the FLNFN-GA controller are, respectively, 48.4033, 85.7310, 50.7233 and 53.5194, which values are given in the third row of Table 3.4. The proposed FLNFN-MODE controller has a smaller  $SAE_p$  value than the other controllers.



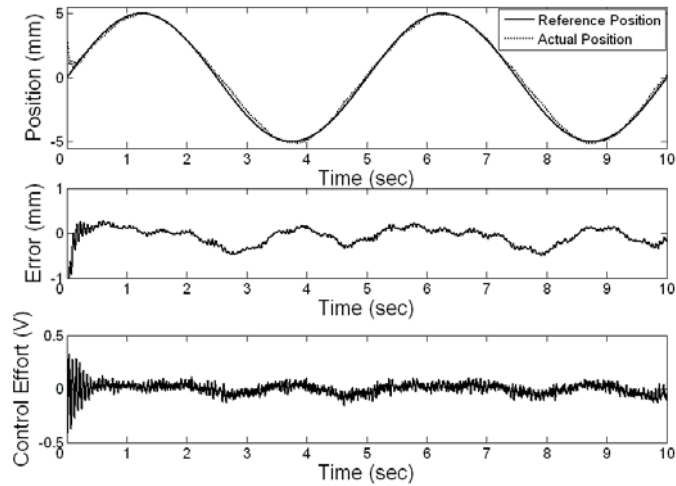
(a)



(b)

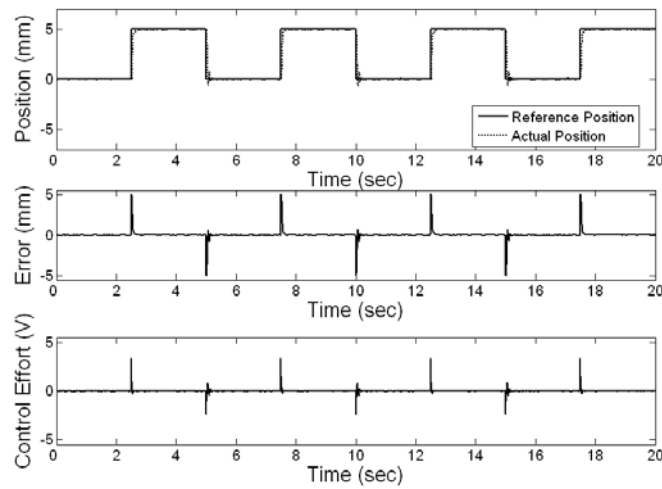


(c)

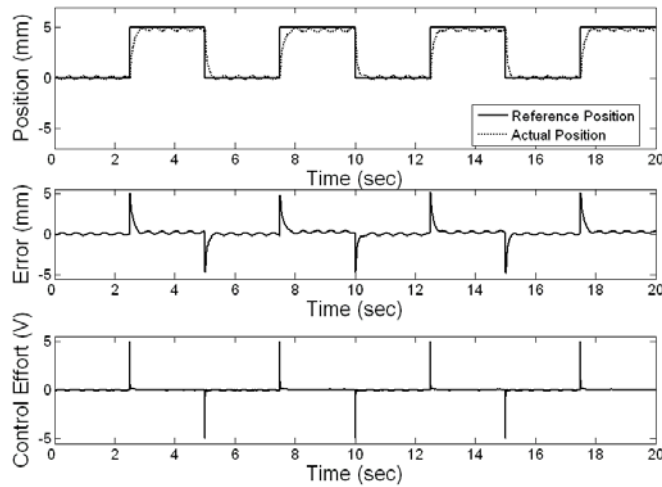


(d)

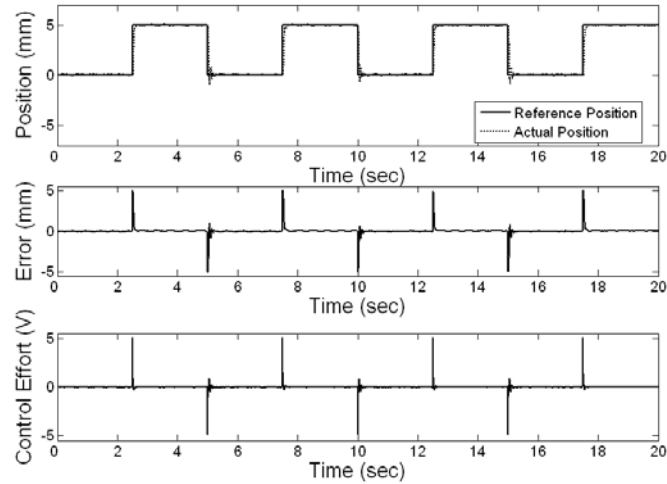
Figure 3.17: (a)-(d) Experimental results of FLNFN-MODE controller, PID controller, FLNFN-DE controller and FLNFN-GA controller due to periodic sinusoidal command for reference position and actual position, tracking error and control effort.



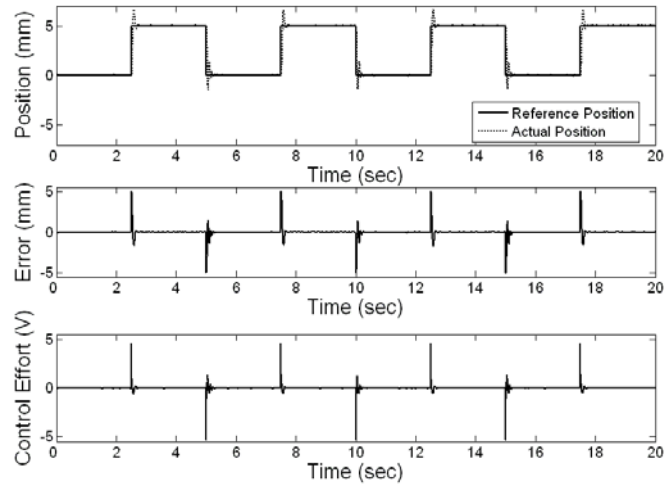
(a)



(b)



(c)

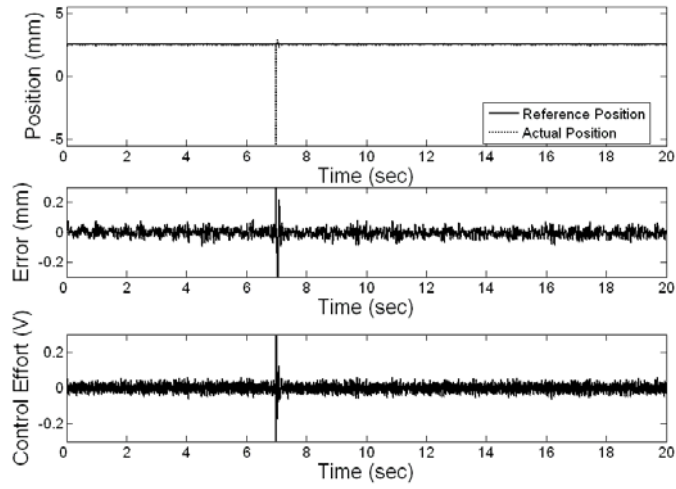


(d)

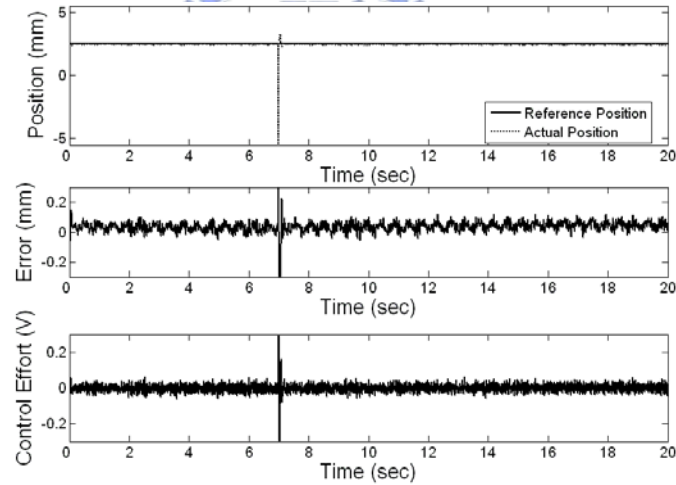
Figure 3.18: (a)-(d) Experimental results of FLNFN-MODE controller, PID controller, FLNFN-DE controller and FLNFN-GA controller due to periodic square command for reference position and actual position, tracking error and control effort.

The third experimentation is performed to demonstrate the noise-rejection ability of the four controllers when some unknown impulse noise is imposed on the process. One impulse noise value,  $-8mm$ , is added to the plant output at the 7<sup>th</sup> second. A set-point of  $2.5mm$  is adopted in this experimental case. The FLNFN-MODE controller can recover from the disturbance quickly after the occurrence of the impulse noise, as shown in Fig. 3.19(a). Figures 3.19(b)-(d) present the behaviors of the other three controllers under the influence of impulse noise. The  $SAE_p$  values of the FLNFN-MODE controller, the PID controller, the

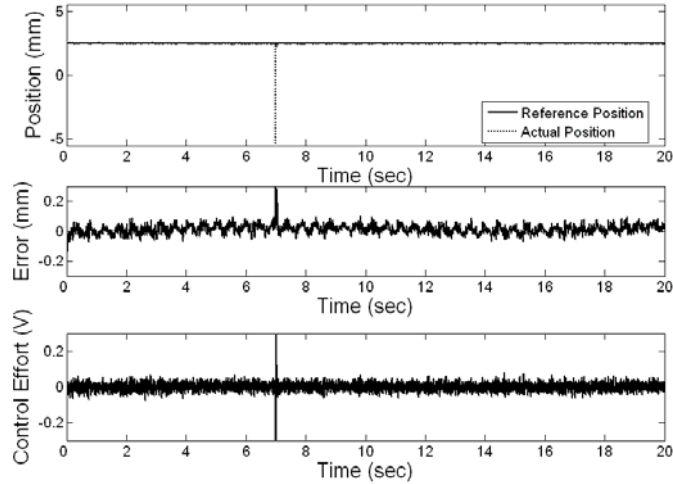
FLNFN-DE controller, and the FLNFN-GA controller are, respectively, 9.2709, 12.7345, 10.1515 and 10.8771, which are shown in the fourth row of Table 3.4. The FLNFN-MODE controller performs quite well.



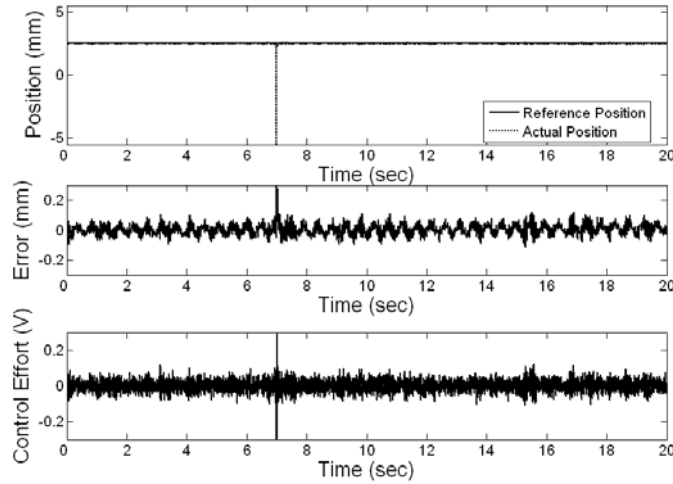
(a)



(b)



(c)

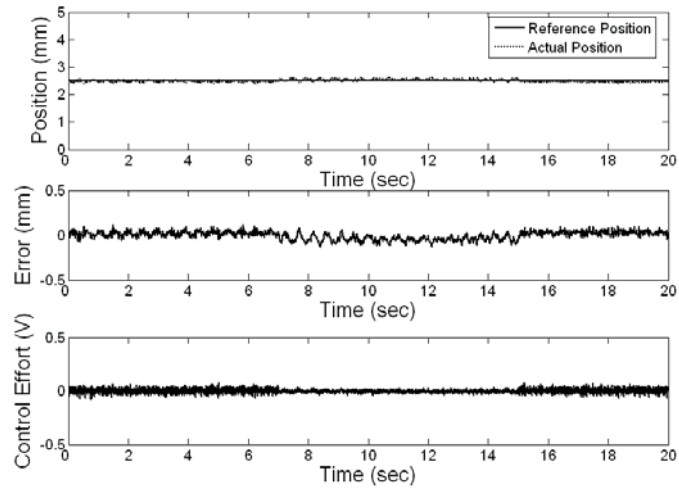


(d)

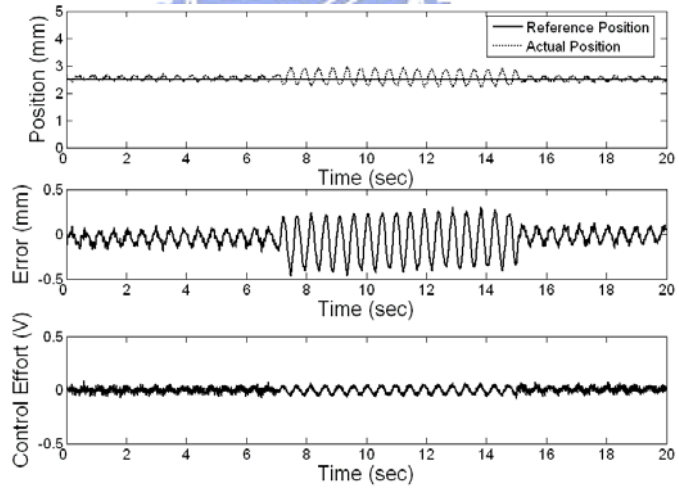
Figure 3.19: (a)-(d) Behavior of the FLNFN-MODE controller, PID controller, FLNFN-DE controller and FLNFN-GA controller under impulse noise in a magnetic levitation system for reference and actual positions, tracking error, and control effort.

One common characteristic of many industrial-control processes is that their parameters tend to change in an unpredictable way. The signal  $0.6 * u(t - 0.005)$  is added to the plant input between the 7<sup>th</sup> second and the 15<sup>th</sup> second in the fourth experiment to test the robustness of the four controllers. A set-point of  $2.5\text{mm}$  is adopted in this fourth experiment. Figures 3.20(a)-(d) present the behaviors of the FLNFN-MODE controller, the PID controller, the FLNFN-DE controller, and the FLNFN-GA controller when in the plant dynamics change. The  $SAE_p$  values of the FLNFN-MODE controller, the PID controller, the FLNFN-DE

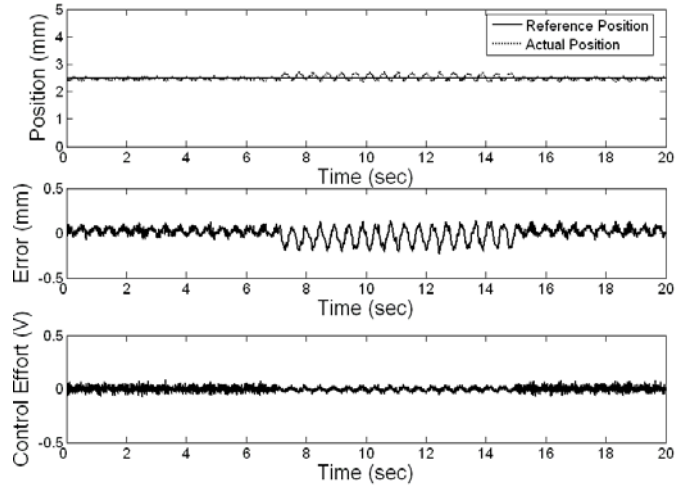
controller, and the FLNFN-GA controller are, respectively, 7.9469, 24.0004, 11.0672 and 14.0844, which values are shown in the fifth row of Table 3.4. The results present the favorable control and disturbance rejection capabilities of the trained FLNFN-MODE controller in the magnetic levitation system.



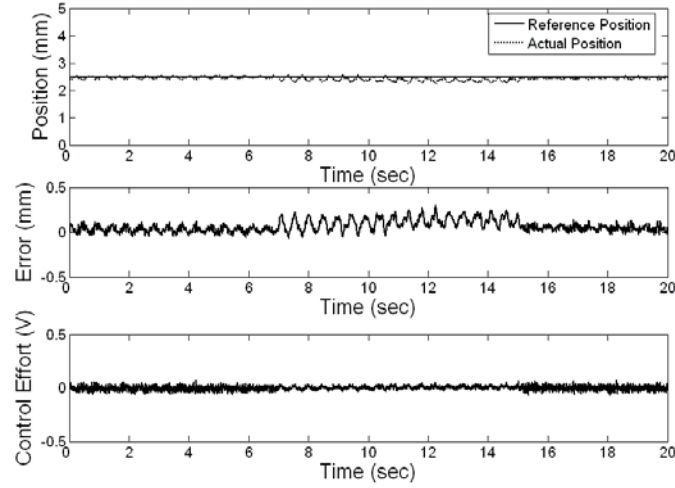
(a)



(b)



(c)



(d)

Figure 3.20: (a)-(d) Behavior of the FLNFN-MODE controller, PID controller, FLNFN-DE controller, and FLNFN-GA controller when a change occurs in the magnetic levitation system for reference and actual positions, tracking error, and control effort.

Table 3.4: Comparison of performance of various controllers to control of the magnetic levitation system with a 0.1s sampling rate.

$SAE_p = \sum  P^{ref} - P $	FLNFN-MODE Controller	PID Controller	FLNFN-DE Controller	FLNFN-GA Controller
Tracking sinusoidal wave	12.9002	27.70017	13.9169	15.1572
Tracking square wave	48.4033	85.7310	50.7233	53.5194
Influence of Impulse Noise	9.2709	12.7345	10.1515	10.8771
Effect of Change in Plant Dynamics	7.9469	24.0004	11.0672	14.0844



### 3.4 Summary

This dissertation proposes a functional-link-based neuro-fuzzy network based on a modified differential evolution (FLNFN-MODE) for nonlinear system control. The FLNFN-MODE controller adopts a nonlinear combination of input variables to the consequent part of fuzzy rules and uses a modified differential evolution to optimize the system parameters. We applied the FLNFN-MODE controller to the planetary train type inverted pendulum system and the magnetic levitation system in the VisSim. The experimental results demonstrate that the FLNFN-MODE controller obtains a smaller SAE value than the generally used FLNFN-DE, FLNFN-GA, and PID controllers for solving nonlinear control problems.



## Chapter 4

### **A Rule-Based Symbiotic Modified Differential Evolution for the FLNFN Model**

In this chapter, a rule-based symbiotic modified differential evolution (RSMODE) is proposed for the FLNFN model. The proposed RSMODE learning algorithm consists of the initialization phase to generate initial rule-based subpopulation, and the parameter learning phase to adjust the FLNFN parameters. The initialization phase can determine the number of rule-based subpopulation which satisfies the fuzzy partition of input variables. Initially, there is not any subpopulation. The rule-based subpopulation is automatically generated from training data by entropy measure. The parameter learning phase combines two strategies including a subpopulation symbiotic evolution (SSE) and a modified differential evolution (MODE). The SSE in which each individual represents a single fuzzy rule differs from original symbiotic evolution. Each subpopulation allows the rule itself to evolve. The MODE adopts a method to effectively search between the best individual and randomly chosen individuals. Therefore, the MODE not only explores the search space by randomly chosen individuals but also exploits the search capability of a near global optimal solution by the best-so-far individual.

## 4.1 A Basic Concept of Symbiotic Evolution

Elements of any ecological system live in an intricate web of interdependence. When two species of organisms live in close physical contact with each other, their relationship is called symbiotic. In a symbiotic relationship at least one of the organisms directly benefits from its close association with the other organism. There are three major forms of symbiotic relationships: mutualism, commensalisms, and parasitism [69].

*Mutualism* – A reciprocal relationship in which two different species live in a symbiotic way where both species benefit and are dependent upon the relationship, that is, both species benefit by the relationship.

*Commensalism* – A relationship in which one species derives food or shelter from another species without seriously harming that organism or providing any benefits in return, that is, one species benefits while the other species is not affected.

*Parasitism* – A relationship between two species in which one species (the parasite) nourishes itself to the detriment of the other species (the host), that is, one species benefits and the other is harmed.

Many researchers [70]-[74] have adopted the concept of mutualisms to develop symbiotic evolution. In addition, recent development [75] in the concept of commensalisms has provided a multi-swarm cooperative particle swarm optimizer method by the phenomenon of symbiosis in natural ecosystems.

## 4.2 A Rule-Based Symbiotic Modified Differential Evolution

This section represents the proposed rule-based symbiotic modified differential evolution (RSMODE). The RSMODE learning algorithm comprises the initialization phase and the parameter learning phase. The initialization phase uses the entropy measure that determines proper input space partitioning, finds the mean and variance of the Gaussian membership

function and the number of rules. Next, the initial rule-based subpopulation is created according to a range of the mean and variance of the membership function. The parameter learning phase consists of a subpopulation symbiotic evolution (SSE) and a modified differential evolution (MODE). Each individual in each subpopulation evolves separately using a modified differential evolution. But in order to evaluate each individual, the individual is composed a fuzzy system using other individuals (rules) in other subpopulations. The detailed flowchart of the proposed RSMODE learning algorithm is presented in Fig. 4.1.

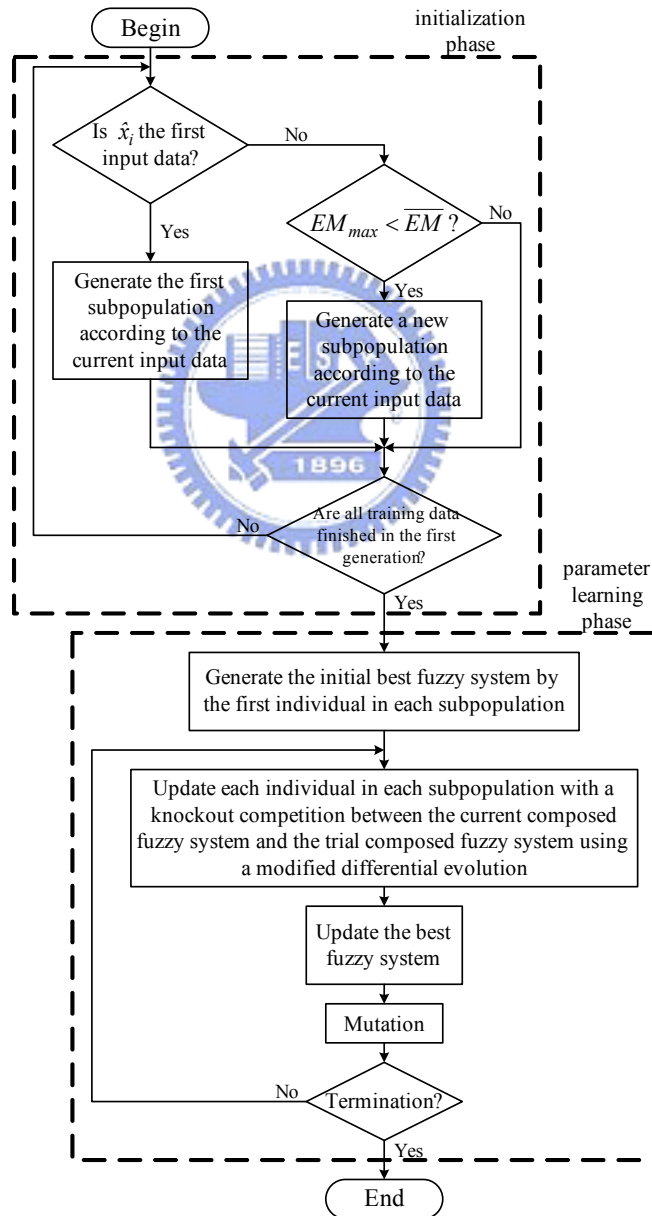


Figure 4.1: Flowchart of the RSMODE learning algorithm.

### 4.2.1 Initialization Phase

In this dissertation, we finish initialization phase from training data in the first generation. This subsection introduces the production of initial rule-based subpopulation, covering the coding and initialization steps. The coding step involves the membership functions and the fuzzy rules of a fuzzy system that represent individuals and are suitable for subpopulation symbiotic evolution. The initialization step assigns the number of subpopulation before the evolution process begins.

#### A. Coding Step

The first step in RSMODE learning algorithm is the coding of a fuzzy rule into an individual. Figure 4.2 shows an example of a fuzzy rule coded into an individual where  $i$  and  $j$  are the  $i$ th dimension and the  $j$ th rule. Figure 4.2 describes a fuzzy rule given by Eq. (2.3), where  $m_{ij}$  and  $\sigma_{ij}$  are the mean and variance of a Gaussian membership function, respectively, and  $w_{kj}$  represents the corresponding link weight of the consequent part that is connected to the  $j$ th rule node. In this dissertation, a real number represents the position of each

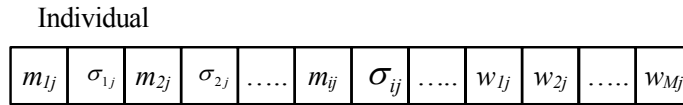


Figure 4.2: Coding a fuzzy rule into an individual in the RSMODE learning algorithm.

#### B. Initialization Step

For training data, finding the optimal solution is difficult because the range of training data is wide. Therefore, the data must be normalized. Let training data be transformed to the interval of  $[0, 1]$ :

$$\hat{x}_i = \frac{\hat{x}'_i - \hat{x}'_{i\_min}}{\hat{x}'_{i\_max} - \hat{x}'_{i\_min}} \quad (4.1)$$

where  $\hat{x}_i$  is the value after normalization;  $\hat{x}'_i$  is the vector of the  $i$ th dimension to be normalized;  $\hat{x}'_{i\_min}$  is the minimum value of vector  $\hat{x}'_i$ ;  $\hat{x}'_{i\_max}$  is the maximum value of vector  $\hat{x}'_i$ .

Before the RSMODE method is designed, the individuals that will constitute  $R$  initial subpopulation must be created. The first step in initialization phase is to create the initial first individual in each subpopulation to satisfy the fuzzy rule partition of input variables. The fuzzy rule partition strategy can determine whether a new rule should be extracted from the training data and determine the number of fuzzy rules in the universal of discourse of each input variable, since one cluster in the input space corresponds to one potential fuzzy logic rule, in which  $m_{ij}$  and  $\sigma_{ij}$  represent the mean and variance of that cluster, respectively. For each incoming data  $\hat{x}_i$ , the rule firing strength can be regarded as the degree to which the incoming data belongs to the corresponding cluster. Entropy measure between each data point and each membership function is calculated based on a similarity measure. A data point of closed mean will has lower entropy. Therefore, the entropy values between data points and current membership functions are calculated to determine whether or not to add a new rule into the initial first individual and create a new rule-based subpopulation space. For computational efficiency, the entropy measure can be calculated using the firing strength from  $\mu_{ij}^{(2)}(\hat{x}_i)$  as follow;

$$EM_j = -\sum_{i=1}^N D_{ij} \log_2 D_{ij} \quad (4.2)$$

where  $D_{ij} = \exp(u_{ij}^{(2)}(\hat{x}_i)^{-1})$  and  $EM_j \in [0,1]$ . According to Eq. (4.2), the measure is used to generate a new fuzzy rule and new functional link bases for new incoming data is described as follows. The maximum entropy measure

$$EM_{\max} = \max_{1 \leq j \leq R} EM_j \quad (4.3)$$

is determined, where  $R$  is the number of existing rules. If  $EM_{\max} \leq \overline{EM}$ , then a new rule and a new rule-based subpopulation space are generated, where  $\overline{EM} \in [0,1]$  is a prespecified threshold.

Once a new rule has been generated, the next step is to assign the initial first individual in the new rule-based subpopulation by the initial mean and variance to the new membership function and the corresponding link weight. Hence, the mean, variance and weight for the new rule are set as follows;

$$m_{ij} = \hat{x}_i \quad (4.4)$$

$$\sigma_{ij} = \sigma_{init} \quad (4.5)$$

$$w_{kj} = random[-1,1] \quad (4.6)$$

where  $\hat{x}_i$  is the current input data and  $\sigma_{init}$  is a prespecified constant.

The second step is to create other individuals in each subpopulation according to a range of the initial first individual. The following formulations show the production of the other individuals.

$$\text{Mean: } Individual[d] = m_{ij} + random[0,1] \times \sigma_{ij}, \text{ where } d=1,3,\dots,2 \times N-1 \quad (4.7)$$

$$\text{Variance: } Individual[d] = 2 \times random[0,1] \times \sigma_{ij}, \text{ where } d=2,4,\dots,2 \times N \quad (4.8)$$

$$\text{Other parameters: } Individual[d] = random[-1,1], \text{ where } d > 2 \times N \quad (4.9)$$

where  $d$  is the site of each individual and  $m_{ij}$  and  $\sigma_{ij}$  are the corresponding mean and variance, respectively, of the initial first individual.

## 4.2.2 Parameter Learning Phase

The parameter learning combines two strategies including a subpopulation symbiotic evolution (SSE) and a modified differential evolution (MODE). Each subpopulation allows

the individual (rule) itself to evolve by evaluating the composed fuzzy system. Figure 4.3 shows the structure of the individual in the rule-based symbiotic modified differential evolution. The parameter learning process is described step-by-step below.

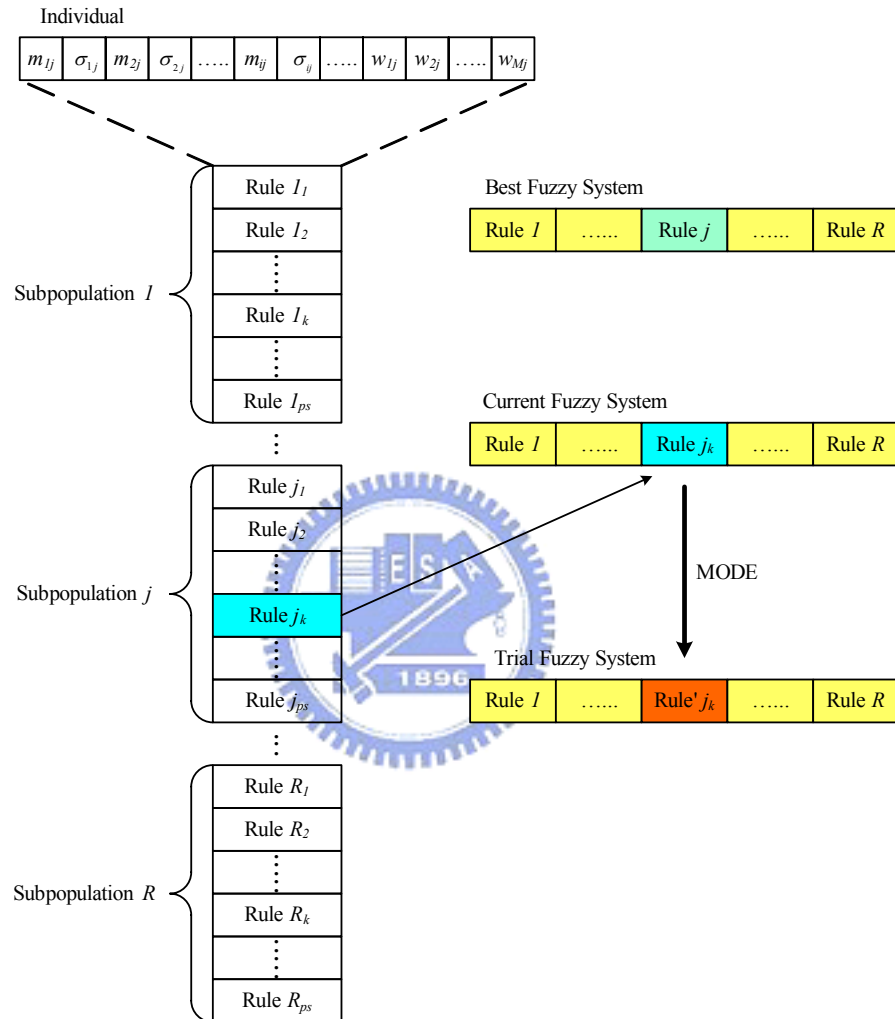


Figure 4.3: Structure of the individual in the RSMODE learning algorithm.

*Step 1: Generate the Initial Best Fuzzy System*

In this step, we orderly select the first individual from each subpopulation, and compose a fuzzy system as the initial best fuzzy system.

*Step 2: Update Each Individual in Each Subpopulation using MODE*

In order to update each individual in each subpopulation, we use a modified differential



evolution to select the better individual to the next step. Figure 4.4 gives an example of the MODE process. Hence, this step comprises of three components - parent choice phase, offspring generation phase and survivor selection phase.

*Step 2.1: Parent Choice Phase*

Each individual in the current generation is allowed to breed through mating with other randomly selected individuals from the subpopulation. Specifically, for each current individual  $x_{k,g}$ ,  $k=1, 2, \dots, PS$ , where  $g$  denotes the current generation and  $PS$  denotes the population size, three other random individuals  $x_{r_1,g}$ ,  $x_{r_2,g}$  and  $x_{r_3,g}$  are selected from the subpopulation such that  $r_1, r_2$ , and  $r_3 \in \{1,2,\dots,PS\}$  and  $k \neq r_1 \neq r_2 \neq r_3$ . This way, a parent pool of four individuals is formed to breed an offspring.

*Step 2.2: Offspring Generation Phase*

After choosing the parents, MODE applies a differential operation to generate a mutated individual  $v_{k,g+1}$ , according to the following equation:

$$v_{k,g+1} = x_{r_1,g} + (1-F) \cdot (x_{r_2,g} - x_{r_3,g}) + F \cdot (x_{best} - x_{r_1,g}) \quad (4.10)$$

where  $F$ , commonly known as scaling factor, is defined as  $\frac{g}{G}$  to control the rate at which the subpopulation evolves,  $g$  denotes the current generation,  $G$  is the maximum number of generations, and  $x_{best}$  is the corresponding parameter of the best fuzzy system. To complement the differential operation search strategy, then uses a crossover operation, often referred to as discrete recombination, in which the mutated individual  $v_{k,g+1}$  is mated with  $x_{k,g}$  and generates the offspring  $u_{k,g+1}$ . The element of trial individual  $u_{k,g+1}$  are inherited from  $x_{k,g}$  and  $v_{k,g+1}$ , determined by a parameter called crossover probability ( $CR \in [0,1]$ ), as follows:

$$u_{kd,g+1} = \begin{cases} v_{kd,g+1}, & \text{if } Rand(d) \leq CR \\ x_{kd,g}, & \text{if } Rand(d) > CR \end{cases} \quad (4.11)$$

where  $d = 1, 2, \dots, D$  denotes the  $d$ th element of individual vectors.  $Rand(d) \in [0, 1]$  is the  $d$ th evaluation of a random number generator. For searching in nonseparable and multimodal landscapes  $CR = 0.9$  is a good choice [55].

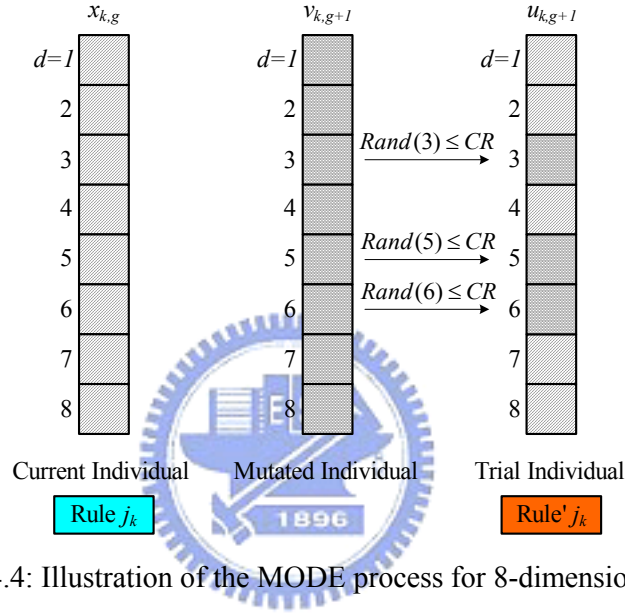


Figure 4.4: Illustration of the MODE process for 8-dimensional vector.

### Step 2.3: Survivor Selection Phase

MODE applies selection pressure only when selecting survivors. First, the current composed fuzzy system embeds the current individual  $x_{i,g}$  into the best fuzzy system and the trial composed fuzzy system embeds the trial individual  $u_{i,g+1}$  into the best fuzzy system. Second, a knockout competition is played between the current composed fuzzy system and the trial composed fuzzy system. Then, the corresponding individual of the winner is selected deterministically based on objective function values and promoted to the next phase. In this dissertation, we adopt a fitness function (i.e., objective function) to evaluate the performance of these composed fuzzy systems. The fitness function is defined as follows.

$$F = \frac{1}{1 + \sqrt{\frac{1}{N_t} \sum_{l=1}^{N_t} (y_l - \bar{y}_l)^2}} \quad (4.12)$$

where  $y_l$  represents the model output of the  $l$ th data;  $\bar{y}_l$  represents the desired output of the  $l$ th data, and  $N_t$  represents the number of the training data.

*Step 3: Update the Best Fuzzy System*

Compare the fitness value of the current composed fuzzy system, the trial composed fuzzy system and the best fuzzy system. If the fitness value of the current composed fuzzy system exceeds those of the best fuzzy system, then the best fuzzy system is replaced with the current composed fuzzy system. If the fitness value of the trial composed fuzzy system exceeds those of the best fuzzy system, and then the best fuzzy system is replaced with the trial composed fuzzy system.

*Step 4: Mutation*

After the above process yielded offspring, no new information is introduced to the each subpopulation at the site of an individual. As a source of new sites, mutation should be used sparingly because it is a random search operator. In the following simulations, a mutation rate was set to  $1/(2*N+M)$ , meaning that, on average, only one trial parameter is mutated, where  $N$  is the number of input variables,  $M$  is the number of basis function of FLNFN and  $2*N+M$  is the length of each individual. Mutation is an operator that randomly alters the allele of a element. The mutation adopted in MODE to yield diversity. The individual suffers from a mutation to avoid falling in a local optimal solution and to ensure the searching capacity of approximate global optimal solution. Figure 4.5 shows the mutation of an individual. The mutation value is generated according to Eqs. (4.7)-(4.9), where  $m_{ij}$  and  $\sigma_{ij}$  are the corresponding mean and variance, respectively, of the current individual. Following the mutation step, a new individual can be introduced into the each subpopulation.

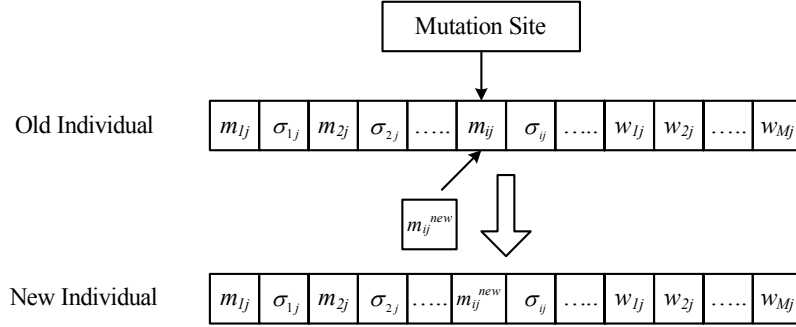


Figure 4.5: A mutation operation in the rule-based symbiotic modified differential evolution.

### 4.3 Experimental Results

This dissertation evaluated the performance of the proposed FLNFN controller using a rule-based symbiotic modified differential evolution (FLNFN-RSMODE) for nonlinear control systems. This section presents several examples and compares the performance with that of the FLNFN-RSDE controller, the FLNFN-DE controller and the FLNFN-GA controller. In the nonlinear control system problems, FLNFN-RSMODE is adopted to design controllers in three simulations - water bath temperature control system, ball and beam system and backing up the truck. Table 4.1 presents the parameter settings before training used in the three computer simulations for the RSMODE. In the RSDE and DE, the population size is set to 50, the maximum number of generation is set to 2000, and the crossover rate is set to 0.9. In the GA, the population size is set to 50, the maximum number of generation is set to 2000, the crossover rate is set to 0.5, and the mutation rate is set to 0.3.

Table 4.1: Parameter settings before training.

Parameter	Value
Population Size	50
Maximum Number of Generation	2000
Crossover Rate	0.9
Mutation Rate	$1/(2*N+M)$
Coding Type	Real Number

## Example 1: Control of Water Bath Temperature System

The description of the system is the same as Example 1 of Section 2.3. In initialization phase, four subpopulations are generated. This dissertation compares the FLNFN-RSMODE controller to the FLNFN-RSDE controller, the FLNFN-DE controller and the FLNFN-GA controller. Each of these controllers is applied to the water bath temperature control system. The performance measures include the set-points regulation, the influence of impulse noise, and a large parameter variation in the system, and the tracking capability of the controllers.

Figure 4.6 plots the learning curves of the best performance of the FLNFN-RSMODE controller for the fitness value, the FLNFN-RSDE controller, the FLNFN-DE controller and the FLNFN-GA controller, after the learning process of 2000 generations. Figure 4.7(a) presents the regulation performance of the FLNFN-RSMODE controller. The regulation performance was also tested using the FLNFN-RSDE controller, the FLNFN-DE controller and the FLNFN-GA controller. Figure 4.7(b) plots the error curves of the FLNFN-RSMODE controller, the FLNFN-RSDE controller, the FLNFN-DE controller and the FLNFN-GA controller. Figure 4.8(a) and (b) present the behaviors of the FLNFN-RSMODE controller, the FLNFN-RSDE controller, the FLNFN-DE controller and the FLNFN-GA controller under the influence of impulse noise, and the corresponding errors, respectively. Figure 4.9(a) presents the behaviors of the FLNFN-RSMODE controller when in the plant dynamics change. Figure 4.9(b) presents the corresponding errors of the FLNFN-RSMODE controller, the FLNFN-RSDE controller, the FLNFN-DE controller and the FLNFN-GA controller. Figure 4.10(a) presents the tracking performance of the FLNFN-RSMODE controller. Figure 4.10(b) presents the corresponding errors of the FLNFN-RSMODE controller, the FLNFN-RSDE controller, the FLNFN-DE controller, and the FLNFN-GA controller. The aforementioned simulation results, presented in Table 4.2, demonstrate that the proposed FLNFN controller outperforms other controllers.

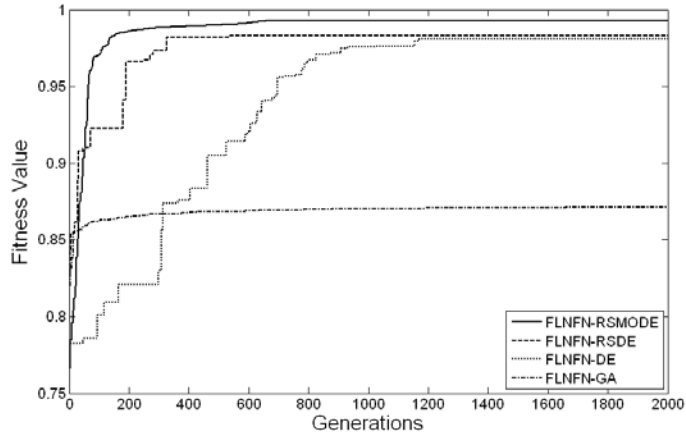
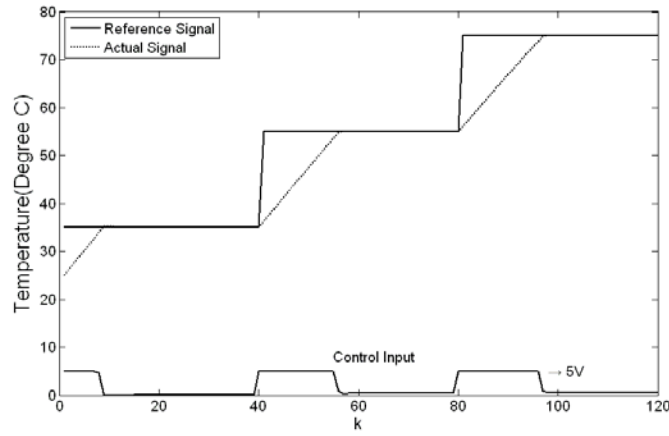
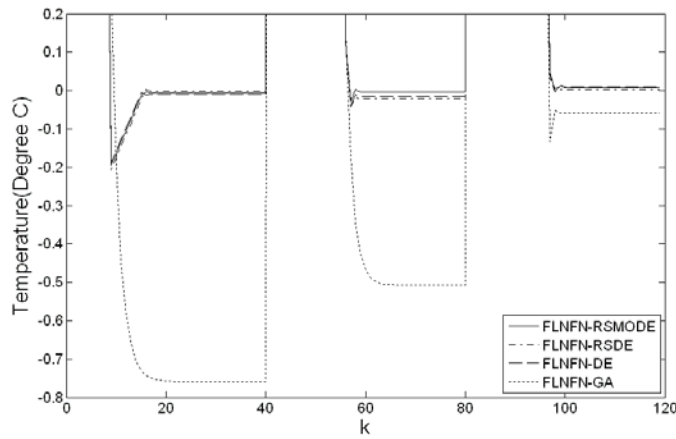


Figure 4.6: Learning curves of best performance of the FLNFN-RSMODE, FLNFN-RSDE, FLNFN-DE and FLNFN-GA in Example 1.



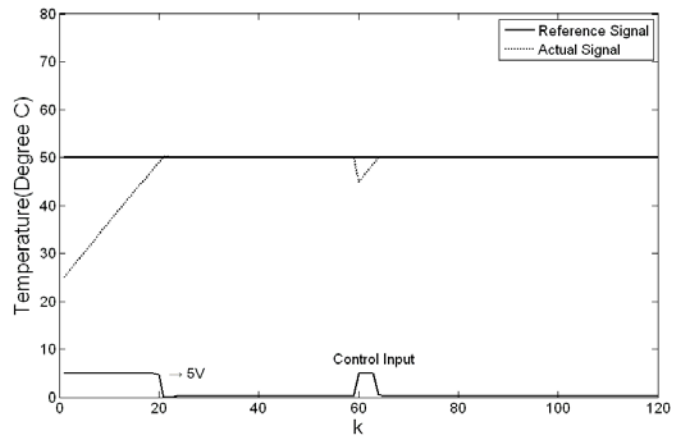
(a)



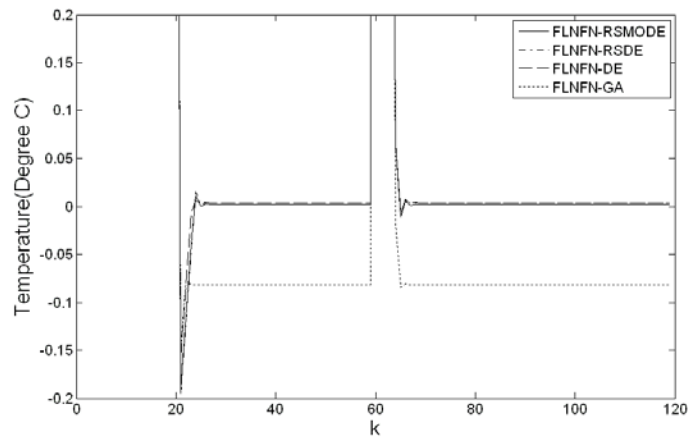
(b)

Figure 4.7: (a) Final regulation performance of FLNFN-RSMODE controller in water bath system. (b) Error curves of the FLNFN-RSMODE controller, FLNFN-RSDE controller, the

FLNFN-DE controller, and FLNFN-GA controller.

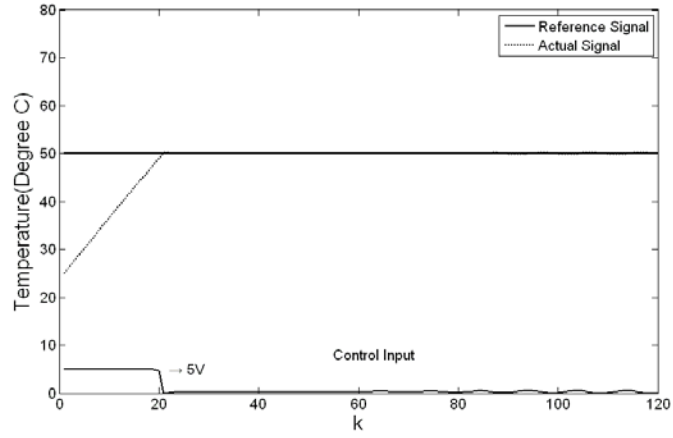


(a)

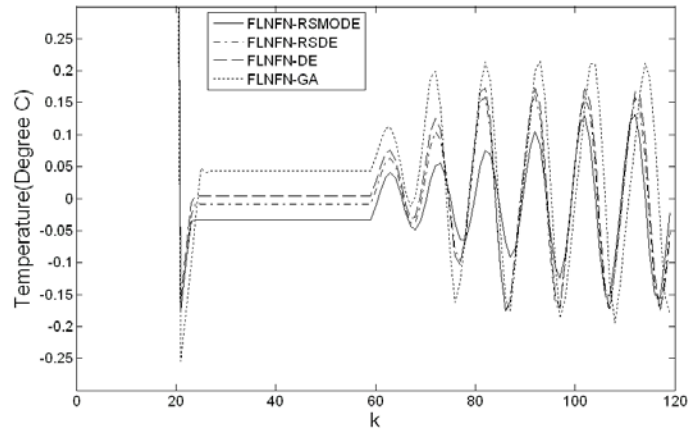


(b)

Figure 4.8: (a) Behavior of FLNFN-RSMODE controller under impulse noise in water bath system. (b) Error curves of FLNFN-RSMODE controller, FLNFN-RSDE controller, the FLNFN-DE controller and FLNFN-GA controller.

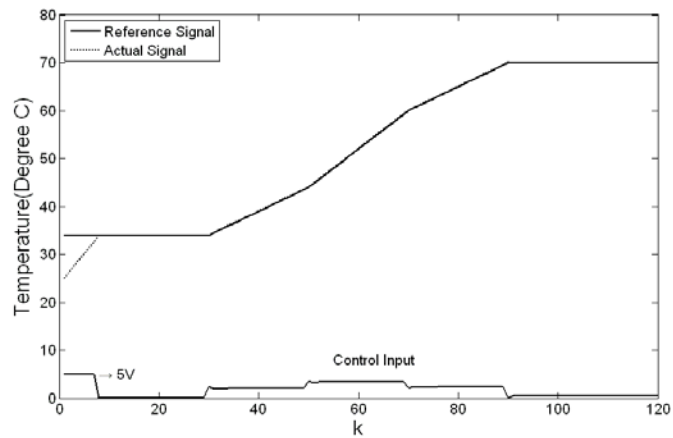


(a)



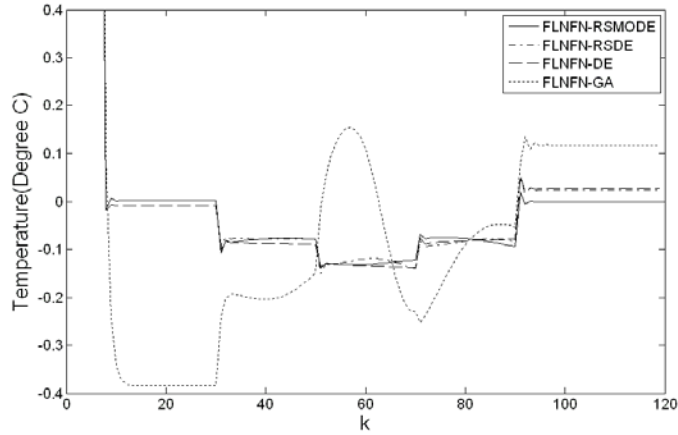
(b)

Figure 4.9: (a) Behavior of FLNFN-RSMODE controller when a change occurs in the water bath system. (b) Error curves of FLNFN-RSMODE controller, FLNFN-RSDE controller, the FLNFN-DE controller, and FLNFN-GA controller.



(a)





(b)

Figure 4.10: (a) Tracking of FLNFN-RSMODE controller when a change occurs in the water bath system. (b) Error curves of FLNFN-RSMODE controller, FLNFN-RSDE controller, the FLNFN-DE controller, and FLNFN-GA controller.

Table 4.2: Comparison of performance of various controllers to control of water bath temperature system.

$SAE = \sum_{k=1}^{120}  y_{ref}(k) - y(k) $	FLNFN-RSMODE Controller	FLNFN-RSDE Controller	FLNFN-DE Controller	FLNFN-GA Controller
Regulation Performance	352.66	352.81	352.91	372.85
Influence of Impulse Noise	270.46	270.76	270.65	282.21
Effect of Change in Plant Dynamics	262.63	263.21	263.25	270.66
Tracking Performance	41.73	42.56	42.92	62.02

## Example 2: Control of the Ball and Beam System

The description of the system is the same as Example 1 of Section 2.3. In initialization phase, 14 subpopulations are generated. This example was simulated 30 times. Figure 4.11 plots the learning curves of the best performance of the FLNFN-RSMODE controller for the fitness value, the FLNFN-RSDE controller, the FLNFN-DE controller and the FLNFN-GA controller, after the learning process of 2000 generations. The FLNFN-RSMODE controller after learning was tested under the following four initial conditions;  $x(0) = [2.4, -0.1, 0.6, 0.1]^T$ ,  $[1.6, 0.05, -0.5, -0.05]^T$ ,  $[-1.6, -0.05, 0.5, 0.05]^T$  and  $[-2.4, 0.1, -0.6, -0.1]^T$ . Figure 4.12

plots the output responses of the closed-loop ball and beam system controlled by the FLNFN-RSMODE controller and the FLNFN-RSDE controller. These responses approximate those of the controller under the four initial conditions. In this figure, the curves of the FLNFN-RSMODE controller tend quickly to stabilize. Figure 4.13 also shows the behavior of the four states of the ball and beam system, starting for the initial condition  $[-2.4, 0.1, -0.6, -0.1]^T$ . In this figure, the four states of the system decay gradually to zero. The results show the perfect control capability of the trained FLNFN-RSMODE controller. The performance of the FLNFN-RSMODE controller is compared with that of the FLNFN-RSDE controller, the FLNFN-DE controller and the FLNFN-GA controller. Table 4.3 presents the comparison results. The results demonstrate that the proposed FLNFN-RSMODE controller outperforms other controllers.

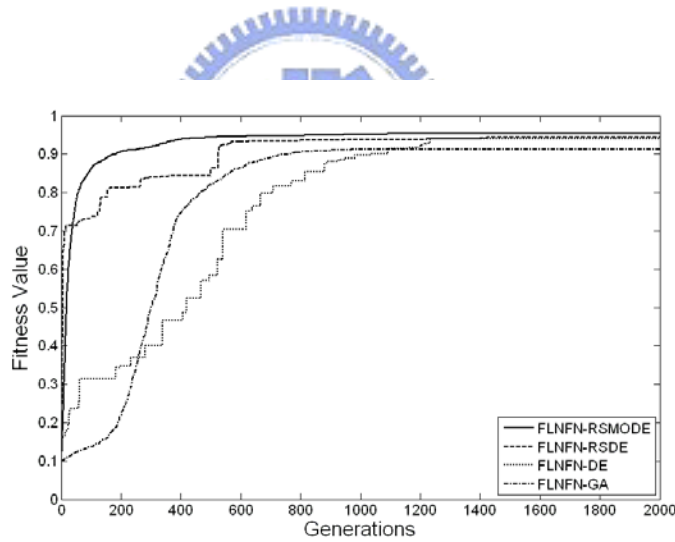


Figure 4.11: Learning curves of best performance of the FLNFN-RSMODE, FLNFN-RSDE, FLNFN-DE, and FLNFN-GA in Example 2.

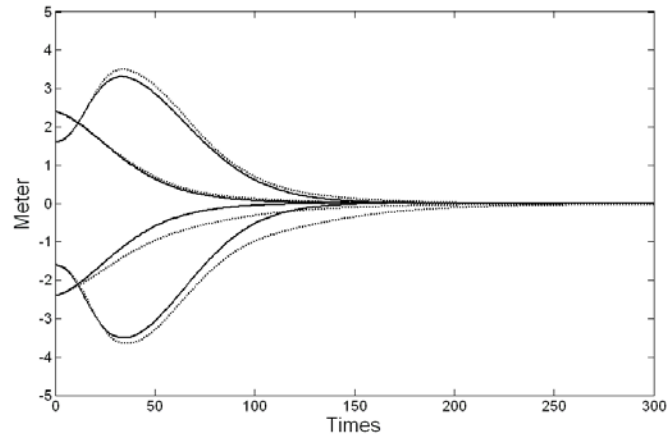


Figure 4.12: Responses of ball and beam system controlled by FLNFN-RSMODE controller (solid curves) and FLNFN-RSDE controller (dotted curves) under four initial conditions.

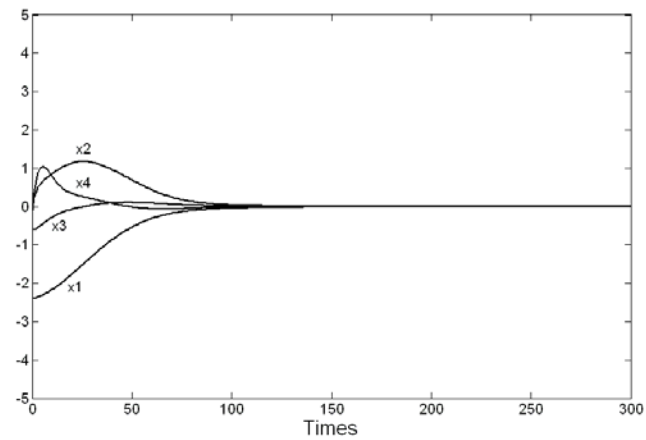


Figure 4.13: Responses of four states of the ball and beam system under the control of the FLNFN-RSMODE controller.

Table 4.3: Comparison of performance of various controllers to control of ball and beam system.

Method	FLNFN-RSMODE	FLNFN-RSDE	FLNFN-DE	FLNFN-GA
Fitness Value (Ave)	0.9041	0.8737	0.8516	0.8287
Fitness Value (Best)	0.9653	0.9447	0.9441	0.9131

### Example 3: Control of Backing Up the Truck

Backing a truck into a loading dock is difficult. It is a nonlinear control problem for

which no traditional control method exists [76]. Figure 4.14 shows the simulated truck and loading zone. The truck position is exactly determined by three state variables  $\phi$ ,  $x$  and  $y$ , where  $\phi$  is the angle between the truck and the horizontal, and the coordinate pair  $(x, y)$  specifies the position of the center of the rear of the truck in the plane. The steering angle  $\theta$  of the truck is the controlled variable. Positive values of  $\theta$  represent clockwise rotations of the steering wheel and negative values represent counterclockwise rotations. The truck is placed at some initial position and is backed up while being steered by the controller. The objective of this control problem is to use backward only motions of the truck to make the truck arrive in the desired loading dock  $(x_{desired}, y_{desired})$  at a right angle ( $\phi_{desired}=90^\circ$ ). The truck moves backward as the steering wheel moves through a fixed distance ( $d_f$ ) in each step. The loading region is limited to the plane  $[0,100] \times [0,100]$ .

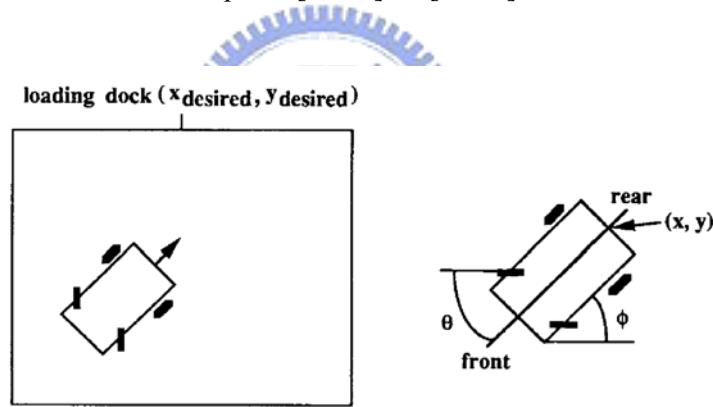


Figure 4.14: Diagram of simulated truck and loading zone.

The input and output variables of the FLNFN-RSMODE controller must be specified. The controller has two inputs, truck angle  $\phi$  and cross position  $x$ . When the clearance between the truck and the loading dock is assumed to be sufficient, the  $y$  coordinate is not considered as an input variable. The output of the controller is the steering angle  $\theta$ . The ranges of the variables  $x$ ,  $\phi$  and  $\theta$  are as follows.

$$0 \leq x \leq 100 \quad (4.13)$$

$$-90^\circ \leq \phi \leq 270^\circ \quad (4.14)$$

$$-30^\circ \leq \theta \leq 30^\circ \quad (4.15)$$

The equations of backward motion of the truck are,

$$\begin{aligned} x(k+1) &= x(k) + d_f \cos \theta(k) + \cos \phi(k) \\ y(k+1) &= y(k) + d_f \sin \theta(k) + \sin \phi(k) \\ \phi(k+1) &= \tan^{-1} \left[ \frac{l \sin \phi(k) + d_f \cos \phi(k) \sin \theta(k)}{l \cos \phi(k) - d_f \sin \phi(k) \sin \theta(k)} \right] \end{aligned} \quad (4.16)$$

where  $l$  is the length of the truck. Equation (4.16) yields the next state from the present state.

Learning involves several attempts, each starting from an initial state and terminating when the desired state is reached; the FLNFN is thus trained. In initialization phase, 7 subpopulations are generated. This example was simulated 30 times. The fitness value of the FLNFN-RSMODE is approximately 0.9746 and the learning curve of FLNFN-RSMODE is compared with those obtained using the FLNFN-RSDE, FLNFN-DE, and FLNFN-GA, as shown in Fig. 4.15. In Fig. 4.16, the solid curves are the training paths and the dotted curves are the paths that the truck runs under the control of the proposed controller. As this figure shown, the FLNFN-RSMODE controller can smooth the training paths. Figures 4.17(a)-(d) plot the trajectories of the moving truck controlled by the FLNFN-RSMODE controller, starting at initial positions  $(x, y, \phi) =$  (a)  $(40, 20, -30^\circ)$ , (b)  $(10, 20, 150^\circ)$ , (c)  $(70, 20, -30^\circ)$  and (d)  $(80, 20, 150^\circ)$ , after the training process has been terminated. The considered performance indices include the best fitness and the average fitness value. Table 4.4 compares the results. According to these results, the proposed FLNFN-RSMODE controller outperforms various existing methods.

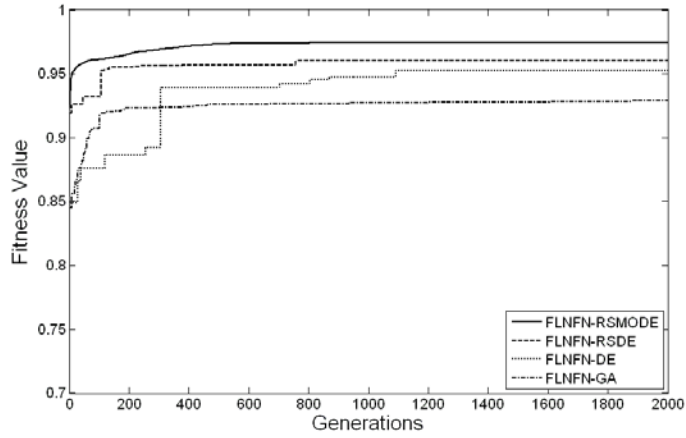


Figure 4.15: Learning curves of best performance of the FLNFN-RSMODE, FLNFN-RSDE, FLNFN-DE and FLNFN-GA in Example 3.

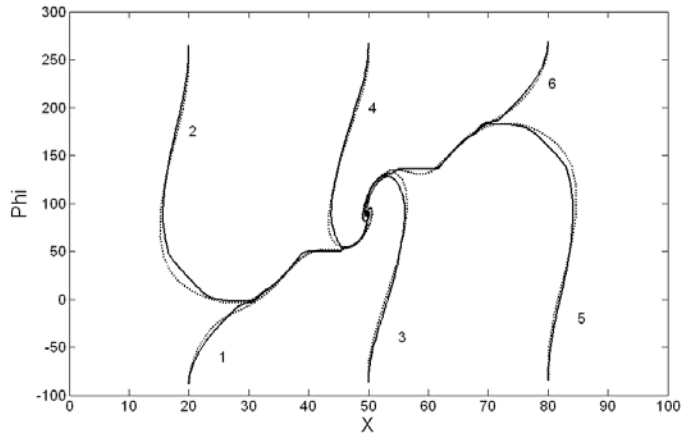
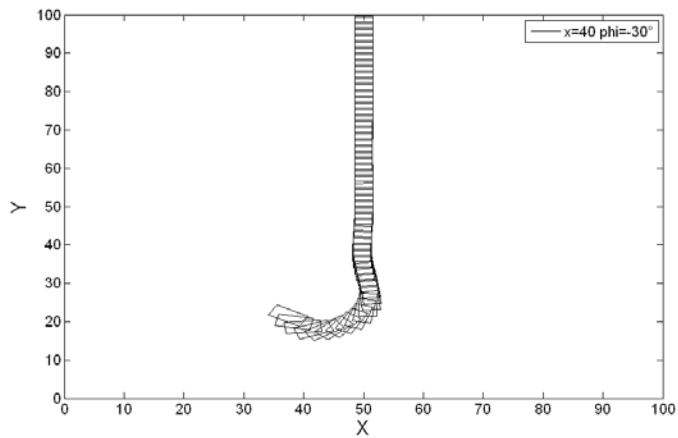
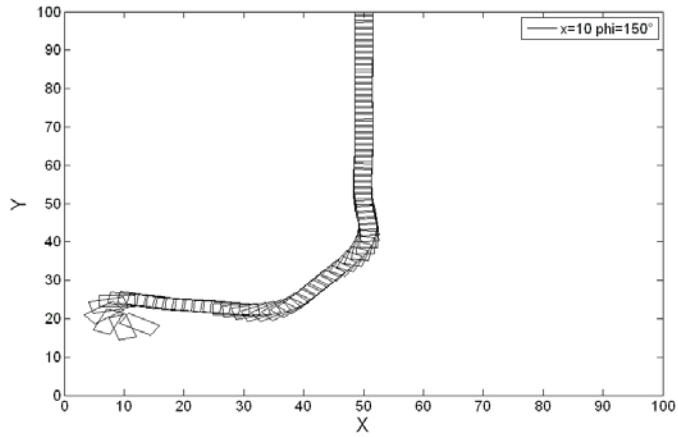


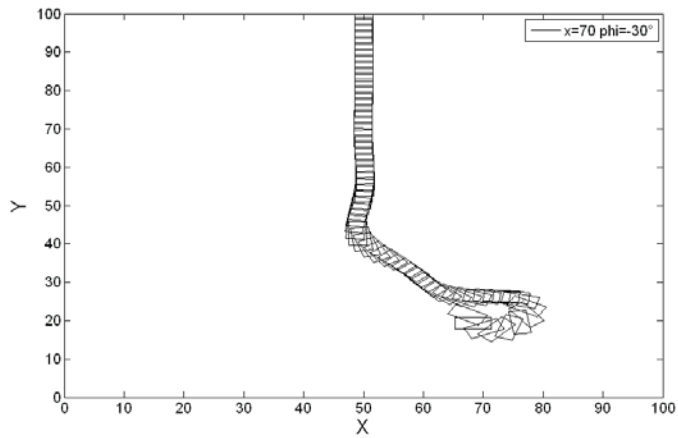
Figure 4.16: The moving trajectories of the truck where the solid curves represent the six sets of training trajectories and the dotted curves represent the moving trajectories of the truck under the FLNFN-RSMODE controller.



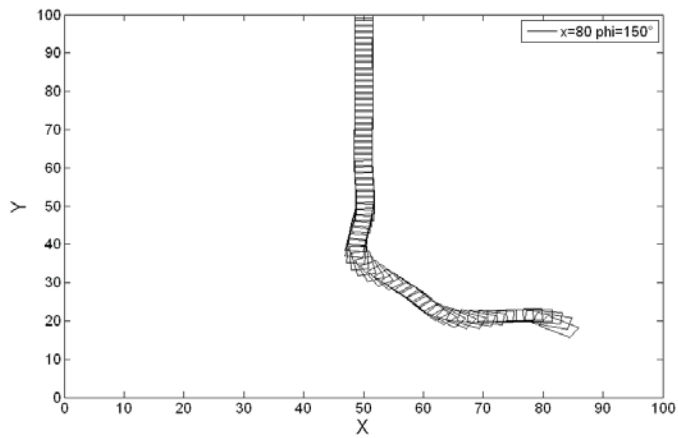
(a)



(b)



(c)



(d)

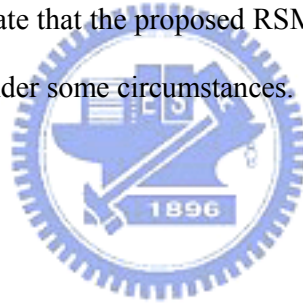
Figure 4.17: Trajectories of truck, starting at four initial positions under the control of the FLNFN-RSMODE after learning using training trajectories.

Table 4.4: Comparison of performance of various controllers to control of backing up the truck.

Method	FLNFN-RSMODE	FLNFN-RSDE	FLNFN-DE	FLNFN-GA
Fitness Value (Ave)	0.9110	0.8939	0.8846	0.8421
Fitness Value (Best)	0.9746	0.9604	0.9527	0.9286

#### 4.4 Summary

This dissertation proposes a rule-based symbiotic modified differential evolution (RSMODE) for the FLNFN model. The proposed RSMODE learning algorithm consists of initialization phase to generate initial rule-based subpopulation, and parameter learning phase to adjust the FLNFN parameters. The proposed RSMODE learning algorithm allows that each individual in each subpopulation evolves separately using a modified differential evolution. The experimental results demonstrate that the proposed RSMODE can obtain a better performance than other existing methods under some circumstances.





# Chapter 5

## Conclusion and Future Works

This dissertation proposes a functional-link-based neuro-fuzzy network (FLNFN) structure for nonlinear system control. The FLNFN model uses a functional link neural network to the consequent part of the fuzzy rules. The FLNFN model can automatically be constructed and the FLNFN parameters can be adjusted by performing online structure/parameter learning schemes concurrently. The advantages of the proposed FLNFN model are summarized below.

1) The consequent of the fuzzy rules of the proposed model is a nonlinear combination of input variables. This study uses the functional link neural network to the consequent part of the fuzzy rules. The local properties of the consequent part in the FLNFN model enable a nonlinear combination of input variables to be approximated more effectively.

2) The online learning algorithm can automatically construct the FLNFN model. No rules or memberships exist initially. They are created automatically as learning proceeds, as online incoming training data are received and as structure and parameter learning are performed.

3) The FLNFN model is proven to be a universal approximator by Stone–Weierstrass theorem and its convergence properties are proven by the Lyapunov theorem in the Appendix.

4) As demonstrated in section 2.3, the proposed FLNFN model is a more adaptive and effective controller than the other models.

Furthermore, in order to prevent the FLNFN parameters to quickly reach the local optima. This dissertation also proposes a rule-based symbiotic modified differential evolution (RSMODE) for the FLNFN model. The proposed RSMODE comprises multi-subpopulation that uses each individual represents a single fuzzy rule and each individual in each subpopulation evolves separately using a modified differential evolution. The advantages of the proposed RSMODE method are summarized as follows:

1) The RSMODE can automatically determine the number of subpopulation. No subpopulation exists initially. They are generated automatically using entropy measure which satisfies the fuzzy partition of input variables.

2) The RSMODE adopt a subpopulation symbiotic evolution strategy which uses the rule-based subpopulation to evolve separately.

3) The RSMODE adopt a modified differential evolution strategy to effectively search between the current best individual and randomly chosen individuals.

Moreover, the proposed FLNFN model and its related learning algorithms can obtain better simulation results than alternative methods in some circumstances, for example achieving higher design accuracy in many nonlinear control problems. We shall address the two issues of the FLNFN model and its related learning algorithms. First, we always require that training data be sufficient and proper. However, there is no procedure or rule suitable for all cases in choosing training data. One rule of thumb is that training data should cover the entire expected input space and then during the training process select training-vector pairs randomly from the set. Second, we believe that the proposed FLNFN-RSMODE and FLNFN-MODE are a more adaptive and effective controller than the FLNFN-BP for high-order nonlinear or overly complex systems.

Two advanced topics on the proposed FLNFN model should be addressed in future research. First, the FLNFN model will tend to apply high-order nonlinear or overly complex systems if it can suitably adopt the consequent part of a nonlinear combination of input

variables, and a functional expansion of multiple trigonometric polynomials. Therefore, it should be analyzed to use how many trigonometric polynomials for functional expansion in future. Second, there are some parameters in the RSMODE method influence the accuracy and complexity of the final FLNFN and training duration. These parameters should be automatically selected using an effective method in future. Khosla *et al.* [77] presented a systematic based on Taguchi approach reasoning scheme for identifying the strategy parameters for the evolutionary algorithm. The Taguchi approach provides systematic, simple and efficient methodology using fractional factorial design to study a large number of parameters with only a few well-defined experimental sets.



# Appendix

## A. Proof of the Universal Approximator Theorem

The Stone-Weierstrass theorem [56] is adopted to prove the universal approximator theorem. For a clear description in the FLNFN model, only the multi-input single-output (MISO) function  $f : x \in \mathfrak{R}^N \rightarrow y \in \mathfrak{R}$  is considered. The FLNFN is expressed as

$$y(x) = \frac{\sum_{j=1}^R \hat{y}_j u_j^{(3)}(x)}{\sum_{j=1}^R u_j^{(3)}(x)} \quad (\text{A.1})$$

*Theorem A1: Stone-Weierstrass Theorem:* Let  $A$  be a set of real continuous functions on a compact set  $U$ . If 1)  $U$  is an algebra that if  $f_1, f_2 \in A$ , and  $c \in \mathfrak{R}$ , then  $f_1 + f_2 \in A$ ,  $f_1 \cdot f_2 \in A$ , and  $cf_1 \in A$ ; 2)  $A$  separates points on  $U$ , meaning that for  $x, y \in U$ ,  $x \neq y$ , there exists  $f_1 \in A$  such that  $f_1(x) \neq f_1(y)$ , and 3)  $A$  vanishes at no point of  $U$ , meaning that for each  $x \in U$  there exists  $f_1 \in A$  such that  $f_1(x) \neq 0$ , then the uniform closure of  $A$  consists of all real continuous functions on  $U$ .

*Lemma A1:* Let  $Y$  be the family of function  $y : \mathfrak{R}^N \rightarrow \mathfrak{R}$  defined in Eq.(A.1); then  $Y \rightarrow U$ , where  $U$  is a compact set.

*Proof of Lemma A1:* Here, the membership function is

$$0 < \mu_{A_{ij}}(x) = \exp\left[-\frac{(x_i - m_{ij})^2}{\sigma_{ij}^2}\right] \leq 1.$$

Therefore, the continuous function  $u_j^{(3)}$  is closed and bounded for all  $x \in \mathfrak{R}^N$ . That is,  $Y \subset U$ .

*Proof of Theorem A1:* First, we prove that  $Y$  is algebra. Let  $f_1, f_2 \in Y$ , such that they can be written as

$$\begin{aligned}
f_1(x) &= \frac{\sum_{j_1=1}^{R1} \hat{y}1_{j_1} u1_{j_1}^{(3)}}{\sum_{j_1=1}^{R1} u1_{j_1}^{(3)}} \\
&= \frac{\sum_{j_1=1}^{R1} (w1_{1,j_1} \phi1_1 + \dots + w1_{M,j_1} \phi1_M) \cdot \left[ \prod_{i_1=1}^{N1} \exp \left[ -\frac{(x_{i_1} - m1_{i_1,j_1})^2}{\sigma1_{i_1,j_1}^2} \right] \right]}{\sum_{j_1=1}^{R1} \left[ \prod_{i_1=1}^{N1} \exp \left[ -\frac{(x_{i_1} - m1_{i_1,j_1})^2}{\sigma1_{i_1,j_1}^2} \right] \right]}
\end{aligned} \tag{A.2}$$

$$\begin{aligned}
f_2(x) &= \frac{\sum_{j_2=1}^{R2} \hat{y}2_{j_2} u2_{j_2}^{(3)}}{\sum_{j_2=1}^{R2} u2_{j_2}^{(3)}} \\
&= \frac{\sum_{j_2=1}^{R2} (w2_{1,j_2} \phi2_1 + \dots + w2_{M,j_2} \phi2_M) \cdot \left[ \prod_{i_2=1}^{N2} \exp \left[ -\frac{(x_{i_2} - m2_{i_2,j_2})^2}{\sigma2_{i_2,j_2}^2} \right] \right]}{\sum_{j_2=1}^{R2} \left[ \prod_{i_2=1}^{N2} \exp \left[ -\frac{(x_{i_2} - m2_{i_2,j_2})^2}{\sigma2_{i_2,j_2}^2} \right] \right]}
\end{aligned} \tag{A.3}$$

where  $\hat{y}1_j$  and  $\hat{y}2_j \in \mathfrak{R}$ ,  $\forall j$ .

Therefore,

$$f_1 + f_2(x) = \frac{\sum_{j_1=1}^{R1} \sum_{j_2=1}^{R2} (\hat{y}1_{j_1} + \hat{y}2_{j_2}) \cdot \left( \prod_{i=1}^N u1_{j_1}^{(3)} u2_{j_2}^{(3)} \right)}{\sum_{j_1=1}^{R1} \sum_{j_2=1}^{R2} \left( \prod_{i=1}^N u1_{j_1}^{(3)} u2_{j_2}^{(3)} \right)}. \tag{A.4}$$

Since  $u1_j^{(3)}$  and  $u2_j^{(3)}$  are Gaussian in form, i.e., this can be verified by straightforward algebraic operations; hence, Eq.(A.4) is in the same form as Eq.(A.1), so that  $f_1 + f_2 \in Y$ .

Similarly, we have

$$f_1 \cdot f_2(x) = \frac{\sum_{j_1=1}^{R1} \sum_{j_2=1}^{R2} (\hat{y}1_{j_1} \cdot \hat{y}2_{j_2}) \cdot \left( \prod_{i=1}^N u1_{j_1}^{(3)} u2_{j_2}^{(3)} \right)}{\sum_{j_1=1}^{R1} \sum_{j_2=1}^{R2} \left( \prod_{i=1}^N u1_{j_1}^{(3)} u2_{j_2}^{(3)} \right)} \tag{A.5}$$

which is also in the same form as Eq.(A.1); hence,  $f_1 \cdot f_2 \in Y$ , Finally, for arbitrary  $c \in \mathfrak{R}$

$$c \cdot f_1(x) = \frac{\sum_{j1=1}^{R1} (c \cdot \hat{y}1_{j1}) \cdot (\prod_{i=1}^N u1_{j1}^{(3)})}{\sum_{j1=1}^{R1} (\prod_{i=1}^N u1_{j1}^{(3)})} \quad (\text{A.6})$$

which is again in the form of Eq.(A.1); hence,  $c \cdot f_1 \in Y$ . Therefore,  $Y$  is an algebra.

Next,  $Y$  is proven to separate points on  $U$ , by constructing a required  $f$ ;  $f \in Y$  is specified such that  $f(x') \neq f(y')$  for arbitrarily given  $x', y' \in U$  with  $x' \neq y'$ . Two fuzzy rules in the form of Eq.(2.3) are chosen for the fuzzy rule base.

Let  $\underline{x}' = (x'_1, x'_2, \dots, x'_N)$  and  $\underline{y}' = (y'_1, y'_2, \dots, y'_N)$ . If  $x'_i \neq y'_i$ , then two fuzzy rules can be chosen as the fuzzy rule base. Furthermore, let the Gaussian membership functions be

$$\mu_{A_{i1}}(x_i) = \exp\left[-\frac{(x_i - x'_i)^2}{\sigma^2}\right] \quad (\text{A.7})$$

$$\mu_{A_{i2}}(x_i) = \exp\left[-\frac{(x_i - y'_i)^2}{\sigma^2}\right] \quad (\text{A.8})$$

Then,  $f$  can be expressed as

$$f = \frac{\hat{y}1 \left[ \prod_{i=1}^N \exp\left[-\frac{(x_i - x'_i)^2}{\sigma_{i1}^2}\right] \right] + \hat{y}2 \left[ \prod_{i=1}^N \exp\left[-\frac{(x_i - y'_i)^2}{\sigma_{i2}^2}\right] \right]}{\left[ \prod_{i=1}^N \exp\left[-\frac{(x_i - x'_i)^2}{\sigma_{i1}^2}\right] \right] + \left[ \prod_{i=1}^N \exp\left[-\frac{(x_i - y'_i)^2}{\sigma_{i2}^2}\right] \right]} \quad (\text{A.9})$$

where  $\hat{y}1$  and  $\hat{y}2$  are outputs of the local FLNN model calculated for output  $y$  and rule *Rule-1*, *Rule-2* and  $\hat{y}_j \in \mathfrak{R}$ ,  $\forall j$ . With this system,

$$f(\underline{x}') = \frac{\hat{y}1 + \hat{y}2 \left[ \prod_{i=1}^N \exp\left[-\frac{(x'_i - y'_i)^2}{\sigma_{i2}^2}\right] \right]}{1 + \left[ \prod_{i=1}^N \exp\left[-\frac{(x'_i - y'_i)^2}{\sigma_{i2}^2}\right] \right]} \quad (\text{A.10})$$

$$f(\underline{y}') = \frac{\hat{y}2 + \hat{y}1 \left[ \prod_{i=1}^N \exp \left[ -\frac{(y'_i - x'_i)^2}{\sigma_{i1}^2} \right] \right]}{1 + \left[ \prod_{i=1}^N \exp \left[ -\frac{(y'_i - x'_i)^2}{\sigma_{i1}^2} \right] \right]} \quad (\text{A.11})$$

Since  $\underline{x}' \neq \underline{y}'$ , some  $i$  must exist such that  $x'_i \neq y'_i$ ; hence  $f(\underline{x}') \neq f(\underline{y}')$ . Therefore,  $Y$  separates points on  $U$ .

Finally,  $Y$  is proven to vanish at no point of  $U$ . By Eq.(A.1),  $u_j^{(3)}(x)$  is constant and does not equal zero. That is, for all  $x \in \mathfrak{R}^N$ ,  $u_j^{(3)}(x) > 0$ . If  $u_j^{(3)}(x) > 0, (j = 1, 2, \dots, R)$ , then  $y > 0$  for any  $x \in \mathfrak{R}^N$ . That is, any  $y \in Y$  with  $u_j^{(3)}(x) > 0$  can serve as the required  $f$ .

In summary, the FLNFN model is a universal approximator, using the *Stone-Weierstrass theorem* and the fact that  $Y$  is a continuous real set on  $U$  proves the theorem.

## B. Proof of Convergence Theorem

*Theorem B1:* Let  $\eta_w$  be the learning rate parameter of the FLNFN weight, and let  $P_{w\max}$  be defined as  $P_{w\max} \equiv \max_k \|P_w(k)\|$ , where  $P_w(k) = \partial y / \partial w_{kj}$  and  $\|\cdot\|$  is the Euclidean norm in  $\mathfrak{R}^N$ . The convergence is guaranteed if  $\eta_w$  is chosen as  $\eta_w = \lambda / (P_{w\max})^2 = \lambda / R$ , in which  $\lambda$  is a positive constant gain, and  $R$  is the number of rules in the FLNFN model.

*Proof of Theorem B1:* Since

$$P_w(k) = \frac{\partial y}{\partial w_{kj}} = \frac{u_j^{(3)} \phi_k}{\sum_{j=1}^R u_j^{(3)}} \quad (\text{B.1})$$

and  $u_j^{(3)} \phi_k / \sum_{j=1}^R u_j^{(3)} \leq 1$ , the following result holds;

$$\|P_w(k)\| \leq \sqrt{R}. \quad (\text{B.2})$$

Then, a discrete Lyapunov function is selected as

$$V(k) = \frac{1}{2} e^2(k). \quad (\text{B.3})$$

The change in the Lyapunov function is obtained as

$$\begin{aligned} \Delta V(k) &= V(k+1) - V(k) \\ &= \frac{1}{2} [e^2(k+1) - e^2(k)] \end{aligned} \quad (\text{B.4})$$

The error difference can be represented as [23]

$$\begin{aligned} e(k+1) &= e(k) + \Delta e(k) \\ &= e(k) + \left[ \frac{\partial e(k)}{\partial w_{kj}} \right]^T \Delta w_{kj} \end{aligned} \quad (\text{B.5})$$

where  $\Delta e$  and  $\Delta w_k$  represent the output error change and the weight change in the output layer, respectively. Equations (2.17) and (B.5) yield

$$\frac{\partial e(k)}{\partial w_{kj}} = \frac{\partial e(k)}{\partial y} \frac{\partial y}{\partial w_{kj}} = P_w(k) \quad (\text{B.6})$$

$$e(k+1) = e(k) - P_w^T(k) \eta_w e(k) P_w(k). \quad (\text{B.7})$$

Then,

$$\begin{aligned} \|e(k+1)\| &= \|e(k) \cdot [1 - \eta_w P_w^T(k) P_w(k)]\| \\ &\leq \|e(k)\| \cdot \|1 - \eta_w P_w^T(k) P_w(k)\| \end{aligned} \quad (\text{B.8})$$

is true. If  $\eta_w = \lambda / (P_{w\max}^2) = \lambda / R$  is chosen, then the term  $\|1 - \eta_w P_w^T(k) P_w(k)\|$  in Eq.(B.8) is less than 1. Therefore, the Lyapunov stability of  $V > 0$  and  $\Delta V < 0$  is guaranteed. The output error between the reference model and actual plant converges to zero as  $t \rightarrow \infty$ . This fact completes the proof of the theorem.

The following lemmas [25] are used to prove Theorem 2.

*Lemma B1:* Let  $g(h) = h \exp(-h^2)$ , then  $|g(h)| < 1, \forall h \in \mathfrak{R}$ .

*Lemma B2:* Let  $f(h) = h^2 \exp(-h^2)$ , then  $|f(h)| < 1, \forall h \in \mathfrak{R}$ .

*Theorem B2:* Let  $\eta_m$  and  $\eta_\sigma$  be the learning rate parameters of the mean and standard



deviation of the Gaussian function for the FLNFN; let  $P_{m\max}$  be defined as  $P_{m\max} \equiv \max_k \|P_m(k)\|$ , where  $P_m(k) = \partial y / \partial m_{ij}$ ; let  $P_{\sigma\max}$  be defined as  $P_{\sigma\max} \equiv \max_k \|P_\sigma(k)\|$ , where  $P_\sigma(k) = \partial y / \partial \sigma_{ij}$ . The convergence is guaranteed if  $\eta_m$  and  $\eta_\sigma$  are chosen as  $\eta_m = \eta_\sigma = [\eta_w / M] \cdot \left[ |w_{kj}|_{\max} \left( 2 / |\sigma_{ij}|_{\min} \right) \right]^2$ , in which  $|w_{kj}|_{\max} = \max_k |w_{kj}(k)|$ ;  $|\sigma_{ij}|_{\min} = \min_k |\sigma_{ij}(k)|$ ;  $|\cdot|$  is the absolute value.

*Proof of Theorem B2:* According to Lemma B1,  $\left[ (x_i - m_{ij}) / \sigma_{ij} \right] \exp \left\{ - \left[ (x_i - m_{ij}) / \sigma_{ij} \right]^2 \right\} < 1$ . The upper bounds on  $P_m(k)$  can be derived as follows;

$$\begin{aligned}
P_m(k) &= \frac{\partial y}{\partial m_{ij}} \\
&= \left( \frac{\partial y}{\partial u_j^{(4)}} \right) \left( \frac{\partial u_j^{(4)}}{\partial u_j^{(3)}} \right) \left( \frac{\partial u_j^{(3)}}{\partial u_{ij}^{(2)}} \right) \left( \frac{\partial u_{ij}^{(2)}}{\partial m_{ij}} \right) \\
&< \left| \sum_{j=1}^R \sum_{k=1}^M w_{kj} \phi_k \right| \left| \left( \frac{2}{\sigma_{ij}} \right) \left( \frac{x_i - m_{ij}}{\sigma_{ij}} \right) \exp \left[ - \left( \frac{x_i - m_{ij}}{\sigma_{ij}} \right)^2 \right] \right| \\
&< \left| \sum_{j=1}^R \sum_{k=1}^M w_{kj} \phi_k \right| \left| \frac{2}{\sigma_{ij}} \right| \\
&< \sqrt{RM} |w_{kj}|_{\max} \left( \frac{2}{|\sigma_{ij}|_{\min}} \right)
\end{aligned} \tag{B.9}$$

where  $\phi_k \in [0,1]$ , for  $k = 1, 2, \dots, M$ . Thus,

$$\|P_m(k)\| < \sqrt{RM} |w_{kj}|_{\max} \left( \frac{2}{|\sigma_{ij}|_{\min}} \right). \tag{B.10}$$

The error difference can also be represented as [23]

$$\begin{aligned}
e(k+1) &= e(k) + \Delta e(k) \\
&= e(k) + \left[ \frac{\partial e(k)}{\partial m_{ij}} \right]^T \Delta m_{ij}
\end{aligned} \tag{B.11}$$

where  $\Delta m_{ij}$  represents the change of the mean of the Gaussian function in the membership

function layer. Equation (2.18) and (B.11) yield

$$\frac{\partial e(k)}{\partial m_{ij}} = \frac{\partial e(k)}{\partial y} \frac{\partial y}{\partial m_{ij}} = P_m(k) \quad (\text{B.12})$$

$$e(k+1) = e(k) - P_m^T(k) \eta_m e(k) P_m(k). \quad (\text{B.13})$$

Then,

$$\begin{aligned} \|e(k+1)\| &= \|e(k) \cdot [1 - \eta_m P_m^T(k) P_m(k)]\| \\ &\leq \|e(k)\| \cdot \|1 - \eta_m P_m^T(k) P_m(k)\| \end{aligned} \quad (\text{B.14})$$

is true. If  $\eta_m = \lambda / (P_{m \max})^2 = [\eta_w / M] \cdot [w_{kj}|_{\max} (2/|\sigma_{ij}|_{\min})]^2$  is chosen, then the term  $\|1 - \eta_m P_m^T(k) P_m(k)\|$  in Eq.(B.14) is less than 1. Therefore, the Lyapunov stability of  $V > 0$  and  $\Delta V < 0$  given by Eq.(B.3) and Eq.(B.4), is guaranteed. The output error between the reference model and actual plant converges to zero as  $t \rightarrow \infty$ .

According to Lemma B2,  $\left\{ \left[ (x_i - m_{ij}) / \sigma_{ij} \right]^2 \exp \left\{ - \left[ (x_i - m_{ij}) / \sigma_{ij} \right]^2 \right\} \right\} < 1$ . The upper bounds on  $P_\sigma(k)$  can be derived as follows;

$$\begin{aligned} P_\sigma(k) &= \frac{\partial y}{\partial \sigma_{ij}} \\ &= \left( \frac{\partial y}{\partial u_j^{(4)}} \right) \left( \frac{\partial u_j^{(4)}}{\partial u_j^{(3)}} \right) \left( \frac{\partial u_j^{(3)}}{\partial u_j^{(2)}} \right) \left( \frac{\partial u_j^{(2)}}{\partial \sigma_{ij}} \right) \\ &< \left| \sum_{j=1}^R \sum_{k=1}^M w_{kj} \phi_k \right| \left| \left( \frac{2}{\sigma_{ij}} \right) \left( \frac{x_i - m_{ij}}{\sigma_{ij}} \right)^2 \exp \left[ - \left( \frac{x_i - m_{ij}}{\sigma_{ij}} \right)^2 \right] \right| \\ &< \left| \sum_{j=1}^R \sum_{k=1}^M w_{kj} \phi_k \right| \left| \frac{2}{\sigma_{ij}} \right| \\ &< \sqrt{RM} |w_{kj}|_{\max} \left( \frac{2}{|\sigma_{ij}|_{\min}} \right) \end{aligned} \quad (\text{B.15})$$

where  $\phi_k \in [0,1]$ , for  $k = 1, 2, \dots, M$ . Thus,

$$\|P_\sigma(k)\| < \sqrt{RM} |w_{kj}|_{\max} \left( \frac{2}{|\sigma_{ij}|_{\min}} \right). \quad (\text{B.16})$$

The error difference can be represented as

$$\begin{aligned}
e(k+1) &= e(k) + \Delta e(k) \\
&= e(k) + \left[ \frac{\partial e(k)}{\partial \sigma_{ij}} \right]^T \Delta \sigma_{ij}
\end{aligned} \tag{B.17}$$

where  $\Delta \sigma_{ij}$  represents the change of the variance of the Gaussian function in the membership function layer. Equation (2.19) and (B.17) yield

$$\frac{\partial e(k)}{\partial \sigma_{ij}} = \frac{\partial e(k)}{\partial y} \frac{\partial y}{\partial \sigma_{ij}} = P_{\sigma}(k) \tag{B.18}$$

$$e(k+1) = e(k) - P_{\sigma}^T(k) \eta_{\sigma} e(k) P_{\sigma}(k). \tag{B.19}$$

Then,

$$\begin{aligned}
\|e(k+1)\| &= \|e(k) \cdot [1 - \eta_{\sigma} P_{\sigma}^T(k) P_{\sigma}(k)]\| \\
&\leq \|e(k)\| \cdot \|1 - \eta_{\sigma} P_{\sigma}^T(k) P_{\sigma}(k)\|
\end{aligned} \tag{B.20}$$

is true. If  $\eta_{\sigma} = \lambda / (P_{\sigma \max})^2 = [\eta_w / M] \cdot [w_{kj}]_{\max} (2 / |\sigma_{ij}|_{\min})^2$  is chosen, then the term  $\|1 - \eta_{\sigma} P_{\sigma}^T(k) P_{\sigma}(k)\|$  in Eq. (B.20) is less than 1. Therefore, the Lyapunov stability of  $V > 0$  and  $\Delta V < 0$  given by Eq. (B.3) and Eq. (B.4) is guaranteed. The output error between the reference model and actual plant converges to zero as  $t \rightarrow \infty$ . This fact completes the proof of the theorem.

# Bibliography

- [1] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System*, NJ: Prentice-Hall, 1996.
- [2] S. Mitra and Y. Hayashi, "Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework," *IEEE Trans. Neural Networks*, Vol. 11, No. 3, pp. 748-768, May 2000.
- [3] F. Sun, Z. Sun, L. Li, and H. X. Li, "Neuro-Fuzzy Adaptive Control Based on Dynamic Inversion for Robotic Manipulators," *Fuzzy Sets and Systems*, Vol. 134, No. 1, pp. 117-133, Feb. 2003.
- [4] L. X. Wang and J. M. Mendel, "Generating Fuzzy Rules by Learning from Examples," *IEEE Trans. on Syst., Man, and Cybern.*, Vol. 22, No. 6, pp. 1414-1427, Nov/Dec. 1992.
- [5] C. J. Lin and C. T. Lin, "An ART-Based Fuzzy Adaptive Learning Control Network," *IEEE Trans. Fuzzy Systems*, Vol. 5, No. 4, pp. 477-496, Nov. 1997.
- [6] W. S. Lin, C. H. Tsai, and J. S. Liu, "Robust Neuro-Fuzzy Control of Multivariable Systems by Tuning Consequent Membership Functions," *Fuzzy Sets and Systems*, Vol. 124, No. 2, pp. 181-195, Dec. 2001.
- [7] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modeling and Control," *IEEE Trans. on Syst., Man, Cybern.*, Vol. 15, pp. 116-132, 1985.
- [8] C. Li and C. Y. Lee, "Self-Organizing Neuro-Fuzzy System for Control of Unknown Plants," *IEEE Trans. Fuzzy Systems*, Vol. 11, No. 1, pp. 135-150, Feb. 2003.
- [9] J.-S. R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System," *IEEE Trans. on Syst., Man, and Cybern.*, Vol. 23, pp. 665-685, 1993.
- [10] C. F. Juang and C. T. Lin, "An On-Line Self-Constructing Neural Fuzzy Inference Network and Its Applications," *IEEE Trans. Fuzzy Systems*, Vol. 6, No.1, pp. 12-31, Feb. 1998.

- [11] S. Halgamuge and M. Glesner, "Neural Networks in Designing Fuzzy Systems for Real World Applications," *Fuzzy Sets and Systems*, Vol. 65, pp. 1-12, 1994.
- [12] N. Kasabov, "Learning Fuzzy Rules and Approximate Reasoning in Fuzzy Neural Networks and Hybrid Systems," *Fuzzy Sets and Systems*, Vol. 82, pp. 135-149, 1996.
- [13] D. Nauck and R. Kruse, "A Neuro-Fuzzy Method to Learn Fuzzy Classification Rules From Data," *Fuzzy Sets and Systems*, Vol. 89, pp. 277-288, 1997.
- [14] S. Paul and S. Kumar, "Subsethood-Product Fuzzy Neural Inference System (SuPFuNIS)," *IEEE Trans. Neural Networks*, Vol. 13, No. 3, pp. 578-599, 2002.
- [15] J. S. Wang and C. S. G. Lee, "Self-Adaptive Neuro-Fuzzy Inference Systems for Classification Applications," *IEEE Trans. Fuzzy Systems*, Vol. 10, No. 6, pp. 790-802, 2002.
- [16] C. J. Lin and C. T. Lin, "Reinforcement Learning for an ART-Based Fuzzy Adaptive Learning Control Network," *IEEE Trans. Neural Networks*, Vol. 7, No. 3, pp. 709-731, June, 1996.
- [17] F. J. Lin, C. H. Lin, and P. H. Shen, "Self-Constructing Fuzzy Neural Network Speed Controller for Permanent-Magnet Synchronous Motor Drive," *IEEE Trans. Fuzzy Systems*, Vol. 9, No. 5, pp. 751-759, Oct. 2001.
- [18] C. T. Chao, T. J. Chen, and C. C. Teng, "Simplification of Fuzzy-Neural Systems Using Similarity Analysis," *IEEE Trans. Syst., Man, Cybern.*, Vol. 26, No. 2, pp. 344-354, 1996.
- [19] C. J. Lin and C. H. Chen, "Identification and Prediction Using Recurrent Compensatory Neuro-Fuzzy Systems," *Fuzzy Sets and Systems*, Vol. 150, pp. 307-330, 2005.
- [20] C. H. Chen, C. J. Lin, and C. T. Lin, "A Functional-Link-Based Neuro-Fuzzy Network for Nonlinear System Control," accepted to appear in *IEEE Trans. on Fuzzy Systems*, 2008.
- [21] H. Takagi, N. Suzuki, T. Koda, and Y. Kojima, "Neural Networks Designed on

- Approximate Reasoning Architecture and Their Application,” *IEEE Trans. Neural Networks*, Vol. 3, pp. 752-759, Sept. 1992.
- [22] E. Mizutani and J.-S. R. Jang, “Coactive Neural Fuzzy Modeling,” in *Proc. Int. Conf. Neural Networks*, pp. 760-765, 1995.
- [23] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*, MA: Addison-Wesley, 1989.
- [24] J. C. Patra and R. N. Pal, “A Functional Link Artificial Neural Network for Adaptive Channel Equalization,” *Signal Process.*, Vol. 43, pp. 181-195, May 1995.
- [25] J. C. Patra, R. N. Pal, B. N. Chatterji, and G. Panda, “Identification of Nonlinear Dynamic Systems Using Functional Link Artificial Neural Networks,” *IEEE Trans. on Syst., Man, and Cybern.*, Vol. 29, Apr. 1999.
- [26] Y. H. Pao, S. M. Phillips, and D. J. Sobajic, “Neural-Net Computing and Intelligent Control Systems,” *International Journal of Control*, Vol. 56, No. 2, pp. 263-289, 1992.
- [27] E. Sanchez, T. Shibata, L. A. Zadeh, *Genetic Algorithms and Fuzzy Logic Systems: Soft Computing Perspectives*, Singapore: World Scientific, 1997.
- [28] O. Cordon, F. Herrera, F. Hoffmann, L. Magdalena, *Genetic Fuzzy Systems-Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*, Singapore: World Scientific, 2001.
- [29] P. P. Angelov, *Evolving Rule-Based Models: A Tool for Design of Flexible Adaptive Systems*, Physica-Verlag, Heidelberg, 2002.
- [30] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena, “Ten Years of Genetic Fuzzy Systems: Current Framework and New Trends,” *Fuzzy Sets and Systems*, Vol. 141, No. 1, pp. 5-31, 2004.
- [31] F. Herrera, “Genetic Fuzzy Systems: Status, Critical Considerations and Future Directions,” *International Journal of Computational Intelligence Research*, Vol. 1, No. 1, pp. 59-67, 2005.
- [32] B. Carse, T. C. Fogarty, and A. Munro, “Evolving Fuzzy Rule Based Controllers Using

- Genetic Algorithms,” *Fuzzy Sets Syst.*, Vol. 80, pp. 273-293, June 1996.
- [33] F. Herrera, M. Lozano, and J. L. Verdegay, “Tuning Fuzzy Logic Controllers by Genetic Algorithms,” *Int. J. Approx. Reas.*, Vol. 12, pp. 299-315, Apr./May 1995.
- [34] A. Homaifar and E. McCormick, “Simultaneous Design of Membership Functions and Rule Sets for Fuzzy Controllers Using Genetic Algorithms,” *IEEE Trans. Fuzzy Syst.*, Vol. 3, pp. 129-139, May 1995.
- [35] J. Velasco, “Genetic-Based On-Line Learning for Fuzzy Process Control,” *Int. J. Intell. Syst.*, Vol. 13, pp. 891-903, 1998.
- [36] H. Ishibuchi, T. Nakashima, and T. Murata, “Performance Evaluation of Fuzzy Classifier Systems for Multidimensional Pattern Classification Problems,” *IEEE Trans. Syst., Man, Cybern.-Part B: Cybern.*, Vol. 29, pp. 601-608, 1999.
- [37] O. Cordon, M. J. del Jesus, F. Herrera, and M. Lozano, “MOGUL: A Methodology to Obtain Genetic Fuzzy Rule-Based Systems under the Iterative Rule Learning Approach,” *Int. J. Intell. Syst.*, Vol. 14, pp. 1123-1153, 1999.
- [38] A. Gonzalez and R. Perez, “SLAVE: A Genetic Learning System Based on an Iterative Approach,” *IEEE Trans. Fuzzy Syst.*, Vol. 27, pp. 176-91, Apr. 1999.
- [39] M. Russo, “FuGeNeSys: A Fuzzy Genetic Neural System for Fuzzy Modeling,” *IEEE Trans. Fuzzy Systems*, Vol. 6, pp. 373-388, 1998.
- [40] I. F. Chung, C. J. Lin, and C. T. Lin, “A GA-Based Fuzzy Adaptive Learning Control Network,” *Fuzzy Sets and Systems*, Vol. 112, No. 1, pp. 65-84, 2000.
- [41] G. Alpaydin, G. Dandar, and S. Balkir, “Evolution-Based Design of Neural Fuzzy Networks Using Self-Adapting Genetic Parameters,” *IEEE Trans. Fuzzy Systems*, Vol. 10, No. 2, pp. 211-221, 2002.
- [42] S. H. Stewart, S. Taylor, J. M. Baker, F. Hoffmann, and G. Pfister, “Evolutionary Design of a Fuzzy Knowledge Base for a Mobile Robot,” *International Journal of Approximate Reasoning*, Vol. 17, No. 4, pp. 447-469, Nov. 1997.

- [43] A. Parodi and P. Bonelli, "A New Approach to Fuzzy Classifier Systems," *Proc. of 5th Int. Conf. Genetic Algorithms*, pp. 223-230, 1993.
- [44] L. Castillo, A. Gonzalez, and R. Perez, "Including a Simplicity Criterion in the Selection of the Best Rule in a Genetic Fuzzy Learning Algorithm," *Fuzzy Sets and Systems*, Vol. 120, No. 2, pp. 309-321, 2001.
- [45] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting Fuzzy If-Then Rules for Classification Problems Using Genetic Algorithms," *IEEE Trans. Fuzzy Systems*, Vol. 3, No. 3, pp. 260-270, 1995.
- [46] H. Ishibuchi, T. Murata, and I. B. Turksen, "Single-Objective and Two-Objective Genetic Algorithms for Selecting Linguistic Rules for Pattern Classification Problems," *Fuzzy Sets and Systems*, Vol. 89, No. 2, pp. 135-150, 1997.
- [47] H. Ishibuchi, T. Nakashima, and T. Murata, "Three-Objective Genetics-Based Machine Learning for Linguistic Rule Extraction," *Information Sciences*, Vol. 136, No. 1-4, pp. 109-133, 2001.
- [48] H. Ishibuchi and Y. Nojima, "Analysis of Interpretability-Accuracy Tradeoff of Fuzzy Systems by Multiobjective Fuzzy Genetics-Based Machine Learning," *International Journal of Approximate Reasoning*, Vol. 44, No. 1, pp. 4-31, 2007.
- [49] S. Mitra and S. K. Pal, "Fuzzy Multi-Layer Perceptron, Inferencing and Rule Generation," *IEEE Trans. Neural Networks*, Vol. 6, No. 1, pp. 51-63, 1995.
- [50] S. Mitra and S. K. Pal, "Fuzzy Self-Organization, Inferencing, and Rule Generation," *IEEE Trans. Syst., Man, Cybern. A*, Vol. 26, No. 5, pp. 608-620, 1996.
- [51] S. Mitra, K. M. Konwar, and S. K. Pal, "Fuzzy Decision Tree, Linguistic Rules and Fuzzy Knowledge-Based Network: Generation and Evaluation," *IEEE Trans. Syst., Man, Cybern. C*, Vol. 32, No. 4, 2002.
- [52] S. K. Pal, S. Mitra, and P. Mitra, "Rough-Fuzzy MLP: Modular Evolution, Rule Generation, and Evaluation," *IEEE Trans. Knowledge and Data Engineering*, Vol. 15, No.



- 1, pp. 328-339, 2003.
- [53] R. Storn and K. V. Price, "Differential Evolution-A Simple and Efficient Heuristic for Global Optimization Over Continuous spaces," *J. Global Opt.*, Vol. 11, No. 4, pp. 341-359, Dec. 1997.
- [54] R. Storn, "System Design by Constraint Adaptation and Differential Evolution," *IEEE Trans. Evolutionary Computation*, Vol. 3, No. 1, pp. 22-34, Apr. 1999.
- [55] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Germany: Springer-Verlag, 2005.
- [56] W. Rudin, *Principles of Mathematical Analysis*, 3rd ed., McGraw-Hill, NewYork, 1976.
- [57] L. X. Wang and J. M. Mendel, "Fuzzy Adaptive Filters, with Application to Nonlinear Channel Equalization," *IEEE Trans. Fuzzy Systems*, Vol. 1, No. 3, pp. 161-170, Aug. 1993.
- [58] C. C. Ku and K. Y. Lee, "Diagonal Recurrent Neural Networks for Dynamic Systems Control," *IEEE Trans. Neural Networks*, Vol. 6, pp. 144-156, Jan. 1995.
- [59] Y. C. Chen and C. C. Teng, "A Model Reference Control Structure Using a Fuzzy Neural Network," *Fuzzy Sets and Systems*, Vol. 73, pp. 291-312, 1995.
- [60] J. Tanomaru and S. Omatu, "Process Control by On-Line Trained Neural Controllers," *IEEE Trans. on Ind. Electron.*, Vol. 39, pp. 511-521, 1992.
- [61] J. Hauser, S. Sastry, and P. Kokotovic, "Nonlinear Control via Approximate Input-Output Linearization: the Ball and Beam Example," *IEEE Transactions on Automatic Control*, Vol. 37, pp. 392-398, 1992.
- [62] K. S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. Neural Networks*, Vol. 1, No. 1, pp. 4-27, Mar. 1990.
- [63] D. Psaltis, A. Sideris, and A. Yamamura, "A Multilayered Neural Network Controller," *IEEE Contr. Syst.*, Vol. 8, pp. 17-21, 1988.

- [64] C. L. Phillips and H. T. Nagle, *Digital Control System Analysis and Design*, Prentice Hall, 1995.
- [65] K. J. Astrom and B. Wittenmark, *Adaptive Control*, MA: Addison-Wesley, 1989.
- [66] C. H. Chen, C. J. Lin, and C. T. Lin, "An Efficient Quantum Neuro-Fuzzy Classifier Based on Fuzzy Entropy and Compensatory Operation," *Soft Computing*, Vol. 12, No. 6, pp. 567-583, Apr. 2008.
- [67] Cho Chieh Tech. Enterprise Ltd. <http://www.chochieh.com.tw/>
- [68] Feedback Instruments Limited <http://www.fb.com/>
- [69] A. E. Douglas, *Symbiotic Interactions*, Oxford University Press, Oxford, 1994.
- [70] R. E. Smith, S. Forrest and A. S. Perelson, "Searching for Diverse, Cooperative Populations with Genetic Algorithms," *Evolutionary Computation*, Vol. 1, No. 2, pp. 127-149, 1993.
- [71] D. E. Moriarty and R. Miikkulainen, "Efficient Reinforcement Learning through Symbiotic Evolution," *Mach. Learn.*, Vol. 22, pp. 11-32, 1996.
- [72] C. F. Juang, J. Y. Lin and C. T. Lin, "Genetic Reinforcement Learning through Symbiotic Evolution for Fuzzy Controller Design," *IEEE Trans. Syst., Man, Cybern. B*, Vol. 30, No. 2, Apr. 2000.
- [73] M. Jamei, M. Mahfouf, and D. A. Linkens, "Elicitation and Fine-Tuning of Fuzzy Control Rules Using Symbiotic Evolution," *Fuzzy Sets and Systems*, Vol. 147, No. 1, pp. 57-74, Oct. 2004.
- [74] C. H. Chen, C. J. Lin, and C. T. Lin, "Using an Efficient Immune Symbiotic Evolution Learning for Compensatory Neuro-Fuzzy Controller," accepted to appear in *IEEE Trans. on Fuzzy Systems*, 2008.
- [75] B. Niu, Y. Zhu, X. He, and H. Wu, "MCPSO: A Multi-Swarm Cooperative Particle Swarm Optimizer," *Applied Mathematics and Computation*, Vol. 185, pp. 1050-1062, 2007.

- [76] D. Nguyen and B. Widrow, "The Truck Backer-Upper: An Example of Self-Learning in Neural Network," *IEEE Conf. Syst. Mag.*, Vol. 10, No. 3, pp. 18-23, 1990.
- [77] A. Khosla, S. Kumar, and K. K. Aggarwal, "Identification of Strategy Parameters for Particle Swarm Optimizer through Taguchi Method," *Journal of Zhejiang University: Science*, Vol. 7, No. 12, pp. 1989-1994, 2006.



# Vita

## 博士候選人學經歷資料

姓名：陳政宏 (Cheng-Hung Chen)

性別：男

生日：民國 68 年 2 月 23 日

出生地：高雄市

論文題目：

中文：以函數鏈結為基礎之類神經模糊網路及其應用

英文：A Functional-Link-Based Neuro-Fuzzy Network and Its Applications

學歷：

- 民國 91 年 6 月，朝陽科技大學資訊工程系畢業
- 民國 93 年 6 月，朝陽科技大學資訊工程系碩士班畢業
- 民國 97 年 7 月，國立交通大學電機與控制工程學系博士班，提博士論

文口試

# Publication List

## 著作目錄

姓名：陳政宏(Cheng-Hung Chen)

已刊登或被接受之期刊論文：

- [1] **Cheng-Hung Chen**, Cheng-Jian Lin, and Chin-Teng Lin, “A Functional-Link-Based Neuro-Fuzzy Network for Nonlinear System Control,” accepted to appear in *IEEE Trans. on Fuzzy Systems*, 2008. (2.8 點)
- [2] Cheng-Jian Lin, **Cheng-Hung Chen**, and Chin-Teng Lin, “A Hybrid of Cooperative Particle Swarm Optimization and Cultural Algorithm for Neural Fuzzy Networks and Its Prediction Applications,” accepted to appear in *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 2008. (2 點)
- [3] **Cheng-Hung Chen**, Cheng-Jian Lin, and Chin-Teng Lin, “Using an Efficient Immune Symbiotic Evolution Learning for Compensatory Neuro-Fuzzy Controller,” accepted to appear in *IEEE Trans. on Fuzzy Systems*, 2008. (2.8 點)
- [4] **Cheng-Hung Chen**, Cheng-Jian Lin, and Chin-Teng Lin, “An Efficient Quantum Neuro-Fuzzy Classifier Based on Fuzzy Entropy and Compensatory Operation,” *Soft Computing*, Vol. 12, No. 6, pp. 567-583, Apr. 2008. (1.4 點)

研討會論文：

- [1] **Cheng-Hung Chen**, Chin-Teng Lin, and Cheng-Jian Lin, “A Novel Recurrent Neuro-Fuzzy System and Its Applications,” *Cross-Strait Workshop on Controls*, Taipei, Taiwan, R.O.C., pp. 69-74, Nov. 22-26, 2007.
- [2] **Cheng-Hung Chen**, Chin-Teng Lin, and Cheng-Jian Lin, “A Functional-Link-Based Fuzzy Neural Network for Temperature Control,” *The First IEEE Symposium on Foundations of Computational Intelligence*, Honolulu, Hawaii, USA, pp. 53-58, Apr. 1-5, 2007.
- [3] **Cheng-Hung Chen**, Cheng-Jian Lin, and Chin-Teng Lin, “A Recurrent Functional-Link-Based Neural Fuzzy System and Its Applications,” *The First IEEE Symposium on Computational Intelligence in Image and Signal Processing*, Honolulu, Hawaii, USA, pp. 415-420, Apr. 1-5, 2007.
- [4] **Cheng-Hung Chen**, Cheng-Jian Lin, and Chin-Teng Lin, “A Self-Constructing Compensatory Neural Fuzzy System for Nonlinear System Control,” *The 14th National Conference on Fuzzy Theory and Its Applications*, Kaohsiung, Taiwan, R.O.C., Dec. 14-15, 2006.
- [5] **Cheng-Hung Chen**, Chin-Teng Lin, and Cheng-Jian Lin, “Identification and Prediction

- Using Recurrent Compensatory Neuro-Fuzzy Systems,” *The 14th National Conference on Fuzzy Theory and Its Applications*, Kaohsiung, Taiwan, R.O.C., Dec. 14-15, 2006.
- [6] **Cheng-Hung Chen**, Chin-Teng Lin, and Cheng-Jian Lin, “A Novel Neuro-Fuzzy Inference System for Skin Color Detection,” *The 19th IPPR Conference on Computer Vision, Graphics and Image Processing*, Taoyuan, Taiwan, R.O.C., Aug. 13-15, 2006.
- [7] Chin-Teng Lin and **Cheng-Hung Chen**, “An Entropy-Based Neuro-Fuzzy Inference System for Classification Applications,” *The First Taiwan Software Engineering Conference*, Taipei, Taiwan, R.O.C., pp. 189-194, June 3-4, 2005.
- [8] **Cheng-Hung Chen** and Chin-Teng Lin, 2005, “Identification of Chaotic System Using Recurrent Compensatory Neuro-Fuzzy Systems,” *IEEE Int’l Workshop on Cellular Neural Networks and their Applications*, Hsinchu, Taiwan, R.O.C., pp. 15-18, May 28-30, 2005.

