

การควบคุมแบบโมเดลพรีดิกทีฟร่วมกับข่ายงานนิวรัลสำหรับกระบวนการกำจัดสนิมเหล็ก



นางสาวชिरา ดาวสุด

สถาบันวิทยบริการ

จุฬาลงกรณ์มหาวิทยาลัย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต

สาขาวิชาวิศวกรรมเคมี ภาควิชาวิศวกรรมเคมี

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2549

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

NEURAL NETWORK BASED MODEL PREDICTIVE CONTROL FOR A STEEL PICKLING PROCESS



Miss Wachira Daosud

สถาบันวิทยบริการ
A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Engineering Program in Chemical Engineering

Department of Chemical Engineering

Faculty of Engineering

Chulalongkorn University

Academic year 2006

Copyright of Chulalongkorn University

Thesis Title NEURAL NETWORK BASED MODEL PREDICTIVE CONTROL FOR
A STEEL PICKLING PROCESS
By Miss Wachira Daosud
Filed of study Chemical Engineering
Thesis Advisor Associate Professor Paisan Kittisupakorn, Ph.D.
Thesis Co-advisor Associate Professor Mohamed Azlan Hussain, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment
of the Requirements for the Doctoral Degree

Direk Lavansiri Dean of the Faculty of Engineering
(Professor Direk Lavansiri, Ph.D.)

THESIS COMMITTEE

Piyasan Praserttham Chairman
(Professor Piyasan Praserttham, Dr.Ing.)

Paisan Kittisupakorn Thesis Advisor
(Associate Professor Paisan Kittisupakorn, Ph.D.)

Azlan Hussain Thesis Co-advisor
(Associate Professor Mohamed Azlan Hussain, Ph.D.)

Montree Wongsri Member
(Assistant Professor Montree Wongsri, D.Sc.)

Suphot Phatanasri Member
(Suphot Phatanasri, D.Eng.)

Wirat Vanichsriaratana Member
(Assistant Professor Wirat Vanichsriaratana, Ph.D.)

วชิรา ดาวสุด : การควบคุมแบบโมเดลพรีดิกทีฟที่พร้อมทั้งช่วยงานนิวัลสำหรับกระบวนการกำจัด
สนิมเหล็ก. (NEURAL NETWORK BASED MODEL PREDICTIVE CONTROL FOR A STEEL
PICKLING PROCESS) อ. ที่ปรึกษา : รศ. ดร. ไพศาล กิตติศุภกร, อ.ที่ปรึกษาร่วม : ASSOC.
PROF. MOHAMED AZLAN HUSSAIN, Ph.D. 167หน้า.

งานวิจัยนี้เสนอการควบคุมแบบโมเดลพรีดิกทีฟที่พร้อมทั้งช่วยงานนิวัล เพื่อควบคุมกระบวนการ
แบบไม่เป็นเชิงเส้นหลายตัวแปร ซึ่งกระบวนการที่ใช้ในการศึกษาคือกระบวนการกำจัดสนิมเหล็กด้วยกรด
ซึ่งเป็นกระบวนการในอุตสาหกรรมพื้นฐานของประเทศไทย สิ่งที่ต้องการควบคุมในกระบวนการนี้คือ
ความเข้มข้นของกรดไฮโดรคลอริกในถังสารละลายกรดกำจัดสนิม โดยในถังของสารละลายกรดจะมี
พฤติกรรมเป็นแบบไม่เป็นเชิงเส้น และมีผลกระทบซึ่งกันและกันของตัวแปรหลายตัวแปรที่เกี่ยวข้องใน
ระบบ ในการจำลองแบบจำลองของกระบวนการ ได้มีการจำลองกระบวนการกำจัดสนิมเหล็กด้วยกรดโดย
ใช้ช่วยงานนิวัล โดยใช้ข้อมูลที่ได้จากการจำลองกระบวนการของแบบจำลองทางคณิตศาสตร์ของ
กระบวนการกำจัดสนิมเหล็กด้วยกรด ในการฝึกช่วยงานนิวัลจะใช้วิธีของเลเวนเบิร์ก-มาร์ควอธ
แบบจำลองช่วยงานนิวัลที่ได้จะใช้ในอัลกอริทึมของการควบคุมแบบโมเดลพรีดิกทีฟ เพื่อใช้ทำนาย
พฤติกรรมของกระบวนการในอนาคตสำหรับหาค่าตัวแปรปรับกระบวนการที่เหมาะสมโดยใช้วิธีการ
ออฟติไมซ์แบบซีกเซสซีฟควอดราติกโปรแกรมมิง (SQP) ตัวควบคุมที่ได้จะนำมาใช้ในการควบคุม
กระบวนการกำจัดสนิมเหล็กด้วยกรดในหลายๆกรณีศึกษา เช่น กรณีการเปลี่ยนแปลงเซ็ทพอยท์ กรณีที่มี
ตัวรบกวนระบบ กรณีที่มีความผิดพลาดของแบบจำลองและกรณีที่เกี่ยวข้องกับความถี่รบกวน จากผลการ
จำลองแสดงให้เห็นว่าการควบคุมแบบโมเดลพรีดิกทีฟที่พร้อมทั้งช่วยงานนิวัลให้สมรรถนะการควบคุมที่
ดีกว่าเมื่อเปรียบเทียบกับตัวควบคุมแบบพีไอ

นอกจากการประยุกต์ใช้เทคนิคการควบคุมแบบโมเดลพรีดิกทีฟที่พร้อมทั้งช่วยงานนิวัลกับ
กระบวนการกำจัดสนิมเหล็กแล้ว งานวิจัยนี้ยังเสนอการควบคุมกระบวนการกำจัดสนิมเหล็กโดยใช้ตัว
ควบคุมแบบนิวัล (NNDIC) และการควบคุมแบบนิวัลร่วมกับตัวควบคุมแบบพีไอ (Dual Mode) ผลการ
จำลองแสดงให้เห็นว่าการควบคุมแบบนิวัลร่วมกับตัวควบคุมแบบพีไอสามารถกำจัดออฟเซ็ทที่เกิดขึ้น
จากการควบคุมแบบนิวัลได้อย่างเดียวได้และให้สมรรถนะการควบคุมที่ดีกว่าเมื่อเปรียบเทียบกับ
การควบคุมแบบ NNDIC และพีไอ

ภาควิชาวิศวกรรมเคมี.....
สาขาวิชาวิศวกรรมเคมี.....
ปีการศึกษา ..2549.....

ลายมือชื่อนิสิต.....
ลายมือชื่ออาจารย์ที่ปรึกษา..... Patsam Wittisupakorn
ลายมือชื่ออาจารย์ที่ปรึกษาร่วม..... Alathasan

4371816921 : MAJOR CHEMICAL ENGINEERING

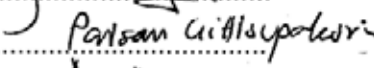
KEY WORD: MODEL PREDICTIVE CONTROL / NEURAL NETWORK / A STEEL PICKLING PROCESS / MODELING / DUAL MODE CONTROL

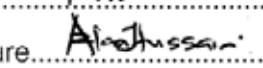
WACHIRA DAOSUD : NEURAL NETWORK BASED MODEL PREDICTIVE CONTROL FOR A STEEL PICKLING PROCESS. THESIS ADVISOR : ASSOC. PROF. PAISAN KITTISUPAKORN, Ph.D., THESIS COADVISOR : ASSOC. PROF. MOHAMED AZLAN HUSSAIN, Ph.D., 167 pp.

A neural network based model predictive control strategy (NNMPC) is developed for a multivariable nonlinear system in this research. A steel pickling process which is a fundamental industry in Thailand and has long existed and served the country's steel demand is chosen as the case study. The hydrochloric acid concentrations in the acid baths are the controlled variables. The baths exhibit common features in an industrial systems including nonlinear dynamics and interaction among variables. In the modeling, multiple-input single-output multilayer feedforward neural network models are developed using input-output data sets obtaining from mathematical model simulation. The Levenberg-Marquardt algorithm is used to train the neural network process models. In the control algorithm, the neural network models are used to predict the future process response in a model predictive control (MPC) algorithm for searching the optimal control actions via the Successive quadratic programming (SQP). The proposed algorithm is tested to control the steel pickling process in simulation for several cases such as set point tracking, disturbance, model mismatch and presence of noise. The results of NNMPC show better performance in control of the system over conventional PI controller in most cases.

In addition implementation of inverse neural network (InvNN) controller and Dual Mode (DM) strategy to a steel pickling process is investigated. Simulation results have demonstrated that DM control strategy give good control results for the steel pickling process and remove the offset when compared to the NNDIC and conventional PI controller.

Department ..Chemical Engineering..... Student's signature..... 

Field of study ..Chemical Engineering..... Advisor's signature..... 

Academic year ..2006..... Co-advisor's signature..... 

ACKNOWLEDGEMENTS

I would like to express my gratitude to my thesis advisor, Associate Professor Paisan Kittisupakorn, for his inspiration, encouragement and supporting throughout my thesis. I would like to thank my thesis co-advisor, Associate Professor Mohamed Azlan Hussain, for helping and providing valuable advice.

I wish to especially thank the other members of my thesis committee, Professor Piyasan Prasertdam, Dr. Montree Wongsri and Dr. Suphot Phatanasri, for their time and useful comments on this thesis.

I gratefully acknowledged the financial support from the Thailand Research Fund (TRF). Many thanks to all my friends, colleagues in my research group and staffs in the Department of Chemical Engineering, Chulalongkorn University and University of Malaya for their friendship and assistance.

Finally, I would like to express my deepest gratitude to my beloved parents and my brother for the endless love, inspiration and encouragement.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CONTENT

	PAGE
ABSTRACT (IN THAI).....	iv
ABSTRACT (IN ENGLISH).....	v
ACKNOWLEDGEMENTS.....	vi
CONTENT.....	vii
LIST OF TABLES.....	x
LIST OF FIGURES.....	xi
NOMENCLATURE.....	xviii
CHAPTER	
I INTRODUCTION.....	1
1.1 Research Objectives.....	2
1.2 Contributions.....	3
1.3 Dissertation Overview.....	3
II LITERATURE REVIEWS.....	5
2.1 Neural Network.....	5
2.1.1 Applications of neural network.....	6
2.2 Model Predictive Control.....	11
2.2.1 Applications of MPC.....	13
III THEORY OF NEURAL NETWORK AND MODEL PREDICTIVE CONTROL.....	15
3.1 Neural Network Introduction.....	15
3.2 Neural Network Architectures.....	18
3.2.1 Models of a neuron.....	18
3.2.2 Types of activation function.....	19
3.2.3 Feedforward neural networks.....	22
3.2.4 Recurrent neural networks.....	24
3.3 Neural Network Training.....	25
3.3.1 Supervised training.....	25

	PAGE
3.3.2 Unsupervised training.....	25
3.3.3 Reinforcement training.....	26
3.4 Feedforward Multilayer Perceptron.....	26
3.5 Training Algorithm.....	30
3.5.1 The delta rule.....	30
3.5.2 The backpropagation method.....	31
3.5.3 Levenberg-Marquardt method.....	36
3.6 Model Predictive Control.....	37
3.6.1 MPC algorithms.....	39
3.6.2 Formulation of model predictive control Problem.....	41
3.6.3 Model predictive control strategy.....	42
3.6.4 Advantages and Disadvantages of MPC.....	43
 IV NEURAL NETWORK MODELING AND INVERSE NEURAL NETWORK MODELING FOR A STEEL PICKLING PROCESS.....	45
4.1 A Steel Pickling Process.....	45
4.2 Neural Network Modeling.....	49
4.3 Procedure for Obtaining Neural Network Forward Models.....	49
4.4 Identification of Neural Network Inverse Models...	57
4.5 Procedure for Obtaining Neural Network Inverse Models.....	58
4.6 The Minimum MSE Method.....	60
4.7 Simulation Results.....	60
 V MODEL PREDICTIVE CONTROL BASED ON NEURAL NETWORK FOR A STEEL PICKLING PROCESS.....	67
5.1 Neural Network Model Based Predictive Control....	67
5.2 Simulation Results.....	71

	PAGE	
VI	NEURAL NETWORK INVERSE MODEL BASED CONTROLLER FOR THE CONTROL OF A STEEL PICKLING PROCESS.....	87
6.1	Neural network direct inverse control strategy (NNDIC).....	87
6.2	Simulation Results and Discussion of NNDIC.....	92
6.2.1	Nominal case.....	92
6.2.2	Disturbance case.....	96
6.2.3	Model mismatch case.....	103
6.2.4	Noise case.....	110
6.3	Dual Mode Control based on Neural Network Inverse Model Strategy.....	113
6.4	Results and Discussion of Dual Mode Control.....	113
6.4.1	Nominal case.....	114
6.4.2	Disturbance case.....	116
6.4.3	Model mismatch case.....	119
VII	CONCLUSIONS.....	122
7.1	A Steel Pickling Process Modeling.....	122
7.2	Neural Network Direct Inverse Control (NNDIC) and Dual Mode Control (DM).....	123
7.3	Neural Network based Model Predictive Control (NNMPC).....	124
	REFERENCES.....	125
	APPENDICES.....	135
	APPENDIX A. MATLAB R2006a – Neural Network Modeling	136
	APPENDIX B. MATLAB R2006a – The Control of NNMPC	139
	VITA.....	148

LIST OF TABLES

		PAGE
Table 4.1	MSE value for different number of hidden nodes in the neural network forward models of the 5%, 10% and 15% HCl baths and the first rinsing bath.....	61
Table 4.2	MSE value for different number of hidden nodes in the neural network inverse models of the 5%, 10% and 15% HCl baths and the first rinsing bath.....	62
Table 5.1	Performance comparison between NNMPC and PI control under the disturbance case.....	72
Table 5.2	Performance comparison between NNMPC and PI control under the model mismatch case.....	76
Table 5.3	Performance comparison between NNMPC and PI control under the noises case.....	83
Table 6.1	Performance comparison between NNDIC and PI control under the nominal case.....	96
Table 6.2	Performance comparison between NNDIC and PI control under the disturbance case (15% increase of the concentration, C_{20}).....	102
Table 6.3	Performance comparison between NNDIC and PI control under the model mismatch case (15% increase of the reaction rate, k).....	109
Table 6.4	Performance of NNDIC under the noise case.....	112
Table 6.5	Performance comparison between Dual mode control, NNDIC and PI control under the nominal case.....	116
Table 6.6	Performance comparison between DM control, NNDIC and PI control under the disturbance case (15% increasing of the concentration, C_{20}).....	117
Table 6.7	Performance comparison between DM control, NNDIC and PI control under the model mismatch case (15% increase of the reaction rate, k).....	120

LIST OF FIGURES

		PAGE
Figure 3.1	The primitive yet powerful human brain.....	16
Figure 3.2	Dendrites, axons and synapse in a biological neuron.....	17
Figure 3.3	Nonlinear model of a neuron.....	21
Figure 3.4	(a) Threshold function.	22
	(b) Piecewise-linear function.....	22
	(c) Sigmoid function for varying slope parameter a	22
Figure 3.5	Feedforward neural networks.....	23
Figure 3.6	Recurrent neural networks.....	24
Figure 3.7	(a) Architecture of a three-layer perceptron.....	29
	(b) A block diagram representation of the perceptron.....	29
Figure 3.8	MPC Strategy.....	43
Figure 4.1	Flow diagram of pickling baths.....	48
Figure 4.2	Flow diagram of rinsing baths.....	48
Figure 4.3	Procedure for obtaining forward and inverse neural network models.....	51
Figure 4.4	Training data set of 5% HCl bath	
	(a) Manipulated variable (Flow rate F2).....	52
	(b) Concentration of 5% HCl bath (C1).....	52
Figure 4.5	Training data set of 10% HCl bath	
	(a) Manipulated variable (Flow rate F3).....	53
	(b) Concentration of 10% HCl bath (C2).....	53
Figure 4.6	Training data set of 15% HCl bath	
	(a) Manipulated variable (Flow rate F5).....	54
	(b) Concentration of 15% HCl bath (C3).....	54
Figure 4.7	Training data set of the first rinsing bath	
	(a) Manipulated variable (Flow rate F6).....	55
	(b) Concentration of the first rinsing bath (C4).....	55
Figure 4.8	Structure for the training of the forward neural network model	57
Figure 4.9	Structure for the training of the inverse neural network model	59

Figure 4.10	The validation result of 5% HCl bath neural network model (structure 5-4-1).....	63
Figure 4.11	The validation result of 10% HCl bath neural network model (structure 9-4-1).....	63
Figure 4.12	The validation result of 15% HCl bath neural network model (structure 7-8-1).....	64
Figure 4.13	The validation result of 1 st rinsing bath neural network model (structure 7-4-1).....	64
Figure 4.14	The validation result of 5% HCl bath inverse neural network model (structure 5-4-1).....	65
Figure 4.15	The validation result of 10% HCl bath inverse neural network model (structure 9-8-1).....	65
Figure 4.16	The validation result of 15% HCl bath inverse neural network model (structure 7-12-1).....	66
Figure 4.17	The validation result of 1 st rinsing bath inverse neural network model (structure 7-16-1).....	66
Figure 5.1	The neural network model of 5% HCl bath (structure 5-4-1)..	69
Figure 5.2	The neural network model of 10% HCl bath (structure 9-4-1)	69
Figure 5.3	The neural network model of 15% HCl bath (structure 7-8-1)	70
Figure 5.4	Flow diagram of a steel pickling process control.....	70
Figure 5.5	Multivariable NNMPC strategy.....	71
Figure 5.6	NNMPC control for HCl acid concentration	
	(a) 5% HCl bath.....	73
	(b) 10% HCl bath.....	73
	(c) 15% HCl bath.....	74
Figure 5.7	PI control for HCl acid concentration	
	(a) 5% HCl bath.....	74
	(b) 10% HCl bath.....	75
	(c) 15% HCl bath.....	75
Figure 5.8	Concentration control in 5% HCl bath under the disturbance case	
	(a) NNMPC.....	77
	(b) PI control.....	77

Figure 5.9	Concentration control in 10% HCl bath under the disturbance case	
	(a) NNMPC	78
	(b) PI control.....	78
Figure 5.10	Concentration control in 15% HCl bath under the disturbance case	
	(a) NNMPC.....	79
	(b) PI control.....	79
Figure 5.11	Concentration control in 5% HCl bath under the model mismatch case	
	(a) NNMPC.....	80
	(b) PI control.....	80
Figure 5.12	Concentration control in 10% HCl bath under the model mismatch case	
	(a) NNMPC.....	81
	(b) PI control.....	81
Figure 5.13	Concentration control in 15% HCl bath under the model mismatch case	
	(a) NNMPC.....	82
	(b) PI control.....	82
Figure 5.14	Concentration control in 5% HCl bath under the noise case	
	(a) NNMPC.....	84
	(b) PI control.....	84
Figure 5.15	Concentration control in 10% HCl bath under the noise case	
	(a) NNMPC.....	85
	(b) PI control.....	85
Figure 5.16	Concentration control in 15% HCl bath under the noise case	
	(a) NNMPC.....	86
	(b) PI control.....	86
Figure 6.1	The inverse neural network controller of 5% HCl bath (structure 5-4-1).....	88
Figure 6.2	The inverse neural network controller of 10% HCl bath (structure 9-8-1).....	88

Figure 6.3	The inverse neural network controller of 15% HCl bath (structure 7-12-1).....	89
Figure 6.4	The inverse neural network controller of 1 st rinsing bath (structure 7-16-1).....	90
Figure 6.5	Flow diagram of pickling baths control system.....	91
Figure 6.6	Flow diagram of rinsing baths control system.....	91
Figure 6.7	Neural network direct inverse model control strategy.....	92
Figure 6.8	Concentration control in 5% HCl bath under the nominal case (a) NNDIC.....	93
	(b) PI control.....	93
Figure 6.9	Concentration control in 10% HCl bath under the nominal case (a) NNDIC.....	94
	(b) PI control.....	94
Figure 6.10	Concentration control in 15% HCl bath under the nominal case (a) NNDIC.....	94
	(b) PI control.....	95
Figure 6.11	Concentration control in 1 st rinsing bath under the nominal case (a) NNDIC.....	95
	(b) PI control.....	95
Figure 6.12	Concentration control in 5% HCl bath under the disturbance case (15% increase of the concentration, C_{20}) (a) NNDIC.....	97
	(b) PI control.....	97
Figure 6.13	Concentration control in 10% HCl bath under the disturbance case (15% increase of the concentration, C_{20}) (a) NNDIC.....	97
	(b) PI control.....	98
Figure 6.14	Concentration control in 15% HCl bath under the disturbance case (15% increase of the concentration, C_{20}) (a) NNDIC.....	98
	(b) PI control.....	98

Figure 6.15	Concentration control in 1 st rinsing bath under the disturbance case (15% increase of the concentration, C_{20})	
	(a) NNDIC.....	99
	(b) PI control.....	99
Figure 6.16	Concentration control in 5% HCl bath under the disturbance case (15% decrease of the concentration, C_{20})	
	(a) NNDIC.....	99
	(b) PI control.....	100
Figure 6.17	Concentration control in 10% HCl bath under the disturbance case (15% decrease of the concentration, C_{20})	
	(a) NNDIC.....	100
	(b) PI control.....	100
Figure 6.18	Concentration control in 15% HCl bath under the disturbance case (15% decrease of the concentration, C_{20})	
	(a) NNDIC.....	101
	(b) PI control.....	101
Figure 6.19	Concentration control in 1 st rinsing bath under the disturbance case (15% decrease of the concentration, C_{20})	
	(a) NNDIC.....	101
	(b) PI control.....	102
Figure 6.20	Concentration control in 5% HCl bath under the model mismatch case (15% increase of the reaction rate, k)	
	(a) NNDIC.....	104
	(b) PI control.....	104
Figure 6.21	Concentration control in 10% HCl bath under the model mismatch case (15% increase of the reaction rate, k)	
	(a) NNDIC.....	104
	(b) PI control.....	105
Figure 6.22	Concentration control in 15% HCl bath under the model mismatch case (15% increase of the reaction rate, k)	
	(a) NNDIC.....	105
	(b) PI control.....	105

Figure 6.23	Concentration control in 1 st rinsing bath under the model mismatch case (15% increase of the reaction rate, k)	
	(a) NNDIC.....	106
	(b) PI control.....	106
Figure 6.24	Concentration control in 5% HCl bath under the model mismatch case (15% decrease of the reaction rate, k)	
	(a) NNDIC.....	106
	(b) PI control.....	107
Figure 6.25	Concentration control in 10% HCl bath under the model mismatch case (15% decrease of the reaction rate, k)	
	(a) NNDIC.....	107
	(b) PI control.....	107
Figure 6.26	Concentration control in 15% HCl bath under the model mismatch case (15% decrease of the reaction rate, k)	
	(a) NNDIC.....	108
	(b) PI control.....	108
Figure 6.27	Concentration control in 1 st rinsing bath under the model mismatch case (15% decrease of the reaction rate, k)	
	(a) NNDIC.....	108
	(b) PI control.....	109
Figure 6.28	Concentration control by NNDIC under the noise case with the disturbance in C_{20} (+15%)	
	(a) 5% HCl Bath.....	110
	(b) 10% HCl Bath.....	110
	(c) 15% HCl Bath.....	111
	(d) 1 st rinsing Bath.....	111
Figure 6.29	Concentration control by NNDIC under the noise case with the model mismatch in k (+15%)	
	(a) 5% HCl Bath.....	111
	(b) 10% HCl Bath.....	111
	(c) 15% HCl Bath.....	112
	(d) 1 st rinsing Bath.....	112

Figure 6.30	Concentration control in 15% HCl bath under the nominal case	
	(a) DM control.....	115
	(b) NNDIC.....	115
	(c) PI control.....	115
Figure 6.31	Concentration control in 15% HCl bath under the disturbance case (15% increase of the concentration, C_{20})	
	(a) DM control.....	117
	(b) NNDIC.....	118
	(c) PI control.....	118
Figure 6.32	Concentration control in 15% HCl under the model mismatch case (15% increase of reaction rate, k)	
	(a) DM control.....	120
	(b) NNDIC.....	121
	(c) PI control.....	121

NOMENCLATURES

CHAPTER III

a	slope parameter of the sigmoid function
b	bias
h	equality constraint functions
J	Jacobian matrix
k	inequality constraint functions
n	number of neurons
o	output vector
P	length of the prediction horizon
t_0	current time
u	value of the internal potential
v_k	induced local field of the neuron
w	synaptic weight
W	interconnection matrix of the synaptic weights
x	input vector
x_j	input of synapse j
y	output vector
y_k	output of neuron k

CHAPTER IV

A	area of operating tank [= $7.29 \times 10^{-2} \text{ m}^2$]
C	concentration of HCl [mol/l]
C_{20}	20% by weight concentration of HCl
F	volumetric rate [l/min]
h	height of operating tank [m]
k	reaction rate constant [= $3.267 \times 10^{-4} \text{ mol}/(1 \text{ min})$]
q	amount of acid solution that stuck with samples [= $2 \times 10^{-3} \text{ l/min}$]
t	time [min]
V	volume of operating tank [m^3]

GREEK SYMBOLS

α	momentum
δ	error
φ	activation function
η	learning coefficient
Ψ	nonlinear bounded activation function
Ψ	diagonal nonlinear operator with (typically identical) sigmoid activation functions
Θ	bias
Γ	weighting parameter

SUPERSCRIPT

[s]	the s-th layer
-----	----------------

SUBSCRIPTS

w	water
sp	setpoint



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER I

INTRODUCTION

Chemical manufacturing processes present many challenging control problems, including: nonlinear dynamic behavior; multivariable interactions between manipulated and controlled variables; unmeasured state variables; unmeasured and frequency disturbances; high-order and distributed processes; uncertain and time-varying parameters; un-modeled dynamics; constraints on manipulated and state variables; and (variable) dead time on inputs and measurements. A number of control approaches and algorithms that are able to handle some of the above process characteristics have been presented in the academic literature in recent years. Bequette (1991) gives a review of various approaches such as internal model approaches, differential geometric approaches, reference system synthesis techniques, including internal decoupling and Generic Model Control (GMC), Model Predictive Control (MPC) and also various special and ad hoc approaches. Many of these approaches are not able to handle the various process characteristics and requirements met in industrial applications and some of the approaches can only be applied for special classes of models.

Model Predictive Control appears to be the only general approach which can handle most of the common process characteristics and industrial requirements in a satisfactory way. It also seems to be the approach which are most suitable for the development of general and application independent software, which is essential for the development of cost-effective applications. The key of the successful use of MPC in solving the process problem is the accurate model. However, chemical processes have been traditionally controlled using linear system analysis and design tools even though they are the inherent nonlinear process.

Recently, neural networks have been successfully applied in identification and controlling nonlinear processes. Neural networks offer alternative nonlinear models for MPC of industrial systems (Lightbody et al., 1997; Doherty et al., 1997; Henson, 1998; Hussain, 1999). Different ways of neural models being embedded in MPC

systems were reviewed by two recent surveys (Yu and Gomm, 2002; Lightbody and Irwin, 1997). It is noted that while neural network modeling and control techniques are investigated for nonlinear systems, the current methods proposed and tested by simulations and some implementations to laboratory rigs are mainly for single-input single-output (SISO) systems (Gomm et al., 1997; Lennox et al., 1998). Applications of neural networks in chemical process modeling and MPC have been investigated for SISO systems (Rohani et al., 1999; Daosud et al., 2005; Psychogios et al., 1991; Elman et al., 1990). Very few investigations into neural control for multiple-input multiple-output (MIMO) chemical processes have been reported.

The work presented in this dissertation is focused on the implementation of a model predictive control based on neural network (NNMPC) technique to control a MIMO chemical process. A steel pickling process which is the highly nonlinear dynamic behavior, multivariable and interaction between variables is chosen to represent such a system. Since the success of MPC is largely depend on the availability of models of the process to be controlled, this research concentrates on the development of neural network model for describing the dynamics of the steel pickling process. The developed neural network model is then used in MPC algorithm.

1.1 Research Objectives

The overall objective of this research concentrates on the application of model predictive control based on neural network in an industrial chemical process. The steel pickling process is chosen for an industrial case study. The neural network models of this process have been developed from simulation data of the process. The developed process models are also used for control purpose. From the view of this objective, this research can be divided into two sections:

1. Developing the process models for describing the steel pickling process behavior by using neural network strategy.

2. An implementation of a model predictive control based on neural network technique in industrial chemical process, the steel pickling process.
 - Investigating the performance of model predictive control based on neural network for the control of the steel pickling process.
 - Comparing the performance of model predictive control based on neural network with the other control strategy such as inverse neural network and the convention control technique.

1.2 Contributions

The main contributions of this dissertation are:

1. The models for describing the behavior of the steel pickling process (MIMO process) have been developed based on neural network technique.
2. The model predictive control based on neural network technique has been developed for the tighter control of MIMO process and highly nonlinear, steel pickling process.
3. The inverse neural network controllers have been developed for the control of the steel pickling process.

1.3 Dissertation Overview

This dissertation is organized as follows. Chapter 2 reviews the literature for work related to history, concepts and background of artificial neural network and model predictive control and their applications as studied by previous researchers.

Chapter 3 discusses the artificial neural network and model predictive control strategy. Since neural network model is used in MPC algorithm for NNMPC technique, the structure of neural network and the formulation of MPC problem are provided in this chapter.

Chapter 4 begins with the process description of a steel pickling process studied in this work. The mathematical model of the steel pickling process derived from mass balances is developed. Forward and inverse neural networks model are presented in order to use as process model and controller of the steel pickling process.

Chapter 5 describes the implementation of NNMPC to control the steel pickling process. The neural network models developed in Chapter 4 are used here to design the NNMPC controller. To evaluate the performance of NNMPC, results are compared with a traditional PI controller. Simulation studies of the NNMPC and PI controllers are demonstrated and discussed.

Chapter 6 describes the implementation of inverse neural network (InvNN) to control the steel pickling process. A Dual Mode (DM) control strategy is presented in order to remove some offset obtaining from InvNN control. To evaluate the performance of DM, results are compared with the InvNN and a traditional PI controller. Simulation studies of the DM, InvNN and PI controllers are demonstrated and discussed.

Chapter 7 gives a conclusion of this dissertation.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER II

LITERATURE REVIEWS

2.1 Neural Network

Many of the concepts behind artificial neural networks (ANNs) were first discussed by biologists in the nineteenth century. For instance, William James suggested in 1890 that the activity at one point in the brain was due to the culminative activities of other points in the brain. This is a core feature of an artificial neuron developed 50 years later (i.e. the McCulloch-Pitts neuron). It was not until the 1940's that significant theoretical advances were made (e.g. McCulloch and Pitts, 1943; Hebb, 1949). In the 1960s researchers were able to dispose of their cumbersome analogue computer hardware and pursue their work through digital computer simulation. This led to the perceptron (Rosenblatt, 1962), which is still widely used. But the 1960s also heralded a lull in ANN research, usually attributed to the work of Minsky and Papert (1969), which highlighted substantial limitations of the single layer perceptron network. In the late 1970s interest in neural computing was rekindled and this interest was accelerated in the 1980s due to numerous theoretical advancements (i.e. Hopfield, 1982; Hinton et al., 1984; Rumelhart et al., 1986; Kohonen, 1988; Grossberg, 1988).

A good review of the history of neural computation can be found in Widrow (1990) or from the excellent collection of early papers, by Anderson and Rosenfeld (1988). Almost 50 different types of neural network architectures are developed, although only some of them are in common use. There are numerous introductory level books to neural computation. A more detailed presentation and analysis can be found for example in Rumelhart and McClelland (1986) or Herz et al. (1991). Since then neural network control has progressed rapidly and also real industrial applications have been reported, e.g. Widrow et al. (1994), especially in Japan, e.g. Asakawa and Takagi (1994). A computer survey (INSPEC) considering the experimental or practical applications of the neural network control in 1994 resulted

262 journal articles and 869 conference papers. Of course only a portion of all these were real on-line tests of neural network control. The fuzzy control was excluded from the search to reduce the number of articles, although Neuro-fuzzy systems form an entity and it is often hard to distinguish between all the approaches, except by the title.

2.1.1 Applications of neural network

Neural network has been used widely in system identification and control. They are used to model all types of processes regardless of whether the processes are linear or non-linear (Pham and Oh, 1999). A feedforward neural network has been used successfully by Petrova et al. (1998) in modeling a fermentation process where the network is trained to predict the specific growth rate. They successfully used the neural network in performing system identification to obtain the kinetic models of the specific growth rate and the specific consumption rate. These parameters are the most important process properties in fermentation process modeling.

Mei and Chen (1997) have studied the use of feedforward neural network in fermentation process to harvest four metabolic products consisting of acetic acid, acetoin, ethanol, and 2, 3-butanediol where 2, 3-BDL is the target product. A neural network model is trained to provide step ahead prediction of the process. Performance of neural network prediction in the bio-reaction systems is studied by varying the size of its learning interval. The sizes of data sampling interval were found to contribute vastly on the prediction. They have concluded that the identification results from the neural network with higher step prediction are more reliable and produced better results, improving significantly the identification of the system.

The use of artificial neural networks for modeling in High Performance Liquid Chromatography (HPLC) testing method development for amiloride and methylclothiazide separation was studied by Agatonavic-Kustrin et al. (1998). The independent input variables were pH and methanol percentage and the outputs were the capacity factors. The results were compared to well-known statistical method consisting of multiple nonlinear regression analysis. The study had shown that the networks were able to predict the experimental responses more accurately than the

conventional regression analysis. The neural network showed to be a very powerful tool in HPLC testing method development that give better results to those that are obtained by the multiple regression technique.

Studies have taken place in order to model and understand the dynamic properties of power plant process behavior. Lu and Hogg (2000) utilized a 200 MWatt oil-fired drum-type boiler-turbine-generator unit power plant and a derived mathematical model in order to obtain its process model. They have used the neural network modeling by implementing a supervised training of neural networks to improve training time. They have also looked into practical aspects for selecting the testing and validation data to ensure sufficient excitations covering its proper dynamic behavior. They have extracted and mapped out the power plant open-loop dynamics for use in their neural network training in order to obtain the model in load and no-load operation conditions of the power plant.

A blast furnace, which is a common process device in manufacturing of iron and steel is modeled using artificial feedforward neural network in a research carried out by Radhakrishnan and Mohamed (2000). Their work successfully exhibited that neural network is capable of identifying the optimum process operating properties. The neural network is also used as the estimator of such complex process to estimate the unknown or hard to measure parameter in the process. Real time process data equivalent to six months of process operational period was used as the training data in order to train the networks. Initially, they have to deal with 51 possible network inputs but they have reduced them to 35 through correlation analysis method. By varying the numbers of hidden layers, they have shown that the training produced better convergence profiles. The neural model worked together with an expert system to predict the blast temperature, humidity and pressure trim and used for the stove control system. Performance controller is evaluated from the quantity of sulfur in the hot metal before and after the implementation of the expert system. Purer hot metal were obtained from the process, which incorporate the neural network control system.

Online identification of a pilot scale fluidized-bed coal gasification unit using neural networks was performed by Nougues et al. (2000) where neural network process models of the coal gasification unit were obtained. They implemented the

neural network model and trained them in serial and parallel to various system identification strategies. The neural network models are the used in a model based control strategy. They have shown that the neural network can be used to model a nonlinear system such as this process and the network can be easily incorporated with other system such as the expert system. They have also used neural networks to predict the possible defects for the system.

Taking example of a heat transport system, Leger et al. (1997) have demonstrated the feasibility of neural networks in implementing fault diagnostic to the system. The neural network is trained in order to try to minimize the false signals, which occurs in normal process operations. The main objective is to capture the error at its roots. This is done with the help of signature pattern evaluation of the fault diagnosis. Once the neural fault diagnostic system is up and running, the operator can pin point directly where the error and fault originated.

Neural network in control applications, there are many neural network implemented as process controllers in various types of strategies in control application. A feedforward neural network neural network is shown to have vast capability in the control engineering field. Despite of this, many are still in simulation and trial stages. Simulation studies have proven that the neural network is capable of being a feasible and reliable process controller. However, the only ultimate test platform is its online application in any process. In many industrial processes, conventional proportional-integral-derivative, on-off, continuous and programmable logic controllers (PLC) still dominate real world applications. Dealing with them raises the controller tuning issue. The tuning parameters normally involve using the closed loop tuning method, the Ziegler-Nichols or the Cohen-Coon method that are normally tedious and time consuming.

Chan et al. (1995) conducted a study to have neural network contributing to the conventional proportional-integral-derivative controller tuning process. According to these investigations, it is not necessary to have detail knowledge about the process to pre-tune the controller and they proposed a method based on neural network to fine-tune the controller continuously. This is done by monitoring the performance of its closed loop criteria. Performance objective such as the normalized peak rise time,

overshoot, peak-to-peak height and its final error are selected as inputs to the network. Once the trained neural network is implemented, it adjusts the proportional and integral values, which are directly used in the process as the newly suggested tuning parameters.

Nikravesh et al. (2000) studied the application of neural network in controlling various chemical processes. In their work, neural networks have been applied to various nonlinear systems such as the continuous stirred tank (CSTR) and the neutralization process. Neural network is used to model the process while its mathematical inverse, which act as a controller is used to regulate the process. It is shown that the neural network controller exhibited better performance in controlling the non-isothermal CSTR over the conventional controllers with faster rise time and less offsets.

Control algorithm based on neural networks has also been applied by Acosta et al. (1999) to a PUMA robot manipulator arm that has five degrees of freedom movements. This control strategy exhibited the use of a neural network in providing crucial parameters to the PID controllers used to control joint angles and velocities. A decentralized model has been assumed, where a controller is attached with each joint and a separate neural network is used to adjust the parameters of each controller. Neural network is trained to predict the best modification to the parameters of the conventional controllers in order to regulate the robot manipulator joint angles and its velocities. The results have shown better response in achieving the desired robotic manipulator joint angle and velocities in order to accomplish desired manipulator coordinates as given by the user.

Neural network control can be implemented in various control strategies such as in the direct inverse and internal model-based control. Hussain (1999) has discussed possible applications of these techniques to chemical process using neural network approach. The internal model based control structure that consists of the inverse neural network model as the controller and a parallel forward is modified and adaptive properties are introduced into the strategy by Liew et al. (1999). They have utilized this idea and simulated the strategy to control biomass concentration in a fermentation unit. In this adaptive scheme, the sliding window learning method is

modified from the original idea from Breusegem et al. (1991). The control strategy exhibited great improvement in the control results especially in rejecting process disturbances.

Neural network control approach was carried out online by Zammareno and Vega (1997) and was applied to a melter unit commonly used in sugar production industries. In their work, the neural network was trained to capture the process dynamics and implemented in the model predictive control strategy.

Dutta and Rhinehart (1999) worked on a 6-stage, lab-scale, atmospheric distillation column separating a methanol-water mixture. Two separate neural network models were utilized in inverse model based scheme to predict the boil up and reflux for the process. Experimental results showed that the neural network control outperformed the available advanced control strategy for the distillation column using model predictive control that was proposed by Gupta and Rhinehart (1994).

Online implementation of inverse model based and conventional neural network internal model control strategies have been reported by Hussain and Kershenbaum (2000) using a pilot scale reactor system. They have successfully show and validated the simulation results to control the system's temperature by using neural network. They have tested the inverse model based control strategies including the internal model based control using neural network. The controller showed good performance when tested for setpoint tracking and load disturbance rejection.

Neural network control of a plant for xylose production used in paper manufacturing factories was proposed and implemented by Alvarez et al. (1999). Xylose was obtain from hard wood hemicellulose and was processed in a reactor. Heat is released to the surrounding due to the process being highly exothermic. Neural network control was used to manage and regulate the reactor temperature. Quality of the xylose produced is highly dependent to the reactor process temperature. The online results showed an improvement in the temperature stabilization time as compared to that obtained using a classic PID controller.

2.2 Model Predictive Control

Model predictive control (MPC), a control algorithm that uses an optimizer to solve for the control trajectory over a future time horizon based on a dynamic model of the process, has become a standard control technique in the process industries over the past two decades. MPC is commonly used for constrained multiple-input/multiple-output (MIMO) control problems which are often encountered in the process industries.

Model predictive control has appeared in several branches of the control literature during the past thirty years. The concept of using an open loop optimal control computation to synthesize a feedback controller is so natural that it probably occurred to many researchers in the optimal control field in the late 1950s and 1960s. In their textbook on optimal control, Lee and Markus describe the approach while pointing out that current (as of 1967) hardware and software make real time implementation of the controller difficult. In their review, Garcia et al. (1989) cite Propoi (1963) as the first to introduce, explicitly, the finite moving horizon.

In the electrical engineering literature, model predictive control is usually called receding (or moving) horizon control. Although this name is clearly more descriptive of the general approach, we will also refer to it as model predictive control since this name has become entrenched in the chemical engineering literature. Kleinman (1970) uses the finite horizon concept to find a state feedback gain that stabilizes linear time invariant systems. Thomas (1975) formulates a quadratic objective function penalizing only the input with the constraint that the state must be zero at the end of the horizon. He shows this formulation results in a state feedback that stabilizes linear time invariant systems. Kwon and Pearson (1977) generalize these results by considering the linear time varying system and using the standard quadratic performance objective including the constraint that $x(t_0+T)=0$. They show that the receding horizon controller can stabilize linear time varying systems. Kwon et al. (1983) also consider linear time varying systems with the quadratic objective. They show the solution to this problem is also stabilizing state feedback law. Mayne and Michalska (1990) consider the quadratic objective function with final time constraint

for the nonlinear system. They show that, under certain conditions, the receding horizon controller stabilizes the nonlinear system.

The MPC framework is also used in aerospace engineering applications. Most publications concern aircraft trajectory optimization (Brusch, 1974; Johnson, 1975). Attention is presently being given to solving nonlinear, constrained optimal trajectories over finite horizons for real time guidance (Bless and Hodges, 1990; Jansch and Paus (1990); Psiaki and Park, 1990).

The use of MPC in the chemical engineering field started in the process industries in the 1970s under the names of “model predictive heuristic control” or “model algorithmic control” (Richalet et al., 1978; Mehra et al., 1982) and “dynamic matrix control” (Cutler and Ramaker, 1979; Prett and Gillette, 1979). The recent review by Garcia et al. (1989) covers the chemical engineering literature on MPC. The survey paper (Garcia et al. 1989) refers to Model Predictive Control (MPC) as that family of controllers in which there is a direct use of explicit and separate identifiable model. The same process model is also implicitly used to compute the control action in a such way that the control design specifications are satisfied.

Control design methods based on the MPC concept have found a wide acceptance in industrial applications due to their high performance and robustness. There are several variants of model predictive control methods, like Dynamic Matrix Control (DMC), Model Algorithmic Control (MAC) and Internal Model Control (IMC). Nonlinear versions of these have also been developed, for example a nonlinear IMC concept, e.g. Economou et al. (1986). Another, largely independently developed branch of MPC, called Generalized Predictive Control (GPC), is aimed more for adaptive control, e.g. Clarke and Mohtadi (1989). For the current state-of-the-art, see Clarke (1994).

Model predictive control in this sense is a broad area and some confusion is encountered, because the abbreviation MPC is often used to mean receding horizon (RHPC) or long range predictive control (LRPC), where a model is used to predict the process output several steps into the future and the control action is computed at each step by numerical minimization of the prediction errors i.e. no specific controller is

used. This is quite different from the concept where the model is controlled with an implicitly derived specific controller, like in many IMC approaches.

2.2.1 Applications of MPC

Nonlinear model predictive control has been applied to a wide variety of process systems. For instance, Norquay et al. (1999) used a nonlinear Wiener MPC to control overhead composition of a C2 splitter. Simulation studies, using the Wiener model for the plant representation, have shown the Wiener MPC based scheme to be successful in rejecting major disturbances and comparisons with linear IMC and IMC using a logarithmic transformation on the output showed the Wiener based version to be superior, as expected for a nonlinear process.

Ju et al. (2000) proposed a nonlinear MPC to control a fabric filtration process. The control algorithm formulated in a multiple-objective optimization framework takes economic into consideration. The global optimization technique is used to compute a manipulated input profile. Simulation results showed that the proposed MPC is especially suitable to the filtration process where the set point change and process disturbance occur frequently.

Seki et al. (2002) formulated the nonlinear MPC based on a successively linearized nonlinear model and applied it to an industrial polypropylene semi batch reactor process as well as to a high density polyethylene (HDPE) continuous stirred tank reactor process. For the semi batch reactor, the nonlinear MPC successfully prevented thermal runaway of the reactor temperature control. For the continuous reactor, the nonlinear MPC improved the closed loop performance during the grade changeover operation, compared with the conventional linear MPC.

Neural networks have successfully been applied to model based control of nonlinear systems. General guidelines can be found from Nahas et al. (1992), Psychogios and Ungar (1991), Hunt and Sbarbaro (1991) and Ydstie (1990). A nonlinear MPC algorithm has been proposed which extends the capacities of Linear Predictive Controllers to control nonlinear systems by Kaeahan et al. (1997). A neural network was used to model the deviation of the nonlinear system from its linear MPC

model. Proposed algorithm was tested in control of an industrial multi-component high purity distillation column by simulation. Results of NNMPC show high improvement in control of system over linear MPC algorithm.

Lin and Stephen (1998) developed a NNMPC. In order to study the effectiveness of NNMPC, a simulation case of wastewater neutralization process was chosen as a test case. The purpose of neutralization was to adjust the pH value to meet the requirements of the different processing units in the wastewater treatment system. Results obtained show that NNMPC could be considered as a powerful alternative control technique for wastewater neutralization processes.

Wei et al. (2002) proposed MPC strategy based on a feedforward neural network model for an industrial polypropylene process. To infer product properties on-line, a dynamic process model was developed and a recursive prediction error method was used to update the model parameters when there is a significant model prediction error. To obtain optimal control strategy during grade transitions, a nonlinear MPC controller was developed based on a neural network model, which is trained using the input-output data of process model. Performance of the nonlinear controller was compared with a conventional PID controller. Application results indicate that MPC controller can obtain satisfactory performance and consequently results in significant reduction in transition time and product variability.

CHAPTER III

THEORY OF NEURAL NETWORK AND MODEL PREDICTIVE CONTROL

This chapter outlines a commonly used type of neural network (NN) and model predictive control (MPC). NN architecture, a widely studied feedforward neural networks and MPC algorithms are discussed.

Neural network model is like human newborns where it needs to be developed, trained and taught to perform desired tasks. This brings to the method of how we can develop the neural network models. Methods to train them are presented, which will illustrate the procedures in detail to obtain reliable neural network models.

3.1 Neural Network Introduction

The term artificial neural network originates from research which attempted to understand, and proposed simple models of, the operation of the human brain. The neural network is a model, which emulates the operation of our brain that is capable of computing vast amount of information to obtain certain results or actions. Nowadays, computers are still using serial operation in carrying out their assigned tasks but increased performance and speed of the computers and workstations made it possible to achieve near parallel computational power in order to mimic our own brain operation.

Studies to comprehend the organization of brains have spread over the scientific and academic research platforms all around the world. Research has shown that human brains have about 10^{11} neurons and 10^{14} synapses. Each synapse is connected to a different neuron and communicates directly with one out of every hundred million cells, which is a small fraction of the total cells in the human brain

(Hardcastle, 1999). This gives us idea of how complex our brains are. Figure 3.1 shows the simplified anatomy of the human brain.

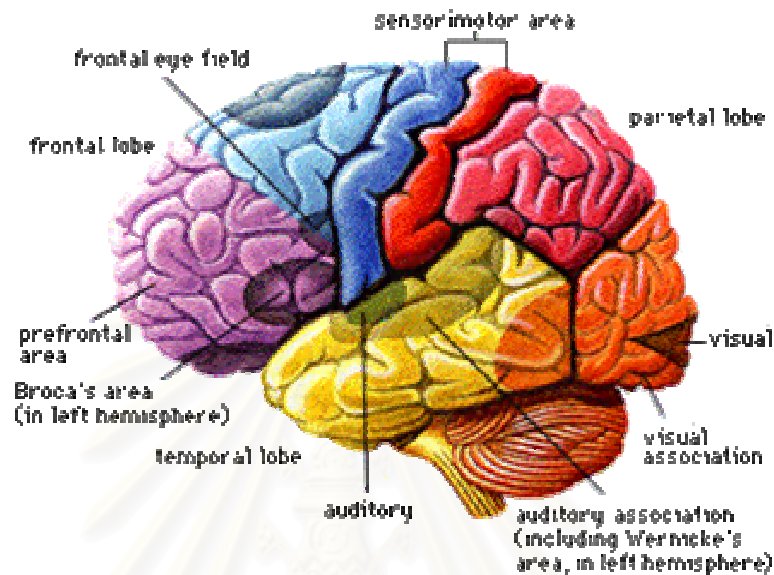


Figure 3.1 - The primitive yet powerful human brain.

Neurons are arranged in a dense mass, also known as the neuropil, in the cranium with only about two tenths of a millimeter separating each cell. On average, each cell has a cell body or soma which is the large and round central body in which almost all the logical functions of the neuron are realized. At one end of the soma, a single axon radiates outward, the bursts into a veritable forest of branches. At the end, a crowd of dendrites extends from the cell. Incoming signals from other neurons pass through the dendrites to the soma, or impinge directly upon the cell body. The cell body computes a weighted average of the incoming pulses and then the axon transmits the results to new cells. Figure 3.2 shows a schematic sketch of the natural sets of neurons that consists of dendrites, axons, synapses and cell body. The axon delivers output of the neuron to connections or other neurons. The dendrite provides large surface area for connections with the other axons from other neurons. The synapse stimulates other neurons to fire and respond. All these activities occur in about one thousandth of a second. These interactions also take place between layers of the

brain. The artificial neural networks work closely to this operating principle of the human brain as signals are propagated through the layers of the networks in order to compute the outputs. The artificial neural network researcher has utilized just a piece of the capabilities of the human brain but yet they can be successfully applied in many applications such as in system identifications, control, prediction and recognition. This also brings us to appreciate more of our brain that is definitely a major gift from the creator himself.

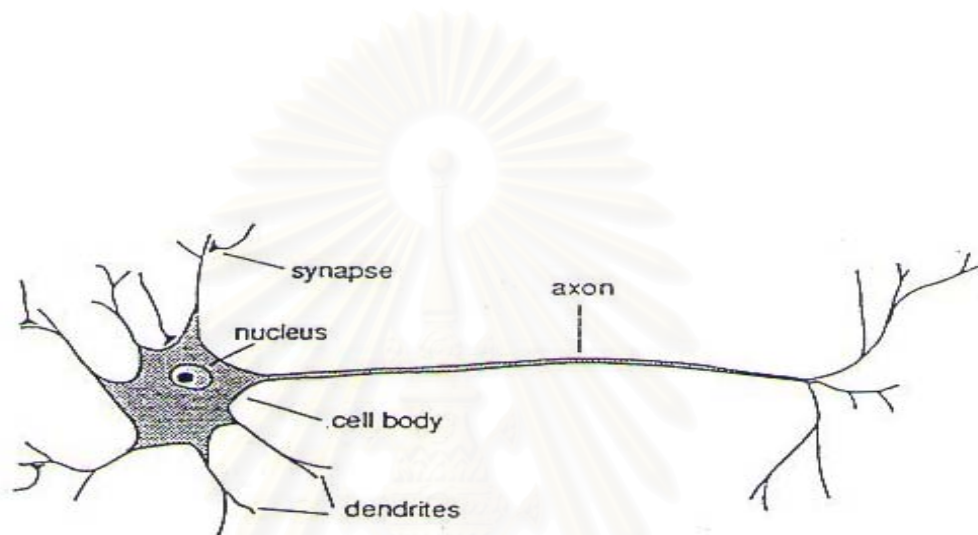


Figure 3.2 - Dendrites, axons and synapse in a biological neuron.

The brain is capable of processing parallel information received from our sensors related to our hearing, sight, touch, feel and smell. Due to this, we can perform walking, reading and listening at the same time. This task requires large amount of data inputs and output to carry out this act. The use of artificial neural network has been motivated by this capability and remarkable performances have been achieved with neural networks in various fields of research areas.

As mentioned before, artificial neural network is a computational model of our brain. It consists of several highly interconnected computational units working in parallel. Electrically, a neuron is equivalent to a few electronic logical gates. Neural networks overcome the knowledge acquisition bottleneck associated with expert

systems by acquiring their knowledge of certain process through training sessions. It also possesses the generalization capability of fuzzy expert systems.

3.2 Neural Network Architectures

Artificial neural networks generally consist of several interconnected processing elements or neurons. Their types are sectioned into various categories. Arrangement and nature of the connections will determine their structure while the manner in which, the connection weights are adjusted and altered to achieve desired outputs by its learning algorithm governs the overall behavior of the network. Hence, neural networks are classified according to their structure and learning algorithms into two basic types of network, i.e. feedforward and recurrent networks (Pham, 1995).

3.2.1 Models of a neuron

A neuron is an information-processing unit that is fundamental to the operation of a neural network. The block diagram of figure 3.3 shows the model of a neuron, which forms the basis for designing (artificial) neural networks. Here we identify three basic elements of the neuronal model (Hakin, 1999) :

1. A set of synapses or connecting links, each of which is characterized by a weight or strength of its own. Specifically, a signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj} . It is important to make a note of the manner in which the subscripts of the synaptic weight w_{kj} are written. The first subscript refers to the neuron in question and the second subscript refers to the input end of the synapse to which the weight refers. Unlike a synapse in the brain, the synaptic weight of an artificial neuron may lie in a range that includes negative as well as positive values.

2. An adder for summing the input signals, weighted by the respective synapses of the neuron; the operations described here constitute a linear combiner.

3. An activation function of limiting the amplitude of the output of a neuron. The activation function is also referred to as a squashing function in that it squashes (limits) the permissible amplitude range of the output signal to some finite value. Typically, the normalized amplitude range of the output of a neuron is written as the closed unit interval $[0,1]$ or alternatively $[-1,1]$.

3.2.2 Types of activation function

The activation function, denoted by $\varphi(v)$, defines the output of a neuron in terms of the induced local field v . Here we identify three basis types of activation functions:

1. Threshold function. For this type of activation function, described in figure 3.4(a),

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \quad (3.1)$$

In engineering literature, this form of a threshold function is commonly referred to as a Heaviside function. Correspondingly, the output of neuron k employing such a threshold function is expressed as

$$y_k = \begin{cases} 1 & \text{if } v_k \geq 0 \\ 0 & \text{if } v_k < 0 \end{cases} \quad (3.2)$$

where v_k is the induced local field of the neuron; that is,

$$v_k = \sum_{j=1}^m w_{kj} x_j + b_k \quad (3.3)$$

Such a neuron is referred to in the literature as the McCulloch-Pitts model, in recognition of the pioneering work done by McCulloch and Pitts (1943). In this model, the output of a neuron takes on the value of 1 if the induced local field of that

neuron is nonnegative, and 0 otherwise. This statement describes the all-or-none property of the McCulloch-Pitts model.

2. Piecewise-linear function. For the piecewise-linear function described in figure 3.4 (b)

$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2} \\ v, & +\frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \quad (3.4)$$

where the amplification factor inside the linear region of operation is assumed to be unity. This form of an activation function may be viewed as an approximation to a nonlinear amplifier. The following two situations may be viewed as special forms of the piecewise-linear function:

- A linear combiner arises if the linear region of operation is maintained without running into saturation.
- The piecewise-linear function reduces to a threshold function if the amplification factor of the linear region is made infinitely large.

3. Sigmoid function. The sigmoid function, whose graph is s-shaped, is by far the most common form of activation function used in the construction of artificial neural networks. It is defined as a strictly increasing function that exhibits a graceful balance between linear and nonlinear behavior. An example of the sigmoid function is the logistic function, defined by

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \quad (3.5)$$

where a is the slope parameter of the sigmoid function. By varying the parameter a , we obtain sigmoid function of different slopes, as illustrated in figure 3.4 (c). In fact, the slope at the origin equals $a/4$. In the limit, as the slope parameter approaches

infinity, the sigmoid function becomes simply a threshold function. Whereas a threshold function assumes the value of 0 or 1, a sigmoid function assumes a continuous range of values from 0 to 1. Note also that the sigmoid function is differentiable, whereas the threshold function is not.

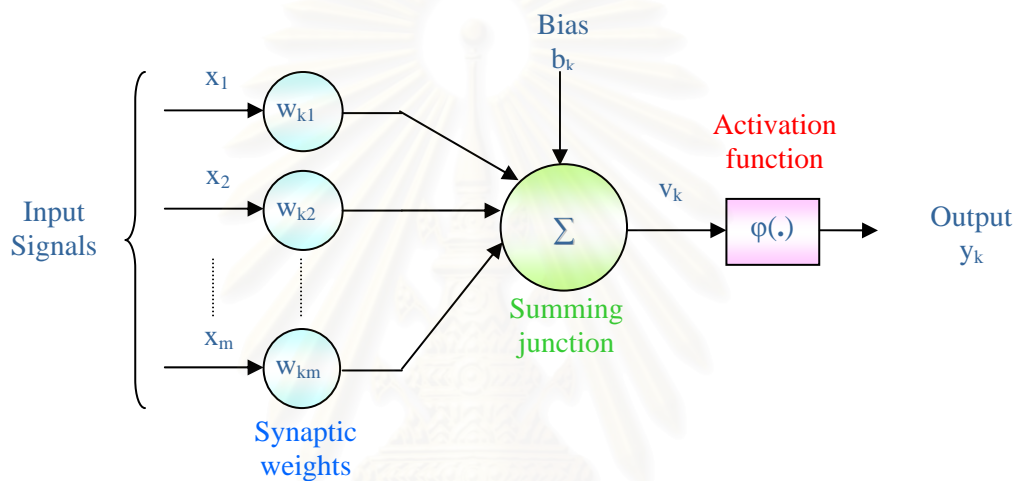


Figure 3.3 - Nonlinear model of a neuron.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

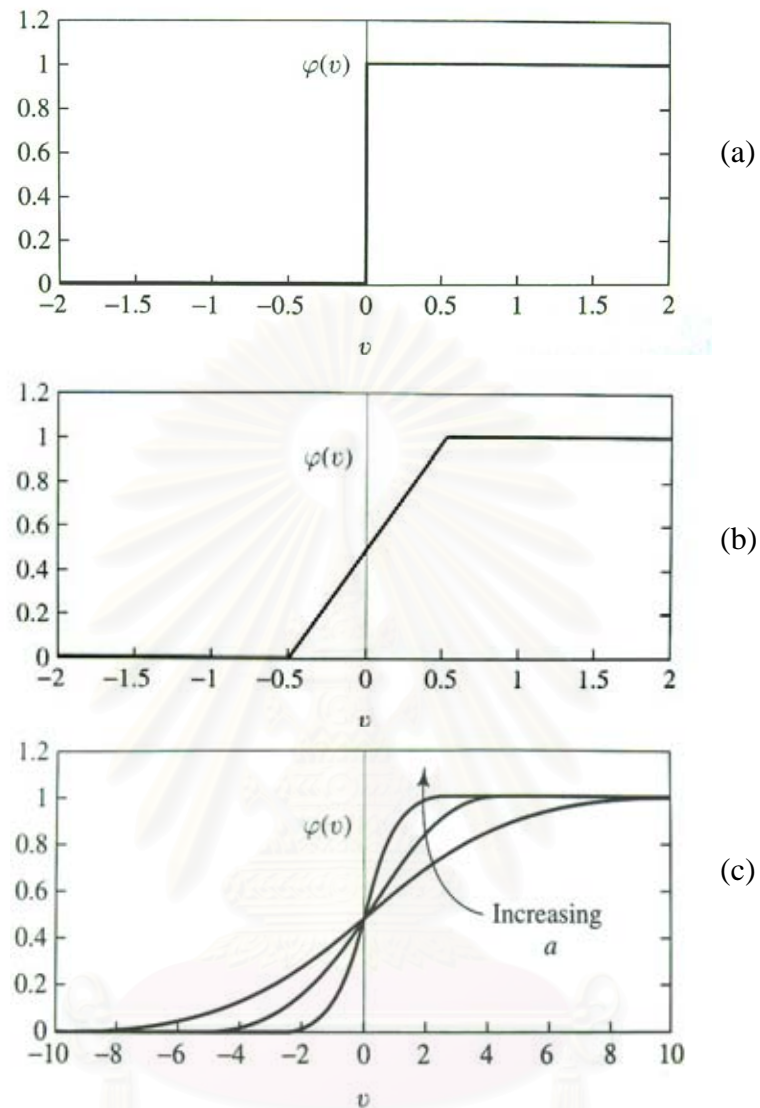


Figure 3.4 - (a) Threshold function. (b) Piecewise-linear function. (c) Sigmoid function for varying slope parameter a .

3.2.3 Feedforward neural networks

An artificial feedforward neural networks (AFNN) consists of various layers. Signal propagates from the input layer to the output layer through unidirectional connections from one layer to another layer. A simple configuration is a two-layer model as shown in figure 3.5. The first layer is the input layer while the second is the hidden and the output layer. These layers are connected to each other by connection

weights to determine the strength of the connections. The weights will have freedom to be changed in an adaptive fashion during the learning or training session. The strength of the connection is adjusted to achieve certain targeted values, which is determined by the training algorithms. A simple example of the feedforward neural network is the multilayer perceptron (MLP). Signals are propagated in the forward direction, from the input to the next hidden and the output layers of the network. There are no connections in the reverse and lateral directions of the network. The learning vector quantization (LVQ) network and the group method of data handling (GMDH) network are other examples of feedforward network (Pham, 1995).

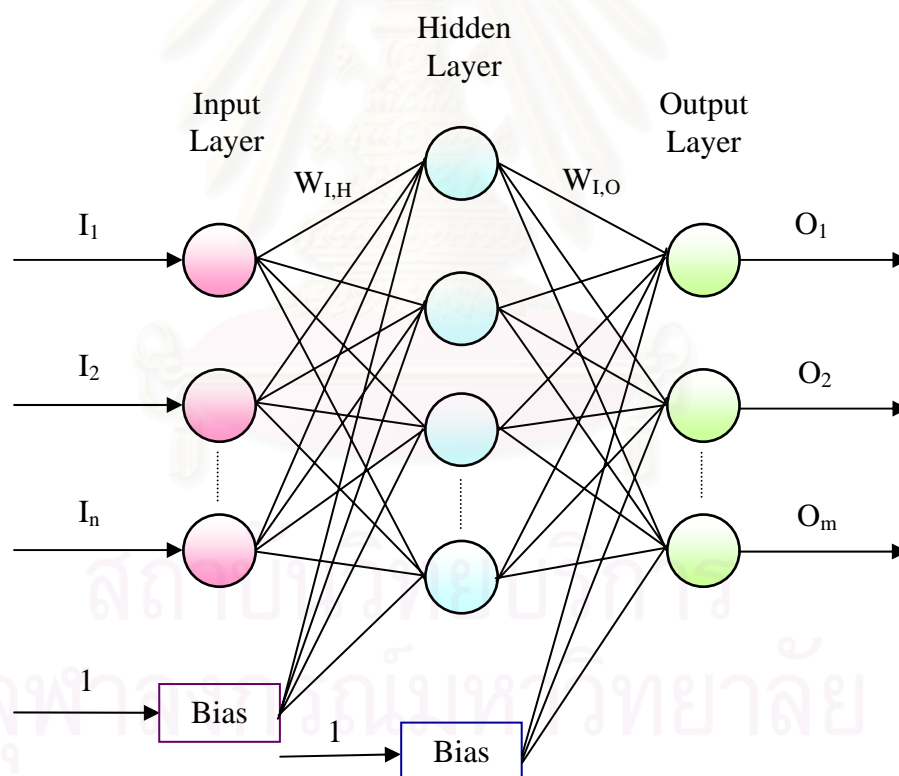


Figure 3.5 - Feedforward neural networks.

3.2.4 Recurrent neural networks

In the recurrent networks, the output of the neurons can be fed back to the same neurons or to the neurons in the other layers. Signal can propagate in both forward and backward directions. Examples of recurrent networks include the Hopfield network, the Elman network and the Jordan network. Recurrent networks have a dynamic memory where their outputs at a given instant reflect the current inputs as well as previous inputs and outputs as shown in figure 3.6. The presence of feedback loops has a profound impact on the learning capability of the network and on its performance. Moreover, the feedback loops involve the use of particular branches composed of unit-delay elements, which results in a nonlinear dynamical behavior, assuming that the neural network contains nonlinear units (Haykin, 1999).

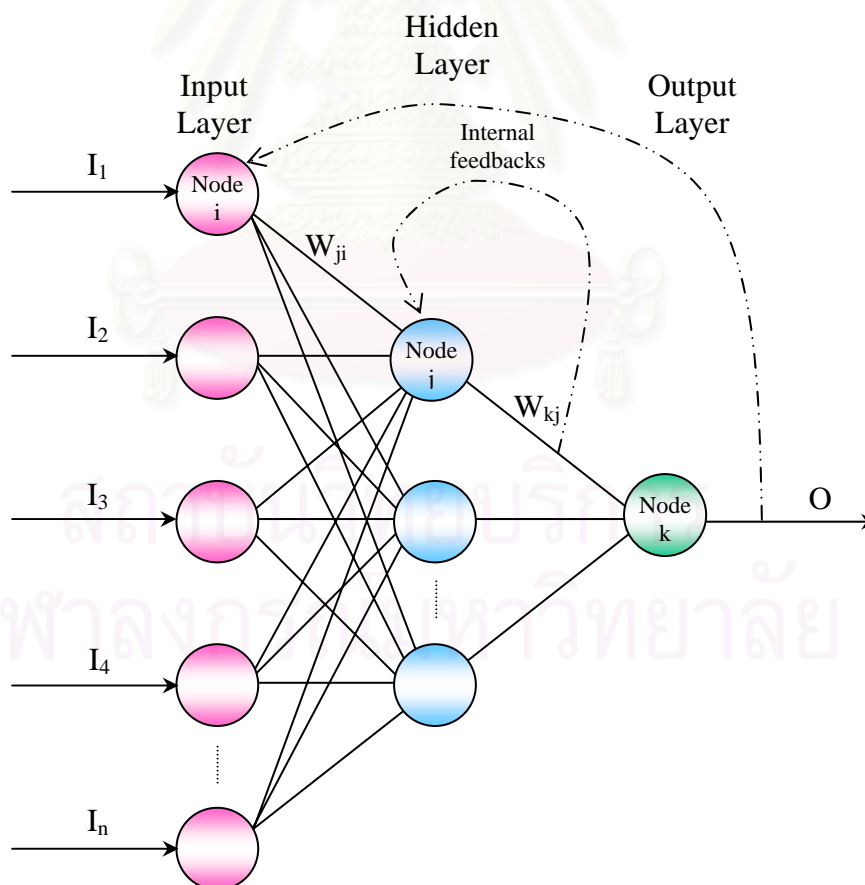


Figure 3.6 - Recurrent neural networks.

3.3 Neural Network Training

Training the network refers to changing its weights and biases, which are altered adaptively to minimize the error between the actual targeted values and that of network output. The training session is carried out repeatedly to minimize this error using various schemes. The training of the neural network is generally into three major categories, which will be described later. Neural network training are sometimes referred to as machine learning algorithms, because changing its connection weights causes the network to learn solutions to a problem. The strength of a connection between the neurons is stored as a weight value for the specific connection. The system stores new knowledge by adjusting these connecting weights. The learning ability of a neural network is determined by its architecture and by the algorithmic method chosen for training. The three general schemes to train the neural network are as follows.

3.3.1 Supervised training

A supervised training is a training algorithm for creating a function from training data, which has been obtained from actual process behavior. The network is trained to map the inputs to the corresponding correct output. As the inputs are applied to the network, the networks are compared to the targets. The training algorithms then used the error between the output and target to adjust the weights and biases of the network in order to move the network outputs closer to the targets. In this method, the trainer knows exactly the desired output. The error between the actual and desired output is used to modify the strengths of the connections i.e. weights between neurons and the training performed until it reaches the required performance. This method of training is used in most applications.

3.3.2 Unsupervised training

In unsupervised training, weights and biases are modified in response to network inputs only. There are no target outputs available. Rather, provision is made for a task- independent measure of the quality of representation that the network is required to learn, and the free parameters of the network are optimized with respect to

that measure. Once the network has become tuned to the statistical regularities of the input data, it develops the ability to form internal representations for encoding features of the input and thereby to create new classes automatically. This method is also used as the pre-processing for the supervised training method to improve training for better convergence. The hidden neurons must find ways to organize themselves. In this approach, no sample outputs are provided to the network against which it can be compared to the predicted performance for a given input vector.

3.3.3 Reinforcement training

This method requires reinforcements from the outside. The connections between the neurons in the hidden layer are randomly arranged and reshuffled as the network approaches its solution in solving the problem. Reinforcement learning is also a form of supervised learning, since it requires a trainer. It may be a training set of data or an observer who grades the performance of the network results. Both unsupervised and reinforcement suffers from relative slowness and inefficiency due to its reliance on random shuffling to determine proper connection weights. Reinforcement learning is closely related to dynamic programming, which was developed by Bellman (1957) in the context of optimal control theory. Dynamic programming provides the mathematical formalism for sequential decision making. By casting reinforcement learning within the framework of dynamic programming, the subject matter becomes all the richer for it, as demonstrated in Bertsekas and Tsitsiklis (1996).

3.4 Feedforward Multilayer Perceptron

There are many different types of neural networks and this work is confined to feedforward ANNs, the multilayer perceptron (MLP), which is employed as nonlinear process models.

Standard multilayer perceptrons (MLP) are a large class of feedforward neural networks with neurons arranged in layers. Generally, all neurons in a layer are connected to all neurons in the adjacent layers through uni-directional links. These

links are represented by synaptic weights. The synaptic weights act as signal multipliers on the corresponding links (interconnections). For example, in a three layer perceptron schematically depicted in figure 3.7 the neurons are grouped in sequentially connected layers; each layer is numbered 0, 1, 2 or 3. The neurons of layer 0 (sometimes called the input layer) do not perform computation, but only feed input signals to the neurons of layer 1 called the first hidden layer. The last layer (layer 3) is the output layer where the response of the network comes from. The neuron layers between the input pattern and the output layer are called hidden layers. Generally, there is no theoretical limit on the number of hidden layers, but usually in practice there will be only one or two hidden layers. It has been shown theoretically that it is sufficient to use a maximum of three layers (two hidden layers and one output layer) to solve an arbitrarily complex pattern classification problem (Rumelhart and McClelland, 1986). Each neuron is connected to all neurons of the two adjacent layers and to no other neurons. Note that connections within a layer or from higher to lower layers are not permitted. The arrows indicate the flow of information (signals). Generally, the multilayer perceptron has a different number of neurons and different synaptic weights for different layers. Each neuron of the MLP is characterized by one output and many inputs which are the outputs of the neurons in the preceding layer.

Let $u_j^{[s]}$ denote the value of the internal potential (signal) of the j -th neuron located in the s -th layer ($s = 1, 2, 3$) (Figure 3.7). The weighted sum of the inputs is computed by the neuron according to the formula

$$u_j^{[s]} = \sum_{i=0}^{n_{s-1}} w_{ji}^{[s]} o_i^{[s-1]} \quad (s = 1, 2, 3; j = 1, 2, \dots, n_s), \quad (3.6)$$

where $w_{ji}^{[s]}$ are the synaptic weights by which the j -th neuron multiplies the inputs $x_i^{[s]} = o_i^{[s-1]}$, $o_i^{[0]} = x_i$, $o_i^{[3]} = y_i$ and n_s is the number of neurons in the s -th layer.

The neuron output is computed by passing the weighted sum of its inputs (i.e. the internal potential $u_j^{[s]}$) by a nonlinear bounded activation function $\Psi_j^{[s]}$, i.e.

$$o_j^{[s]} = \Psi_j^{[s]}(u_j^{[s]}) = \Psi_j^{[s]} \left[\sum_{i=0}^{n_{s-1}} w_{ji}^{[s]} o_i^{[s-1]} \right] \quad (3.7)$$

For convenience and simplicity of consideration the bias term $\Theta_j^{[s]}$ is handled here (as usual) in a manner uniform with the synaptic weights $w_{j0}^{[s]} = \Theta_j^{[s]}$ by considering it as a weight connecting a neuron whose activation is always equal to unity. When processing data input signals are fed into the network by the input neurons and for each layer the outputs are computed successively and fed into the neurons of the next layer up to the output layer. Each layer of the network can be represented by the nonlinear matrix operator (Narendra and Parthasarthy, 1990.)

$$\mathbf{o}^{[s]} = \Psi^{[s]}[\mathbf{W}^{[s]}\mathbf{x}^{[s]}], \quad (3.8)$$

where

$\mathbf{o}^{[s]} = [o_1^{[s]}, o_2^{[s]}, \dots, o_{n_s}^{[s]}]^T$ is the output vector,

$\mathbf{x}^{[s]} = [x_1^{[s]}, x_2^{[s]}, \dots, x_{n_{s-1}}^{[s]}]^T$ is the input vector,

$\mathbf{x}^{[s]} = \mathbf{o}^{[s-1]}$,

$\mathbf{W}^{[s]} = [w_{ji}^{[s]}]_{(n_s+1) \times (n_{s-1}+1)} \in \mathfrak{R}^{(n_s+1) \times (n_{s-1}+1)}$ is the interconnection matrix of the synaptic weights in the s-th layer,

$\Psi^{[s]}[\cdot]$ is a diagonal nonlinear operator with (typically identical) sigmoid activation functions.

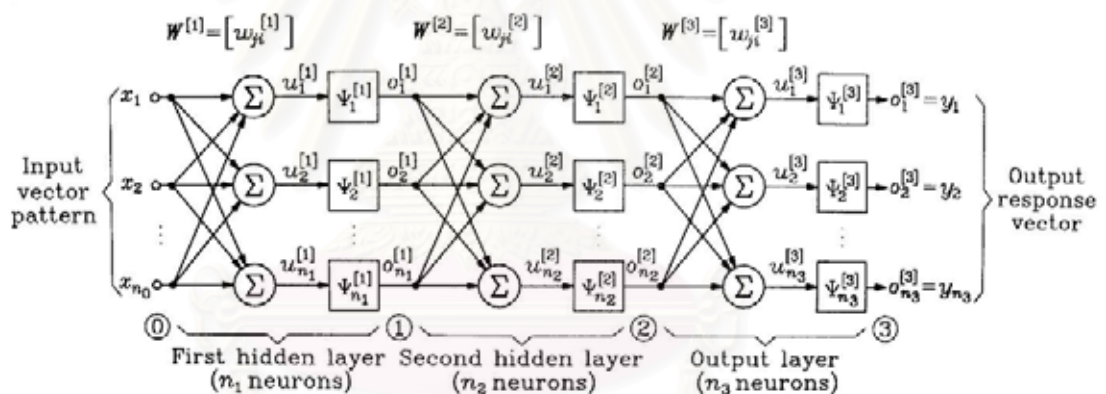
Hence, the input-output mapping of the three layer perceptron can be represented in the compact matrix form

$$\mathbf{y} = \Psi[\mathbf{x}] = \Psi^{[3]}[\mathbf{W}^{[3]}\Psi^{[2]}[\mathbf{W}^{[2]}\Psi^{[1]}[\mathbf{W}^{[1]}\mathbf{x}]]]$$

A functional block diagram representation of the above mapping is shown in figure 3.7(b)

The neurons usually take activity in the normalized range from +1 to -1 (in some applications the range from +1 to 0 is used). It should be noted that all synaptic weights are kept constant or fixed during the computation and their values determine the network behavior and its capability to correctly process (map) the input data or signals. In order to obtain the required network behavior the values of the synaptic weights must be properly computed. This means that the MLP must be trained. Such a computation is called the learning or training process. During the training process information is also propagated back through the network and it is used to update the synaptic weights successively first in the output layer, next in the second hidden layer and in the last step in the first hidden layer.

(a)



(b)

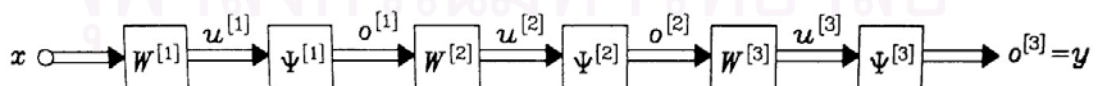


Figure 3.7- (a) Architecture of a three-layer perceptron

(b) A block diagram representation of the perceptron

3.5 Training Algorithm

The purpose of the training algorithm is to enable the ANN to represent a mapping which describes the I/O behavior of a non-linear system. To achieve this, the algorithm attempts to minimise an objective function by adjusting the ANN weight parameters. The objective function is a measure of how well the ANN fits a set of I/O training data patterns which the system has produced. Since in this work, we are concerned with the feedforward neural network, which utilizes the supervised training method, we will describe the training algorithm involved for training the feedforward in the next section.

3.5.1 The delta rule

In order to train a single layer network, one of the earliest supervised learning methods used is the delta rule. In this method, the inputs are presented to the network and the outputs are calculated. These outputs are then compared to the targeted values and the difference between them is calculated to give the error, δ , i.e.

$$\delta = \text{target} - \text{output} \quad (3.9)$$

The change of the weights is proportional to the previous calculated error, the input and a learning coefficient as given below,

$$\Delta i = \eta \delta x_i \quad (3.10)$$

$$w_i(t+1) = w_i(t) + \Delta i \quad (3.11)$$

where x_i is the inputs before the weight changes and η is the learning coefficient with value chosen randomly. In order to obtain better convergence during training, the learning rate coefficient is normally set between the values of 0 to 1. The disadvantage of this rule is that the output of the network must be known in order to calculate and adjust the weights. Due to this, the delta rule is only applicable

efficiently for a single layer network. As for the multi layer network, the backpropagation method is proposed and will be described in the next section.

3.5.2 The backpropagation method

The objective of this training method is to train the weights of a multilayer network in order to obtain the desired and the targeted outputs corresponding to a given set of inputs to the network. The methodology of the conventional backpropagation method is mentioned below (Hussain, 1994):

- Weights and biases are initialized with values between -1 and 1 randomly.
- Inputs are summed and propagated to the hidden layer for a node j as:

$$net_j = \sum_{i=1}^{N_i} W_{ji} O_i^1 + \theta_j \quad (3.12)$$

- Output from node j is given by

$$O_j^2 = f(net_j) \quad (3.13)$$

where f is the transfer function or activation function used in the hidden nodes

- Hidden layer output is propagated to node k at the output layer given as:

$$net_k = \sum_{j=1}^{N_j} W_{kj} O_j^2 + \theta_k \quad (3.14)$$

- Output from the node k is:

$$O_k^3 = f(net_k) \quad (3.15)$$

- Error is calculated at the output layer as:

$$e = \frac{1}{2} \sum_{k=1}^{N_k} (t_k - O_k^3)^2 \quad (3.16)$$

- Weights are adjusted along the negative gradient descent of the error, e as:

$$\Delta w_{kj} = -\eta \frac{\partial e}{\partial w_{kj}} \quad (3.17)$$

where η is the learning rate applied in the training

- Weights in the output and the hidden layers are then corrected using equations below:

$$\Delta w_{ji} = \eta O_j^2 (1 - O_j^2) O_i^1 \sum_{k=1}^{N_k} \delta_k^3 w_{kj} \quad \text{and} \quad (3.18)$$

$$\Delta w_{kj} = \eta (t_k - O_k^3) O_k^3 (1 - O_k^3) O_j^2$$

- A momentum term is then added to equation (3.18) to facilitate convergence and to avoid local minima to occur.

$$\Delta w(t) = \Delta w(t) + \alpha w(t-1) \quad (3.19)$$

where α is the momentum term

An advantage of the BP algorithm is its computational efficiency, since the number of computations per epoch of BP (an ANN is said to have been trained for one epoch when all the training vectors have been processed once) is proportional to $W \times N$, where W is the total number of network weights and N is the data length. On the other hand, BP is a gradient descent type algorithm which moves along the local negative gradient of the cost function and consequently can be slow to converge, particularly when the search encounters a local minimum, a saddle point or a long valley in the cost function surface. While the momentum term does assist the search

when the gradient information is poor, the algorithm is still relatively inefficient and its performance is dependent on the choice of suitable values for the learning gain and momentum.

Numerous algorithms have been proposed to circumvent the problems associated with BP and some of the main categories are outlined :-

- **Refining backpropagation.** Procedures for adjusting the learning gain and momentum terms during training to try to prevent the search from oscillating or becoming trapped in a local minima have been suggested (Battiti, 1989; Le Chun et al., 1993).
- **Conjugate gradient methods.** Gradient descent performs a fixed step length search in the direction of negative gradient and this is not efficient for reasons already discussed. The Conjugate gradient algorithm (Hestenes and Stiefel, 1952; Fletcher and Reeves, 1964) performs a series of searches in directions which are conjugate, or non-interfering, to each other. An outline of the algorithm is :-
 1. Initialise weights. Choose the initial search direction using gradient descent.
 2. Minimise the cost function in the new search direction. This can be done using a line search technique (Luenberger, 1984). Update the network weights.
 3. Stop if a termination criteria is satisfied.
 4. Evaluate a new search direction which is conjugate to all previous search directions. This means that the components of the new search direction that are parallel to all the previous search directions (which are zero since they already have been minimized) are kept fixed.
 5. Go to 2.

For a quadratic cost function the conjugate gradient algorithm will converge to the minimum in at most W steps, where the number of computations for each step, like BP, is in proportion to $N \times W$ (a step is akin to an epoch because this is a batch method). This is a significant improvement in the convergence rate over gradient descent methods with only a small (linear) increase in computational effort. However, for a neural network, the cost function will not generally be quadratic and this can result in degeneration of the conjugacy of the search directions. Hence, it is usually necessary to periodically reset the search direction according to some rule. The conjugate gradient method has been used to train MLP neural networks (Leonard and Kramer, 1990; Johansson et al., 1992; Charalambous, 1992).

- **Quasi-Newton methods** These methods make use of the second derivative, or Hessian, of the cost function. This curvature information gives a better perception of the cost function topology enabling a more efficient choice of search direction. The full Newton method, which is a batch update, can be computationally prohibitive since it necessitates the calculation of the Hessian and the inverse Hessian with the number of computations required per step in proportion to $N \times W^2$ and W^3 respectively. Quasi-Newton methods bypass this problem by building up an approximation to the Hessian inverse using a recursive algorithm which only uses information from the first derivative of the cost function.

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) is a widely used quasi-Newton procedure which has been shown to give superior convergence to gradient descent (Dennis and Schnabel, 1983). However, BFGS requires the storage and update of the approximated Hessian inverse matrix which is of size $W \times W$, and this can lead to prohibitive memory requirements when applied to large networks (Nahas et al., 1992). The limited memory BFGS algorithm considerably reduces the memory and computational requirements but at the cost of less efficient search direction estimates (Robitaille et al., 1996). Limited memory BFGS has been applied to training MLP neural networks (Batti, 1989; Irwin et al., 1994; Robitaille et al., 1996).

- **Stochastic search methods** These methods introduce a random element into the search for the weights to assist the search to escape from local minima in the cost function surface. The methods which have been applied to training neural networks include :-
 - ◆ Simulated annealing (Kilpatrick et al., 1983) was inspired from mathematical models of the energy state of annealing molten metals. The algorithm is controlled by a single parameter, termed temperature. As training progresses, the temperature is reduced slowly to steer the search towards the global minimum with a random excitation providing escape from local minima.
 - ◆ The chemotaxis algorithm (Bremermann and Anderson, 1989) is a similar idea where weights are perturbed by the addition of a random vector generated by a multivariate Gaussian probability density function. The standard deviation of the probability density function is reduced as training proceeds, in an analogous manner to the reduction of temperature in simulated annealing.
 - ◆ A genetic algorithm (Holland, 1975) is a non-linear optimisation technique based on evolutionary principles and has been applied to a variety of problems including the training of neural networks. The individual weights are represented by binary strings which are concatenated into one large string. Training is accomplished by combining subsets of a 'population' of such strings according to genetically inspired rules to form new strings. A new population is formed from some of the 'fittest' strings of the old population (fitness is related to how well the network is trained) and some new random strings. A possible advantage of genetic algorithms is that the algorithm combines elements of directed and random search methods (Brown and Harris, 1994).

In summary, several MLP ANN training algorithms have recently been proposed which do offer faster convergence than BP, but generally this comes at the cost of increased computational complexity and greater memory needs. Furthermore, none of the algorithms seem to overcome one of the most consequential pitfalls of BP which is the possibility of the search converging to a local, rather than the global,

minimum. Recently, alternative training algorithms, which converge faster, have been proposed and these are discussed in the next section.

3.5.3 Levenberg-Marquardt method

Levenberg-Marquardt method is a nonlinear least square optimization algorithm based on Newton's method (Marquardt, 1963). To minimize a function $V(x)$ with respect to the parameter vector x , then Newton's method would be:

$$\Delta x = -[\nabla^2 V(x)]^{-1} \nabla V(x) \quad (3.20)$$

Here, $\nabla^2 V(x)$ is the Hessian matrix and $\nabla V(x)$ is the gradient. Suppose that $V(x)$ is a sum of squares function,

$$V(x) = \sum_{i=1}^N e_i^2(x) \quad (3.21)$$

then it can shown that

$$\nabla V(x) = J^T(x)e(x) \quad (3.22)$$

$$\nabla^2 V(x) = J^T(x)J(x) + S(x) \quad (3.23)$$

Where $J(x)$ is the Jacobian matrix and

$$S(x) = \sum_{i=1}^N e_i(x) \nabla^2 e_i(x) \quad (3.24)$$

For the Gauss-Newton method, it is assumed that $S(x) \approx 0$, and the weight updates in equation (3.20) becomes,

$$\Delta x = [J^T(x)J(x)]^{-1} J^T(x)e(x) \quad (3.25)$$

The Levenberg-Marquardt modification to the Gauss-Newton method is,

$$\Delta x = [J^T(x)J(x) + \mu I]^{-1} J^T(x)e(x) \quad (3.26)$$

The parameter μ is multiplied by some factor β whenever a step would result in an increased $V(x)$. When a step reduces $V(x)$, μ is divided by β . Notice that when μ is large, the algorithm becomes steepest descent with step $1/\mu$, while for smaller μ , the algorithm becomes Gauss-Newton (Pivonka and Zizka, 1996). The Levenberg-Marquardt method interpolates between the approaches based on the maximum neighborhood and in which the truncated Taylor-series gives an adequate representation of the nonlinear model. The method has been found to be advantageous as compared to others. This method is used to train the neural network model in this work.

3.6 Model Predictive Control

The essence of MPC is to optimize, over the manipulated inputs, forecasts of process behavior. The forecasting is accomplished with a process model and therefore, the model is the essential element of an MPC controller. The models are not perfect forecasters, and feedback can overcome some effects of poor models, but starting with a poor process model is akin to driving a car at night without headlights; the feedback may be a bit late to be truly effective.

In general, model predictive control can be divided into two classes: linear model predictive control and nonlinear model predictive control. Linear MPC refers to a family of MPC schemes in which linear models are used to predict the system dynamics even though the dynamics of the system is nonlinear, while nonlinear MPC refers to the general cases in which the dynamic system models, performance objective, and constraints may be in nonlinear function of state, input and output variables.

- **Linear Models :** Historically, the models of choice in early industrial MPC applications were time domain, input/output, step, or impulse response models. Part of the early appeal of MPC for practitioners in the process industrials was undoubtedly the ease of understanding provided by this model form. It has become more common for MPC researchers, however, to discuss linear models in state-space form:

$$\frac{dx}{dt} = Ax + Bu \quad x_{j+1} = Ax_j + Bu_j$$

$$y = Cx \quad y_j = Cx_j$$

In which x is the n -vector of states, y is the p -vector of (measurable) outputs, u is the m -vector of (manipulable) inputs, and t is the continuous-time and j is the discrete-time sample number. Continuous-time models may be more familiar to those with a classical control background in transfer functions, but discrete-time models are very convenient for digital computer implementation. Linear models in the process industries are, by nature, empirical models and identified from input/output data. The idea model form for identification purposes is perhaps best left to the experts in identification theory, but a survey of that literature indicates no disadvantage to using state-space models inside the MPC controller.

The discussion of MPC in state-space form has several advantages, including easy generalization to multivariable systems, ease of analysis of closed-loop properties, and online computation. Furthermore, starting with this model form, the wealth of linear systems theory, the linear quadratic (LQ) regulator theory, Kalman filtering theory, internal model principle, etc., is immediately accessible of use in MPC.

- **Nonlinear Models :** The use of nonlinear models in MPC is motivated by the possibility of improving control by improving the quality of the forecasting. The fundamentals in any process control problem, conservation of mass, momentum and energy, considerations of phase equilibrium, relationships of

chemical kinetics and properties of final products, all introduce nonlinearity into the process description. Determining the setting in which the use of nonlinear models for forecasting delivers improved control performance is an open issue. For continuous process maintained at nominal operating conditions and subject to small disturbances, the potential improvement would appear small. For processes operated over large regions of the state space, semibatch reactors, frequent product grade changes, processes subject to large disturbances, for example, the advantages of nonlinear models appear larger. (Rawlings, 2000).

3.6.1 MPC algorithms

Several reviews and comparative studies of the main MPC algorithms have been published, for example by De Keyser, Van de Welde and Dumortier (1988), Garcia, Prett and Morari (1989), Kramer and Ubehauen (1991) and Qin and Badgwell (1996). Their application particular to the chemical process industry has been described by Eaton and Rawlings (1992). Industrial applications have also been described by Richalet (1993). There are several textbooks on MPC, for instance those by Prett and Garcia (1988), Bitmead, Gevers and Wertz (1990), Soeterboek (1992), Camacho and Bordons (1995), Camacho and Bordons (1999) and Maciejowski (2000). There is also a special feature on MPC recently published in the IEE Computing and Control Engineering Journal (Roberts, 1999).

The various techniques are differentiated by different types of model and performance function employed. Presented from an historical perspective, some of the main algorithms are:-

- Model Algorithmic Control, MAC, initially called Model Predictive Heuristic Control, MPHC, (Richalet et al., 1976). This uses an impulse response model, which is valid only for open-loop stable processes, and minimizes the variance of the error between the output and a reference trajectory computed as a first-order system.

- Dynamic Matrix Control, DMC, (Cutler and Ramaker, 1980). This is similar to MAC but uses a step response model instead of an impulse response model. This technique was originally applied in Shell Oil as early as 1973. The method was extended to include input and output constraint handling using quadratic programming to solve the constrained optimization problem, giving rise to Quadratic Dynamic Matrix Control, QDMC, (Morshedi and Haydel, 1983). The DMC algorithm can also be derived for a general discrete state-space model (Prett and Garcia, 1988).
- Extended Prediction Self Adaptive Control, EPSAC, (Keyser and Cuawenberghe, 1985) uses a discrete (z-transform) transfer function to model the process and a simple control law structure calculated analytically using a quadratic performance function assuming $u(t)$ stays constant from instant t . The process model also includes measurable disturbances.
- Generalised Predictive Control, GPC, (Clarke et al., 1987), using a quadratic performance function, with weighting of control effort, and a auto -regressive moving average with exogenous variables model (ARMAX). It also provides an analytic solution for the optimal control in the absence of constraints.

DMC and GPC are perhaps the most popular techniques. There are several extensions to GPC including techniques with guarantee stability through end-point equality constraints (Clarke and Scattolini, 1992; Mosca and Zhang, 1992), and by stabilising the process prior to the objective function optimization (Kouvaritakis et al., 1992). The GPC technique, using a method known as Constrained Stable Generalised Predictive Control, CSGPC, (Rossiter and Kouvaritakis, 1993) has also been extended to guarantee feasibility and stability when there are input constraints as well as terminal constraints (Rossiter et al., 1996; Rossiter et al., 1997). Although most of the work has been performed in discrete time GPC has also been formulated in continuous time (Demircioglu and Gawthrop; 1991, 1992). A state space model description for GPC controllers has also been developed (Ordys and Clarke, 1993;

Gawthrop et al., 1998). Nonlinear model predictive control based on state space models and the receding horizon concept has also been developed, for example by Mayne and Michalska (1990), who perform a stability analysis, and Balchen et al. (1992). Integration of economic objectives within the performance function has also been performed, (Becerra et al., 1998). Several commercial companies offer software products for implementation of model predictive control, for example MDC (SMOC), Predictive Control (CONNOISSEUR), AspenTech (DMCplus) and Honeywell (RMPCT).

3.6.2 Formulation of model predictive control problem

The problem to be solved by the model predictive controller may be stated as

$$\min_{u(t)} \Phi[u(t), x(t), y(t)] \quad (3.27)$$

subject to

$$\frac{dx}{dt} - f(x, u) = 0$$

$$y - g(x, u) = 0$$

$$h(x, u) = 0$$

$$k(x, u) \geq 0$$

$$x(t_0) = x_0$$

with

$$t \in [t_0, t_0 + P]$$

in which u is the input vector, y is the output vector, and x is the state vector. The time interval is from the current time, t_0 , to some finite time in the future, t_0+P , in which P

is the length of the prediction horizon. The scalar functional Φ is the controller's performance objective, the functions f and g determine the plant model, and h and k are equality and inequality constraint functions that may be specified as further performance objectives. The generality of the performance objective, as opposed to standard integral square error between output and set point, provides the opportunity to design the MPC controller for higher level functions such as energy or waste minimization.

3.6.3 Model predictive control strategy

Model Predictive Control, MPC, usually contains the following three ideas (Camacho and Bordons, 1999):-

1. Explicit use of a model to predict the process output along a future time horizon.
2. Calculation of a control sequence to optimize a performance index.
3. A receding horizon strategy, so that at each instant the horizon is moved towards the future, which involves the application of the first control signal of the sequence calculated at each step.

The strategy is illustrated as shown in figure 3.8 and described as follows (Camacho and Bordons, 1999):-

1. The predicted future outputs $\hat{y}(t+k|t)$, $k=1 \dots P$ for the prediction horizon P are calculated at each instant t using the process model. These depend upon the known values up to instance t (past inputs and outputs), including the current output (initial condition) $y(t)$ and on the future control signals $u(t+k|t)$, $k=0 \dots P-1$, to be calculated. (Note - the notation $x(t+k|t)$ indicates the value of x at time instant $t+k$ calculated at time instant t).

2. The sequence of future control signals is computed to optimize a performance criterion, often to minimise the error between a reference trajectory and the predicted process output. Usually the control effort is included in the performance criterion.
3. Only the current control signal $u(t/t)$ is transmitted to the process. At the next sampling instant $y(t+1)$ is measured and step 1 is repeated and all sequences brought up to date. Thus $u(t+1|t+1)$ is then calculated using the receding horizon concept.

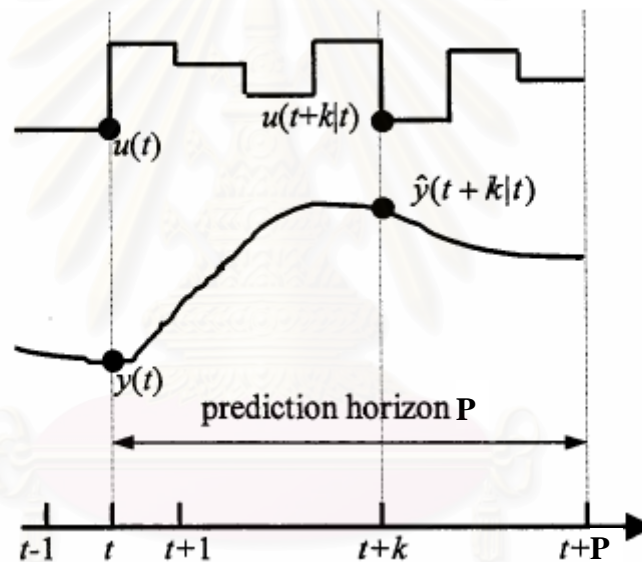


Figure 3.8 - MPC Strategy

3.6.4 Advantages and Disadvantages of MPC

Some of the main advantages are:-

- Concepts are intuitive and attractive to industry.
- Can be used to control a great variety of processes, including those with non-minimum phase, long time delay or open-loop unstable characteristics.

- Can deal with multivariable, multi-input multi-output as well as single-input single-output processes.
- Process constraints can readily be treated within the optimization process.
- Readily applicable to batch processes where the future reference signals are known.
- An open technology which allows for future extensions.

A significant disadvantage is:-

- Requirement of an appropriate model of the process.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER IV

NEURAL NETWORK MODELING AND INVERSE NEURAL NETWORK MODELING FOR A STEEL PICKLING PROCESS

4.1 A Steel Pickling Process

It has been known that many chemical industrial plants cause environmental problems due to the usage of chemicals in their production lines. One such industry is the steel pickling plant which is a fundamental industry in Thailand and has long existed and served the country's steel demand. The steel pickling process utilizes concentrated chemicals in the production lines and the wastewater released from the process contains hazardous materials and usually causes major environmental problems. Therefore, production scheduling and control of this pickling process are inevitably needed to minimize the amount of hazardous material contained in the released wastewater and also to maintain the concentration of acid solution in the tanks in order to obtain the maximum reaction rate at the same time.

The steel pickling process consists of two major steps: pickling and rinsing steps (Kittisupakorn and Kaewpradit, 2003). The purpose of the pickling step is to remove surface oxides (scales) and other contaminants out of metals by an immersion of the metals into an aqueous acid solution. Metals are immersed in pickling baths, containing 5, 10 and 15% by weight of hydrochloric acid (HCl), respectively, in order to remove the scales from the metals. The metals move counter current to the acid stream. The reaction occurring in the pickling baths is as follows:



Drag in-out pickling solution is removed from the metal surface using rinsing water during the rinsing step, which consists of three pure water baths. The metals move opposite to the rinsing water flow. Here, the amount of drag out solution of each bath is assumed to be equal to the amount of drag in solution. It should be noted that the

steel pickling process studied in this work as illustrated in figure 4.1 and 4.2 is one of several existing configurations.

The following assumptions are made for the purpose of this study.

- The system is supposed to be perfectly mixed and isothermal.
- All state variables are measurable directly.
- Density of liquid is assumed to be constant.
- The deterioration of pickling efficiency resulting from iron concentration is considered negligible.

Based on the above assumptions, the mathematical model of the continuous steel pickling process (Figure 4.1 and 4.2) for the change in volume and concentration can be derived for both the pickling and rinsing steps as follows.

- Pickling step (occurring in the 5, 10 and 15% HCl baths)

$$A \frac{dh_1}{dt} = F_2 - F_1 - q \quad (4.2)$$

$$A \frac{dh_2}{dt} = F_3 - F_2 - F_{11} \quad (4.3)$$

$$A \frac{dh_3}{dt} = F_4 + F_5 - F_3 - F_{10} \quad (4.4)$$

$$\frac{d(V_1 C_1)}{dt} = F_2 C_2 - C_1 (F_1 + q) - V_1 r_1 \quad (4.5)$$

$$\frac{d(V_2 C_2)}{dt} = q C_1 + F_3 C_3 - C_2 (F_2 + F_{11} + q) - V_2 r_2 \quad (4.6)$$

$$\frac{d(V_3 C_3)}{dt} = q C_2 + F_5 C_{20} + F_4 C_4 - C_3 (F_3 + F_{10} + q) - V_3 r_3 \quad (4.7)$$

- Rinsing step (occurring in three pure water baths)

$$A \frac{dh_4}{dt} = F_6 - F_4 - F_9 \quad (4.8)$$

$$A \frac{dh_5}{dt} = F_7 - F_6 \quad (4.9)$$

$$A \frac{dh_6}{dt} = F_8 - F_7 \quad (4.10)$$

$$\frac{d(V_4 C_4)}{dt} = qC_3 + F_6 C_5 - C_4(F_4 + F_9 + q) \quad (4.11)$$

$$\frac{d(V_5 C_5)}{dt} = qC_4 + F_7 C_6 - C_5(F_6 + q) \quad (4.12)$$

$$\frac{d(V_6 C_6)}{dt} = qC_5 + F_8 C_w - C_6(F_7 + q) \quad (4.13)$$

The meanings of all these variables are specified in the nomenclature. To complete the mathematical modeling of this continuous process, the expression of the reaction rate, equation (4.1), in the pickling baths needs to be imposed. The reaction is assumed to be first order neglecting liquid diffusion and the deterioration of pickling efficiency. Therefore, the rate of reaction studied here solely depends upon acid concentration as shown below:

$$r = kC \quad (4.14)$$

where k is the reaction rate constant.

The objective of this work is to control the concentration of HCl in all the pickling baths (C_1 , C_2 and C_3) and the pH (or H^+ concentration (C_4)) in the first rinsing bath to a desired set point by manipulating inlet flows F_2 , F_3 , F_5 and F_6 .

Since a neural network-based model is used for the control, we will first describe the procedure for neural network modeling and its use for control in the next section.

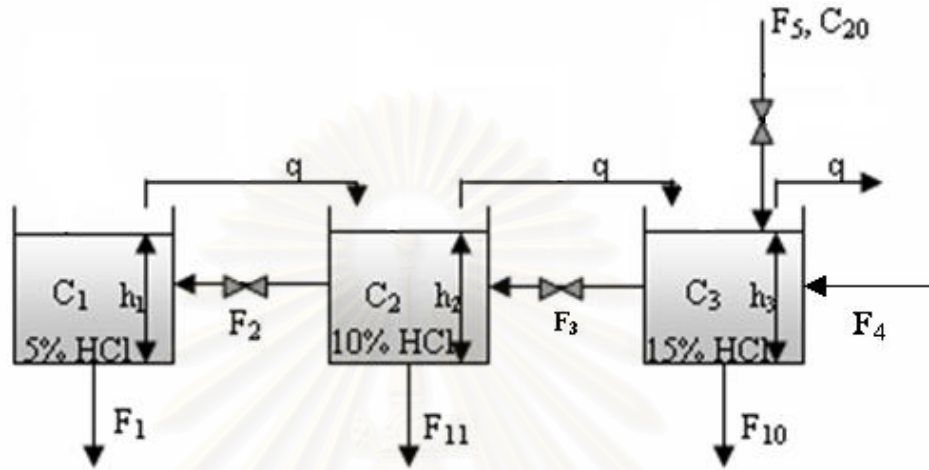


Figure 4.1 - Flow diagram of pickling baths

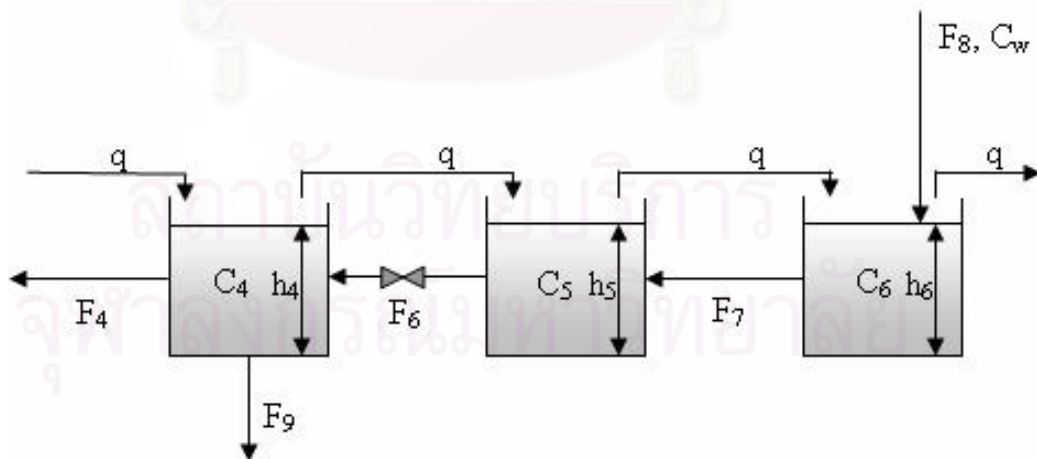


Figure 4.2 - Flow diagram of rinsing baths

4.2 Neural Network Modeling

Neural networks have the advantages of distributed information processing and the inherent potential for parallel computation. They can learn sufficiently accurate models and provide good nonlinear control when model equations are not known or only partial state information is available (Psichogios and Ungar, 1991; Hussain, 2003). Due to their parallel processing capability nonlinear nature and their ability to do without a priori knowledge neural networks can be used successfully to capture dynamic, nonlinear models of complex, multivariable systems. They therefore offer potential benefits in MPC strategies.

Although various types of neural network exist such as multi-layer perception (MLP), radial basis function (RBF) network and recurrent neural network (RNN), they consist of the same basic features: nodes, layers and connection. In this work, multi-layered feedforward network is used for the neural network since it is one of the most popular and successful neural network architectures suited to a wide range of applications in prediction, process modeling and control.

4.3 Procedure for Obtaining Neural Network Forward Models

Forward modeling refers to training the neural network model to predict the plant output, $C(k+1)$. The detailed procedures to find neural network models for the various baths are summarized in figure 4.3.

In the data preparing, training and validation data sets are obtained by selecting appropriate excitation signals (relating to operating condition range and manipulated variable constrains) from the simulation of the steel pickling process models by solving equation (4.2) to (4.13). These equations are solved to obtain the process states according to various changes in the manipulated variables, i.e. flow rates (F_2 , F_3 , F_5 and F_6). Data are selected to define the input and the output to the neural networks. The inputs are relevant data that are used in training to map out the defined output of the network. The training data sets of three pickling baths and the

first rinsing bath are shown in figure 4.4 to 4.7. Mathematically, four neural network models and are expressed as the function of inputs to the model as shown below:

- The pickling Baths
 - 5% HCl Bath :

$$C1(k+1) = f(F2(k-1), F2(k), C2(k-1), C2(k), C1(k))$$
 - 10% HCl Bath :

$$C2(k+1) = f(F2(k-1), F2(k), F3(k-1), F3(k), C1(k-1), C1(k), C3(k-1), C3(k), C2(k))$$
 - 15% HCl Bath :

$$C3(k+1) = f(F3(k-1), F3(k), F5(k-1), F5(k), C2(k-1), C2(k), C3(k))$$
- Rinsing baths
 - 1st Rinsing bath:

$$C4(k+1) = f(F6(k-1), F6(k), C3(k-1), C3(k), C5(k-1), C5(k), C4(k))$$

The inputs of neural network are the past and present values of the variables which effect to the state variable in each bath. The data sets need to be scaled in order to overcome the significant minimum and maximum values used in the training process. Raw process data generated earlier are scaled down to between 0.05-0.95 using the following equations:

$$ValueSD = \left[\frac{(ValueAC - \min value)(0.95 - 0.05)}{(\max value - \min value)} \right] + 0.05 \quad (4.15)$$

The actual value is given by

$$ValueAC = \left[\frac{(ValueSD - 0.05)(\max value - \min value)}{(0.95 - 0.05)} \right] + \min value \quad (4.16)$$

where $ValueSD$ is the scaled down value and $ValueAC$ is the actual value.

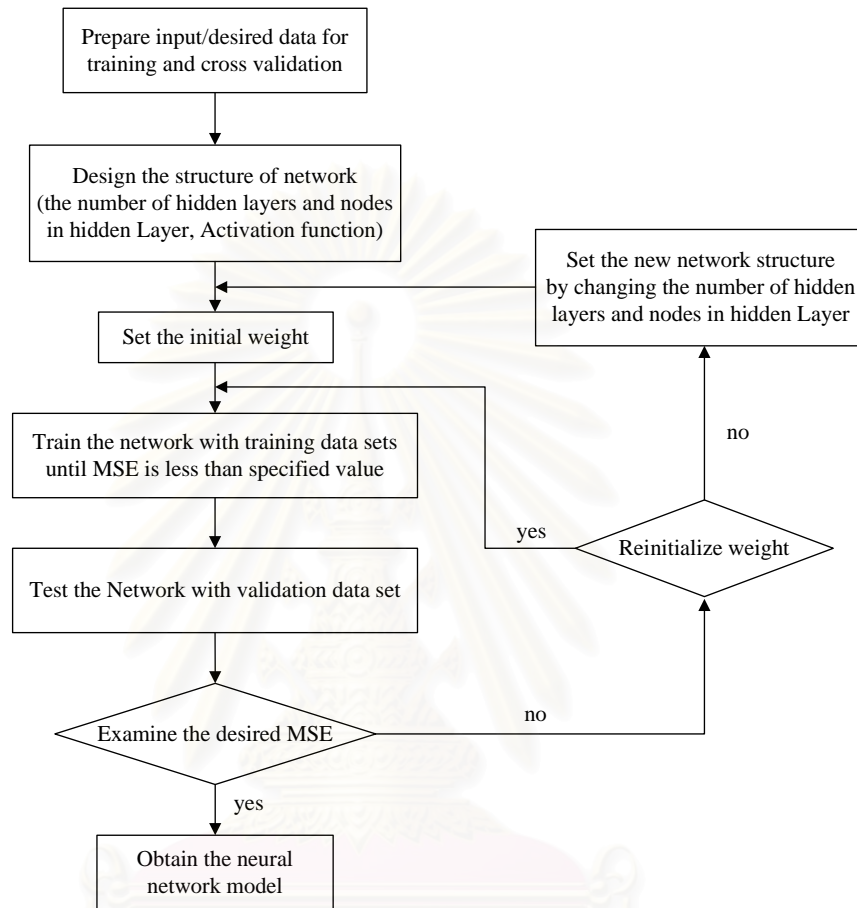
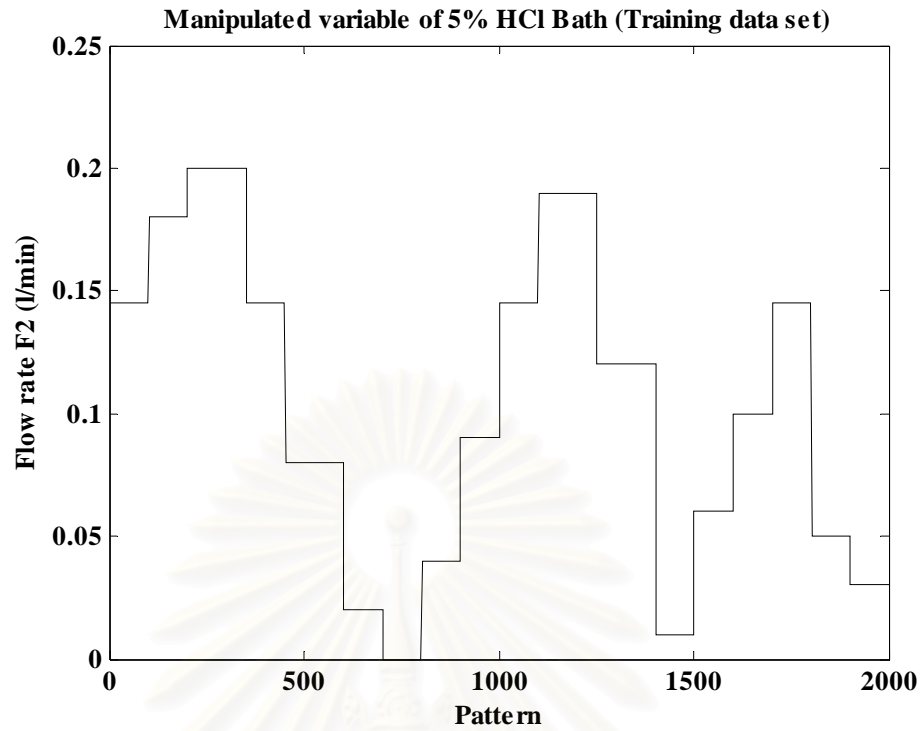
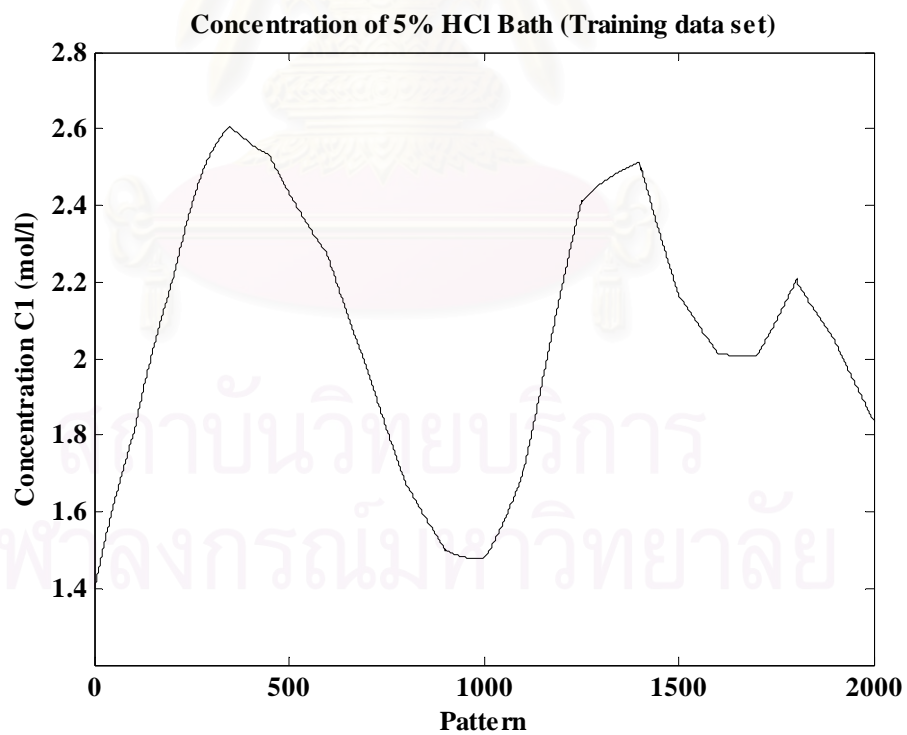


Figure 4.3 - Procedure for obtaining forward and inverse neural network models.

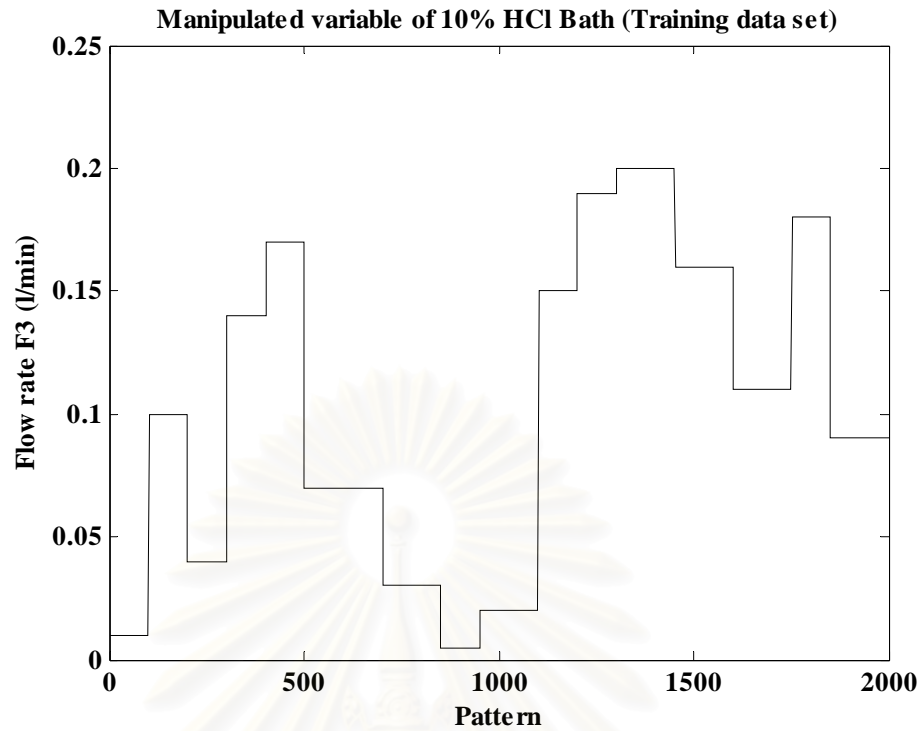


(a)

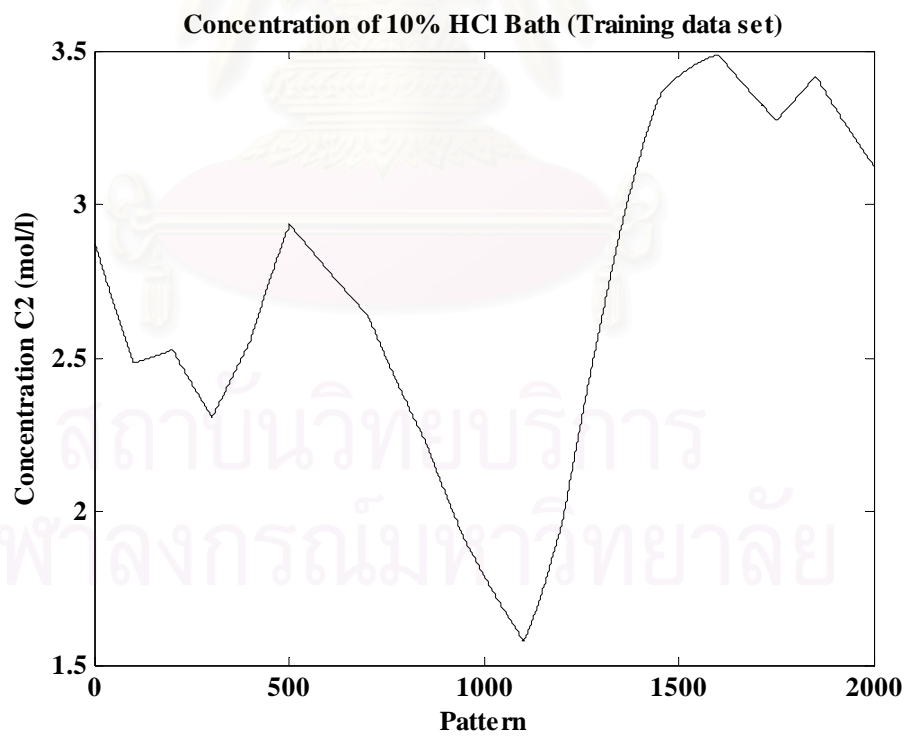


(b)

Figure 4.4 - Training data set of 5% HCl bath (a) Manipulated variable (Flow rate F2)
(b) Concentration of 5% HCl bath (C1)

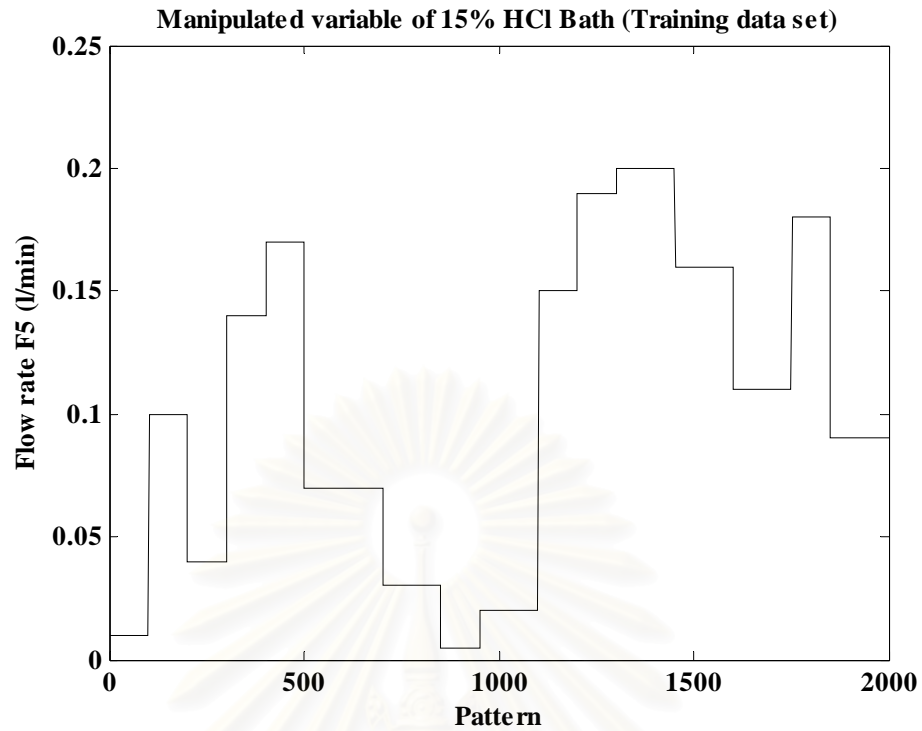


(a)

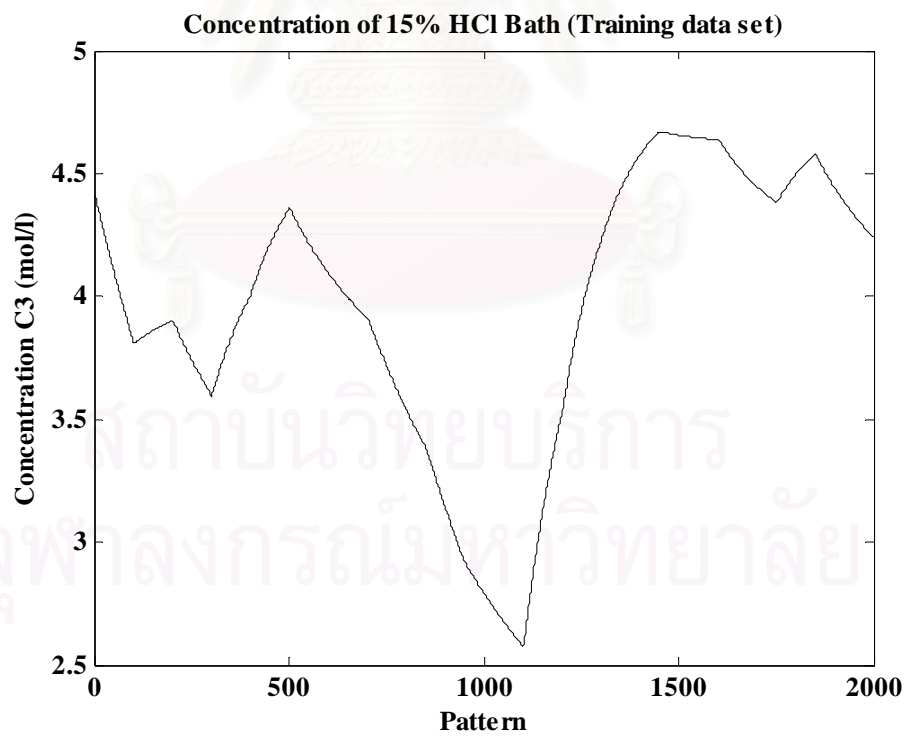


(b)

Figure 4.5 - Training data set of 10% HCl bath (a) Manipulated variable (Flow rate F3) (b) Concentration of 10% HCl bath (C2)

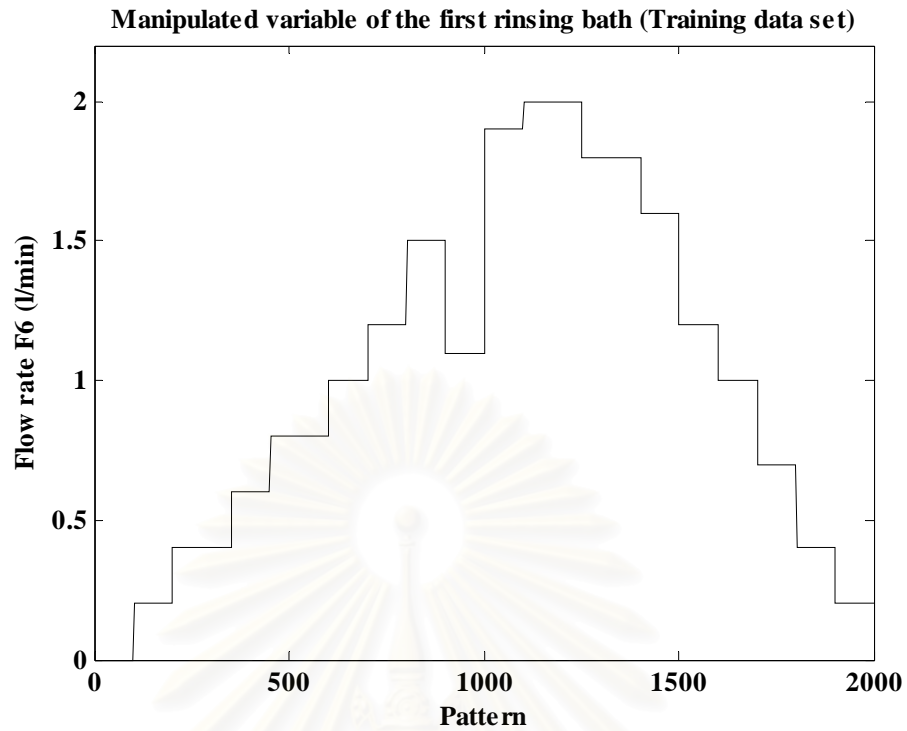


(a)

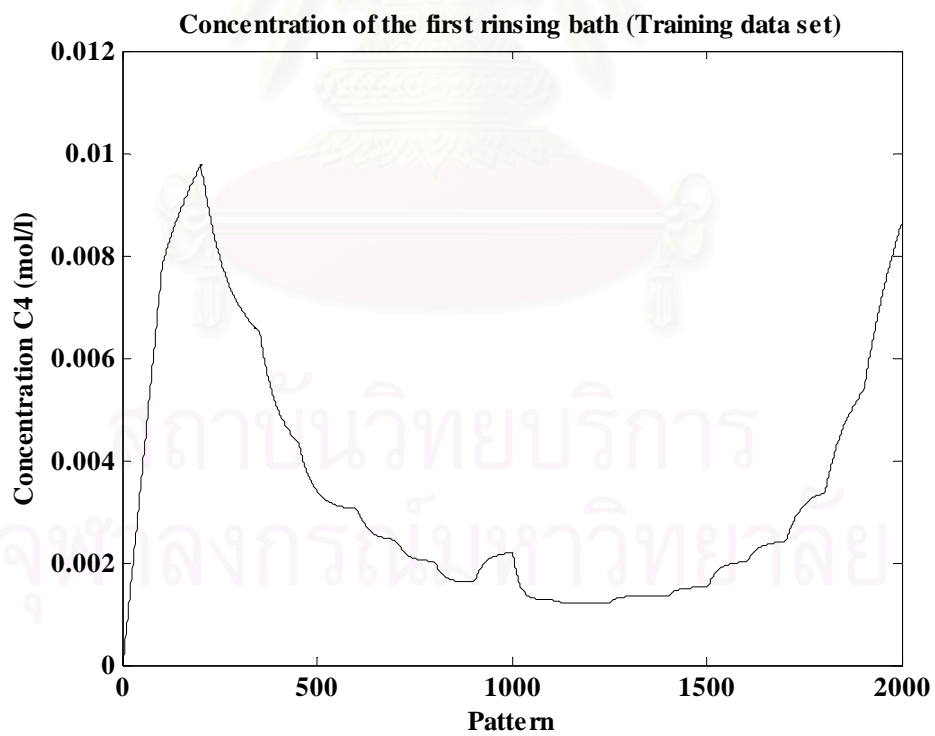


(b)

Figure 4.6 - Training data set of 15% HCl bath (a) Manipulated variable (Flow rate F5) (b) Concentration of 15% HCl bath (C3)



(a)



(b)

Figure 4.7 - Training data set of the first rinsing bath (a) Manipulated variable (Flow rate F6) (b) Concentration of the first rinsing bath (C4)

For neural network design, suitable neural network structure or configuration needs to be selected. The important aspects to consider are the number of hidden nodes, layers and transfer function used in the neural network. In this work, we use the sigmoidal function as the activation function of the nodes in the hidden layer and linear function neurons in its output layer. The defined neural networks are trained with the Levenberg-Marquardt algorithm in the Matlab Neural Network Toolbox where the common objective is to reduce the error between the neural network predicted value and the actual targeted value. Structure for the training of the forward neural network model is shown in figure 4.8. The training stops when the desired mean squared error (MSE) reaches the specified value of 0.001. The MSE is expressed mathematically below:

$$MSE = \frac{1}{n} \sum_{k=1}^n (F_{tg}(k) - F_N(k))^2 \quad (4.17)$$

where n is the number of data, F_{tg} is the target/desired flow value and F_N is the neural network output.

After training, the trained neural networks are validated by validation data sets. If the validation routine is not satisfactory, the neural network is not properly trained and requires more training. This can be done by re-initializing the weights and biases and to re-train the neural network for the next loop. Reconfiguring the neural network architecture can also help to increase the quality of the neural network simply by increasing or decreasing the number of hidden nodes.

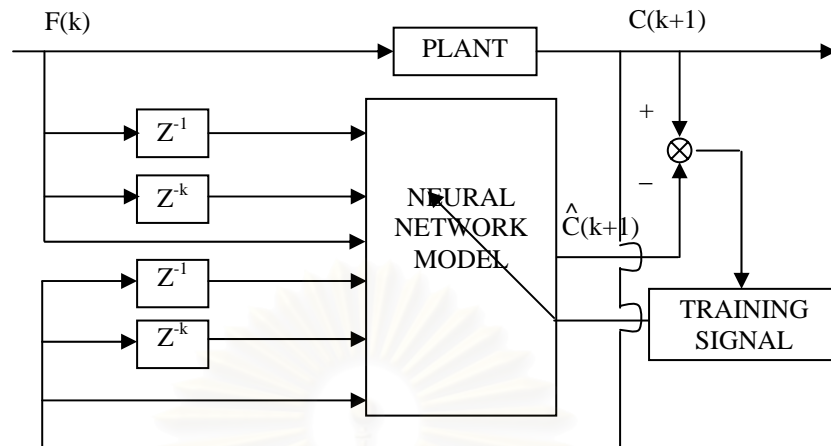


Figure 4.8 - Structure for the training of the forward neural network model.

4.4 Identification of Neural Network Inverse Models

There are several ways to carry out this identification process of the neural network inverse models. The technique used in this work is known as the generalized inverse learning method. Here, the network is fed with the required future or reference output together with the past inputs and the past output variables to predict the current input or control action. The trained network represents the inverse model of the system. The assignment of the input nodes consists of the past and present values of the known flows and concentrations associated with the individual tanks and the desired value of the plant output, $C(k + 1)$, which corresponds to the required set point or reference signal. The output node of the neural network model consists of the manipulated variable for the tank, i.e. flow entering the associated tank. Although various prediction horizons can be used for inverse models, this study concentrates on a simple one-step ahead horizon which assumes that there is no additional time delay between the control action and the output. The trained inverse model is then utilized as the controller in the direct inverse control method, which will be described later.

4.5 Procedure for Obtaining Neural Network Inverse Models

The detailed procedures to find reliable inverse neural network models for the various tanks are summarized in figure 4.3 which has the same steps as the procedures of the forward models but different in some details. The procedures to obtain inverse neural network models of the steel pickling process are described below:

1. Preparing training and validation data sets by selecting appropriate excitation signals from the simulation of the steel pickling process models by solving equation (4.2) to (4.13).
2. Selecting data to define the input and the output to the neural networks. Mathematically, four inverse models are expressed as the function of inputs to the model as shown below:

- Pickling baths

- 5% HCl bath:

$$F2(k) = f^{-1}(F2(k-1), C2(k-1), C2(k), C1(k), C1(k+1))$$

- 10% HCl bath:

$$F3(k) = f^{-1}(F2(k-1), F2(k), F3(k-1), C1(k-1), C1(k), C3(k-1), C3(k), C2(k), C2(k+1))$$

- 15% HCl bath:

$$F5(k) = f^{-1}(F3(k-1), F3(k), F5(k-1), C2(k-1), C2(k), C3(k), C3(k+1))$$

- Rinsing baths

- 1st Rinsing bath:

$$F6(k) = f^{-1}(F6(k-1), C3(k-1), C3(k), C5(k-1), C5(k), C4(k), C4(k+1))$$

3. Scaling the data sets in order to overcome the significant minimum and maximum values used in the training process. Raw process data are scaled down to between 0.05 and 0.95 using equation 4.15 and 4.16

4. Selecting the suitable neural network structure or configuration by considering the number of hidden nodes, layers and transfer function used in the neural network. For neural network inverse models, we use the sigmoidal function as the activation function of the nodes in the hidden layer and linear function in outer layer (only one hidden layer is used in all the networks).
5. Initializing the weights and biases prior to the network training.
6. Training the defined neural networks with the Levenberg–Marquardt method. The training stops when the desired mean squared error (MSE) reaches the specified value of 0.001. Structure for the training of the inverse neural network model is shown in figure 4.9.
7. The validation data sets that are not used during the training session are employed as the performance-monitoring element to achieve the specified validation error where the training is stopped while the final result will be a properly trained neural network.
8. If the validation routine is not satisfactory, the neural network is not properly trained and requires more training by re-initializing the weights and biases and to re-train the neural network for the next loop.

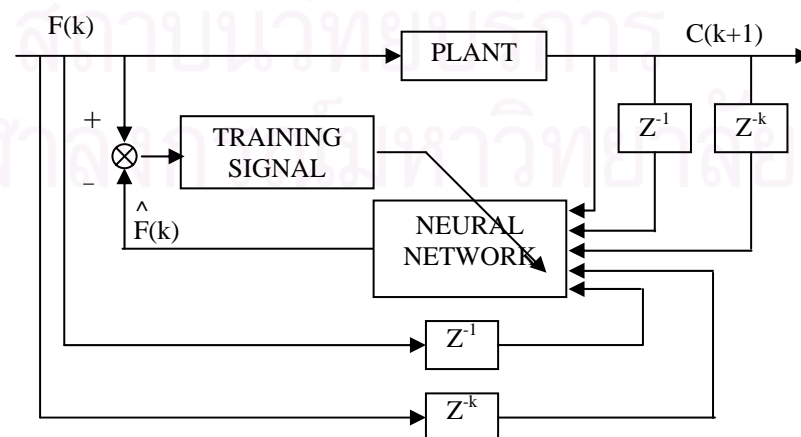


Figure 4.9 - Structure for the training of the inverse neural network model.

4.6 The Minimum MSE Method

In this work, the optimum structures are selected by the minimum MSE method. The hidden nodes are varied from 4 to 20 nodes. The MSE error is then monitored and the one that corresponds to the minimum MSE value is selected for determining the final number of hidden nodes. Table 4.1 shows the MSE values obtained from the neural network forward model and Table 4.2 shows the MSE values obtained from the neural network inverse model for the 5%, 10% and 15% HCl baths and the first rinsing bath using different number of hidden nodes. Based on the minimizing MSE error values, it is found that 4, 4, 8 and 4 hidden nodes appear to be the best to be applied respectively for the 5%, 10% and 15% HCl baths and the first rinsing bath forward models and 4, 8, 12 and 16 hidden nodes appear to be the best to be applied respectively for the 5%, 10% and 15% HCl baths and the first rinsing bath inverse models which will be used as controllers in the control strategy.

4.7 Simulation Results

After training process, the neural networks are validated by the validation data sets for the performance-monitoring. Figure 4.10 to 4.13 show the results of neural network model validation of each bath with optimal neural network structure (5-4-1, 9-4-1, 7-8-1 and 7-4-1 structures, respectively). Figure 4.14 to 4.17 show the results of neural network inverse model validation of each bath with optimal neural network structure (5-4-1, 9-8-1, 7-12-1 and 7-16-1 structures, respectively). The result in these figures indicated that the forward and inverse neural network models predict the concentration of HCl acid solution and flow rate in each bath identically to the validation data.

Table 4.1. MSE value for different number of hidden nodes in the neural network forward models of the 5%, 10% and 15% HCl baths and the first rinsing bath.

Bath	Numbers of hidden nodes	Mean Squared Error (MSE) after validation
5% HCl Bath	4	1.276 x 10⁻⁶
	8	1.919 x 10 ⁻⁵
	12	1.942 x 10 ⁻⁵
	16	1.758 x 10 ⁻⁵
	20	2.140 x 10 ⁻⁵
10% HCl Bath	4	8.069 x 10⁻⁶
	8	1.867 x 10 ⁻⁵
	12	2.194 x 10 ⁻⁵
	16	9.191 x 10 ⁻⁶
	20	5.525 x 10 ⁻⁴
15% HCl Bath	4	7.441 x 10 ⁻⁴
	8	2.437 x 10⁻⁵
	12	7.478 x 10 ⁻⁴
	16	8.654 x 10 ⁻⁴
	20	5.524 x 10 ⁻⁴
1 st Rinsing Bath	4	7.055 x 10⁻⁶
	8	1.406 x 10 ⁻⁵
	12	1.430 x 10 ⁻⁵
	16	1.208 x 10 ⁻⁵
	20	1.187 x 10 ⁻⁵

Table 4.2. MSE value for different number of hidden nodes in the neural network inverse models of the 5%, 10% and 15% HCl baths and the first rinsing bath.

Bath	Numbers of hidden nodes	Mean Squared Error (MSE) after validation
5% HCl Bath	4	7.792 x 10⁻⁴
	8	9.834 x 10 ⁻⁴
	12	9.584 x 10 ⁻⁴
	16	8.701 x 10 ⁻⁴
	20	1.700 x 10 ⁻³
10% HCl Bath	4	1.041 x 10 ⁻⁴
	8	2.381 x 10⁻⁵
	12	1.547 x 10 ⁻⁴
	16	1.904 x 10 ⁻⁴
	20	2.654 x 10 ⁻⁴
15% HCl Bath	4	1.268 x 10 ⁻⁴
	8	3.211 x 10 ⁻⁵
	12	2.088 x 10⁻⁵
	16	2.214 x 10 ⁻⁵
	20	1.082 x 10 ⁻⁴
1 st Rinsing Bath	4	1.208 x 10 ⁻⁵
	8	1.406 x 10 ⁻⁵
	12	1.430 x 10 ⁻⁵
	16	7.055 x 10⁻⁶
	20	1.187 x 10 ⁻⁵

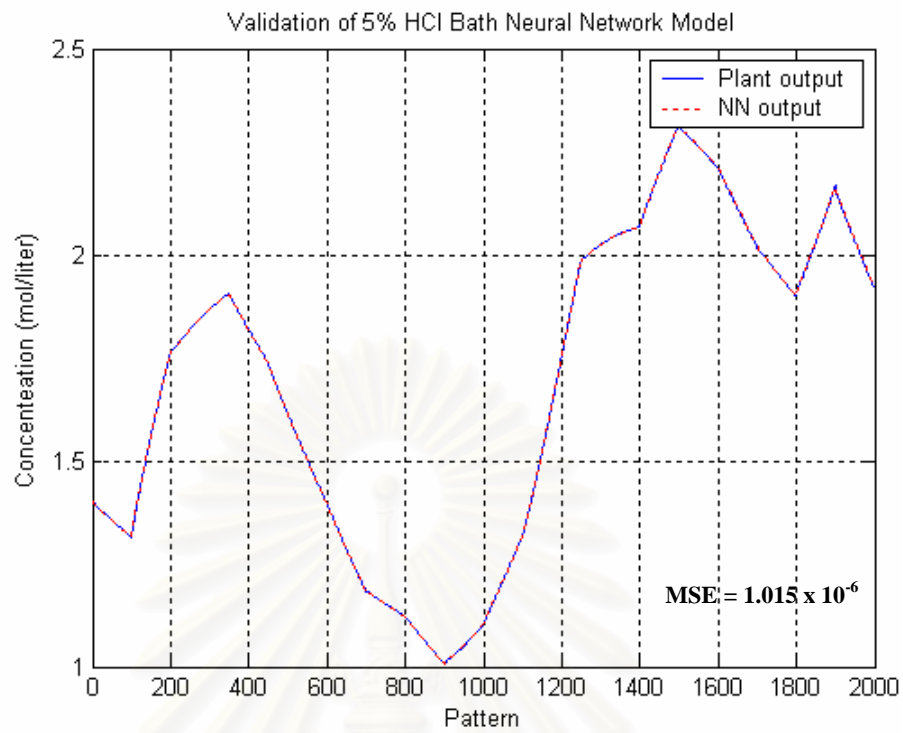


Figure 4.10 - The validation result of 5% HCl bath neural network model
(structure 5-4-1)

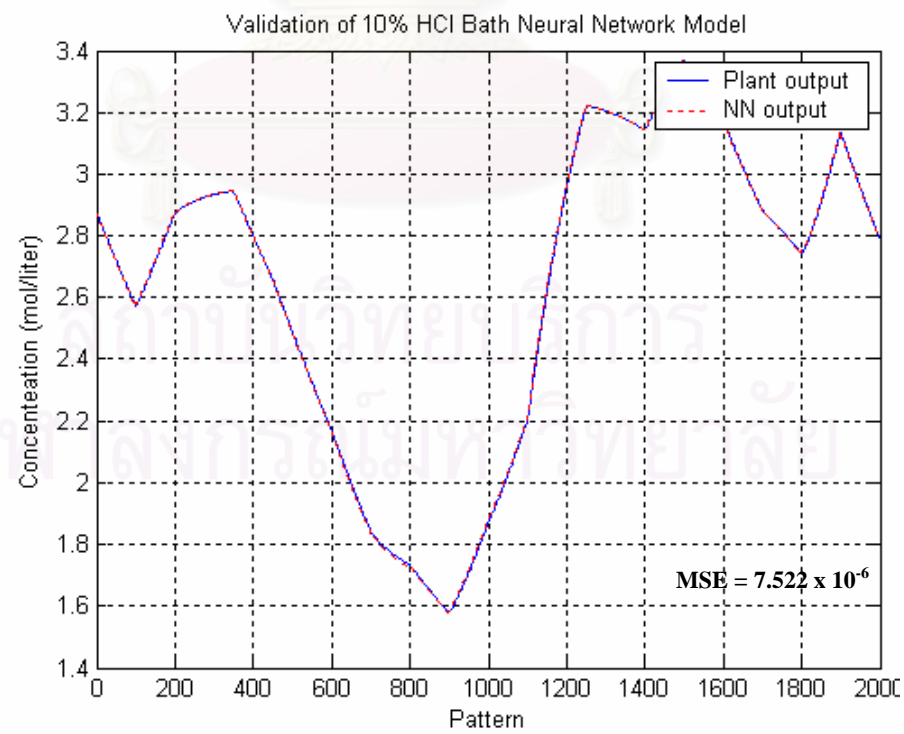


Figure 4.11 - The validation result of 10% HCl bath neural network model
(structure 9-4-1)

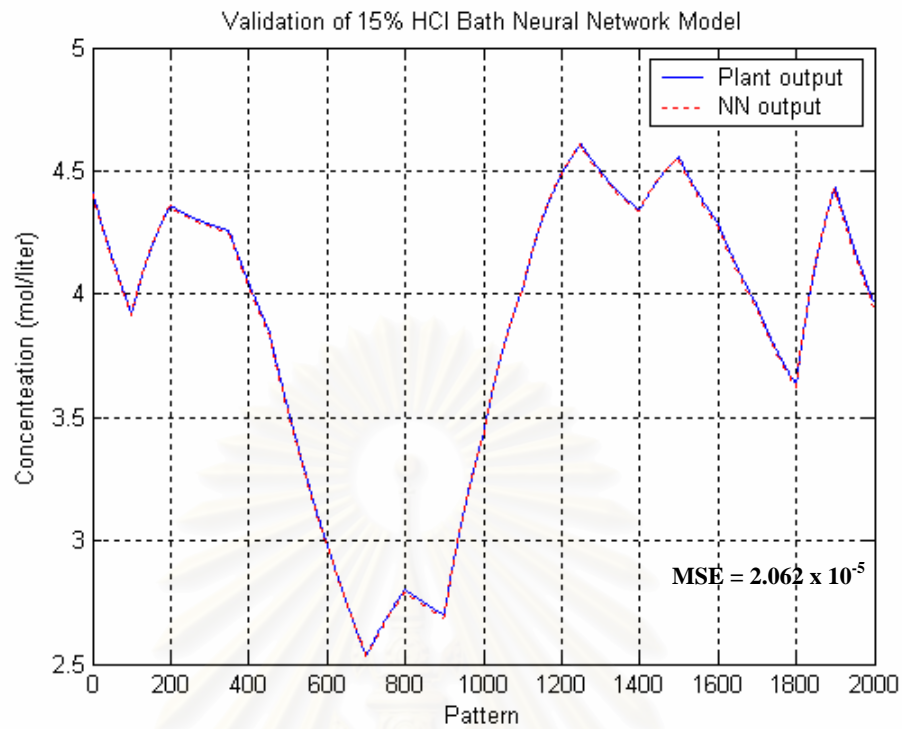


Figure 4.12 - The validation result of 15% HCl bath neural network model
(structure 7-8-1)

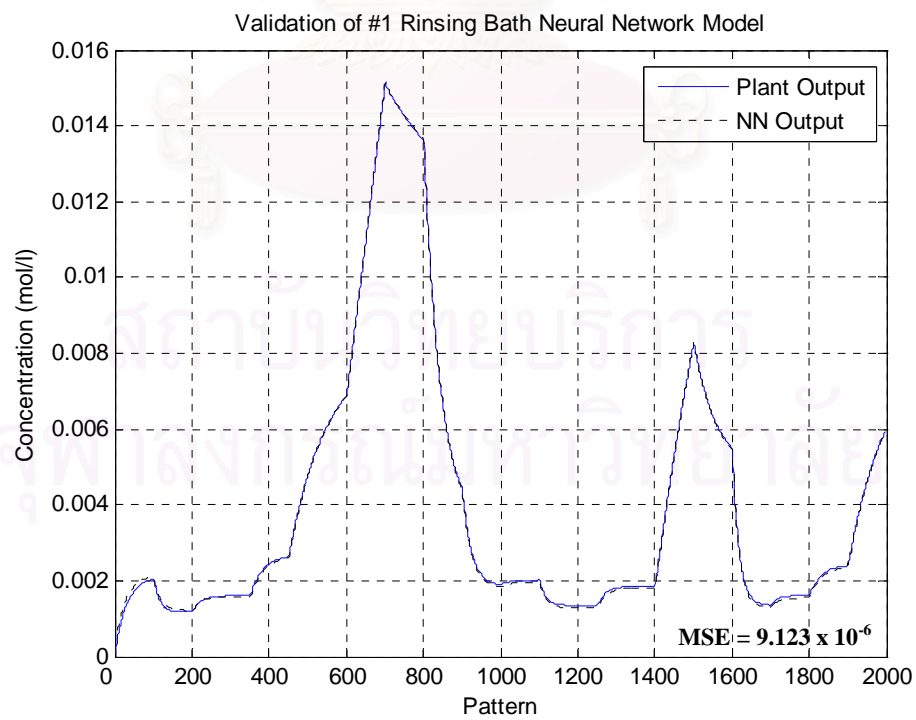


Figure 4.13 - The validation result of 1st rinsing bath neural network model
(structure 7-4-1)

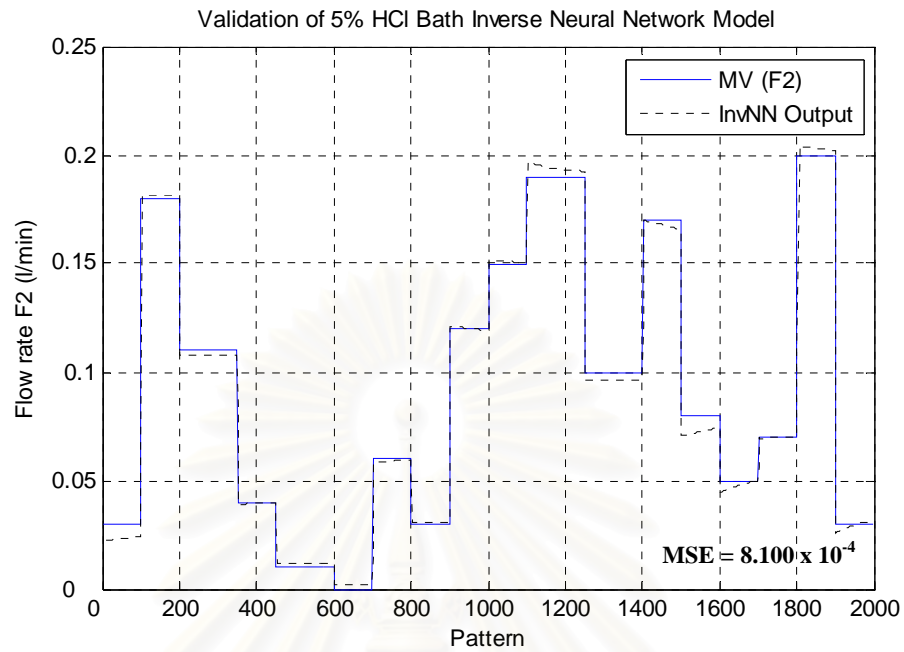


Figure 4.14 - The validation result of 5% HCl bath inverse neural network model (structure 5-4-1)

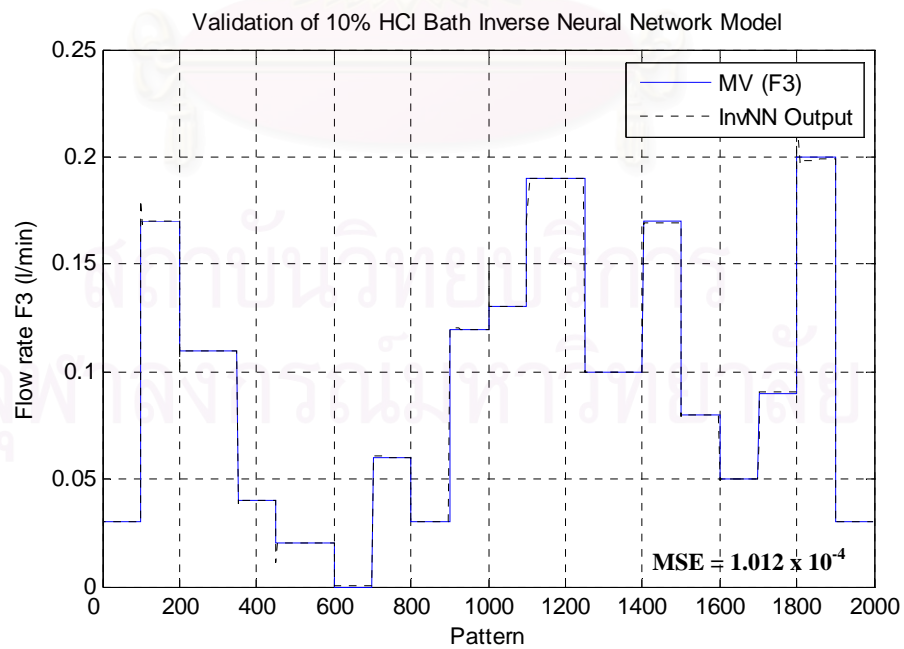


Figure 4.15 - The validation result of 10% HCl bath inverse neural network model (structure 9-8-1)

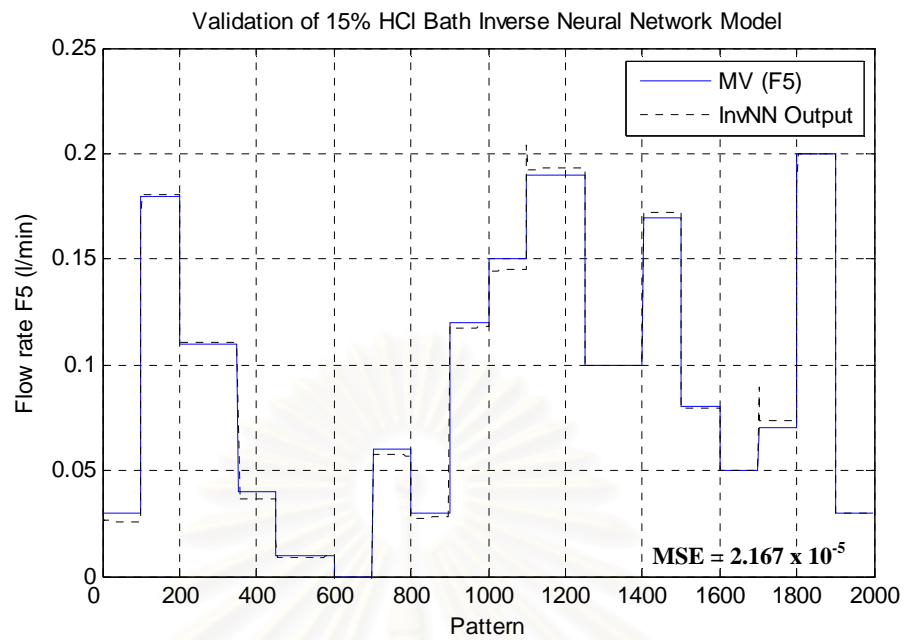


Figure 4.16 - The validation result of 15% HCl bath inverse neural network model (structure 7-12-1)

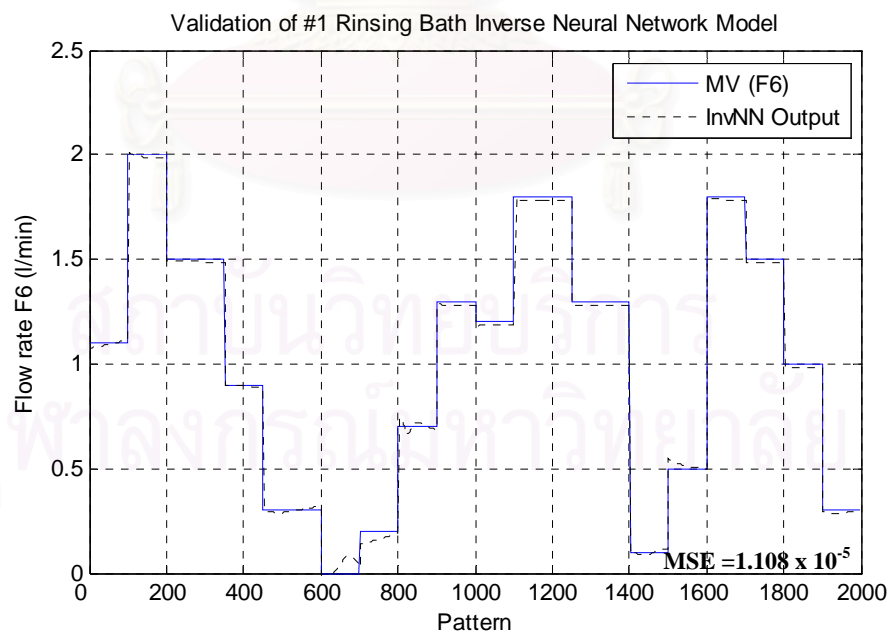


Figure 4.17 - The validation result of 1st rinsing bath inverse neural network model (structure 7-16-1)

CHAPTER V

MODEL PREDICTIVE CONTROL BASED ON NEURAL NETWORK FOR A STEEL PICKLING PROCESS

The model predictive control based on neural network (NNMPC) technique is investigated in this chapter for application to a steel pickling process which is commonly found in the steel industries of Thailand. The process involves removal of surface oxides (scales) and other contaminants out of metals by an immersion of the metals into an aqueous acid solution, which consists of three acid baths in series. Due to the highly nonlinear dynamic behavior, multivariable and interaction between baths cause this process to be difficult to control by conventional controllers. It is, therefore, the aim of this work to apply the neural network model predictive control strategy for controlling such a nonlinear system. To demonstrate the robustness and reliability of the proposed control strategy, tests involving the set point tracking under nominal condition and various disturbances including model mismatch and noise are performed in these studies.

5.1 Neural Network Model Based Predictive Control

As mentioned in Chapter 4, the neural network models of a steel pickling process are developed from data sets that obtained by selecting appropriate excitation signals from the simulation of the steel pickling process models by solving equation (4.2) to (4.13). The neural network models of a steel pickling process are shown in figure 5.1 to 5.3. These neural network models are then used in the control algorithm for controlling the process to the desired objective. Process control configuration of a steel pickling process is shown in figure 5.4.

The neural network MPC strategy developed in this work is shown in figure 5.5. In this approach, a neural network model is used to predict the future process response over the prediction horizon. The predictions are passed to a optimization

routine which attempts to minimize a specified objective function (Equation (5.1)) in searching an optimal control signal $F_j(k)$ at each sample instant. The objective function has a form of predictive control strategy as follows,

$$\min_{F_2, F_3, F_5} \sum_{i=1}^3 \sum_{l=1}^p \left\| \Gamma_l^{C_i} ([C_{spi}(k+l) - C_i(k+l|k)]) \right\|^2 + \sum_{j=2,3,5} \sum_{l=1}^m \left\| \Gamma_l^{F_j} [\Delta F_j(k+l-1)] \right\|^2 \quad (5.1)$$

Subject to

$$(F_j)_{\min} \leq F_j(K+l) \leq (F_j)_{\max}, \quad l=1, \dots, p$$

$$j=2, 3 \text{ and } 5$$

p is a parameter specifying the prediction horizon; C_i is the concentration in each bath, with the i th element specifying the parameter for the corresponding bath. $\Gamma_l^{C_i}$ is weighting parameter used to give different weights to different squared tracking errors. If all variables in equation (5.1) are in a similar range, then the choice of identity parameters may suffice. C_{spi} is the set point to introduce a feedback in order to compensate the system steady state error (Figure 5.5).

The simulations have been done using the neural network models to find a set of suitable control parameters, i.e. values for the parameters, p ; m (control horizon) and $\Gamma_l^{C_i}$. With NN MPC feedback, the control results indicate that longer prediction horizons tend to produce more aggressive control action, more overshoot and faster response. For control horizon (m), shortening the control horizon relative to the prediction horizon tends to produce less aggressive controllers and slower system response. According to this, the choice of p includes an equal number of future predictions of each output in objective function which, with this setting of p is 8 and the control horizon (m) is set as 4. $\Gamma_l^{C_i}$ is chosen as identity vector because the outputs of process are scaled before they are use in the network process model. Successive quadratic programming (SQP) is used to solve the multivariable optimization problem of minimizing objective function in equation (5.1) with respect to F_j and to produce a solution constrained within the process input operating ranges.

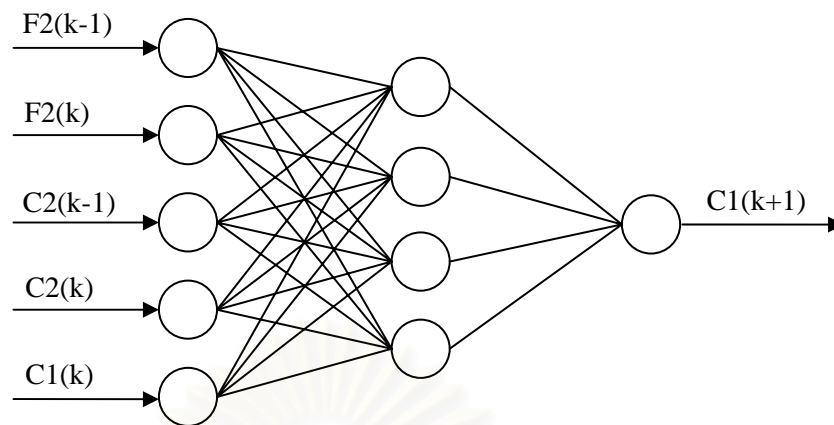


Figure 5.1 - The neural network model of 5% HCl bath (structure 5-4-1)

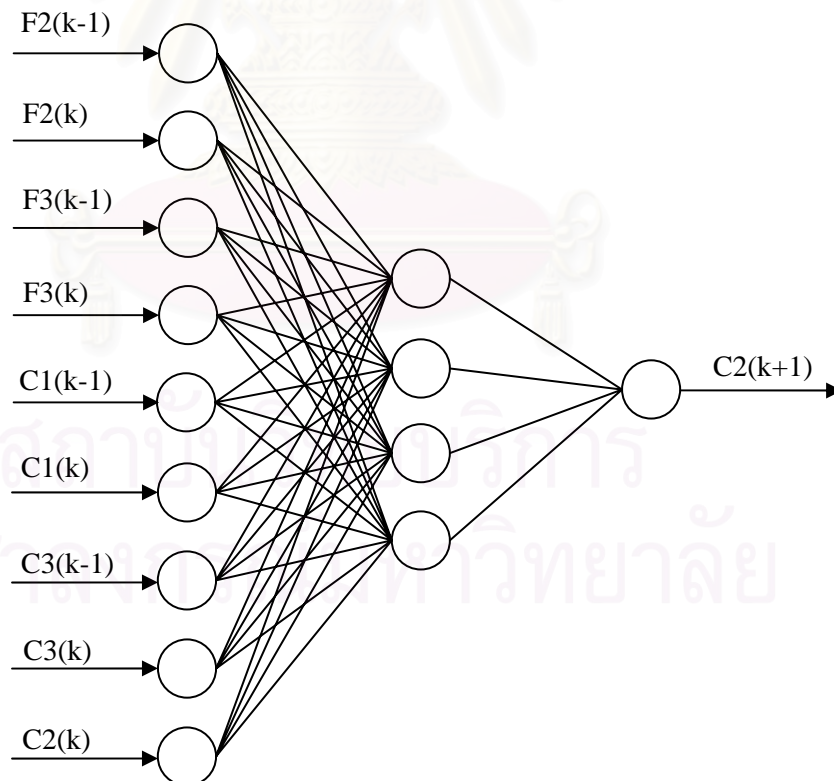


Figure 5.2 - The neural network model of 10% HCl bath (structure 9-4-1)

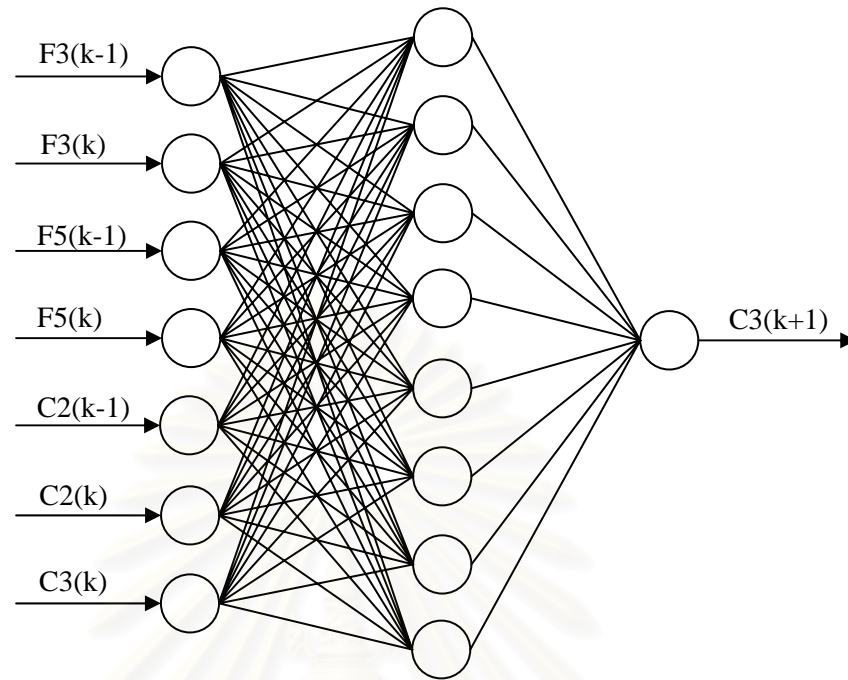


Figure 5.3 - The neural network model of 15% HCl bath (structure 7-8-1)

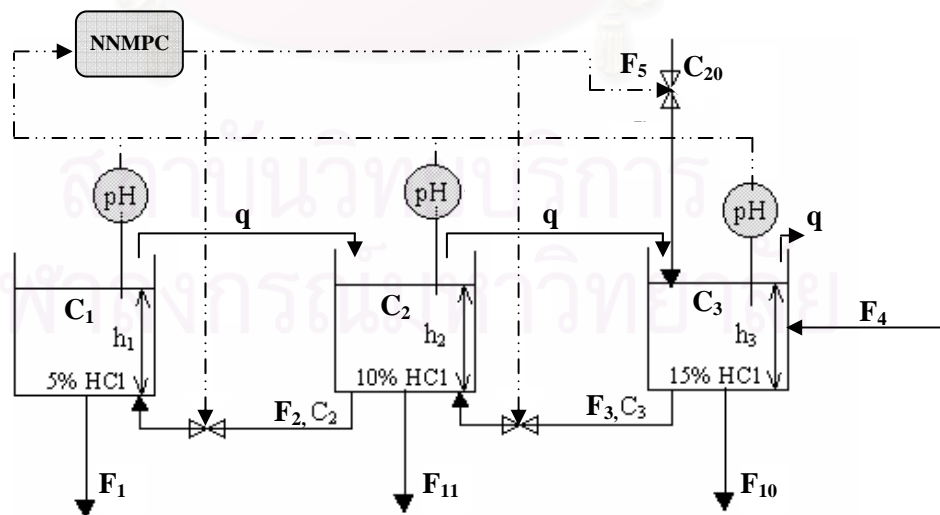


Figure 5.4 - Flow diagram of a steel pickling process control.

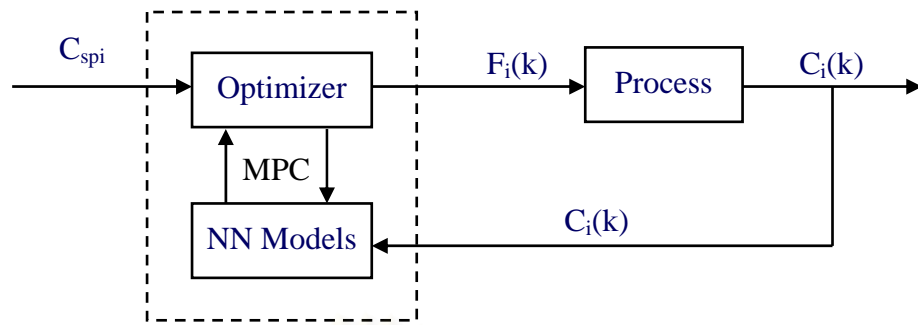


Figure 5.5 - Multivariable NN MPC strategy.

5.2 Simulation Results

The multivariable NN MPC strategy is applied to control the concentration of HCl in the 5% HCl, 10% HCl and 15% HCl bath to the values of 1.40, 2.87 and 4.41 (mol/l) by adjusting the manipulated variables F2, F3 and F5 respectively. They are divided into four cases of control studies, which are the set point tracking case, disturbance case, model mismatch case and noise case, respectively.

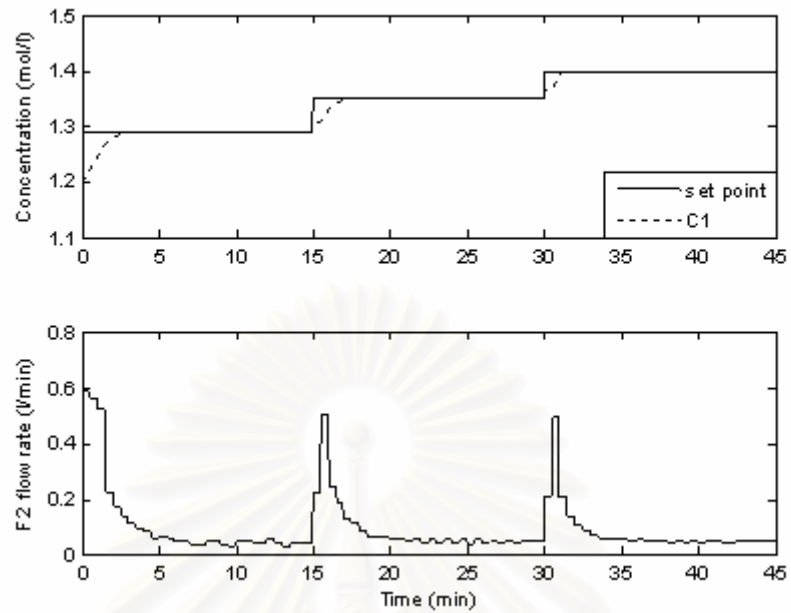
For set point tracking case, the controllers are designed to bring the concentration of HCl in each bath to the desired value. The initial set points are set at 1.29, 2.77 and 4.30 (mol/l) and changed to 1.35 and 1.40 (mol/l) for 5% HCl bath, 2.80 and 2.87 (mol/l) for 10% HCl bath and 4.35 and 4.41 (mol/l) for 15% HCl bath at 15 and 30 min, respectively. The control results of NN MPC in figure 5.6 show that, although there exists effects among input and output variables of acid baths, suitable control has been found to drive the process response to follow the set points without overshoot and oscillations. The satisfactory performance is due to the full representation of nonlinear dynamics of process by neural network models. For comparison, three PI controllers have designed for the four loops within the process. The controllers are designed using the Ziegler-Nichols closed loop method around one operating point and subsequent fine tuning. The maximum of manipulated flow rate is limited at value of 2 (l/min) as reference to pump flow rate limit. The control of HCl concentration in three baths using PI show a poor performance as display in

figure 5.7 because of the nonlinear dynamics of the baths. They show the overshoot of control variables and rigorous adjusting of manipulated variables.

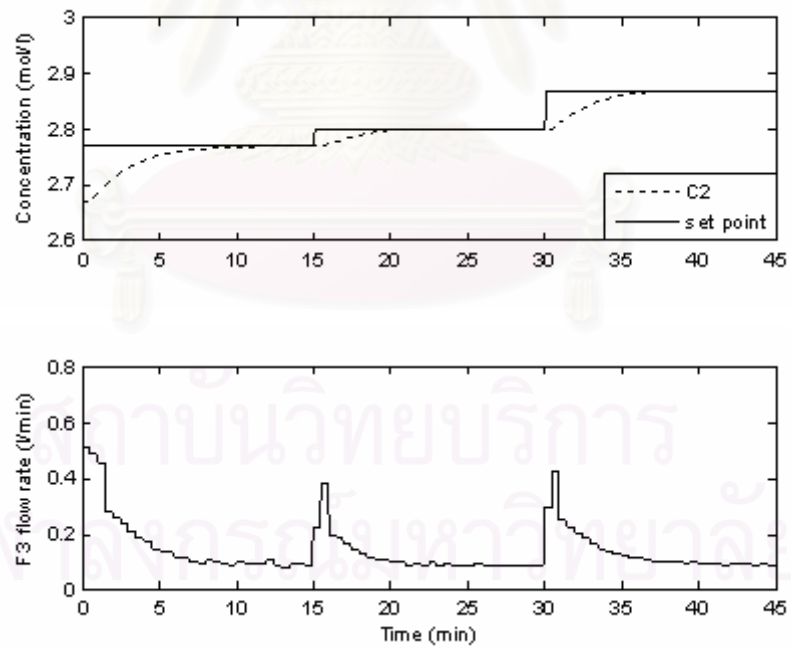
For disturbance case, which is the change in the concentration C_{20} in the stream F5, introduced by random increasing and reducing 15% from its nominal operation values. Figure 5.8 to 5.10 show the results of NNMPC and PI control for the 5% HCl, 10% HCl and 15% HCl bath respectively by random increasing and reducing 15% of C_{20} from its nominal value. It can be seen from figure 5.8 to 5.10 that the NNMPC strategy brought the concentrations to the required value by gradually increasing the flow rate of F2, F3 and F5 while PI control bring the concentrations to the set point by rigorous adjusting of the flow rates and cause the overshoot in process response. Table 5.1 summarizes the IAE values of NNMPC and PI control for the three baths. They indicate that NNMPC has more robustness and give better control performance than PI controllers with smaller IAE error values, when disturbances are present in the system.

Table 5.1 Performance comparison between NNMPC and PI control under the disturbance case.

Bath	IAE Values	
	NNMPC	PI
5% HCl Bath	0.223	0.308
10% HCl Bath	0.266	0.332
15% HCl Bath	0.220	0.421

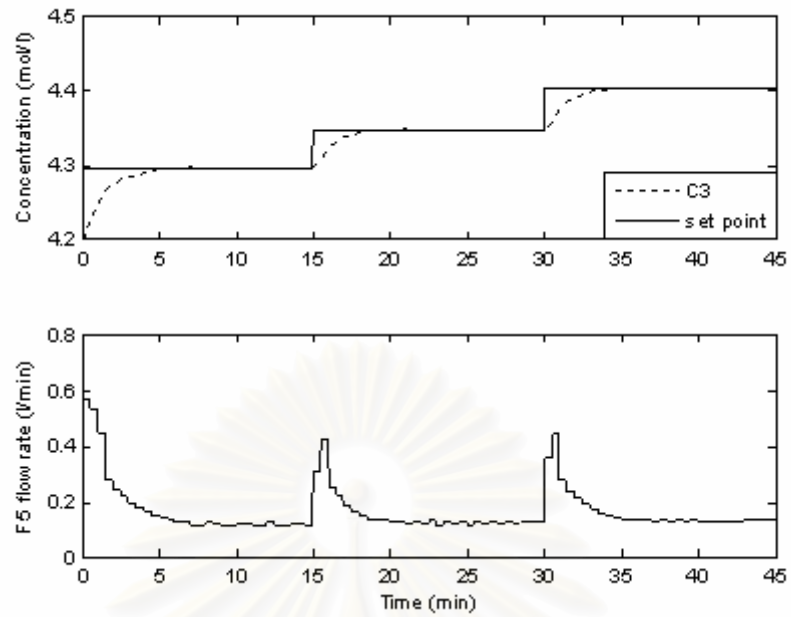


(a)



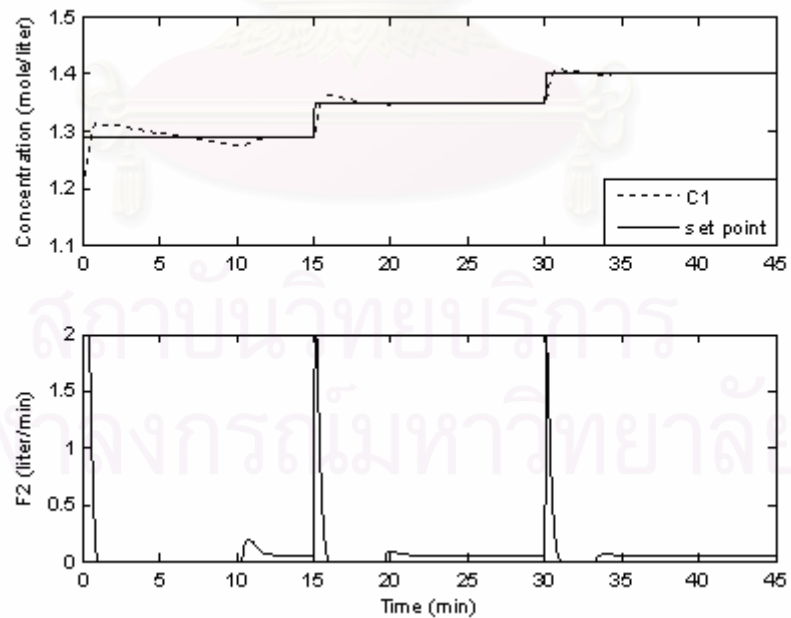
(b)

Figure 5.6 - NN MPC control for HCl acid concentration: (a) 5% HCl bath; (b) 10% HCl bath; and (c) 15% HCl bath



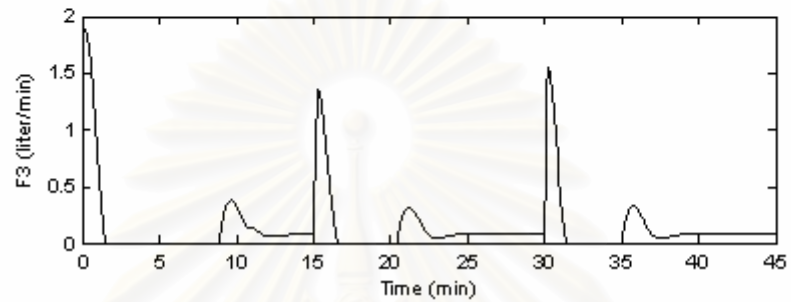
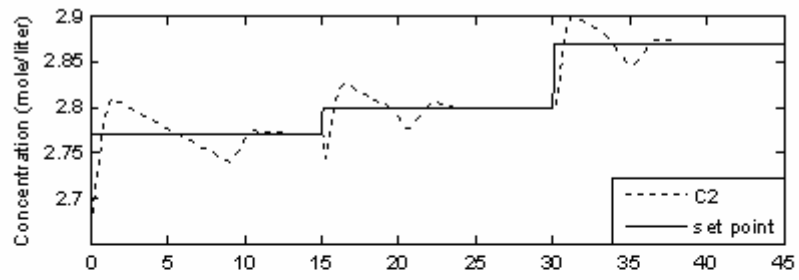
(c)

Figure 5.6(Cont.) - NNMPC control for HCl acid concentration: (a) 5% HCl bath;
(b) 10% HCl bath; and (c) 15% HCl bath

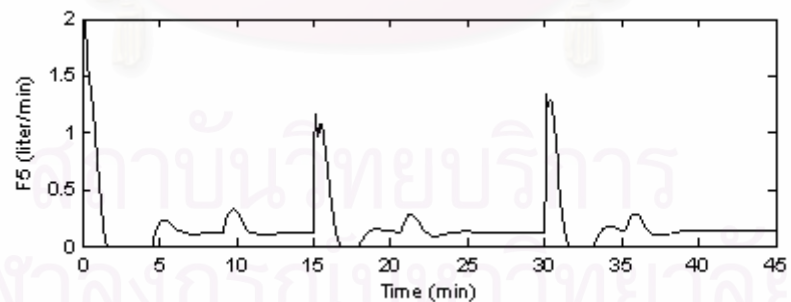
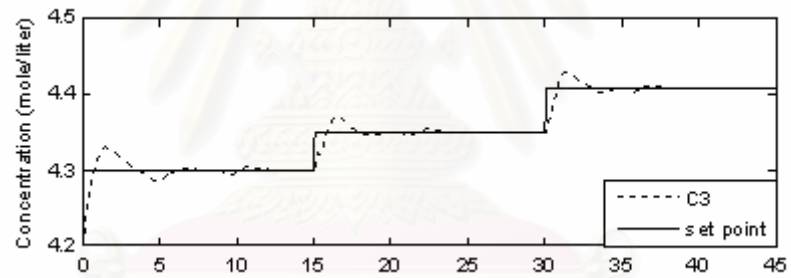


(a)

Figure 5.7 - PI control for HCl acid concentration: (a) 5% HCl bath;
(b) 10% HCl bath; and (c) 15% HCl bath



(b)



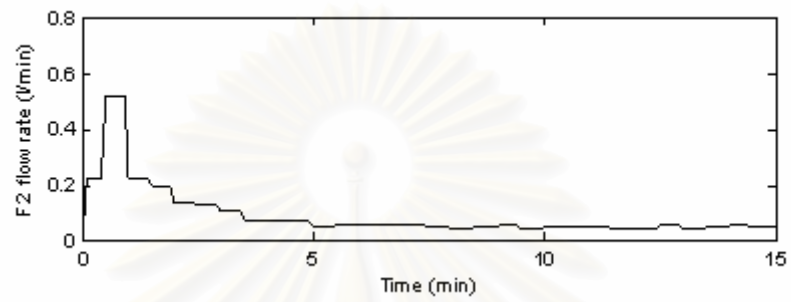
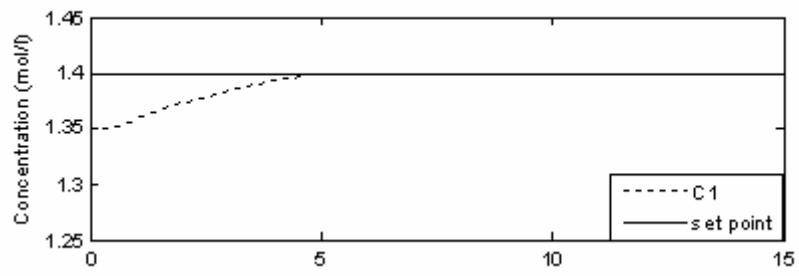
(c)

Figure 5.7(Cont.) - PI control for HCl acid concentration: (a) 5% HCl bath;
(b) 10% HCl bath; and (c) 15% HCl bath

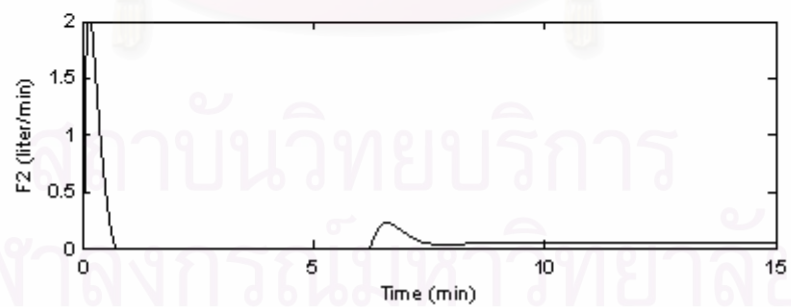
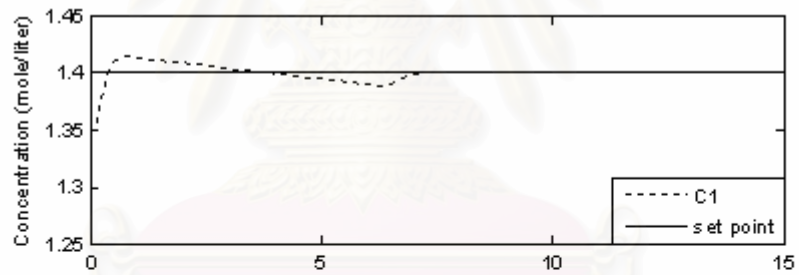
For the model mismatch case, the rate of reaction in acid bath is considered as the model mismatch in parameter. The model mismatch is introduced by randomly increasing and reducing 15% of the kinetic rate constant from its nominal value. Figure 5.11 to 5.13 show the results of NNMPC and PI control for the 5% HCl, 10% HCl and 15% HCl bath by random increasing and reducing 15% of the rate of reaction. Figure 5.11 to 5.13 illustrate that the NNMPC strategy brought the concentrations to the set points by gradually increasing the flow rate of F2, F3 and F5 while PI control bring the concentrations to the set points by rigorous adjusting of the flow rates cause the overshoot in process response and using long time back to the set points. Table 5.2 shows the IAE values of NNMPC and PI control for the three baths. They indicated that NNMPC has more robustness and give better control performance than PI controllers, similar to the disturbance case study.

Table 5.2 Performance comparison between NNMPC and PI control under the model mismatch case.

Bath	IAE Values	
	NNMPC	PI
5% HCl Bath	0.218	0.311
10% HCl Bath	0.266	0.331
15% HCl Bath	0.130	0.420



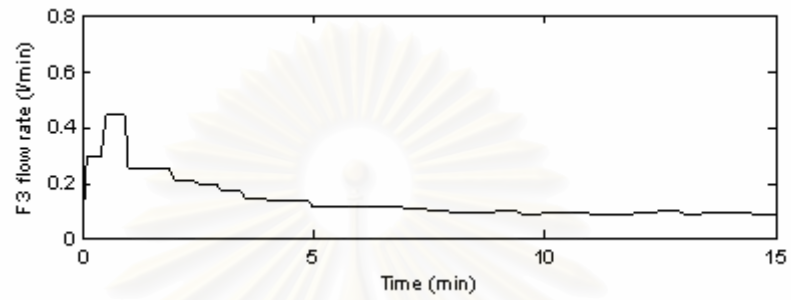
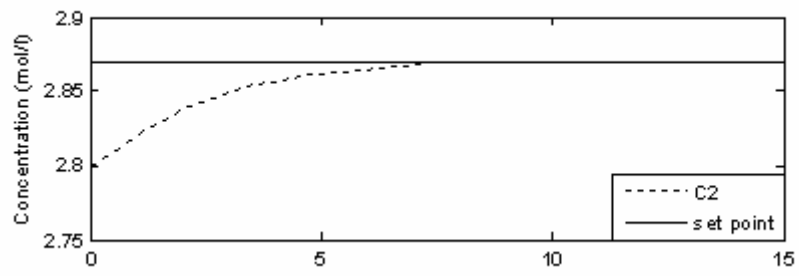
(a)



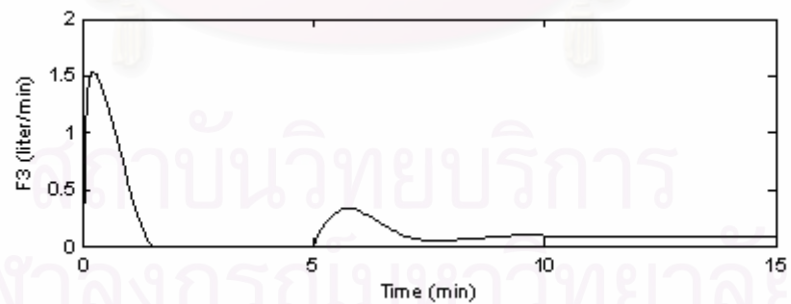
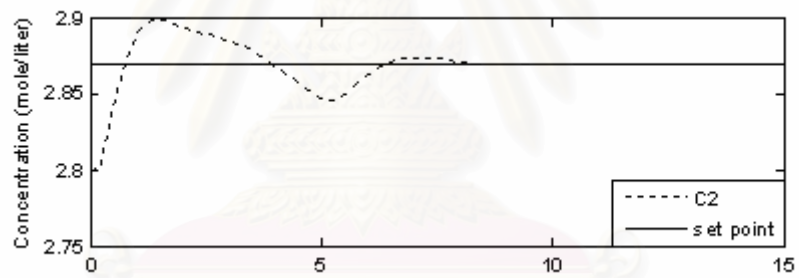
(b)

Figure 5.8 - Concentration control in 5% HCl bath under the disturbance case:

(a) NNMPC (b) PI control.



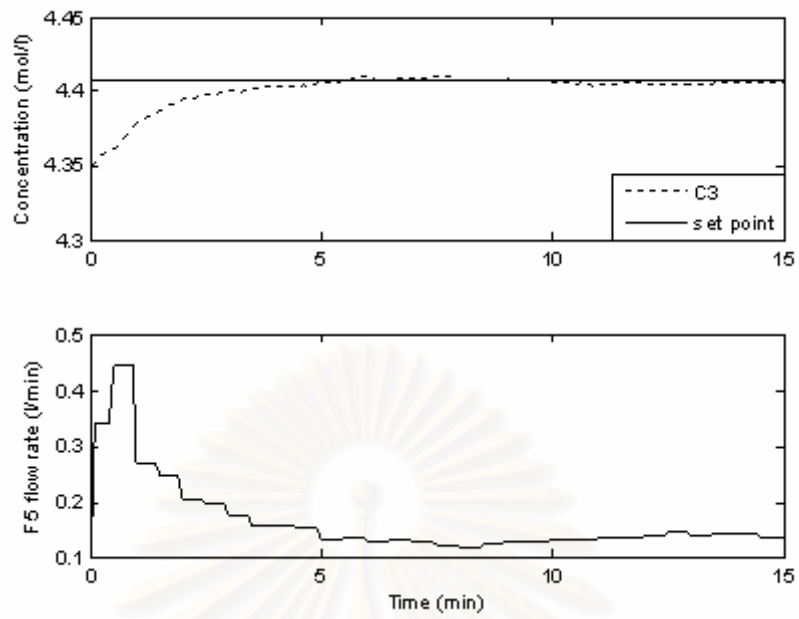
(a)



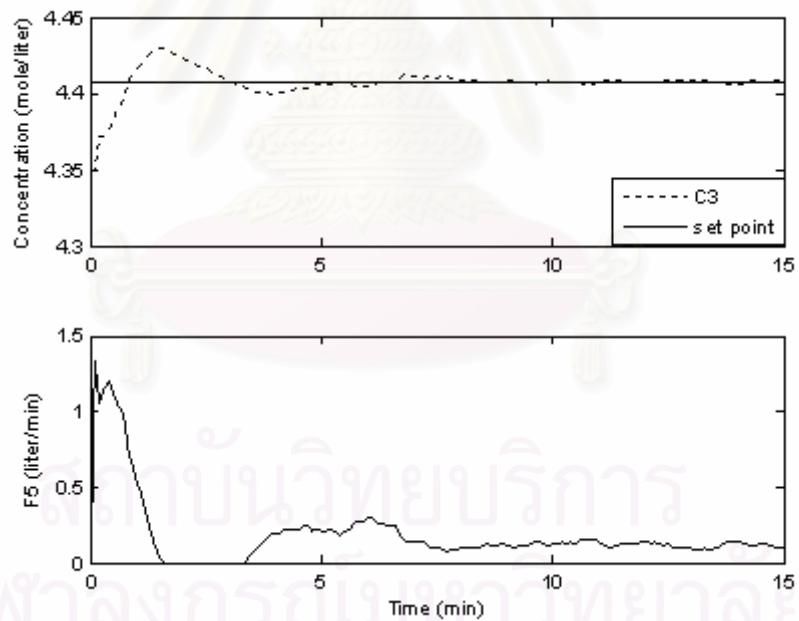
(b)

Figure 5.9 - Concentration control in 10% HCl bath under the disturbance case:

(a) NNMPC (b) PI control.



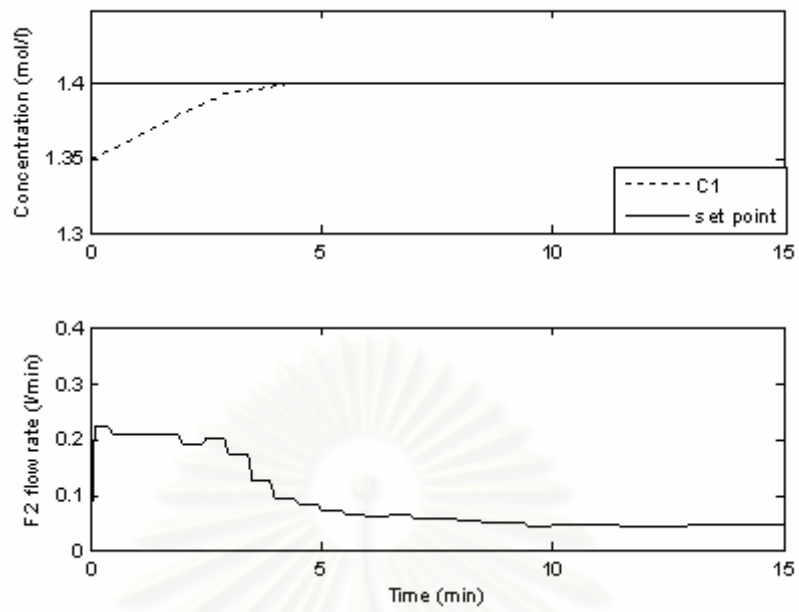
(a)



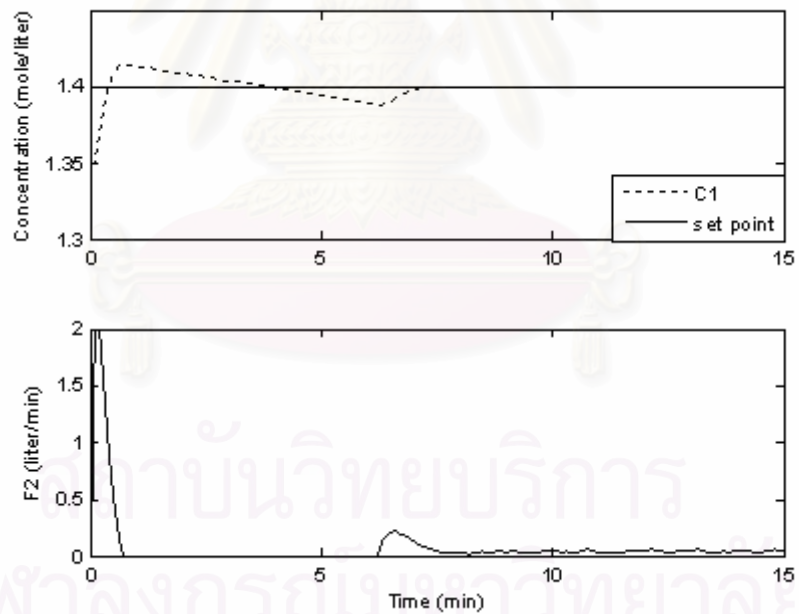
(b)

Figure 5.10 - Concentration control in 15% HCl bath under the disturbance case:

(a) NNMPC (b) PI control.



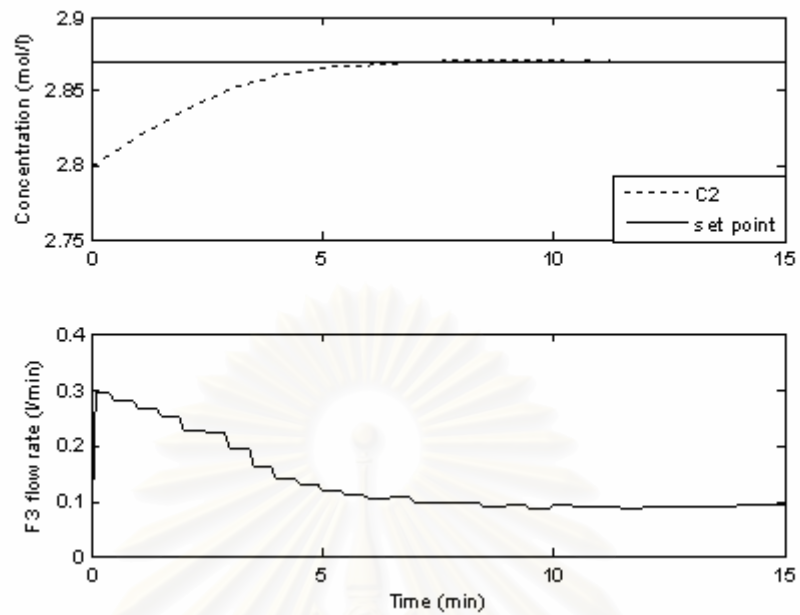
(a)



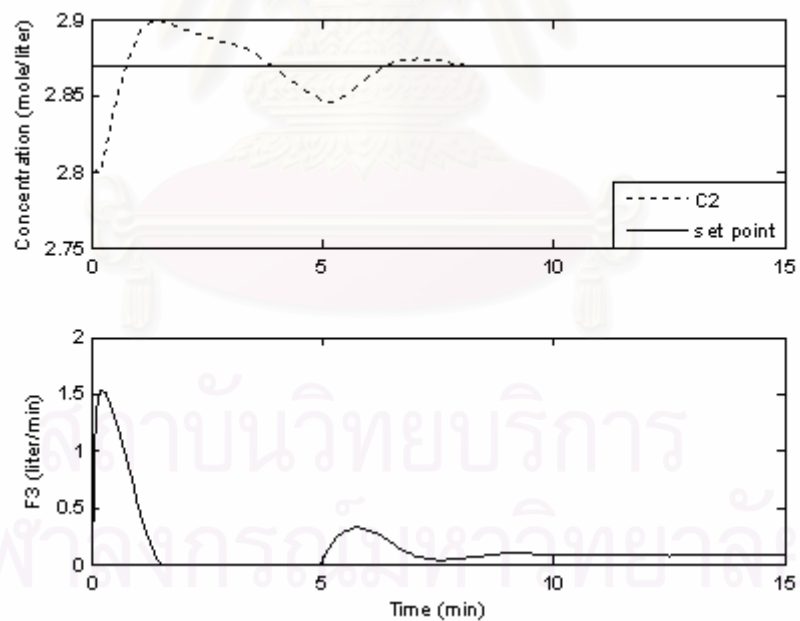
(b)

Figure 5.11 - Concentration control in 5% HCl bath under the model mismatch case:

(a) NNMPC (b) PI control.

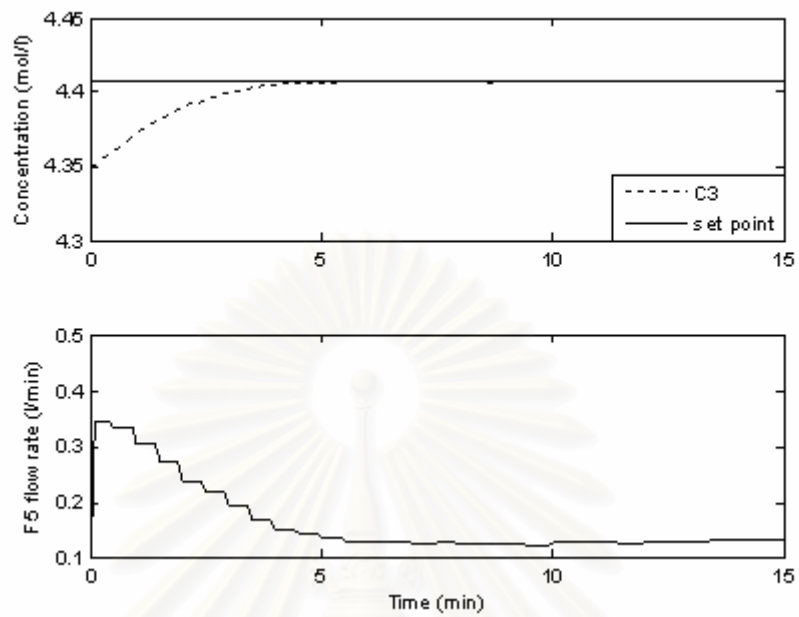


(a)

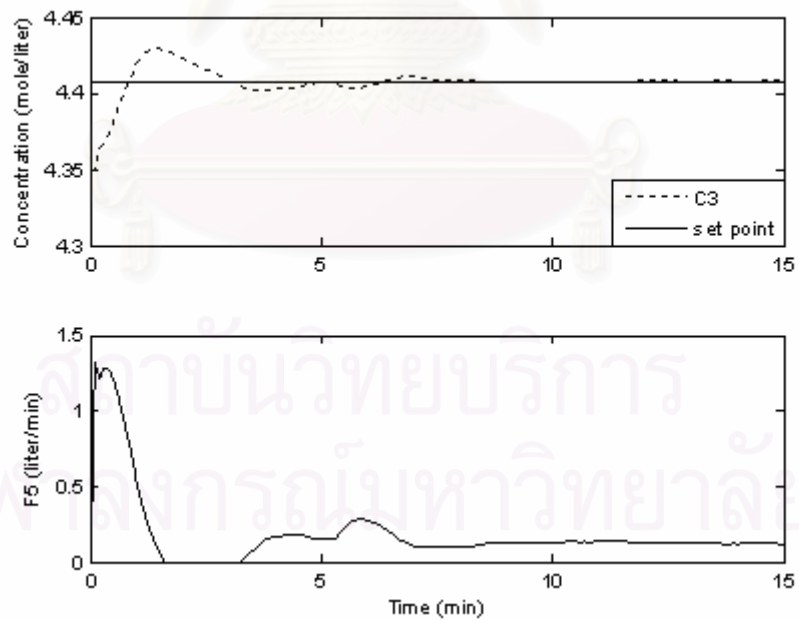


(b)

Figure 5.12 - Concentration control in 10% HCl bath under the model mismatch case:
 (a) NN MPC (b) PI control.



(a)



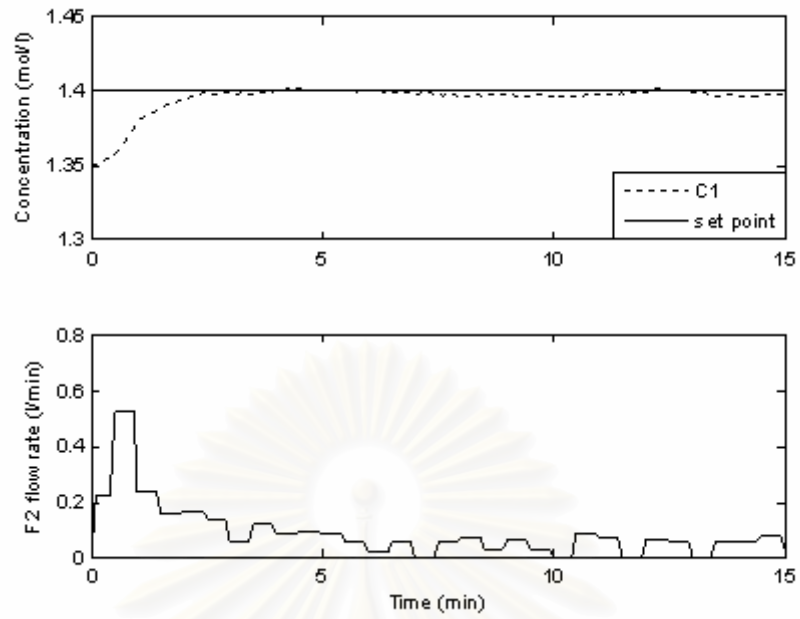
(b)

Figure 5.13 - Concentration control in 15% HCl bath under the model mismatch case:
 (a) NNMPC (b) PI control.

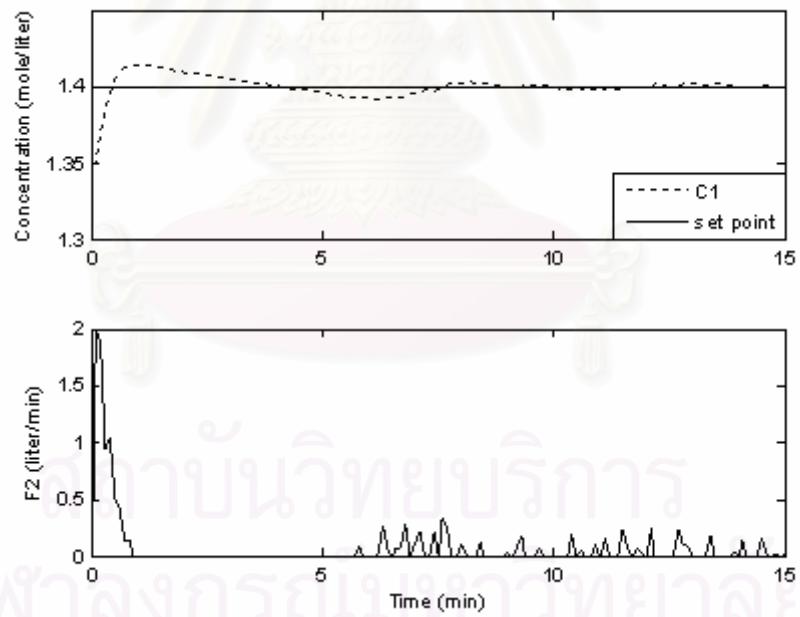
For the noises case, noises accounting to 2% random values from the output measurement, are introduced into the system to further test its robustness and performance of the NNMPC approach under close to real situations. The results in figure 5.14 to 5.16 show that the NNMPC strategy can control the system and bring the concentrations to desired value, while PI control bring the concentrations to the set point by rigorous adjusting of flow rates causing the overshoot in process response and corresponding with noise make the control variable very far from the desired value. Table 5.3 shows the IAE values of NNMPC and PI control for the three baths under noise effects. These results show the robustness and stability of the NNMPC when dealing with noise effects and more robust than PI controllers.

Table 5.3 Performance comparison between NNMPC and PI control under the noises case.

Bath	IAE Values	
	NNMPC	PI
5% HCl Bath	0.350	0.410
10% HCl Bath	0.304	0.471
15% HCl Bath	0.278	0.375

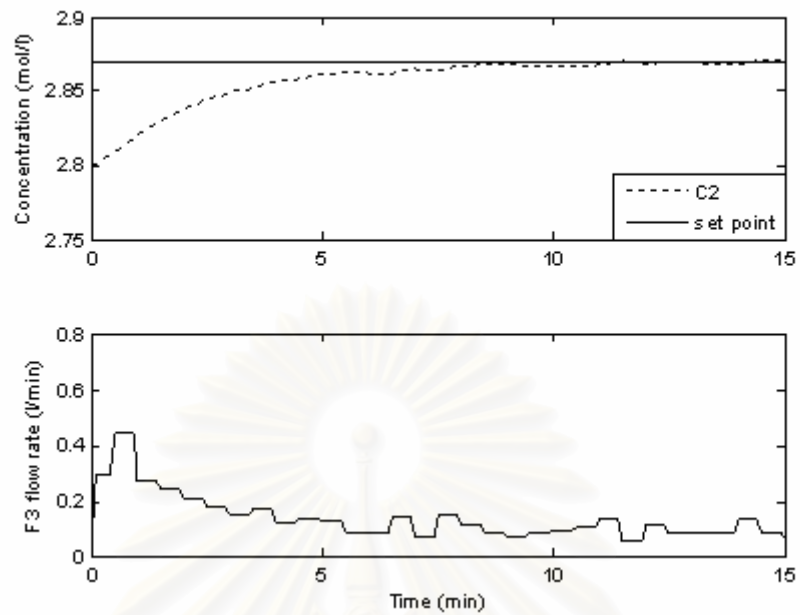


(a)

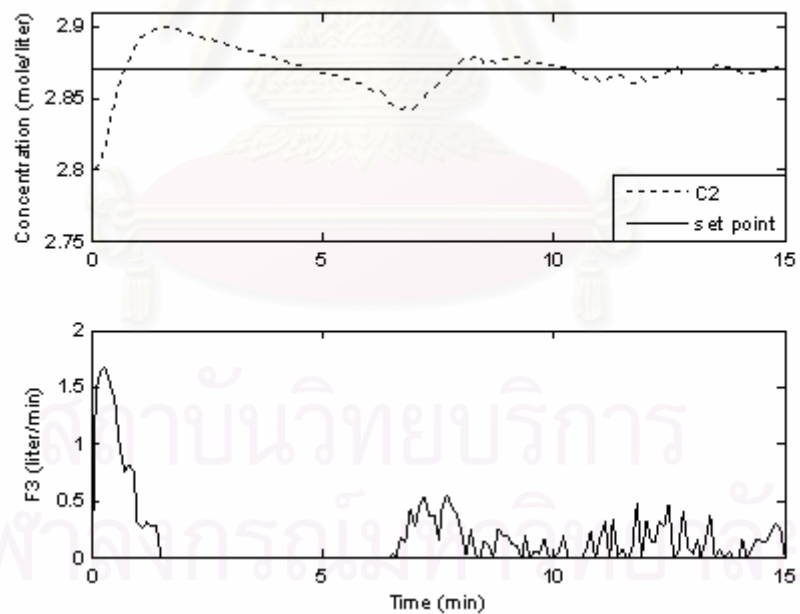


(b)

Figure 5.14 - Concentration control in 5% HCl bath under the noise case: (a) NNMPC
(b) PI control.

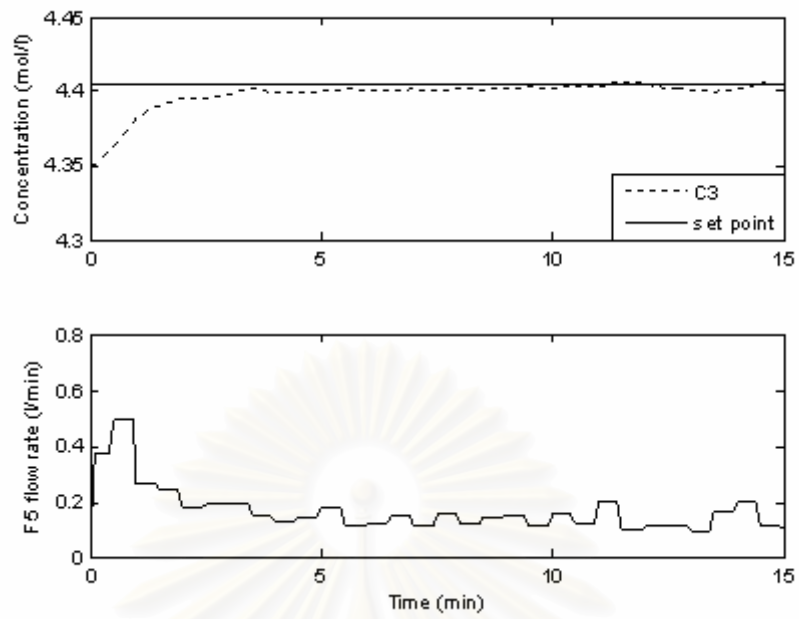


(a)

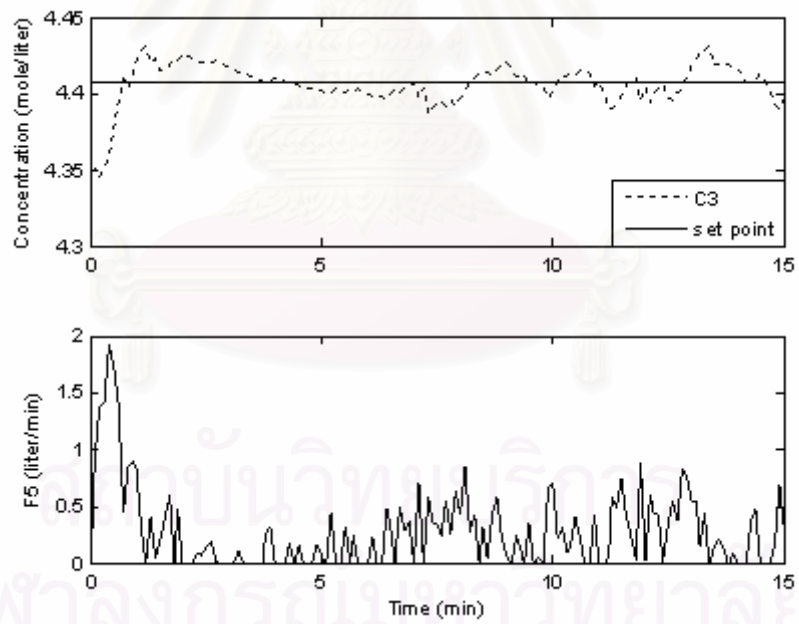


(b)

Figure 5.15 - Concentration control in 10% HCl bath under the noise case:
 (a) NNMPC (b) PI control.



(a)



(b)

Figure 5.16 - Concentration control in 15% HCl bath under the noise case:

(a) NN MPC (b) PI control.

CHAPTER VI

NEURAL NETWORK INVERSE MODEL BASED CONTROLLER FOR THE CONTROL OF A STEEL PICKLING PROCESS

6.1 Neural network direct inverse control strategy (NNDIC)

In this chapter, the neural network direct inverse control (NNDIC) method is used for the control strategy of a steel pickling process (Daosud et al., 2005). This strategy consists of the neural network inverse model that acts as the controller placed in series with the process under control. In this work, the neural network inverse models trained as described in chapter 4 (Figure 6.1 to 6.4) are utilized to predict the manipulated flow rates of each bath to bring the process to desired conditions. Process control configuration of a steel pickling process is shown in figure 6.5 to 6.6

As shown in figure 6.7, the controller predicts the control action, $F(k)$, by having current and past values of the process model state variables and the past control action as well as the required set point as its inputs. The prediction of the controller action, i.e. manipulated variable is normally sufficient to make the value of the controlled variable, $C(k + 1)$, change according to the set point. The control structure is fairly simple and works fairly well in many non-linear plants (Hussain et al., 2003). This control strategy is then implemented in the steel pickling process to control the HCl concentrations in baths, $C1$, $C2$, $C3$ and $C4$, by manipulating the flow, $F2$, $F3$, $F5$ and $F6$. The control performance is tested under the nominal case and with disturbances case, model mismatch and noise added into the process. The simulation results and discussion of these control studies are described in the next section.

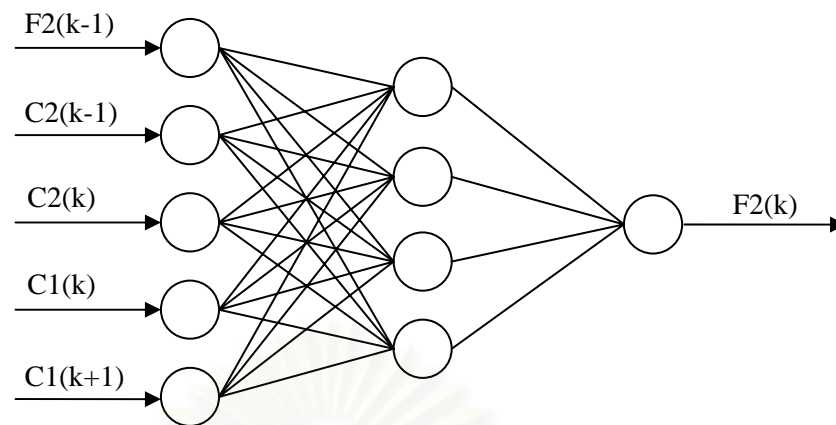


Figure 6.1 - The inverse neural network controller of 5% HCl bath (structure 5-4-1)

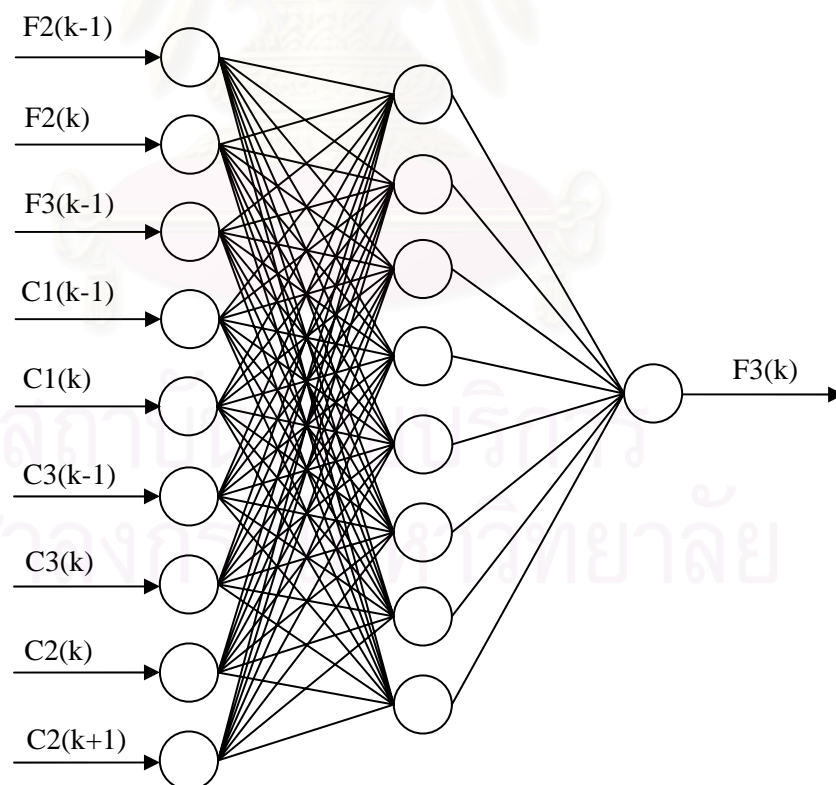


Figure 6.2 - The inverse neural network controller of 10% HCl bath (structure 9-8-1)

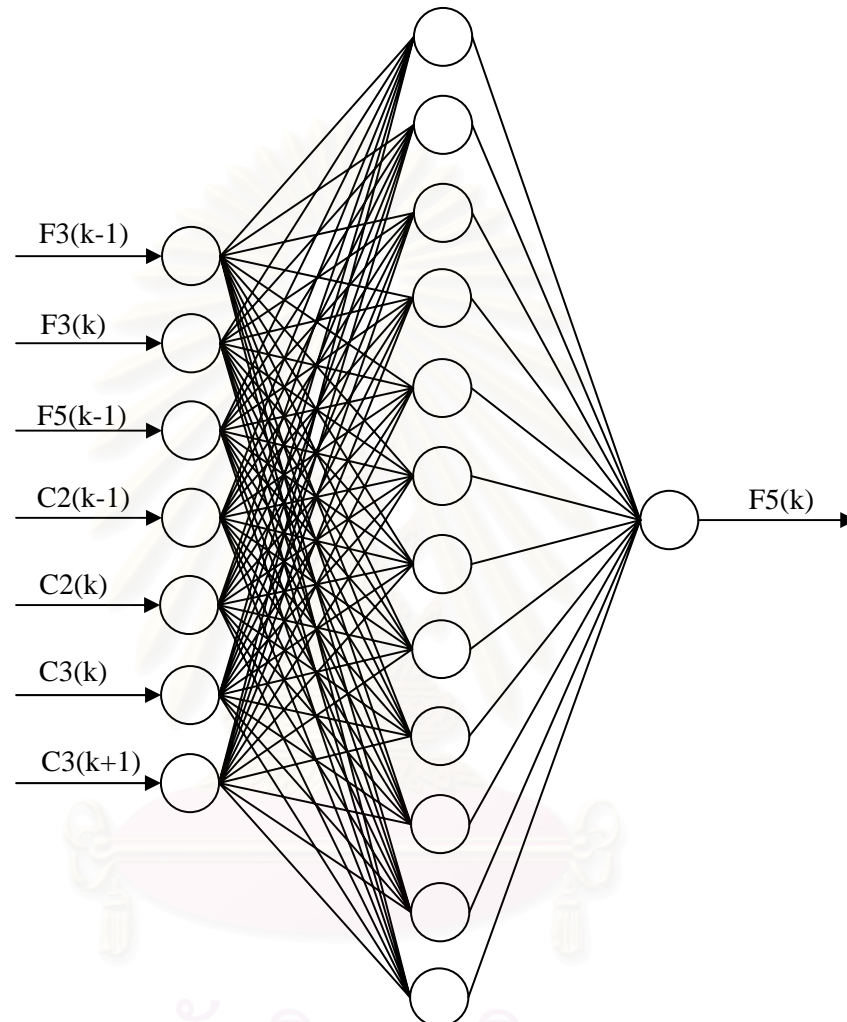


Figure 6.3 - The inverse neural network controller of 15% HCl bath (structure 7-12-1)

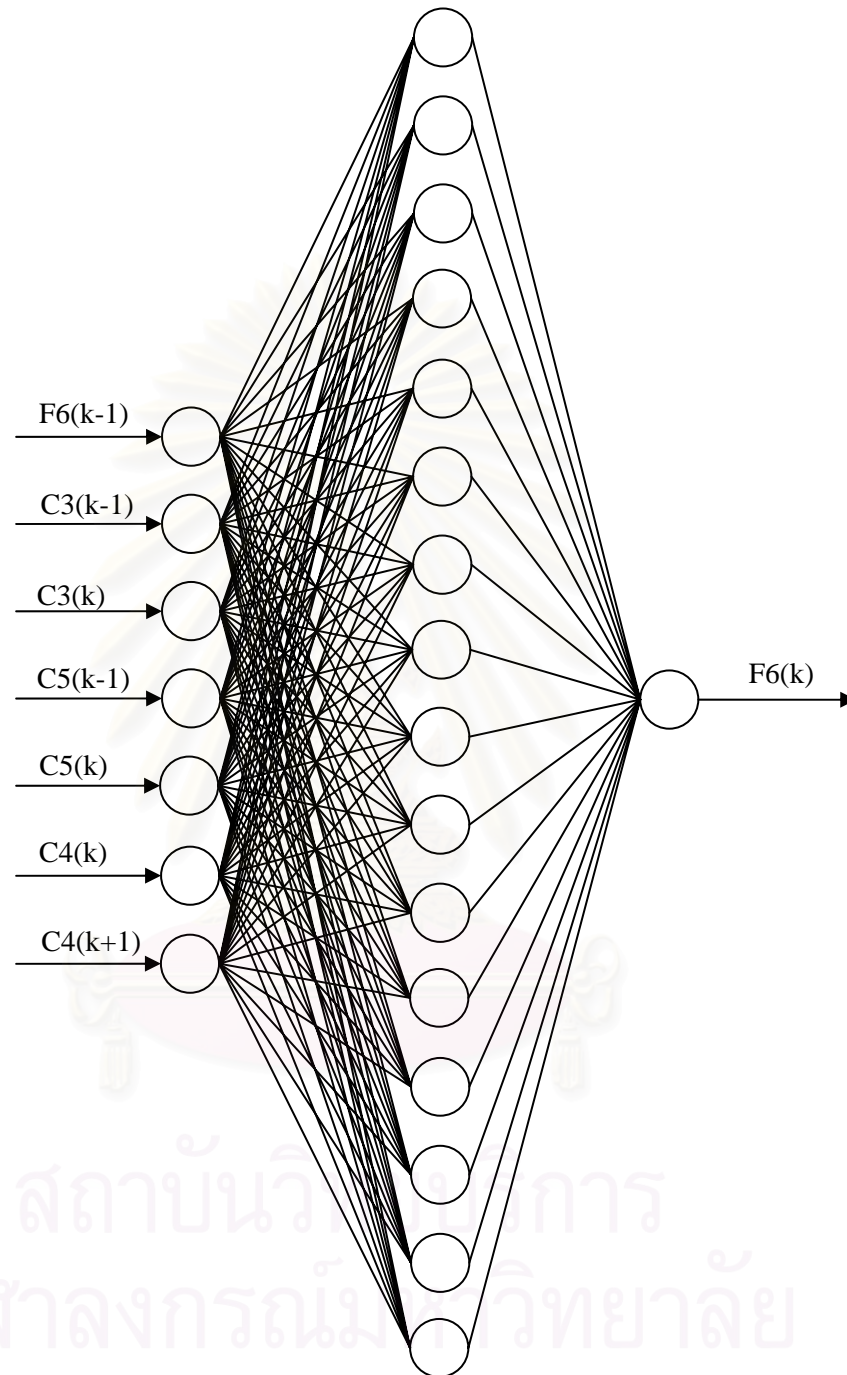


Figure 6.4 - The inverse neural network controller of 1st rinsing bath (structure 7-16-1)

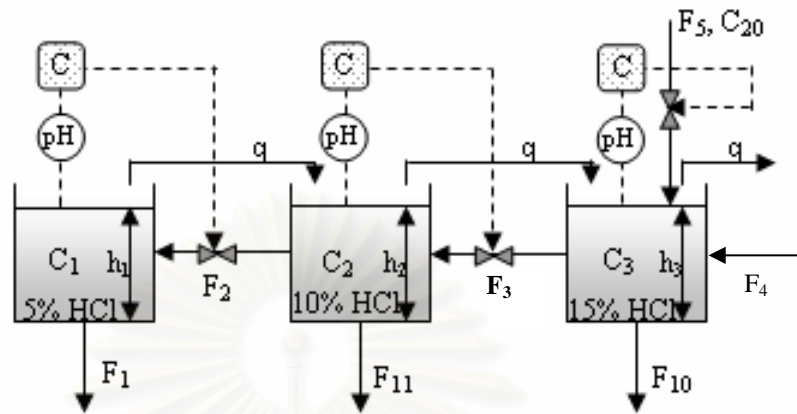


Figure 6.5 - Flow diagram of pickling baths control system.

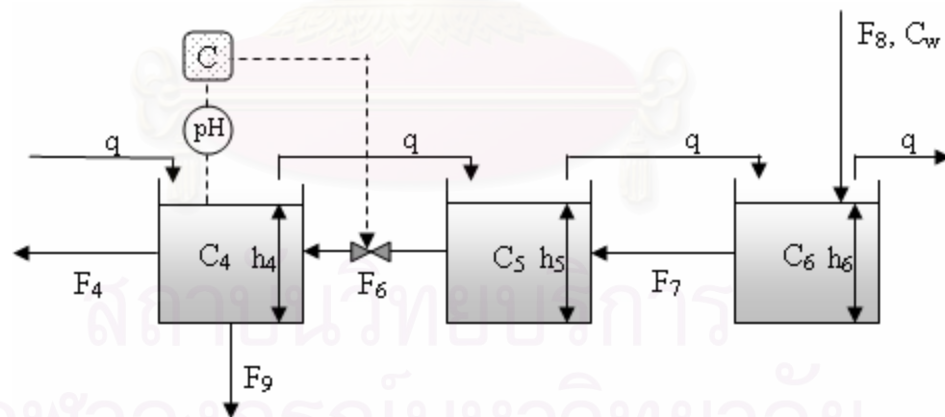


Figure 6.6 - Flow diagram of rinsing baths control system.

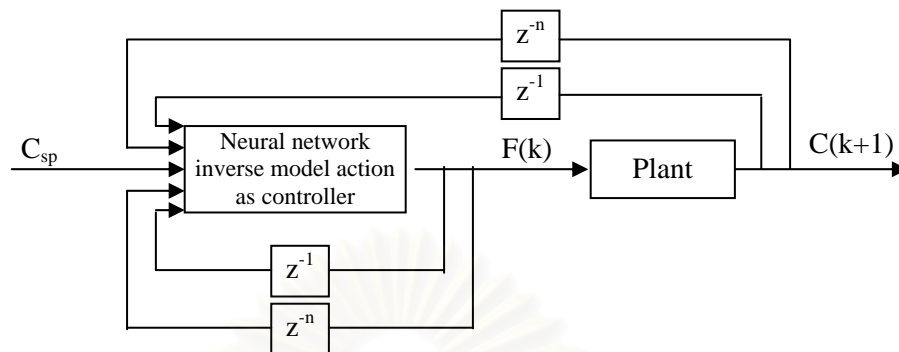


Figure 6.7 - Neural network direct inverse model control strategy.

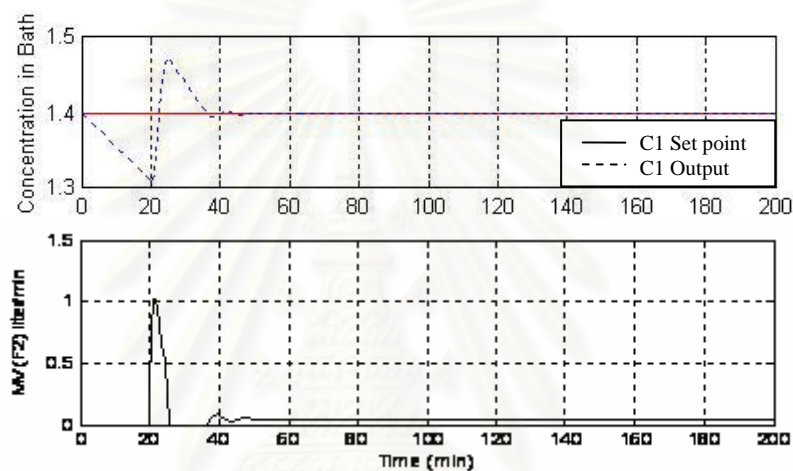
6.2 Simulation Results and Discussion of NNDIC

In the simulation studies, the objective is to control the concentration of HCl in the 5% HCl, 10% HCl, 15% HCl and 1st rinsing bath to the values of 1.40, 2.87, 4.41 and 1×10^{-3} mol/l (pH 3) by adjusting the manipulated variables $F2$, $F3$, $F5$ and $F6$, respectively. They are divided into four cases of control studies, which are the nominal case, disturbance case, model mismatch case and noise case, respectively.

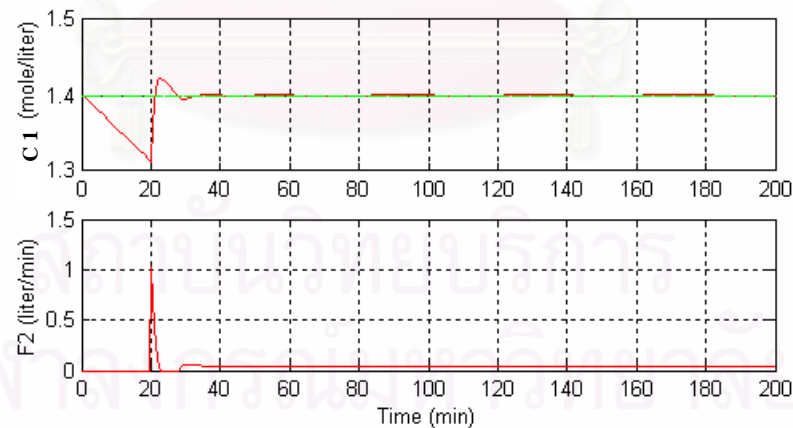
6.2.1 Nominal case

In this case, the controllers are designed to bring the concentration of HCl in each bath to the desired value when the initial condition is set at steady state for 20 min without controller action. Figure 6.8(a), 6.9(a), 6.10(a) and 6.11(a) show the control of HCl concentration in the 5% HCl, 10% HCl, 15% HCl and 1st rinsing bath using NNDIC, respectively and figure 6.8(b), 6.9(b), 6.10(b) and 6.11(b) show them with PI control. The results in these figures indicate that NNDIC can bring the concentrations closely to the set points and give minimal offsets while PI control can bring the controlled variable to the set points without offsets. However, drastic change of the manipulated variable and oscillation at the initial state when starting control for the PI control can be remarkably observed. Their performances are also evaluated

using the integral absolute error (IAE). The IAE results for the nominal case of these four baths are summarized in Table 6.1. They all showed that relatively, the PI controllers give better results than NNDIC in term of lesser IAE values since there are no offsets in using the PI control strategy. It is noted that although, the PI controllers can handle the system well in the nominal case, the control action of the PI method is less smooth than that of the NNDIC method.

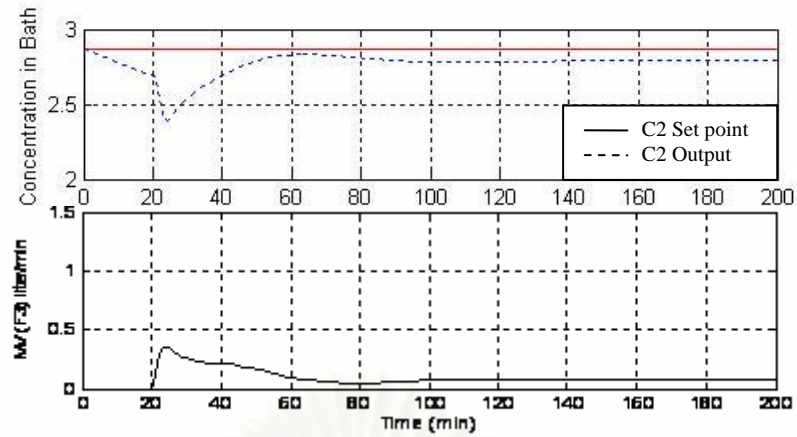


(a)

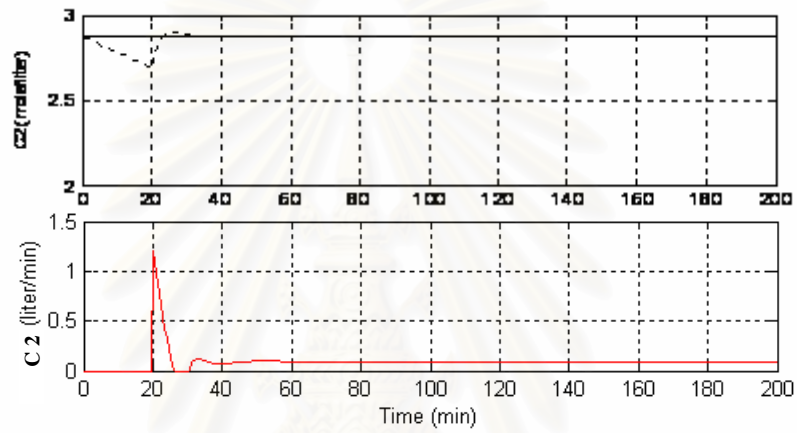


(b)

Figure 6.8 - Concentration control in 5% HCl bath under the nominal case:
(a) NNDIC (b) PI control.

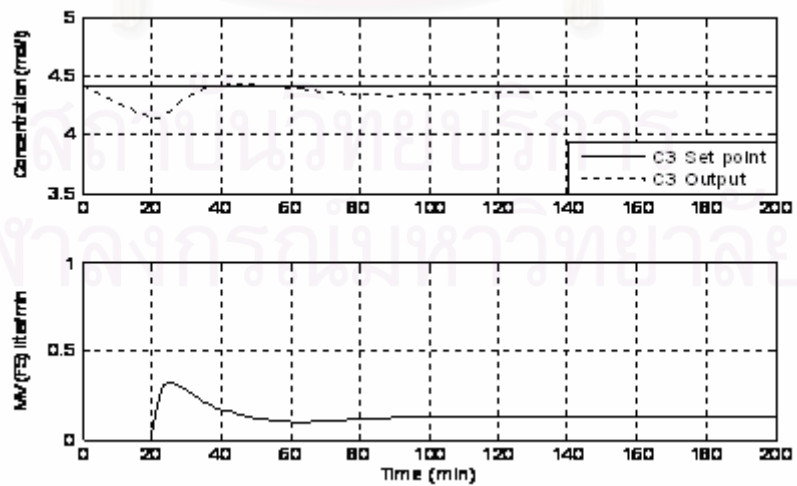


(a)



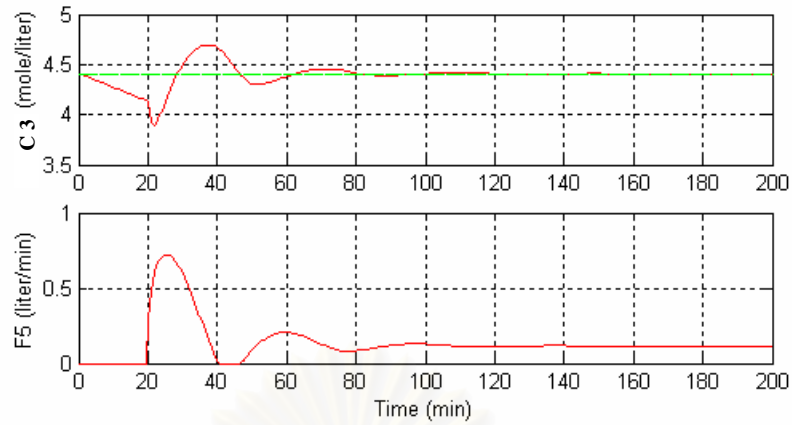
(b)

Figure 6.9 - Concentration control in 10% HCl bath under the nominal case:
(a) NNDIC (b) PI control.



(a)

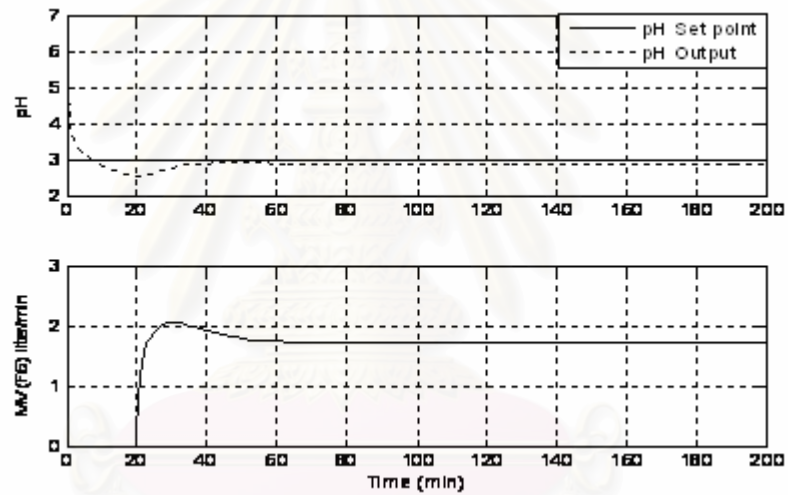
Figure 6.10 - Concentration control in 15% HCl bath under the nominal case:
(a) NNDIC (b) PI control.



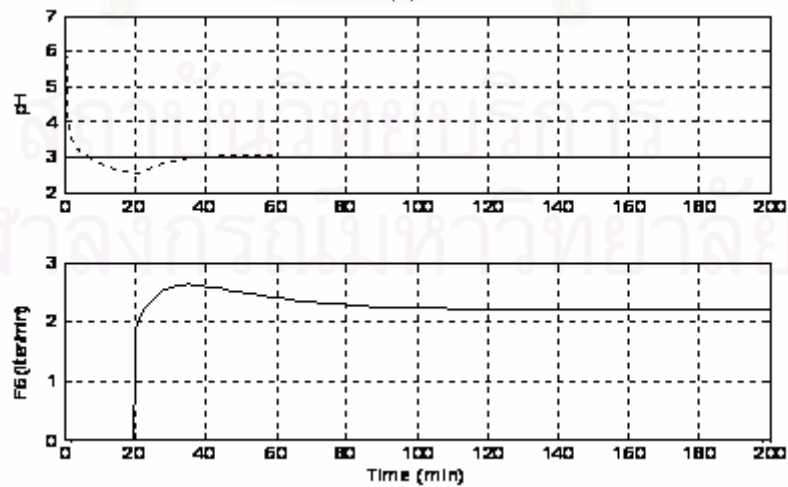
(b)

Figure 6.10 (Cont.) - Concentration control in 15% HCl bath under the nominal case:

(a) NNDIC (b) PI control.



(a)



(b)

Figure 6.11 - Concentration control in 1st rinsing bath under the nominal case:

(a) NNDIC (b) PI control.

Table 6.1 Performance comparison between NNDIC and PI control under the nominal case.

Bath	IAE Values	
	NNDIC	PI
5% HCl Bath	2.109	0.342
10% HCl Bath	37.925	2.607
15% HCl Bath	21.697	13.281
1 st Rinsing Bath	0.274	0.007

6.2.2 Disturbance case

In this case, the disturbance, which is the change in the concentration C_{20} in the stream F_5 , is introduced by increasing and reducing 15% from its nominal operation values. Initially, the process is left under control until $t = 200$ min, at which instant, the disturbance is introduced. During the period $t = 200$ – 300 min, the control action is halted to the last value to allow the process to respond to the new load condition. At $t = 300$ min, the PI and neural network based control action are introduced back into the system. Figure 6.12 to 6.15 show the results of NNDIC and PI control for the 5% HCl, 10% HCl, 15% HCl and 1st rinsing bath by increasing 15% of C_{20} from its nominal value. It can be seen from these figures that when the disturbance is introduced ($t = 200$ – 300 min), the process responds by an increase in the concentration of the baths due to the increase of HCl concentration (C_{20}) to the 15% HCl bath. After $t = 300$ min, the NNDIC strategy can bring back the concentrations close to the required value by gradually decreasing the manipulated flow rates (F_2 , F_3 , F_5 and F_6), while PI control strategy bring the concentrations to the set point with oscillation. In reducing the concentration C_{20} by 15%, the results show that the NNDIC strategy can still control the process and bring the concentrations to their set points but the PI controllers bring the concentrations to the set point with oscillation that can be clearly observed from figure 6.18. Table 6.2 summarizes the IAE values of NNDIC and PI control for the four baths. They indicate that NNDIC has more robustness and give better control performance than PI controllers with much smaller IAE error values, when disturbances are present in the system.

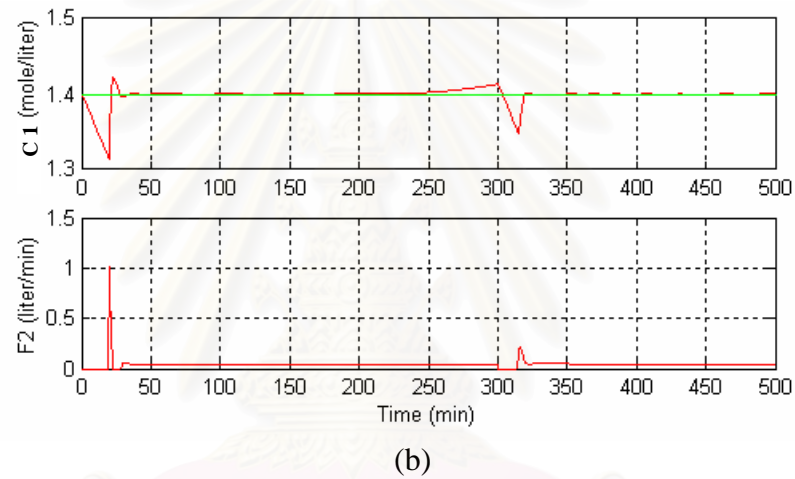
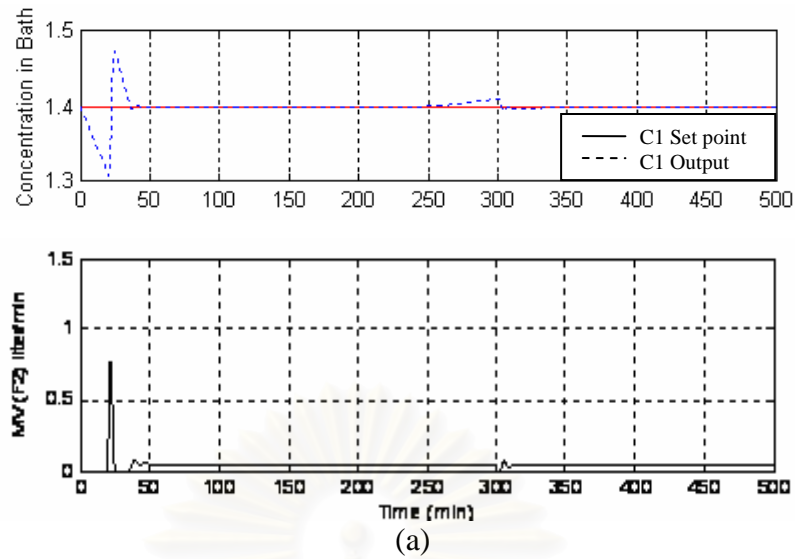


Figure 6.12 - Concentration control in 5% HCl bath under the disturbance case (15% increase of the concentration, C_{20}): (a) NNDIC (b) PI control.

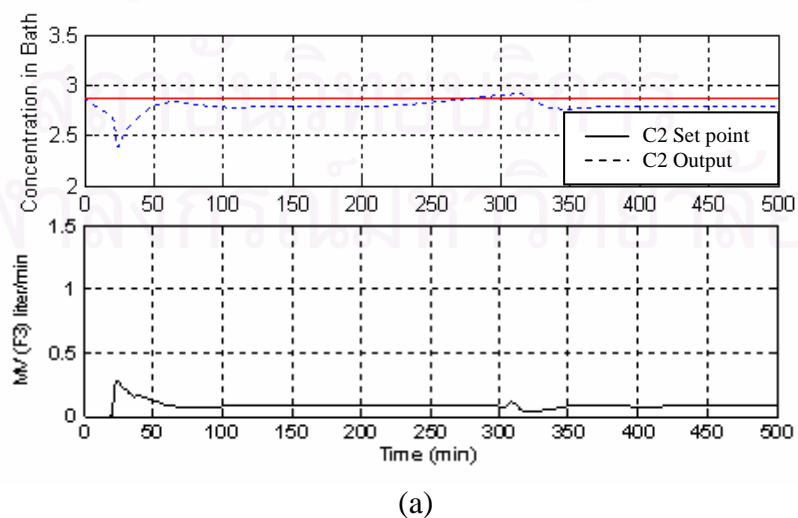


Figure 6.13 - Concentration control in 10% HCl bath under the disturbance case (15% increase of the concentration, C_{20}): (a) NNDIC (b) PI control.

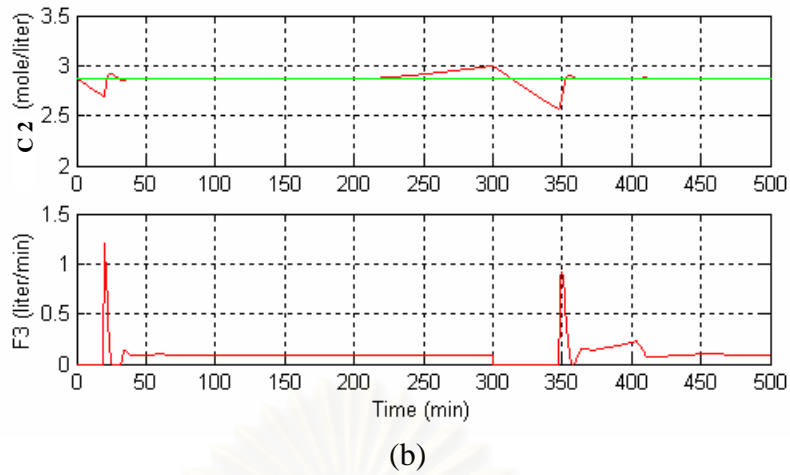


Figure 6.13(Cont.) - Concentration control in 10% HCl bath under the disturbance case (15% increase of the concentration, C_{20}): (a) NNDIC (b) PI control.

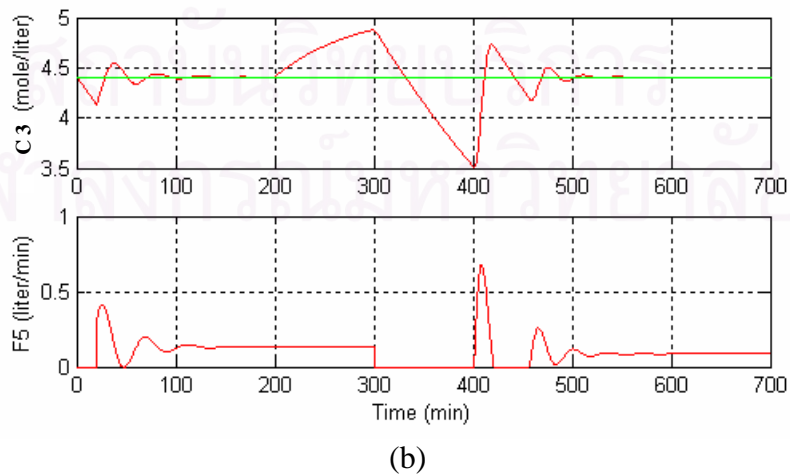
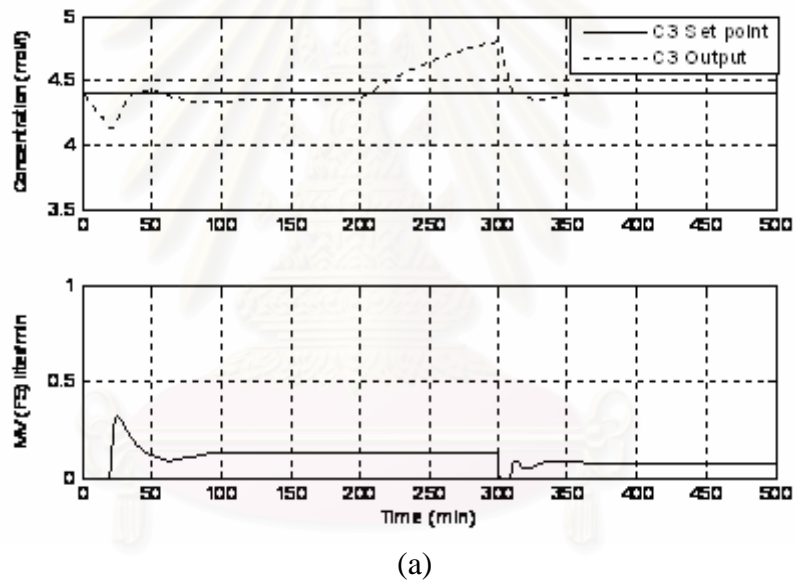
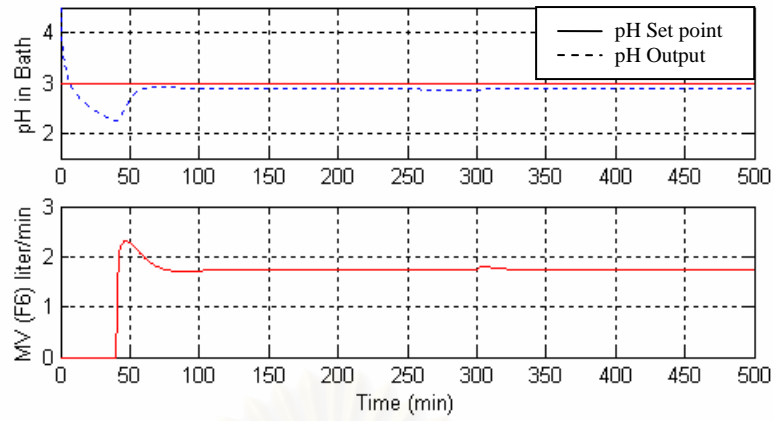
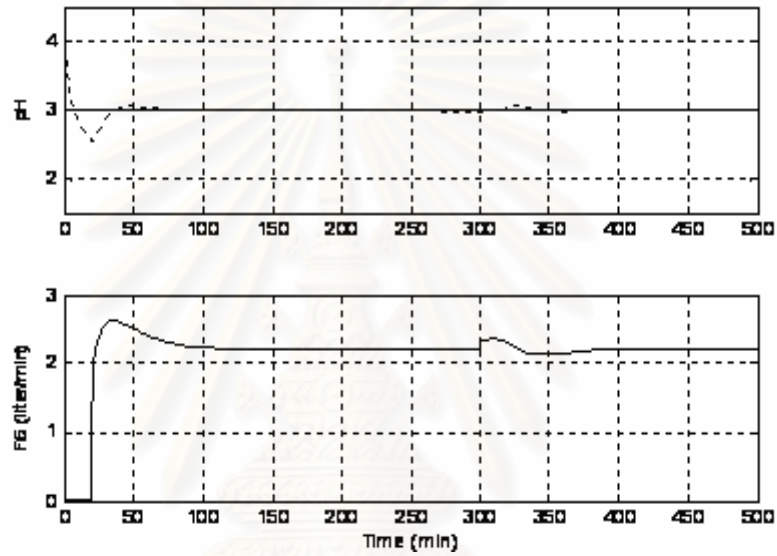


Figure 6.14 - Concentration control in 15% HCl bath under the disturbance case (15% increase of the concentration, C_{20}): (a) NNDIC (b) PI control.

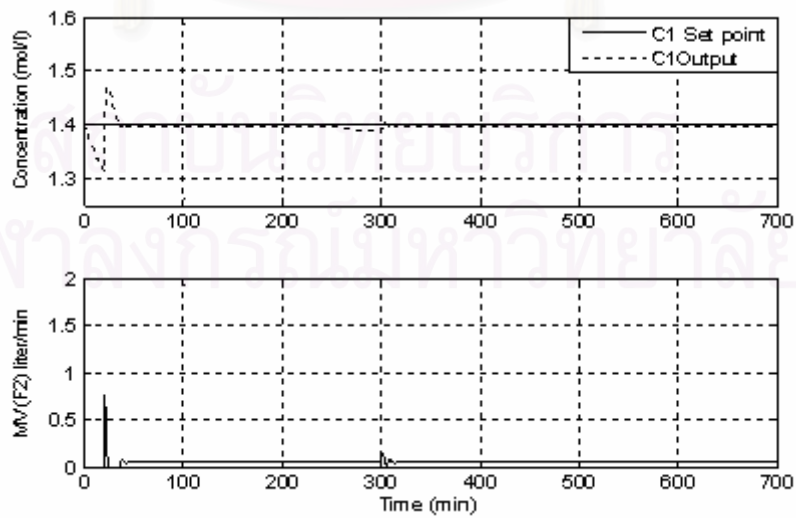


(a)



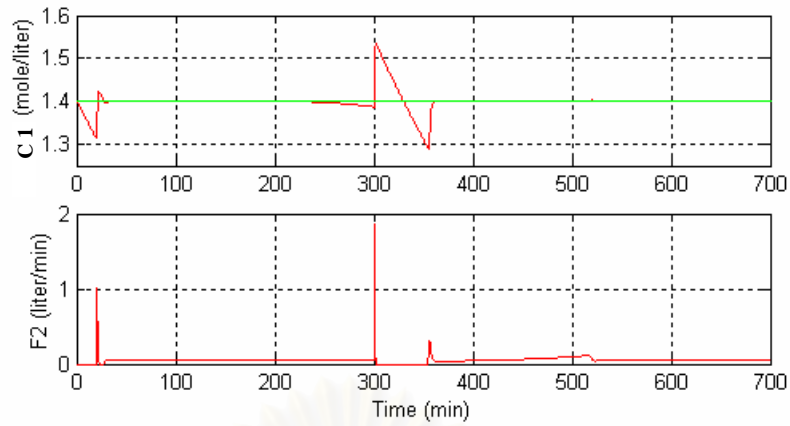
(b)

Figure 6.15 - Concentration control in 1st rinsing bath under the disturbance case (15% increase of the concentration, C_{20}): (a) NNDIC (b) PI control.



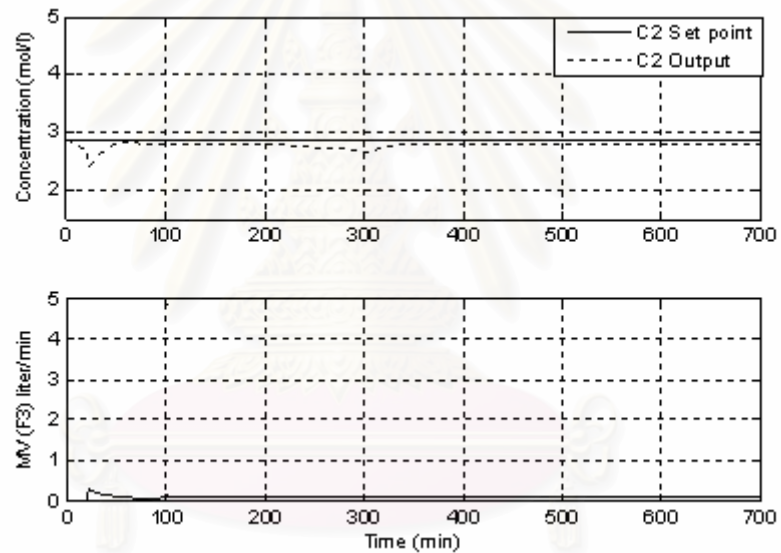
(a)

Figure 6.16 - Concentration control in 5% HCl bath under the disturbance case (15% decrease of the concentration, C_{20}): (a) NNDIC (b) PI control.

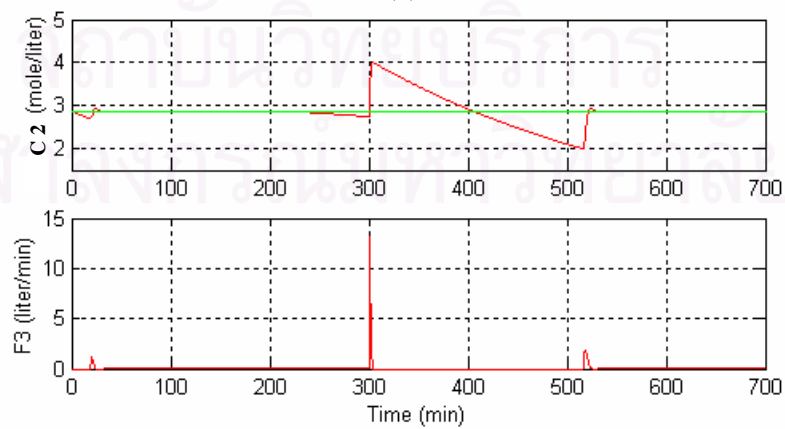


(b)

Figure 6.16 (Cont.) - Concentration control in 5% HCl bath under the disturbance case (15% decrease of the concentration, C_{20}):(a) NNDIC (b) PI control

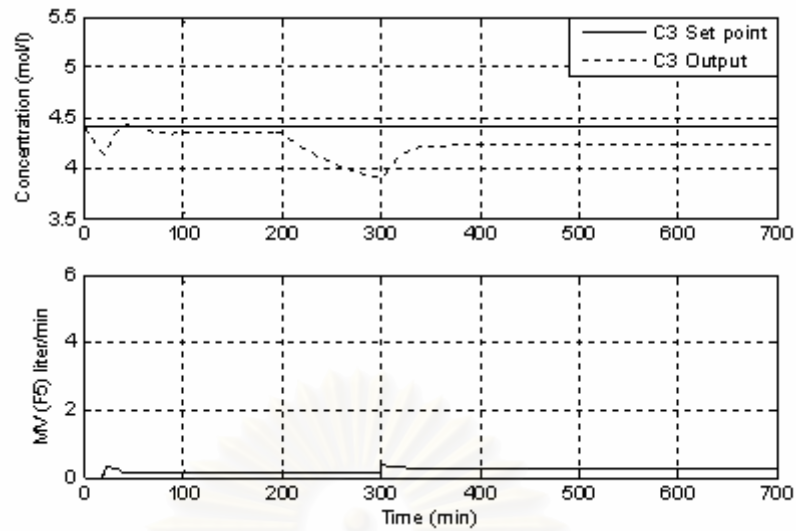


(a)

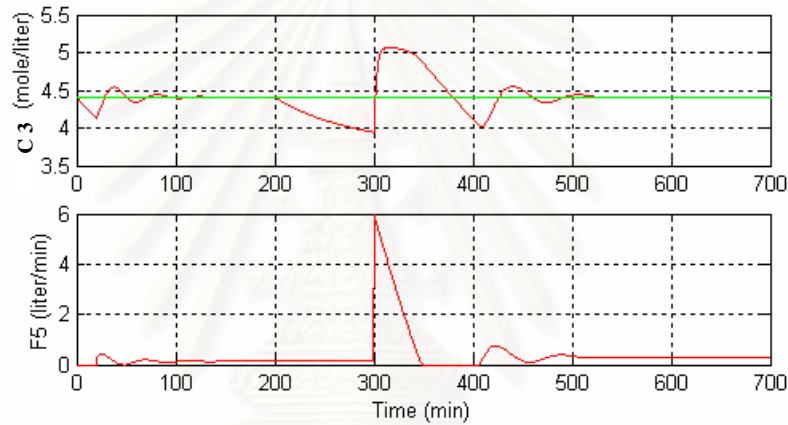


(b)

Figure 6.17 - Concentration control in 10% HCl bath under the disturbance case (15% decrease of the concentration, C_{20}): (a) NNDIC (b) PI control.

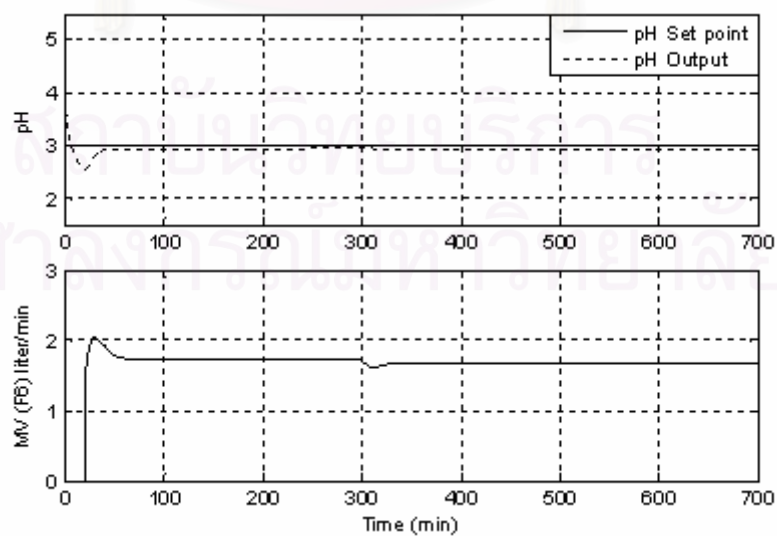


(a)



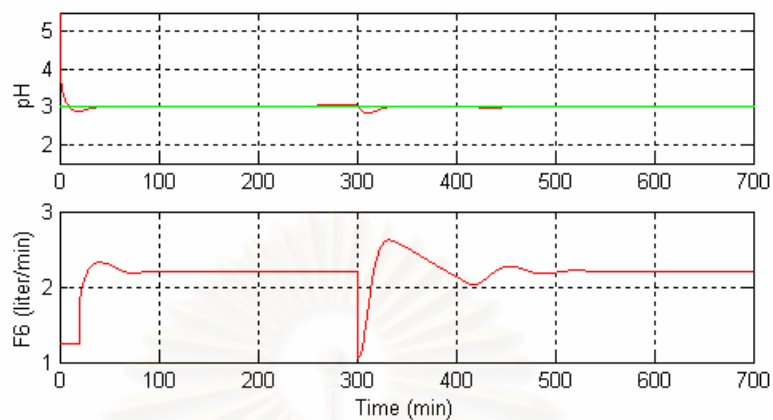
(b)

Figure 6.18 - Concentration control in 15% HCl bath under the disturbance case (15% decrease of the concentration, C_{20}): (a) NNDIC (b) PI control.



(a)

Figure 6.19 - Concentration control in 1st rinsing bath under the disturbance case (15% decrease of the concentration, C_{20}): (a) NNDIC (b) PI control.



(b)

Figure 6.19 (Cont.) - Concentration control in 1st rinsing bath under the disturbance case (15% decrease of the concentration, C_{20}): (a) NNDIC (b) PI control.

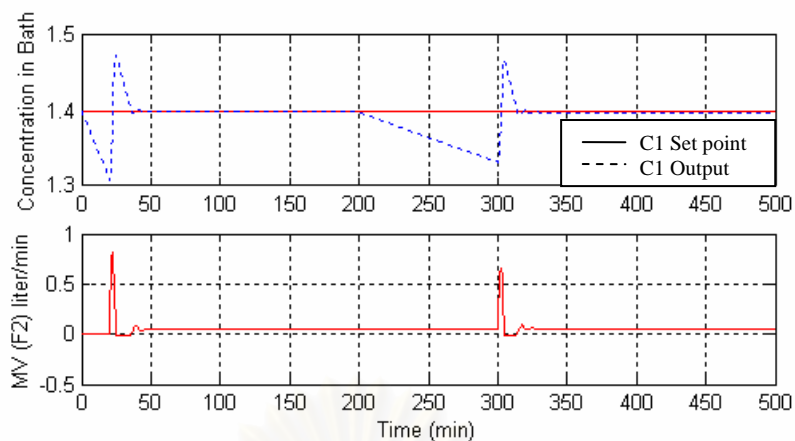
Table 6.2 Performance comparison between NNDIC and PI control under the disturbance case (15% increase of the concentration, C_{20}).

Bath	IAE Values	
	NNDIC	PI
5% HCl Bath	3.508	10.899
10% HCl Bath	77.595	146.633
15% HCl Bath	74.997	702.675
1 st Rinsing Bath	0.444	0.782

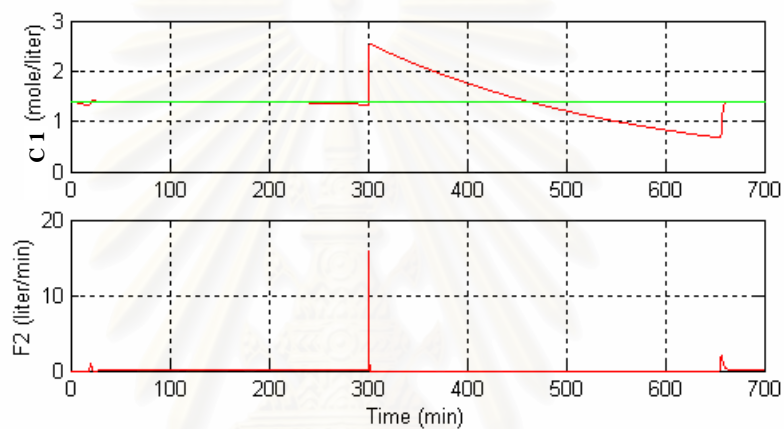
6.2.3 Model mismatch case

The rate of reaction in acid bath is considered as the model mismatch in parameter. The model mismatch is introduced by increasing and reducing 15% of the kinetic rate constant from its nominal value. Initially, the process is left under control until $t = 200$ min, at which instant, the model mismatch is introduced. During the period $t = 200$ – 300 min, the process control action is halted to the latest value to allow the process to respond to the new mismatch condition. At $t = 300$ min, the PI and neural network based control action are introduced back into the system. Figure 6.20 to 6.23 show the results of NNDIC and PI control for 5% HCl, 10% HCl, 15% HCl and 1st rinsing bath by increasing 15% of the rate of reaction. When model mismatch ($t = 200$ – 300 min) is introduced, the process responds by a decrease in concentration in baths due to the increase of the rate of reaction in acid bath. After $t = 300$ min, the NNDIC strategy bring back the concentrations close to their required values by gradually increasing the manipulated flow rates ($F2$, $F3$, $F5$ and $F6$), while PI control strategy bring the concentrations to the set point with drastic change of the manipulated variable and oscillation.

In reducing 15% of the rate of reaction, the results again show that the NNDIC strategy can control the process and bring the concentrations to their set points but PI controller bring the concentrations to the set point with oscillation that can be clearly observed from figure 6.26. Table 6.3 shows the IAE values of NNDIC and PI control for the four baths. They indicate that NNDIC has more robustness and give better control performance than PI controllers, similar to the disturbance case study. The robustness of the NNDIC can be explained by the fact that the obtained NN inverse model for the use in the NNDIC was trained with the wide range of operating conditions whereas the PI controller was tuned based on a nominal condition.

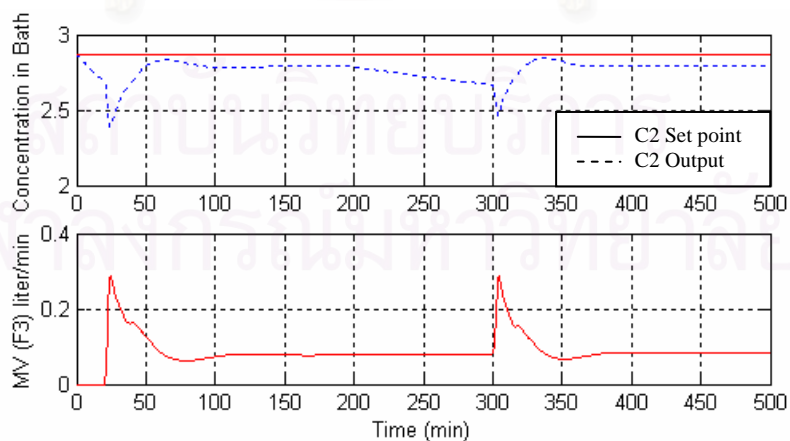


(a)



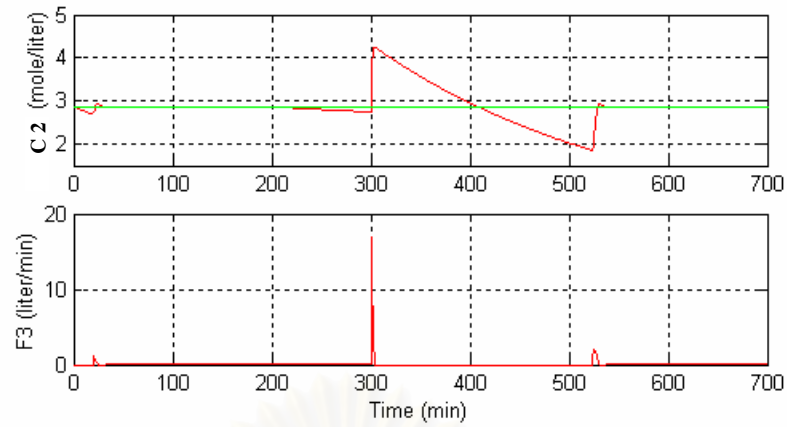
(b)

Figure 6.20 - Concentration control in 5% HCl bath under the model mismatch case (15% increase of the reaction rate, k): (a) NNDIC (b) PI control.



(a)

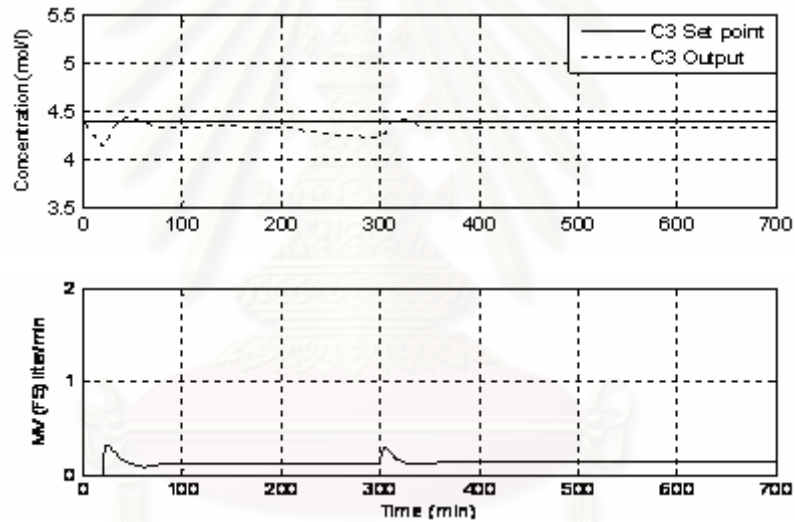
Figure 6.21 - Concentration control in 10% HCl bath under the model mismatch case (15% increase of the reaction rate, k): (a) NNDIC (b) PI control.



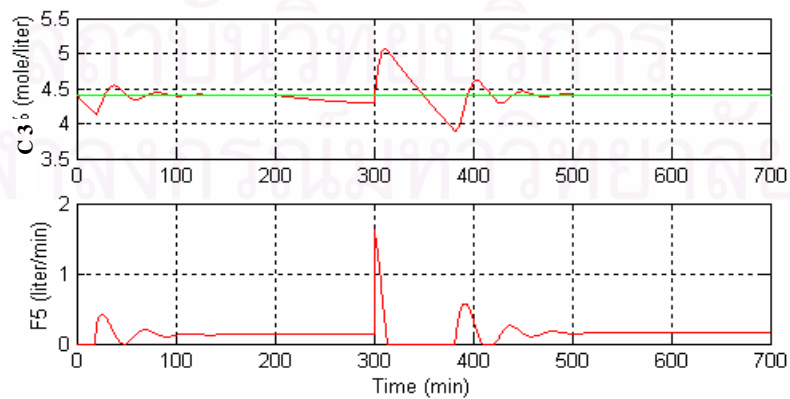
(b)

Figure 6.21 (Cont.) - Concentration control in 10% HCl bath under the model mismatch case (15% increase of the reaction rate, k):

(a) NNDIC (b) PI control.

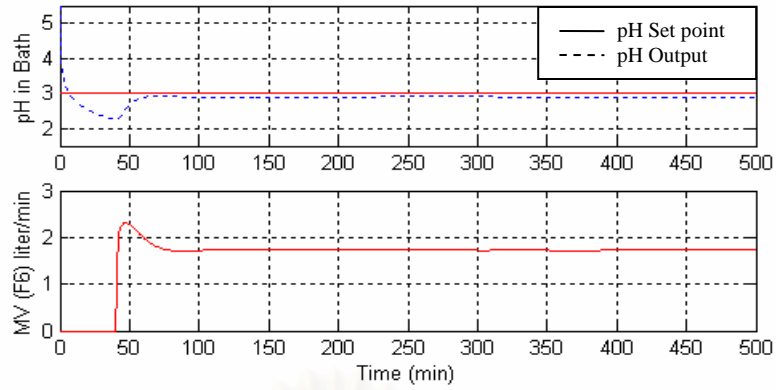


(a)

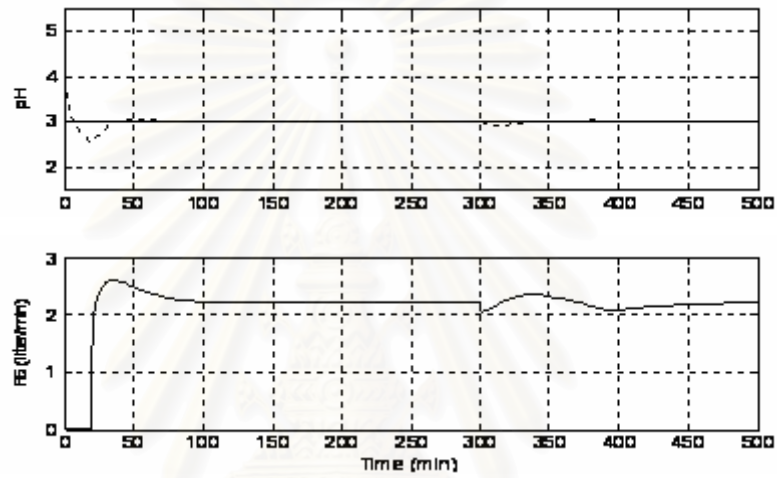


(b)

Figure 6.22 - Concentration control in 15% HCl bath under the model mismatch case (15% increase of the reaction rate, k): (a) NNDIC (b) PI control.

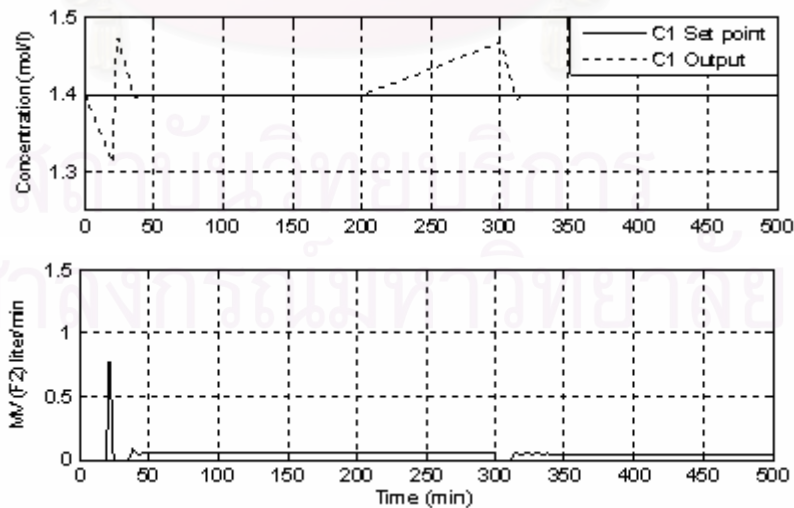


(a)



(b)

Figure 6.23 - Concentration control in 1st rinsing bath under the model mismatch case (15% increase of the reaction rate, k): (a) NNDIC (b) PI control.



(a)

Figure 6.24 - Concentration control in 5% HCl bath under the model mismatch case (15% decrease of the reaction rate, k): (a) NNDIC (b) PI control.

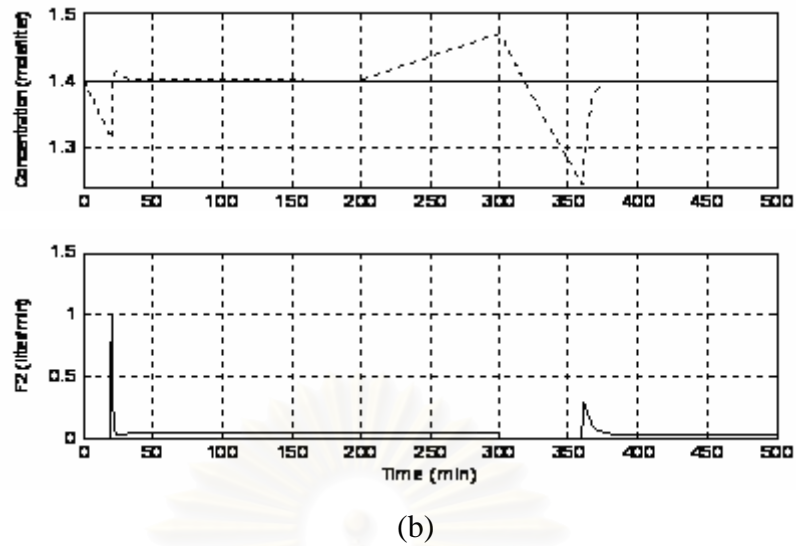


Figure 6.24 (Cont.) - Concentration control in 5% HCl bath under the model mismatch case (15% decrease of the reaction rate, k): (a) NNDIC (b) PI control.

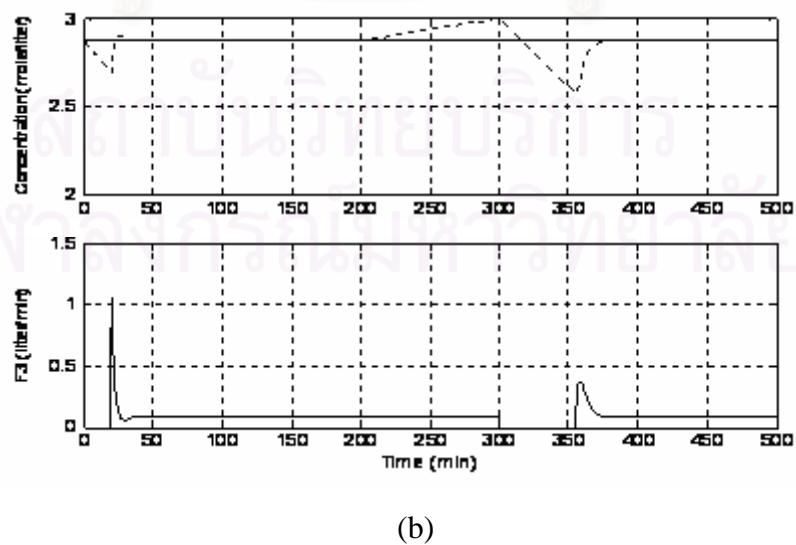
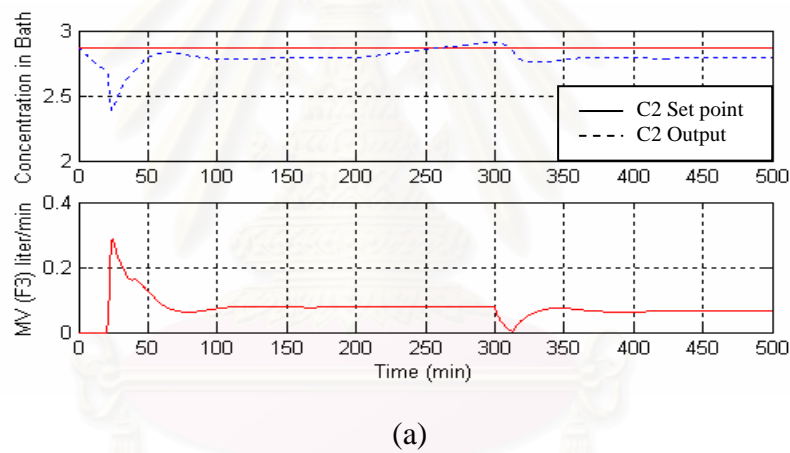
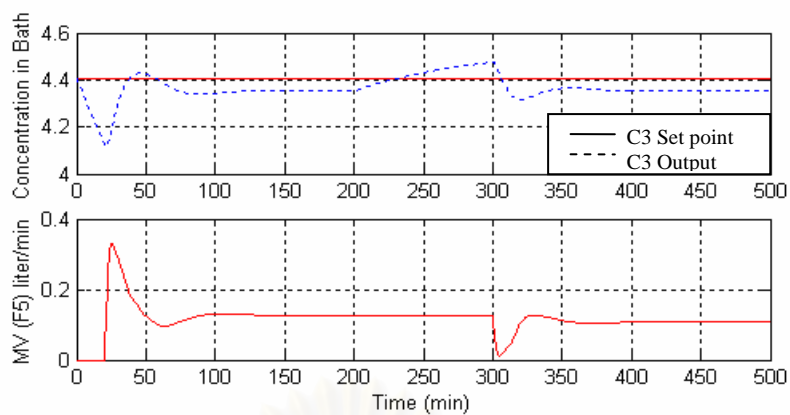
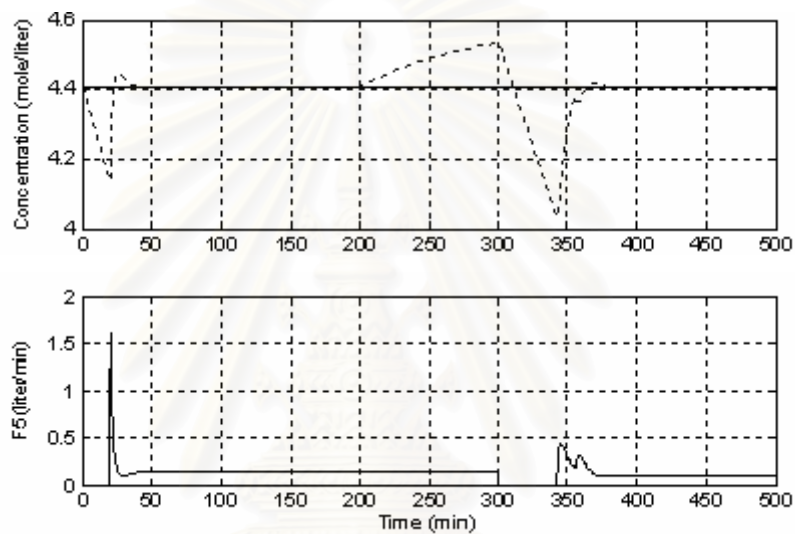


Figure 6.25 - Concentration control in 10% HCl bath under the model mismatch case (15% decrease of the reaction rate, k): (a) NNDIC (b) PI control.

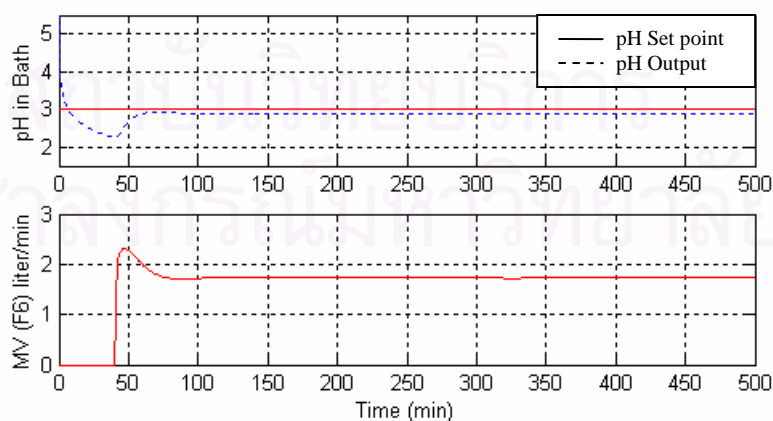


(a)



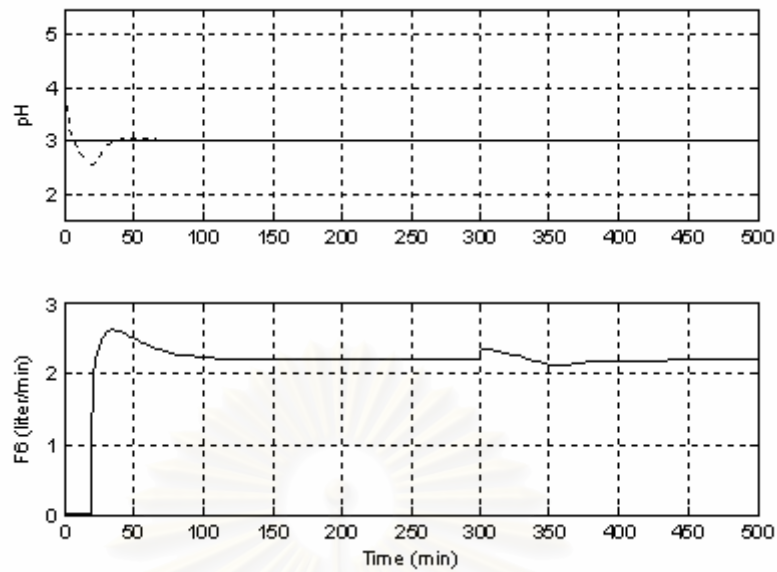
(b)

Figure 6.26 - Concentration control in 15% HCl bath under the model mismatch case (15% decrease of the reaction rate, k): (a) NNDIC (b) PI control.



(a)

Figure 6.27 - Concentration control in 1st rinsing bath under the model mismatch case (15% decrease of the reaction rate, k): (a) NNDIC (b) PI control.



(b)

Figure 6.27(Cont.) - Concentration control in 1st rinsing bath under the model mismatch case (15% decrease of the reaction rate, k):
(a) NNDIC (b) PI control.

Table 6.3 Performance comparison between NNDIC and PI control under the model mismatch case (15% increase of the reaction rate, k).

Bath	IAE Values	
	NNDIC	PI
5% HCl Bath	11.116	181.503
10% HCl Bath	101.759	244.167
15% HCl Bath	73.782	209.544
1 st Rinsing Bath	0.424	0.696

6.2.4 Noise case

Noises, accounting to 2% random values from the output measurement, are introduced into the system to further test its robustness and performance of the NNDIC approach under close to real situations. They are introduced under the model mismatch and disturbance cases as mentioned previously. The steps are the same as in the model mismatch and disturbance case, i.e. during the period $t = 200\text{--}300$ min, the process control action is halted to the latest value to allow the process to respond to the model mismatch and disturbance load conditions. At $t = 300$ min, the neural network-based control action is introduced back into the system. The results in figure 6.28 and 6.29 show that the NNDIC strategy can control the system in the both cases (model mismatch with noise and disturbance with noise) and bring the concentrations to desired set points in all these baths. Table 6.4 shows the IAE values of NNDIC for the four baths in both cases under noise effects. These results show the robustness and stability of the NNDIC when dealing with noise and disturbance effects simultaneously.

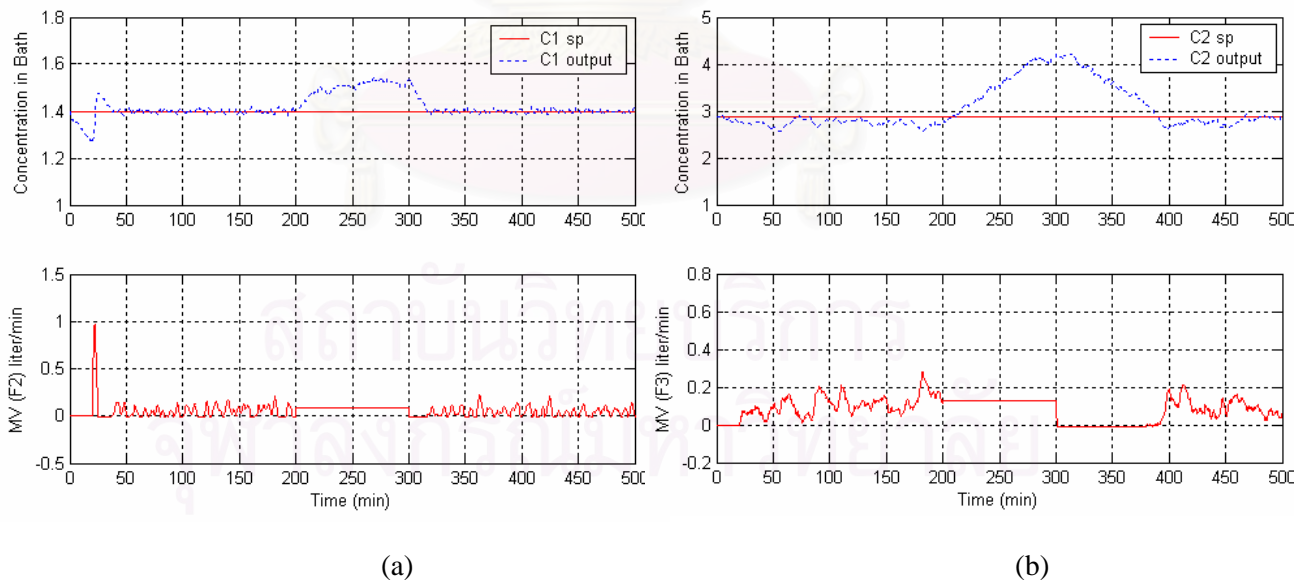


Figure 6.28 - Concentration control by NNDIC under the noise case with the disturbance in C_{20} (+15%): (a) 5% HCl Bath (b) 10% HCl Bath (c) 15% HCl Bath (d) 1st rinsing Bath.

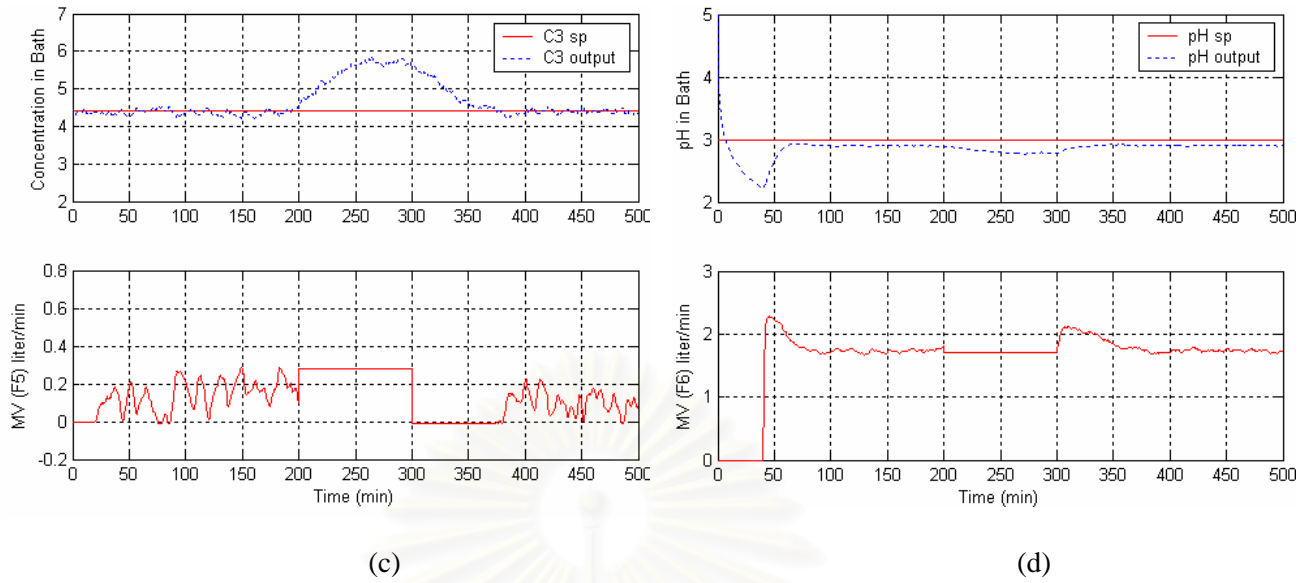


Figure 6.28(Cont.) - Concentration control by NNDIC under the noise case with the disturbance in C_{20} (+15%): (a) 5% HCl Bath (b) 10% HCl Bath (c) 15% HCl Bath (d) 1st rinsing Bath.

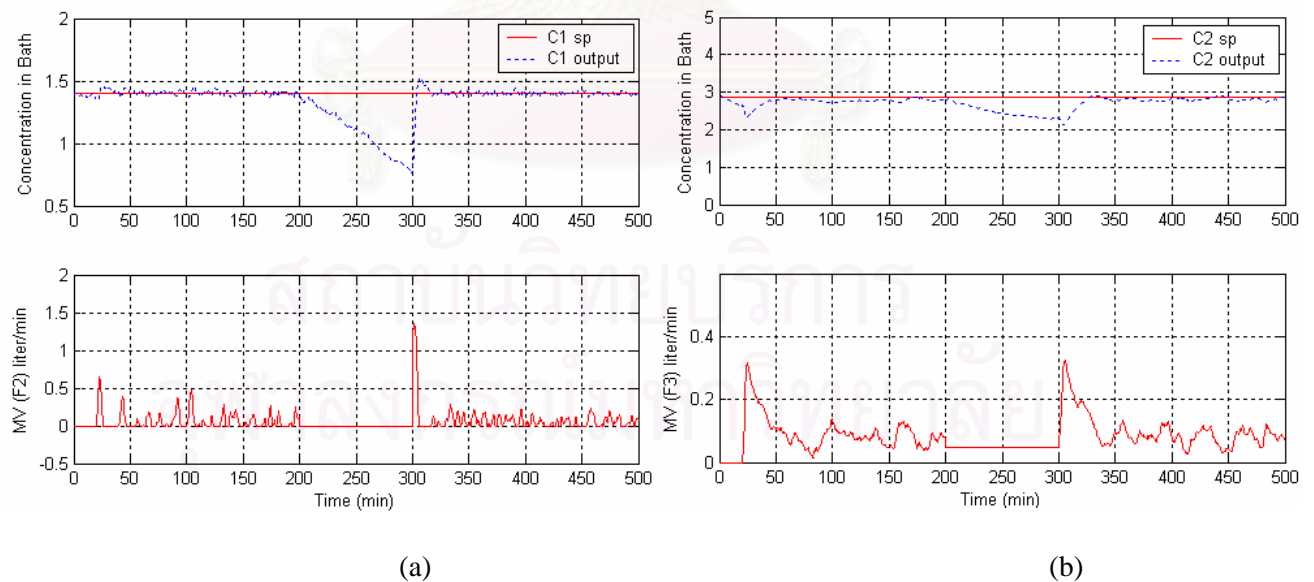


Figure 6.29 - Concentration control by NNDIC under the noise case with the model mismatch in k (+15%): (a) 5% HCl Bath (b) 10% HCl Bath (c) 15% HCl Bath (d) 1st rinsing Bath.

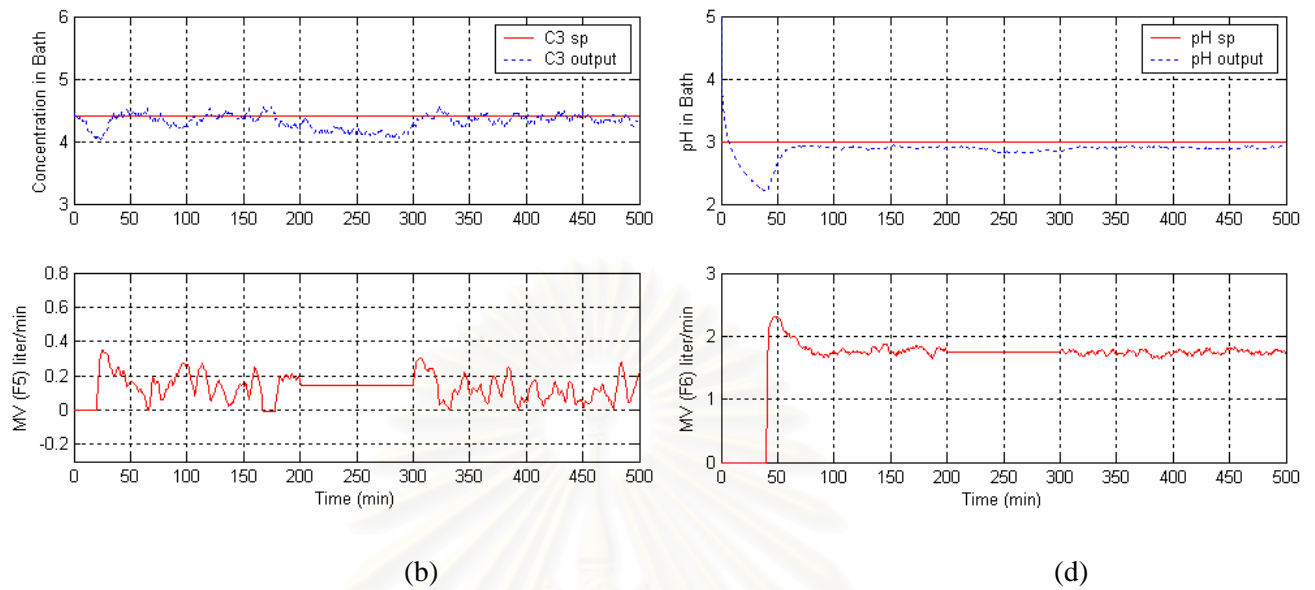


Figure 6.29(Cont.) - Concentration control by NNDIC under the noise case with the model mismatch in k (+15%): (a) 5% HCl Bath (b) 10% HCl Bath (c) 15% HCl Bath (d) 1st rinsing Bath.

Table 6.4 Performance of NNDIC under the noise case.

Bath	IAE Values	
	disturbance with noise	model mismatch with noise
5% HCl Bath	78.139	56.6884
10% HCl Bath	345.742	245.509
15% HCl Bath	317.110	179.934
1 st Rinsing Bath	0.501	0.472

6.3 Dual Mode Control based on Neural Network Inverse Model Strategy

Normally, the implementation of the inverse neural network model to control the process gives some errors between set point and control variable (offset) since an exact inverse model is difficult to obtain as have been proved theoretically (Hussain, 1997) and observed from the previous section. In order to remove this offset and improve the process response, proportional-integral (PI) controller is implemented in this proposed dual mode control strategy which makes use of both inverse neural network and PI controller. The basis concept of the dual mode algorithm can be divided into two modes of operation. That is, in the first mode, the inverse neural network controller is applied whenever error between the state (control variables) and set point lies outside the limit values, E (Equation (6.1)), while a controller in the second mode, the PI controller is employed inside the limit error region to bring the state to the desired set point. In this work the limited values is set of $\pm 3\%$ of the set point in each baths, where E is defined as:

$$|C_{sp} - C(k)| = E \quad (6.1)$$

The main benefit of the dual mode controller is that, under nominal operating condition when the state are located far away from set point, the inverse neural network controller can bring the state to the desired set point without drastic change of the manipulated variable and oscillation. However, when the state are located within limited region (E), PI controller start to control and bring the state to the desired set point without offset which normally occur when controlling using the inverse neural network controller only. In addition, the control action given by the PI controller in dual mode gradually changes and is less drastic as compared when using the conventional controller (PI) alone.

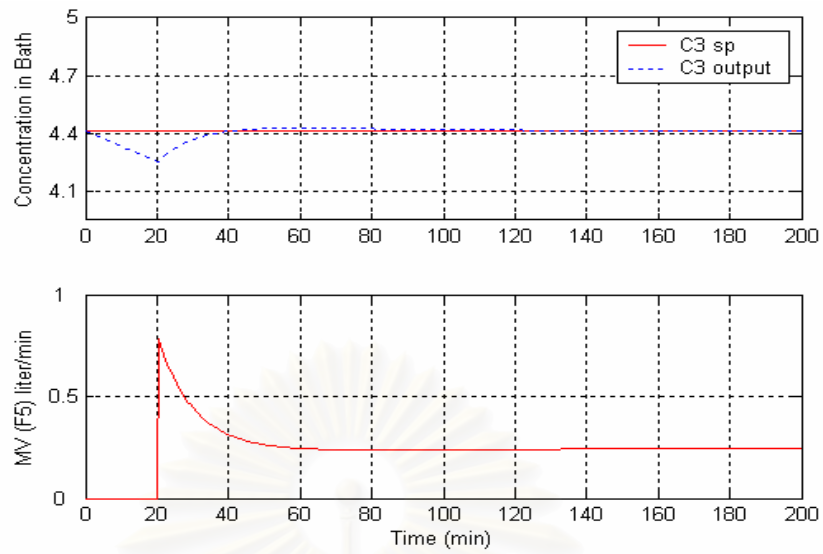
6.4 Results and Discussion of Dual Mode Control

The objective of the simulation studies is to control the concentration of HCl in the 5% HCl, 10% HCl, 15% HCl and 1st rinsing bath to values of 1.40, 2.87, 4.41 and

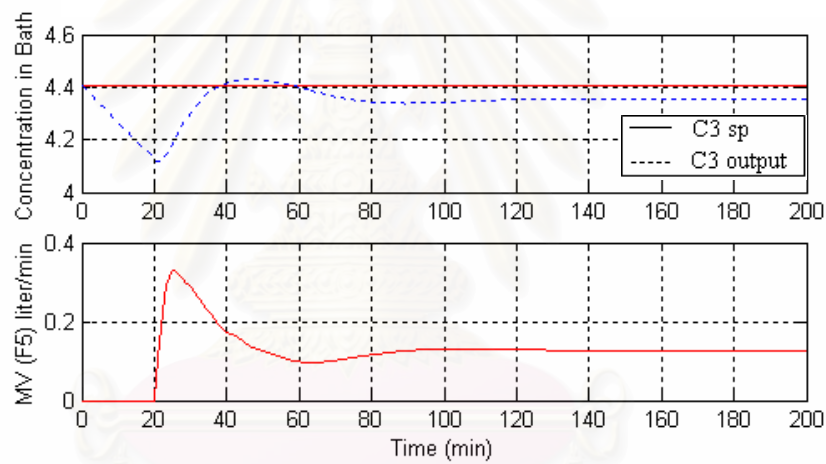
1×10^{-3} mol/liter (pH 3) by adjusting the manipulated variable F2, F3, F5 and F6 respectively. They are divided into three cases of control studies, which are the nominal case, disturbance case and model mismatch case, respectively.

6.4.1 Nominal case

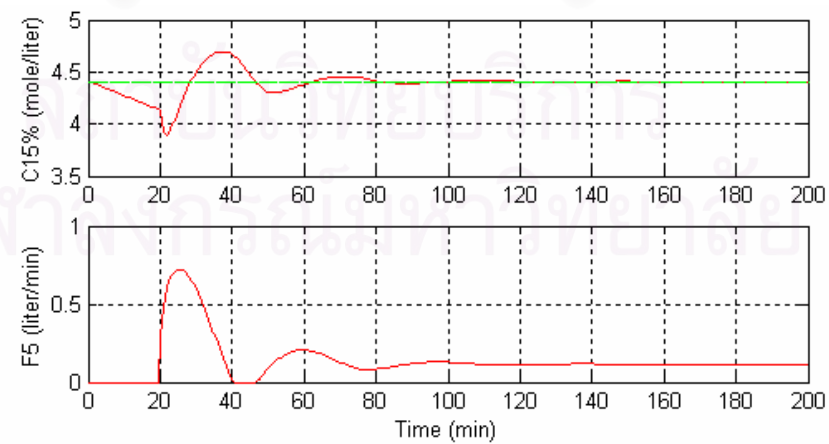
The controllers are designed to bring the concentration of HCl in each bath to the desired value when the initial condition is set at the steady state and leave the process without controller for 20 min after that start to control by controllers. Figure 6.30 (a), (b) and (c) show the control of HCl concentration in 15% HCl bath using dual mode control, NNDIC and PI control, respectively. The results in these figures indicate that dual mode controller can bring the concentration to the desired set point without any offset and oscillation of manipulated variable while the NNDIC bring the concentration closely to the set point with minimal offset. The PI controller bring the concentration to the set point without offset but however there is drastic change of the manipulated variable and large oscillation in the initial state of the control. These control strategies are also applied for the 5% HCl, 10% HCl and 1st rinsing bath; however, only the 15% HCl bath control result is given because this bath has highest HCl concentration and is very difficult to control, from which we can observe the effect of the controllers. Their performances are also evaluated using the Integral Absolute Error (IAE). The IAE results for the nominal case of these four baths are summarized in Table 6.5. They showed that relatively, the PI controllers give better results than dual mode control and NNDIC in term of lesser IAE values for the 5% HCl, 10% HCl and 1st Rinsing Baths. For the 15% HCL bath, the control action of PI controller is very drastic causing overshoot of concentration and therefore higher IAE value is obtained as compared to the DM controller.



(a)



(b)



(c)

Figure 6.30 - Concentration control in 15% HCl bath under the nominal case :

(a) DM control (b) NNDIC (c) PI control

Table 6.5 Performance comparison between Dual mode control, NNDIC and PI control under the nominal case

Bath	IAE Values		
	DM	NNDIC	PI
5% HCl Bath	1.756	2.109	0.342
10% HCl Bath	3.425	37.925	2.607
15% HCl Bath	4.967	21.697	13.281
1 st Rinsing Bath	0.044	0.274	0.007

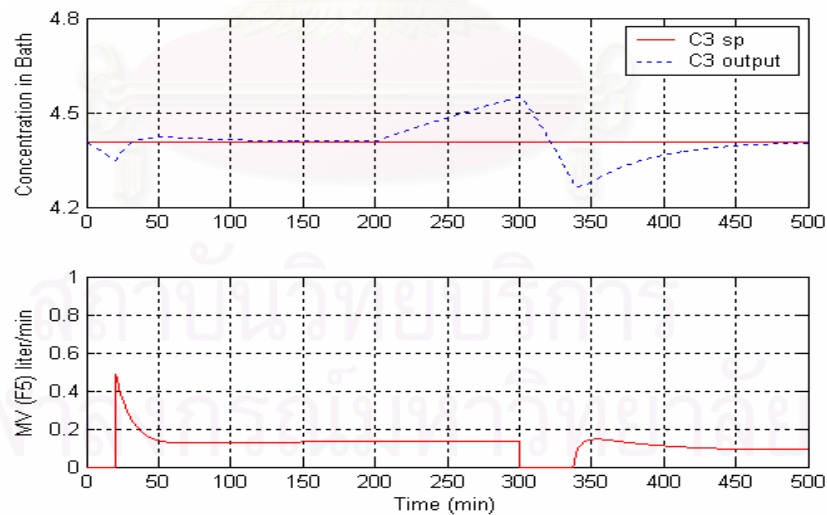
6.4.2 Disturbance case

In this case, the disturbance, which is the change in the concentration of C_{20} in the stream F_5 , is introduced by increasing and reducing 15% from its nominal operation values. Initially, the process is left under control until $t=200$ minutes, at which instant, a disturbance is introduced. During the period $t=200$ to 300 min, the control action is halted to the last value to allow the process to respond to the new load condition. At $t=300$ min, the DM, NNDIC and PI control action are introduced back into the system. Figure 6.31 shows the results of the DM, NNDIC and PI control for 15% HCl bath by increasing 15% of C_{20} from its nominal value. It can be seen from figure 6.31 that when the disturbance is introduced ($t=200$ to 300 min), the process responds by the increase in concentration of the baths due to the increase of HCl concentration (C_{20}) to 15% HCl bath. After $t=300$ min, the DM strategy can bring back the concentrations to the required value without offset and oscillation, while NNDIC strategy bring back the concentrations close to the required value and give offsets and the PI control strategy cannot bring the concentrations to the set point. Relatively, similar results are obtained in the 10% HCl bath and 1st rinsing bath. In 5% HCl bath PI controller can bring the concentration to its set point but there is rigorous adjusting of the F_2 causing overshoot in concentration. In reducing the concentration C_{20} by 15%, the results show that the DM strategy can still control the process and bring the concentrations to their set points, while NNDIC strategy bring the concentrations to their set points with slight offsets and PI controller bring the

concentrations to the set point with oscillation. Table 6.6 summarizes the IAE values of DM control, NNDIC and PI control for the four baths. They indicate that the DM controller has most robustness and give better control performance than the NNDIC controller and PI controller with very much smaller IAE error values, when disturbances are present in the system.

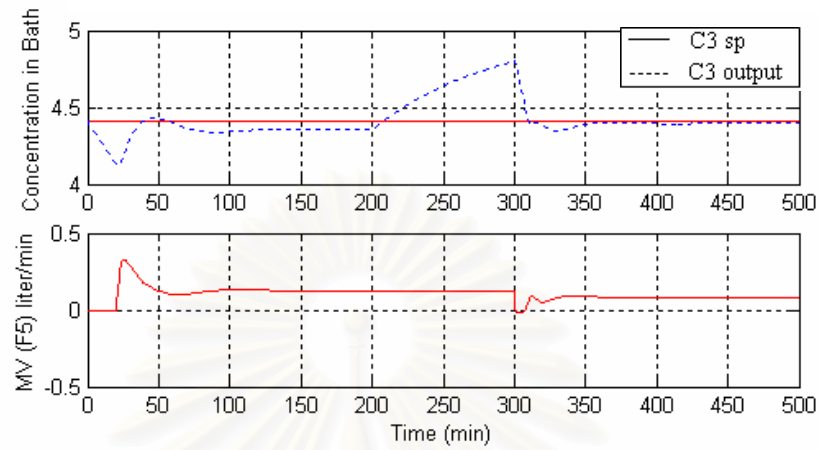
Table 6.6 Performance comparison between DM control, NNDIC and PI control under the disturbance case (15% increasing of the concentration, C_{20})

Bath	IAE Values		
	DM	NNDIC	PI
5% HCl Bath	1.183	3.508	12.252
10% HCl Bath	3.661	77.595	623.919
15% HCl Bath	38.492	74.997	696.903
1 st Rinsing Bath	0.052	0.444	0.087

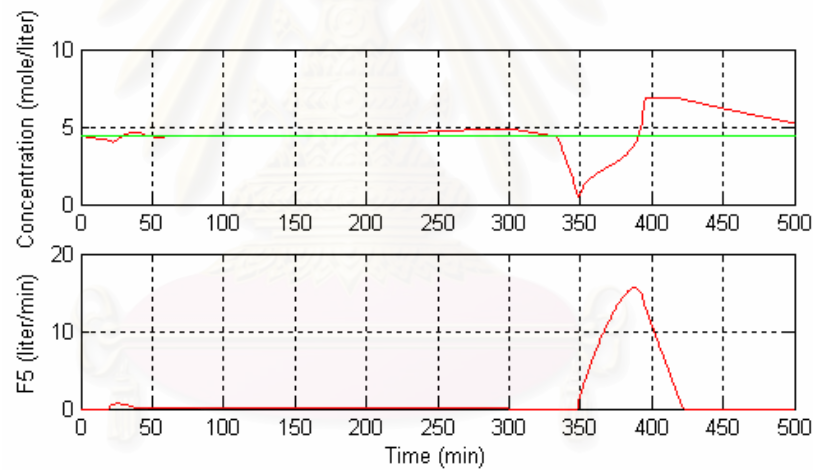


(a)

Figure 6.31 - Concentration control in 15% HCl bath under the disturbance case (15% increase of the concentration, C_{20}) : (a) DM control (b) NNDIC. (c) PI control.



(b)



(c)

Figure 6.31(Cont.) - Concentration control in 15% HCl bath under the disturbance case (15% increase of the concentration, C_{20}): (a) DM control (b) NNDIC. (c) PI control.

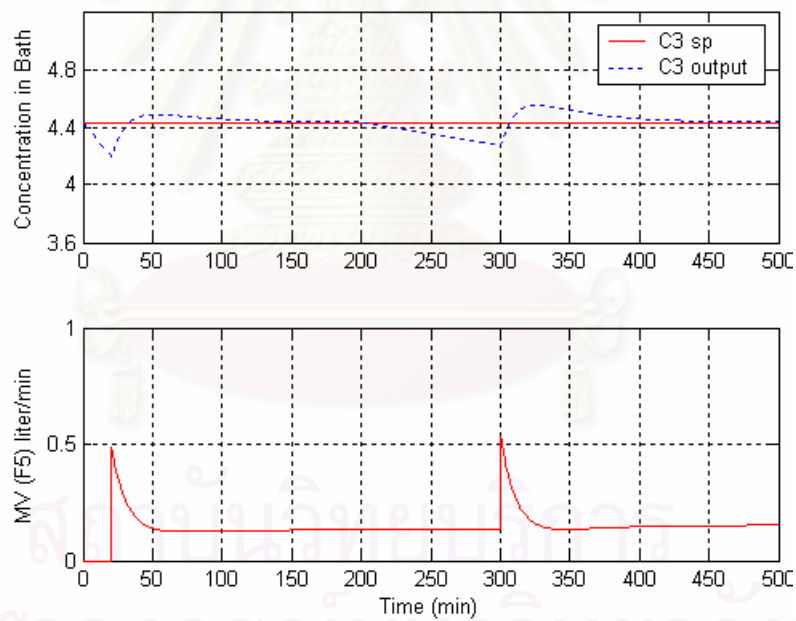
6.4.3 Model mismatch case

In this work, the rate of reaction in acid bath is considered as the model mismatch in the parameter under consideration. The model mismatch is introduced by increasing and reducing 15% of the kinetic rate constant from its nominal value. Initially, the process is left under control until $t=200$ min, at which instant, the model mismatch is introduced. During the period $t=200$ to 300 min, the process control action is halted to the latest value to allow the process to respond to new load condition. At $t=300$ min, the DM, NNDIC and PI control action are introduced back into the system. Figure 6.32 shows the results of DM, NNDIC and PI control for 15% HCl bath by increasing 15% of the rate of reaction. When the model mismatch is introduced ($t=200$ to 300 min), the process responds by a decrease in concentration in baths due to the increase of rate of reaction in acid bath. After $t=300$ min, the DM control bring back the concentrations to their required values without the offset, while NNDIC strategy bring back the concentrations close to their required values and give offsets and PI control strategy bring the concentrations to the set point with drastic change of the manipulated variable and oscillation.

In reducing 15% of the rate of reaction, the results again show that the DM control can control the process and bring the concentrations to their set points, while NNDIC strategy can control the process and bring the concentrations close to their set points with slight offsets and PI control strategy bring the concentrations to the set point with drastic change of the manipulated variable and oscillation. Table 6.7 shows the IAE values of DM control, NNDIC and PI control for the four baths. They again indicate that the DM control has most robustness and give better control performance than NNDIC and PI controller, similar to the disturbance case study.

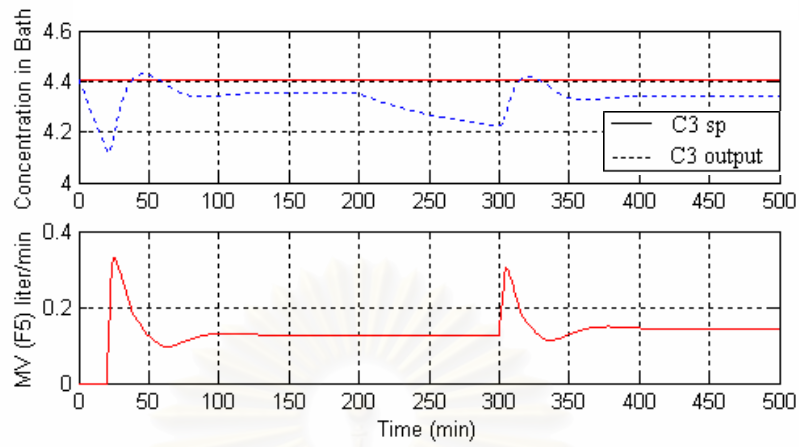
Table 6.7 Performance comparison between DM control, NNDIC and PI control under the model mismatch case (15% increase of the reaction rate, k)

Bath	IAE Values		
	DM	NNDIC	PI
5% HCl Bath	3.998	11.116	136.385
10% HCl Bath	8.162	101.759	220.565
15% HCl Bath	11.665	73.782	289.428
1 st Rinsing Bath	0.049	0.424	0.041

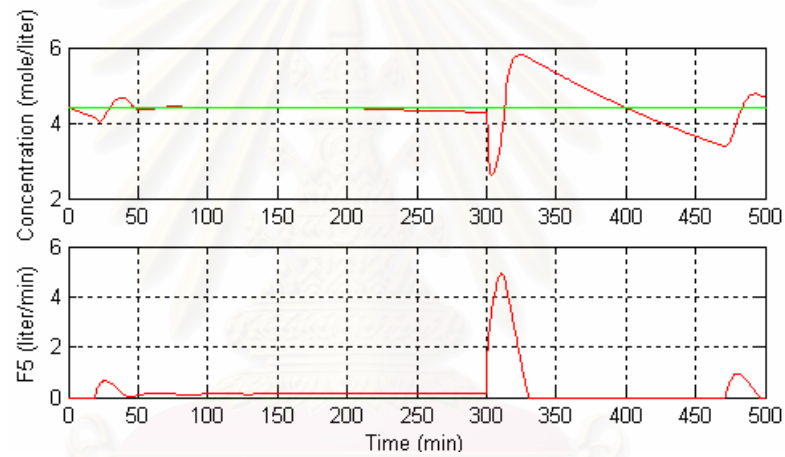


(a)

Figure 6.32 - Concentration control in 15% HCl under the model mismatch case (15% increase of reaction rate, k): (a) DM control (b) NNDIC (c) PI control



(b)



(c)

Figure 6.32(Cont.) - Concentration control in 15% HCl under the model mismatch case

(15% increase of reaction rate, k): (a) DM control.

(b) NNDIC. (c) PI control.

CHAPTER VII

CONCLUSIONS

The objective of this research is to develop and implement an advanced control scheme for the control of a nonlinear multivariable chemical process. Since the real chemical processes are non-linear and multivariable interacting systems, which make them difficult to control by using conventional controllers, model-based advance control techniques are then required to obtain tighter control.

The work presented in this dissertation studies on a model predictive control based on neural network (NNMPC) to control a steel pickling process which has highly nonlinear dynamic behavior and involves multivariable interactions in nature. Since the MPC controller uses a model of controlled process in its algorithm to determine manipulated variables then the modeling of the process is very importance. However, in many cases it is even impossible to obtain a suitable process model due to the complexity of the underlying processes or the lack of knowledge of critical parameters of the models. Therefore, in this work, the neural network is used to develop the model of a steel pickling process. The developed neural network models are then used in the MPC algorithm. In addition to implementation of inverse neural network (InvNN) and Dual Mode controller (DM) to a steel pickling process is investigated. The main issues studied in this research are summarized below.

7.1 A Steel Pickling Process Modeling

For system identification based on neural networks, process data are prepared and used to train and validate neural network models. In this work, neural networks process models based on input-output information have been developed to predict the hydrochloric acid concentration of a steel pickling process which has highly nonlinear dynamic behavior and multivariable interaction. Various neural networks architectures have been trained using Lenvenberg- Marquardt techniques, and the

accuracy of the obtained models has been evaluated using validation data set. The optimal neural network architectures are obtained using minimum MSE technique. The simulation results have shown that the multilayer feedforward neural network models provide sufficiently accurate prediction of acid concentrations and pH values of the process. Therefore, with the obtained neural network models, advanced model based control techniques, i.e., NNMPC, are then applicable to control the steel pickling process.

7.2 Neural Network Direct Inverse Control (NNDIC) and Dual Mode Control (DM)

A neural network direct inverse control (NNDIC) strategy was tested and implemented for controlling the concentrations of pickling and rinsing baths in a steel pickling process which is highly non-linear and involves multivariable interactions in nature. It was observed that the neural network inverse model-based control strategy can bring the controlled variables closely to their set points with minimal oscillations in all cases studied, i.e., nominal case, disturbance case, model mismatch case and noise case. Comparison of performance with the conventional PI controller indicated that NNDIC was more robust than the PI controller and gave better results in cases involving disturbances, model mismatches and noise. The PI gave slightly better results in the nominal case in term of offset rejection but rigorous oscillation was observed as compared to the neural network method.

Nevertheless, inverse neural network model is not the exact model, and then normally the offset occurs when implemented in control of the process as mention above. Therefore, the dual mode is an attractive control methodology for process control application because it can remove the offset and improves robustness of the controller. In this work dual mode control strategy is tested and implemented for controlling concentrations of pickling and rinsing baths in a steel pickling process. Lenvenberg- Marquardt techniques and MSE minimization technique are used for training and choosing the optimal inverse neural network model structures, respectively. It was observed that DM control strategy can bring the control variables to their set points without offset and oscillations in all cases studies, i.e., nominal

case, disturbance case and model mismatch case. Comparison of performance with NNDIC and the conventional PI controller indicated that DM control strategy gave better results in cases involving disturbances, and model mismatches which gave lesser IAE values than NNDIC and PI control. These results show that DM control strategy give good control results for highly nonlinear and multivariable system such as the steel pickling process, and can improve the robustness of control system and remove the offset when compared to the NNDIC and conventional PI control strategy.

7.3 Neural Network based Model Predictive Control (NNMPC)

The implementation of a neural network model based predictive controller, MIMO controller, to a steel pickling process is investigated. The obtained multilayer feedforward neural network models have been employed to predict the future process response in MPC algorithm for controlling the concentrations of pickling in a steel pickling process. It was observed that NNMPC can bring the control variables to their desired set points without oscillations and drastic changing of manipulated variables in all cases studies, i.e., set point tracking case, disturbance case, model mismatch case and noise case. Comparison of performance with the conventional PI controller indicated that NNMPC was more robust than the PI controller and gave better control results in cases involving disturbances, model mismatches and noise. These results show that NNMPC controllers are robust in nature and highly promising to be implemented in such highly nonlinear multivariable systems such as the steel pickling process.

REFERENCES

- Acosta, L., Marichal, G. N., Moreno, L., Rodrigo, J. J., Hamilton, A. and Mendez, J. A., A robotic system based on neural network controllers. Artificial Intelligence in Engineering 13(1999): 393-398
- Agatonovic-Kustin, S., Zecevic, M., Zivanovic, L. J. and Tucker, I. G., Application of artificial neural networks in HPLC method development. Journal of Pharmaceutical and Biomedical Analysis 17(1998): 69-76.
- Alvarez, E., Riverol, Carmen, Navaza, J.M., Control of chemical processes using neural networks: implementation in a plant for xylose production. ISA Transactions 38(1999): 375-382.
- Anderson, J. A. and Rosenfeld (Eds.), E., Neurocomputing: Foundations of Research. Cambridge: MIT Press, 1988.
- Asakawa, K. and Takagi, H., Neural networks in Japan. Communications of the ACM. 37, 3(1994): 106-112.
- Balchen, J. G., Ljungquist, D. and Strand, S., State-space predictive control. Chemical Engineering Science 47(1992): 787-807.
- Batti, R., Accelerated backpropagation learning : two optimization methods. Complex Systems 3(1989): 331-342.
- Becen-a V. M., Roberts, P. D. and Griffiths, G. W., Novel developments in process optimisation using predictive control. J. Proc. Cont 8(1998): 117-138.
- Bequette, B. W., Nonlinear control of chemical processes: A review. Ind. Eng. Chem. Res 30(1991): 1391-1413.
- Bitmead, R. R., Gevers, M. and Wertz, V., Adaptive Optimal Control. The Thinking Man's GPC: Prentice Hall Intemational, 1990.
- Bless, R. R. and Hodges, D. H., Finite element solution of optimal control problems with inequality constraints. Proceedings of the 1990 American Control Conference(May 1990): 242-247.
- Bremermann, H. J. and Anderson, R. W., An alternative to backpropagation: A simple rule for synaptic modification for neural net training and memory. Internal report, Dept. of Mathematics, Univ. of California Berkley, 1989.

- Brown, M. and Harris, C., Neurofuzzy Adaptive Modelling and Control. Prentice-Hall, 1994.
- Brusch, R. G., A nonlinear programming approach to space shuttle trajectory optimization. J Optimization Theory Appl. 13(1974): 94-118.
- Camacho, E. F. and Bordons, C., Model Predictive Control in the Process Industry. Sprhger-Verlag: Berlin, Germany, 1995.
- Camacho, E. F. and Bordons, C., Model Predictive Control. Springer-Verlag. (an update of Camacho and Bordons), 1999.
- Chan, K. C., Leong, S. S., Lin, G. C. I., A neural network PI controller tuner. Artificial intelligence in Engineering 9(1995): 167-176.
- Charalambous, C., Conjugate gradient algorithm for efficient training of artificial neural networks. IEE Proc. G. 139, 3(1992).
- Clarke, D. W. (Eds.), Advances in Model-Based Control. Oxford University Press, 1994.
- Clarke, D. W. and Mohtadi, C., Properties of generalized predictive control. Automatica 25, 6(1989): 859-875.
- Clarke, D. W., Mohtadi, C. and Tuffs, P. S., Generalised Predictive Control - Part I. The basic algorithm. Automatica 23(1987): 137-148.
- Clarke, D. W. and Scattolini, R., Constrained receding horizon predictive control. IEE Proceedings D. 138(1992): 347-354.
- Cutler, C. R, Morshedi, A. and Haydel, J., An industrial perspective on advanced control. AICHE Annual Meeting, Washington, D.C., USA(1983).
- Cutler, C. R. and Ramaker, B. L., Dynamic matrix control-a computer control algorithm. AICHE National Meeting, Houston, TX(April 1979).
- Cutler, C. R and Ramaker, B. L., Dynamic matrix control - a computer control algorithm. Proceedings of the Joint Automatic Control Conference. San Francisco, USA(1980).
- Daosud, W., Thitiyasook, P., Arpornwichanop, A., Kittisupakorn, P. and Hussain M. A., Neural network inverse model-based controller for the control of a steel pickling process. Computers & Chemical Engineering 29, 10(2005): 2049-2264.
- De Keyser RM.C. and Van Cuawenberghe, A.R., Extended prediction self-adaptive control. Proceedings of IFAC Symposium on Identification and system Parameter Estimation. York, VK(1985): 1317-1322.

- De Keyser R.M. C., Van de Welde Ph. G. A. and Dumortier F. G. A., A comparative study of self-adaptive long range predictive control methods. Automatica. 24(1988): 149-163.
- Demircioglu, H. and Gawthrop, P. J., Continuous time generalised predictive control. Automatica. 27(1991): 55-74.
- Demircioglu, H. and Gawthrop, P. J., Multivariable continuous time generalised predictive control. Automatica. 28(1992): 697-713.
- Dennis, J. E. and Schnabel, R. B., Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Prentice-Hall, 1983.
- Doherty, S. K., Gomm, J. B., and Williams, D., Experiment design considerations for non-linear system identification using neural networks. Computers and Chemical Engineering. 21, 3(1997): 327–346.
- Dutta, P. and Rhinehart, R. R., Application of neural network control to distillation and an experimental comparison with other advanced controllers. ISA Transactions. 38(1999): 251-278.
- Eaton, J. W. and Rawlings, J. B., Model predictive control of chemical processes. Chemical Engineering Science 47(1992): 705-720.
- Economou, C. G., Morari, M. and Palsson, B. O., Internal model control 5. Extension to nonlinear systems. Ind. Eng. Chem. Process Des. Dev 25(1986): 403-411.
- Elman, J., Finding structure in time. Cognitive Science. 14(1990): 179-211.
- Fletcher, R. and Reeves, C. M., Function minimization by conjugate gradients. Comput. J. 7(1964): 149-154.
- Garcia, C. E., Prett, D. M. and Morari, M., Model predictive control: theory and practice-a survey. Automatica. 25(1989): 335-348.
- Gawthrop, P. J., Demircioglu, H. and Siller-Alcala I. I., Multivariable continuous-time generalised predictive control: A state-space approach to linear and nonlinear systems. IEE Proceedings D. 145(1998): 241-250.
- Gomm, J. B., Evans, J. T. and Williams, D., Development and performance of a neural network predictive controller. Control Engineering Practice. 5, 1(1997): 49–60.
- Grossberg, S., Nonlinear neural networks: Principles, mechanisms and architectures. Neural Networks 1, 1(1988): 17-62.

- Gupta, A. and Rhinehart, R. R., Experimental comparison of advanced control techniques on a lab-scale distillation column. Proc. Amer. Cont. Conf., Seattle, WA, paper FA 10-4(June 1994): 21-23.
- Hardcastle, V. G., What we don't know about brains, Stud. Hist. Phil. & Biomed. Sci. 1, 30(1999): 69-89.
- Haykin, S., Neural Networks : A Comprehensive Foundation. Second edition. Prentice Hall International, Inc., 1999.
- Hebb, D. O., The Organization of Behaviour. Wiley, 1949.
- Henson, M. A., Nonlinear model predictive control: Current status and future directions. Computers & Chemical Engineering. 23, 2(1998): 187-202.
- Hertz, J., Krogh, A. and Palmer, R. G., Introduction to the Theory of the Neural Computation. Addison-Wesley, 1991.
- Hestenes, M. R. and Stiefel, E., Methods of conjugate gradients for solving linear systems. J. of Research of the National Bureau of Standards. 49, 6(1952): 409-436.
- Hinton, G. E., Sejnowski, T. J. and Ackley, D., Boltzmann machines: Constraint satisfaction networks that learn. Technical Report CMU CS 84 119, Carnegie Mellon University, Pittsburgh, USA., 1984.
- Holland, J. M., Adaption in natural and artificial systems. University of Michigan Press, 1975.
- Hopfield, J. J., Neural networks and physical systems with emergent collective computational abilities. Proc. Of the National Academy of Scientists 79(1982): 2554-2558.
- Hunt, K. J., Sbarbaro, D., Neural networks for nonlinear internal model control. IEE Proceedings-D. 138, 5(1991): 431-438.
- Hussain, M. A., Transfer report, Neural networks for process control. Imperial College of Science and Technology, London, 1994.
- Hussain, M. A., in H. E. Rauch (ed.), Neural network inverse model control strategy discrete the analysis for relative order two system. Artificial intelligence in real-time control., 1997.
- Hussain, M. A., Review of the applications of neural networks in chemical process control-simulation and online implementation. Artificial Intelligence in Engineering 13(1999): 55-68.

- Hussain, M. A., Neural Network Techniques and Application in Chemical Process Control System. CRC Press, 2003.
- Hussain, M. A. and Kershenbaum, L. S., Implementation of an inverse-model-based control strategy using neural networks on a partially simulated exothermic reactor. Trans IChemE. 78, part A(March 2000): 299-311.
- Hussain, M. A., Ng, C. W, Aziz, N. and Mujtaba, I. M., Neural network techniques and applications in chemical process control systems. Intelligent systems Techniques and Applications. 5, N. J: CRC press, 2003.
- Irwin, G. W., Lightbody, G. and McLoone, S. F., Offline training of feedforward neural networks. Proc. Irish DSP and Control Conf., IDSPCC'94, Dublin, 1994.
- Jansh, C. and Paus, M., Aircraft trajectory optimization with direct collocation using movable gidpoints. Proceedings of the 1990 American Control Conference(May 1990): 262-267.
- Johansson, E. M., Dowla, F. U. and Goodman, D. M., Backpropagation learning for multilayer feedforward neural networks using the conjugate gradient method. Int. J. of Neural Systems. 2(1992): 291-301.
- Johnson, I. L., Optimization of the solid-rocket assisted space shuttle ascent trajectory. J. Spacecraft. 12(1975): 765-769.
- Ju, J., Chiu, M. S. and Tien, C., Multiple-objective based model predictive control of pulse jet fabric filters. Trans IChemE. 78 Part A(2000): 581-589.
- Kaeahan, O., Ozgen, C., Hahci, U. and Leblebicioglu, K., Nonlinear model predictive controller using neural network, IEEE.(1997): 690-693.
- Kilpatrick, S., Gelatt, C. D. and Vecchi, M. P., Optimization by simulated annealing. Science. 220(1983): 671-680.
- Kittisupakorn, P. and Kaewpradit, P., Integrated data reconciliation with generic model control for the steel pickling process. Korean J. Chem. Eng. 20(2003): 985.
- Kleinman, D. L., An easy way to stabilize a linear constant system. IEEE Trans. Autom. Control. 15(1970): 692.
- Kohonen, T., An introduction to neural computing. Neural Networks. 1, 1(1988): 3-16.

- Kouvaritakis, B., Rossiter J. A. and Chang A. O. T., Stable generalised predictive control: An algorithm with guaranteed stability. IEE Proceedings D. 139(1992): 349-362.
- Kramer, K. and Ubehauen, H., Predictive adaptive control. Comparison of main algorithms. Proceedings of European Control Conference. Grenoble, France(1991): 327-332.
- Kwon, W. H., Bruckstein, A. M. and Kailath, T., Stabilizing state-feedback design via the moving horizon method. Int. J. Control 37(1983): 631-643.
- Kwon, W. H. and Pearson, A. E., A modified quadratic cost problem and feedback stabilization of a linear system. IEEE Trans. Autom. Control 22(1977): 838-842.
- Le Chun, Y., Simard, P. Y. and Pearlmutter, B., Automatic learning rate maximization by online estimation of the Hessian's eigenvectors. In: Hanson, S. J., Cowan, J. D. and Giles, C. L. (eds), Advances in neural information processing systems. 5(1993): 156-163.
- Leger, R. P., Garland, W. J. and Poehlmann, W. F. S., Fault detection and diagnosis using statistical control charts and artificial neural networks. Artificial intelligence in engineering. 12(1997): 35-47.
- Lennox, B., Montague, G. A., Frith, A. M. and Beaumont, A. J., Non-linear model-based predictive control of gasoline engine air– fuel ratio. Transactions of the Institute of Measurement and Control. 20, 2(1998): 103–112.
- Leonard, J. A. and Kramer, M. A., Improvements of the backpropagation algorithm for training neural networks. Computers Chem. Engng. 14(1990): 337- 341.
- Liew, S. H., Ho, P.Y., Hussain, M. A., and Kittisupakorn, P., Neural networks in adaptive control techniques, Regional Symposium of Chemical Engineering 1999, Songkhla, Thailand, 2(1999): B23-1-B23-6.
- Lightbody, G., and Irwin, G. W., Non-linear control structures based on embedded neural system models. IEEE Transactions on Neural Networks. 8, 3(1997): 553–567.
- Lightbody, G., O'Reilly, P., Irwin, G. W., Kelly, K. and McCormick, J., Neural modelling of chemical plant using MLP and b-spline networks. Control Engineering Practice. 5, 11(1997): 1501–1515.

- Lin E. K. and Stephen, S. M., Wastewater neutralization control using a neural network based model predictive controller, Proceedings of the American Control Conference, Philadelphia, Pennsylvania(June 1998): 3896-3898.
- Lu, S. and Hogg, B. W., Dynamic nonlinear modeling of power plant by physical principles and neural networks. Electrical Power and Energy Systems. 22(2000): 67-78.
- Luenberger, D. G., Linear and Nonlinear Programming. Addison-Wesley, 1984.
- Maciejowski, J., Predictive Control With Constraints. Addison Wesley Longman, 2000.
- Marquardt, An algorithm for least squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Maths.(1963): 434-441.
- Mayne, D. Q. and Michalska, H., Receding horizon control of nonlinear systems. IEEE Trans autom. Control 35(1990): 814-824.
- McCulloch, W. S. and Pitts, W., A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics. 5(1943): 115-133.
- Mehra, R. K., Rouhani, R., Eterno, J., Richalet, J. and Rault, A., Model algorithmic control: review and recent development, Engineering Foundation Conference on Chemical Process Control II.(1982): 287-310.
- Mei-J. S, Cheng-L., H., A neural network study on the dynamic identification of a fermentation system. Bioprocess Engineering. 17(1997): 203-213.
- Minsky, M. and Papert, S., Perceptrons: Expanded Edition. MIT Press, 1969.
- Mosca, E. and Bang, J., Stable redesign of predictive control. Automatica. 28(1992): 1229-1233.
- Nahas, E. P., Henson, M. A. and Seborg, D. E., Nonlinear internal model control strategy for neural network models. Computers Chem. Engng. 16, 12(1992): 1039-1057.
- Narendra, K. S. and Parthasarthy, K., Identification and control of dynamical systems using neural networks. IEEE Trans. Neural Networks. 1(1990): 1-27.
- Nikraves, M., Farrell, A. E. and Stanford, T.G., Control of nonisothermal CSTR with time varying parameters via dynamic neural network control (DNNC). Chemical Engineering Journal. 76(2000): 1-16.
- Norquay, S. J., Palazoglu, A. and Romagnoli, J. A., Application of Wiener model predictive control (WMPC) to an industrial C2-splitter. Journal of Process Control 9(1999): 461-473.

- Nougues, J. M., Pan, Y. G., Velo, E. and Puigjaner, L., Identification of a pilot scale fluidized-bed coal gasification unit by using neural networks. Applied Thermal Engineering. 20(2000): 1561-1575.
- Ordys A. W. and Clarke D. W., A state-space description for GPC controllers. Int. J. Sys. Sci. 29(1993): 1727-1744.
- Petrova, M., Koprinkova, P., Patarinska, T. and Bliznakova, M., Neural network modeling of fermentation processes, specific growth rate model. Bioprocess Engineering 18(1998): 281-287.
- Pham, D. T., An introduction to artificial neural networks, Neural Networks for Chemical Engineerings, A.B. Bulsari (Editor), Computer-Aided Chemical Engineering 6, Chapter 1(1995): 1-19.
- Pham, D. T. and Oh, S. J., Identification of plant inverse dynamics using neural networks. Artificial Intelligence in Engineering. 13(1999): 309-320.
- Pivonka, P. and Zizka, J., Neural controllers in real-process control, Applications of Artificial Intelligence. Advanced Manufacturing Forum. 1, Edited by Nuno J. Mamede and Carlos Pinto-Ferreira, Scitec Publications Ltd, 1996: 255-264.
- Prett, D. M. and Garcia, C. E., Fundamental Process Control. Butterworths, Boston, USA., 1988.
- Prett, D. M. and Gillette, R. D., Optimization and constrained multivariable control of a catalytic cracking unit, AICHE National Meeting, Houston, TX(April 1979).
- Propoi, A. I., Use of linear programming methods for synthesizing sampled-data automatic systems. Autom. Remote Control 24(1963): 837-844.
- Psiaki, M. L. and Park, K., Trajectory optimization for real-time guidance: Part 1, time-varying LQR on a parallel processor, Proceedings of the 1990 American Control Conference.(May 1990): 248-253.
- Psichogios, D. C. and Ungar, L. H., Direct and indirect model based control using artificial neural networks. Ind. Eng. Chem. Res. 30(1991): 2564-2573.
- Qin J. and Badgwell T. A., An overview of industrial model predictive control technology. Proceedings of Chemical Process Control V. Tahoe City, California, AICHE Symposium Series 316, 93(1997): 232-256.
- Radhakrishnan, V. R. and Mohamed, A. R., Neural networks for the identification and control of blast furnace hot metal-quality. Journal of Process Control. 10(2000): 509-524.

- Rawlings, J. B., Tutorial Overview of Model Predictive Control, IEEE Control Systems Magazine.(June 2000): 38-52.
- Richalet, J., Industrial applications of model based predictive control. Automatica. 29(1993): 1251-1274.
- Richalet, J., Rault, A., Testud J. L. and Papon J., Algorithmic control of industrial processes. Proceedings of 4th IFAC Symposium on Identification and System Parameter Estimation, Tbilisi, URSS, 1976.
- Richalet, J., Rault, A., Testud J. L. and Papon J., Model predictive heuristic control: Application to industrial processes. Automatica. 14(1976): 413-428.
- Roberts, P. D., A brief overview of model predictive control. (Special Feature: Model Predictive Control: Editorial). Computing & Control Engineering Journal. 10(1999): 186-188.
- Robitaille, B., Marcos, B., Veillette, M. and Payre, G., Modified quasi-Newton methods for training neural networks. Computers Chem. Engng. 20, 9(1996): 1133-1140.
- Rohani, S., Haeri, M., and Wood, H. C., Modelling and control of a continuous crystallization process—Part 2: Model predictive control. Computers & Chemical Engineering. 23, 3(1999): 279–286.
- Rosenblatt, F., Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. Spartan Books, 1962.
- Rossiter, J. A., Gossner, J. R. and Kouvaritakis, B., Constrained cautious stable predictive control. IEE Proceedings D. 144(1997): 313-323.
- Rossiter, J. A. and Kouvaritakis B., Constrained stable generalised predictive control. IEE Proceedings D. 140(1993): 243-254.
- Rossiter, J. A., Kouvaritakis, B. and Gossner, J. R., Guaranteeing feasibility in constrained stable generalized predictive control. IEE Proceedings D. 143(1996): 463-469.
- Rumelhart, D. E., Hinton, G. E. and Williams R. J., Learning internal representations by error propagation. In: Rumelhart, D.E. and McClelland (eds.), Parallel distributed processing., MIT Press, 1986.
- Rumelhart, D. E. and McClelland, J. L. (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations. MIT Press, 1986.

- Seki, H., Ogawa, M., Ooyama, S., Akamatsu, K., Ohshima, M. and Yang, W., Industrial application of a nonlinear model predictive control to polymerization reactors. Control Engineering Practice. 9(2002): 819-828.
- Soeterboek, R., Predictive Control. A Unifed Approach. Cambridge, UK: Prentice Hall Intemational, 1992.
- Thomas, Y. A., Linear quadratic optimal estimation and control with receding horizon. Electron. Lett. 11(1975): 19-21.
- Wei, J., Xu, Y. and Zhang, J., Neural networks based Model predictive control of an industrial polypropylene process, Proceedings of the 2002 IEEE International Conference on Control Applications, Glasgow, Scotland, U.K.(September 2002): 397-402.
- Widrow, B., 30 years of adaptive neural networks: perceptron, madaline and backpropagation. proc. IEEE.(August 1990): 27.
- Widrow, B., Rumelhart, D. E. and Lehr, M. A., Neural Networks: Applications in Industry, Business and Science. Communications of the ACM. 37, 3(1994): 93-105.
- Ydstie, B. E., Forecasting and Control using Adaptive Connectionistic Networks. Computers Chem. Engng. 4/5(1990): 583-599.
- Yu, D. L. and Gomm, J. B., Implementation of neural network predictive control to a multivariable chemical reactor. Control Engineering Practice. 11(2002): 1315-1323.
- Zamarreno, J. M. and Vega, P., Identification and predictive control of a melter unit used in the sugar industries. Artificial Intelligence in Engineering II.(1997): 365-373.



APPENDICES

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

APPENDIX A

MATLAB R2006a – Neural Network Modeling

This MATLAB source code will create a feedforward neural network with 4 hidden nodes in hidden layer and one output node in output layer. The number of inputs to the network is defined from the size of the input matrix where if input is a matrix of 5 x 2000, the number of input to the network is 5. There are 2000 pairs of training data used in the training of this network.

The transfer functions used in the hidden and output layers are sigmoidal and linear respectively. The network is trained with the Lavenberg-Marquardt training algorithm.

```
%=====
function Modeling
clear
load TrainData.mat
load TestData.mat

%=====
% scale down data
%=====
xmaxin=[2;2;4;4;3];
xminin=[0;0;0;0;0];
dmax=0.95;
dmin=0.05;
xmaxout=[3];
xminout=[0];

TrainInDown1 = ScDown(InTrain1,xmaxin,xminin,dmax,dmin);
TrainOutDown1= ScDown(OutTrain1,xmaxout,xminout,dmax,dmin);
TrainInDown2 = ScDown(InTrain2,xmaxin,xminin,dmax,dmin);
TrainOutDown2= ScDown(OutTrain2,xmaxout,xminout,dmax,dmin);
TestInDown  = ScDown(InTest,xmaxin,xminin,dmax,dmin);
TestOutDown  = ScDown(OutTest,xmaxout,xminout,dmax,dmin);

%=====
%network initialization
%=====
S1=4;
S2=1;
```



```

NetB1=newff(minmax(TrainIn1),[S1 S2],{'tansig' 'purelin'},'trainlm');

NetB1.trainParam.goal=0;
NetB1.trainParam.epochs=1000;
NetB1.trainParam.min_grad=1e-10;
NetB1.trainparam.show=100;

dmse=1e-4;
round=1;
msenn=1;
ssenn=1;
mseR(:,1)=msenn;
sseR(:,1)=ssenn;

%=====
% Network Training
%=====
while msenn>=dmse
    set=(round/2-floor(round/2));
    switch set
    case 0
        input=TrainInDown1;
        target=TrainOutDown1;

    case 0.5
        input=TrainInDown2;
        target=TrainOutDown2;

    end

    [NetB1,tr,Y,E,Pf,Af]=train(NetB1,input,target);

    NNOutputtest=sim(NetB1,TestInDown);
    error=NNOutputtest-TestOutDown;
    msenn=mse(error);
    ssenn=sse(error);

    mseR(:,round)=msenn;
    sseR(:,round)=ssenn;
    [NNTestOut] = ScUp(NNOutputtest,xmaxout,xminout,dmax,dmin);

    if msenn>=3
        NetB1=init(NetB1);
    end

    if round>=15
        msenn=0;
        ssenn=0;
    end
end

```

```
time=0:length(NNTestOut(1,:))-1;  
round=round+1;  
end
```

```
save 4ModelB1.mat NetB1 S1 S2 xmaxin xminin dmax dmin xmaxout xminout
```

```
figure  
plot(time,OutTest,'r',time,NNTestOut,'b:')  
title('Validation of Neural Network Model')  
legend('C1 Target','C1 Output')  
ylabel('Concentration (mol/l)')  
xlabel('Pattern')
```



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

APPENDIX B

MATLAB R2006a – The Control of NNMPC

This MATLAB source code simulates a model predictive control based on neural network (NNMPC) to control a steel pickling process. The developed neural network models are used in the MPC algorithm for prediction of the future process response.

```
%=====
function closeloop
clear
global i p c1sp c2sp c3sp H1 H2 H3 c20 C1 C2 C3 C11 C22 C33 dt PP F2k F3k F5k
i=1;
p=4;
PP=8;

TTime =45;
dt = 0.1;
Numdata = TTime/dt;
samptime = 0.5;
j=0;

% Constant value
A1=0.0729; A2=0.0729; A3=0.0729;
A4=0.0729; A5=0.0729; A6=0.0729;
k1i=0.003267; k2i=0.003267; k3i=0.003267;
c20i=6.034;
cw=1*10^-7;
c1sp=1.29;
c2sp=2.77;
c3sp=4.3;
c1spi(1)=1.29;
c2spi(1)=2.77;
c3spi(1)=4.3;
pH4sp=3;
pH5sp=5;
pH6sp=7;

% Drag in-out
drag=(0.0005/1000);
q=drag;
qest(1)=drag;
```

% Initial values of manipulated variables

F2(1)=0.0465/1000;

F3(1)=0.0916/1000;

F5(1)=0.1235/1000;

F6(1)=0.001/1000 ;

F7(1)=0.001/1000 ;

F8(1)=0.001/1000;

% Initial values of disturbance

F4(1)=0/1000 ;

% Initial values of other variables

F1(1)=F2(1)-q ;

F9(1)=F6(1)-F4(1);

F10(1)=F5(1)+F4(1)-F3(1);

F11(1)=F3(1)-F2(1);

h1(1)=0.205; c1(1)=1.2;

h2(1)=0.205; c2(1)=2.67;

h3(1)=0.205; c3(1)=4.2;

h4(1)=0.205; c4(1)=1*10⁻⁷;

h5(1)=0.205; c5(1)=1*10⁻⁷;

h6(1)=0.205; c6(1)=1*10⁻⁷;

rate1(1)=0; rate2(1)=0; rate3(1)=0;

sumF5=F5(1)*dt; sumF1=F1(1)*dt;

Hwait=waitbar(0,'Simulation in Progress ...');

for k=1:1:Numdata
waitbar(k/Numdata);

if k>=150

c1sp=1.35;

c2sp=2.8;

c3sp=4.35;

end

if k>=300

c1sp=1.4;

c2sp=2.87;

c3sp=4.408;

end

c20=c20

k1=k1i;

k2=k2i;

k3=k3i;

```

% Process
h1(k+1)=h1(k)+(dt/A1)*(F2(k)-F1(k)-q);
h2(k+1)=h2(k)+(dt/A2)*(F3(k)-F2(k)-F11(k));
h3(k+1)=h3(k)+(dt/A3)*(F4(k)+F5(k)-F3(k)-F10(k));

V1=A1*h1(k);   V2=A2*h2(k);   V3=A3*h3(k);

if k>1
    rate1(k)=k1*c1(k); rate2(k)=k2*c2(k); rate3(k)=k3*c3(k);
end

c1(k+1)=c1(k)+(dt/V1)*((F2(k)*c2(k))-((F1(k)+q)*c1(k)))-(dt)*rate1(k);
if c1(k+1)<0 c1(k+1)=1*10^-7; end;
c2(k+1)=c2(k)+(dt/V2)*((q*c1(k))+F3(k)*c3(k))-((F2(k)+q+F11(k))*c2(k))-
    (dt)*rate2(k);
if c2(k+1)<0 c2(k+1)=1*10^-7; end;
c3(k+1)=c3(k)+(dt/V3)*((q*c2(k))+F4(k)*c4(k))-((F3(k)+q+F10(k))*c3(k))-
    (dt)*rate3(k)+(dt/V3)*(F5(k)*c20);
if c3(k+1)<0 c3(k+1)=1*10^-7; end;

H1=h1(k+1);
H2=h2(k+1);
H3=h3(k+1);

C1=c1(k+1);
C2=c2(k+1);
C3=c3(k+1);

C11=c1(k);
C22=c2(k);
C33=c3(k);

F2k=F2(k);
F3k=F3(k);
F5k=F5(k);

if k==j|k==1
    Fopt=MPCNN(p,F2(k),F3(k),F5(k));
    F2(k+1)=Fopt(1);
    F3(k+1)=Fopt(p1+1);
    F5(k+1)=Fopt(p1+p2+1);
    j=j+5;
else
    F2(k+1)=F2(k);
    F3(k+1)=F3(k);
    F5(k+1)=F5(k);
end
end

```

```

% Disturbance
F1(k+1)=F2(k+1)-q;
F4(k+1)=F4(k);
F10(k+1)=F5(k+1)+F4(k+1)-F3(k+1);
F11(k+1)=F3(k+1)-F2(k+1);
c4(k+1)=c4(k);

c1spi(k+1) = c1sp;
c2spi(k+1) = c2sp;
c3spi(k+1) = c3sp;

i=1;
end
close(Hwait)

for k = 1:1:Numdata+1
    ti(k) = (k-1)*dt;
end;

IAE1 = 0;
err1 = c1spi - c1;
err21 = abs(err1)*dt;
IAEc1 = sum(err21);

IAE2= 0;
err2 = c2spi - c2;
err22 = abs(err2)*dt;
IAEc2 = sum(err22);

IAE3 = 0;
err3 = c3spi - c3;
err23 = abs(err3)*dt;
IAEc3 = sum(err23);

figure
subplot(211),plot(ti,c1,'r',ti,c1spi,'-r'),...
ylabel('Concentration (mol/l)'),
legend('C1','set point')
Title('5% HCl Bath')
subplot(212),plot(ti,F2*1000,'-r'),...
ylabel('F2 flow rate (l/min)'),
legend('F2')
Xlabel('Time (min)'),

figure
subplot(211),plot(ti,c2,'r',ti,c2spi,'-r'),...
ylabel('Concentration (mol/l)'),
legend('C2','set point')
Title('10% HCl Bath')

```

```
subplot(212),plot(ti,F3*1000,'-r'),...
ylabel('F3 flow rate (l/min)'),
legend('F3')
xlabel('Time (min)'),
```

```
figure
subplot(211),plot(ti,c3,':r',ti,c3spi,'-r'),...
ylabel('Concentration (mol/l)'),
legend('C3','set point')
Title('15% HCl Bath')
subplot(212),plot(ti,F5*1000,'-r'),...
ylabel('F5 flow rate (l/min)'),
legend('F5')
xlabel('Time (min)'),
```

```
figure
subplot(311),plot(ti,h1,':r'),...
ylabel('Hight (m)'),
Title('Hight')
subplot(312),plot(ti,h2,'-r'),...
ylabel('Hight (m)'),
subplot(313),plot(ti,h3,'-r'),...
ylabel('Hight (m)'),
xlabel('Time (min)'),
```

```
%=====
```

```
function xopt=MPCNN(p,F2mv,F3mv,F5mv)
```

```
int1=ones(p,1)*(0.1/1000);
int2=ones(p,1)*(0.1/1000);
int3=ones(p,1)*(0.1/1000);
lb1=ones(1,p)*0;
ub1=ones(1,p)*2/1000;
lb2=ones(1,p)*0;
ub2=ones(1,p)*10/1000;
lb3=ones(1,p)*0;
ub3=ones(1,p)*2/1000;
```

```
OPTIONS=OPTIMSET('TolFun',1e-6,'TolX',1e-
6,'MaxFunEvals',5000,'MaxIter',5000,);
[xopt,Fval,conv]=fmincon(@obj,[int1;int2;int3],[[],[],[]],[lb1 lb2 lb3],[ub1 ub2
ub3],[[],OPTIONS];
```

```
%=====
```

```
function f = obj(Fin)
load 4ModelB1.mat
load 4ModelB2.mat
load 4ModelB3.mat
```



```
global i p c1sp c2sp c3sp H1 H2 H3 c20 C1 C2 C3 C11 C22 C33 dt PP F2k F3k F5k
clear SimCon
```

```
cc1=C1;
cc2=C2;
cc3=C3;
cc11=C11;
cc22=C22;
cc33=C33;
F22k=F2k;
F33k=F3k;
F55k=F5k;
cc11down=ScDown(cc11,Pi(12,2),Pi(12,1),0.95,0.05);
cc22down=ScDown(cc22,Pi(13,2),Pi(13,1),0.95,0.05);
cc33down=ScDown(cc33,Pi(14,2),Pi(14,1),0.95,0.05);
F22kdown=ScDown(F22k,Pi(2,2),Pi(2,1),0.95,0.05);
F33kdown=ScDown(F33k,Pi(3,2),Pi(3,1),0.95,0.05);
F55kdown=ScDown(F55k,Pi(5,2),Pi(5,1),0.95,0.05);
```

```
for m=1:1:p
    f2(m)=Fin(m);
end
```

```
for m=1:1:p
    f3(m)=Fin(p+m);
end
```

```
for m=1:1:p
    f5(m)=Fin(p+p+m);
end
```

```
for n=1:1:PP
```

```
    if n>p
        f2(n)=f2(n-1);
    end
```

```
    if n>p
        f3(n)=f3(n-1);
    end
```

```
    if n>p
        f5(n)=f5(n-1);
    end
```

```
end
```

```
f=0;
```

```
fin1sum=0;
```

```
fin2sum=0;
```

```
fin3sum=0;
```

```
c1spdown=ScDown(c1sp,Pi(12,2),Pi(12,1),0.95,0.05);
```

```
c2spdown=ScDown(c2sp,Pi(13,2),Pi(13,1),0.95,0.05);
c3spdown=ScDown(c3sp,Pi(14,2),Pi(14,1),0.95,0.05);
```

```
for j=1:1:PP
```

```
SimCon(1,1)=ScDown(f2(1),Pi(2,2),Pi(2,1),0.95,0.05);
SimCon(2,1)=ScDown(f3(1),Pi(3,2),Pi(3,1),0.95,0.05);
SimCon(3,1)=ScDown(f5(1),Pi(5,2),Pi(5,1),0.95,0.05);
SimCon(4,1)=ScDown(cc1,Pi(12,2),Pi(12,1),0.95,0.05);
SimCon(5,1)=ScDown(cc2,Pi(13,2),Pi(13,1),0.95,0.05);
SimCon(6,1)=ScDown(cc3,Pi(14,2),Pi(14,1),0.95,0.05);
SimCon(7,1)=ScDown(10^-7,Ri(15,2),Ri(15,1),0.95,0.05);
```

```
%B1
```

```
SimConB1(1,j)=SimCon(1,j);
if j<=1
SimConB1(2,j)=F22kdown;
else
SimConB1(2,j)=SimCon(1,j-1);
end
SimConB1(3,j)=SimCon(5,j);
if j<=1
SimConB1(4,j)=cc22down;
else
SimConB1(4,j)=SimCon(5,j-1);
end
SimConB1(5,j)=SimCon(4,j);
```

```
%B2
```

```
SimConB2(1,j)=SimCon(1,j);
if j<=1
SimConB2(2,j)=F22kdown;
else
SimConB2(2,j)=SimCon(1,j-1);
end
SimConB2(3,j)=SimCon(2,j);
if j<=1
SimConB2(4,j)=F33kdown;
else
SimConB2(4,j)=SimCon(2,j-1);
end
SimConB2(5,j)=SimCon(4,j);
if j<=1
SimConB2(6,j)=cc11down;
else
SimConB2(6,j)=SimCon(4,j-1);
end
SimConB2(7,j)=SimCon(6,j);
if j<=1
SimConB2(8,j)=cc33down;
```

```

else
    SimConB2(8,j)=SimCon(6,j-1);
end
SimConB2(9,j)=SimCon(5,j);

%B3
SimConB3(1,j)=SimCon(2,j);
    if j<=1
        SimConB3(2,j)=F33kdown;
    else
        SimConB3(2,j)=SimCon(2,j-1);
    end
SimConB3(3,j)=SimCon(3,j);
    if j<=1
        SimConB3(4,j)=F55kdown;
    else
        SimConB3(4,j)=SimCon(3,j-1);
    end
SimConB3(5,j)=SimCon(5,j);
    if j<=1
        SimConB3(6,j)=cc22down;
    else
        SimConB3(6,j)=SimCon(5,j-1);
    end
SimConB3(7,j)=SimCon(7,j);
    if j<=1
        SimConB3(8,j)=SimCon(7,1);
    else
        SimConB3(8,j)=SimCon(7,j-1);
    end
SimConB3(9,j)=SimCon(6,j);

if j+1>p
    SimCon(1,j+1)=ScDown(f2(p),Pi(2,2),Pi(2,1),0.95,0.05);
else
    SimCon(1,j+1)=ScDown(f2(j+1),Pi(2,2),Pi(2,1),0.95,0.05);
end

if j+1>p
    SimCon(2,j+1)=ScDown(f3(p),Pi(3,2),Pi(3,1),0.95,0.05);
else
    SimCon(2,j+1)=ScDown(f3(j+1),Pi(3,2),Pi(3,1),0.95,0.05);
end

if j+1>p
    SimCon(3,j+1)=ScDown(f5(p),Pi(5,2),Pi(5,1),0.95,0.05);
else
    SimCon(3,j+1)=ScDown(f5(j+1),Pi(5,2),Pi(5,1),0.95,0.05);
end

```

```

SimCon(4,j+1)=sim(netB1,SimConB1(:,j));
SimCon(5,j+1)=sim(netB2,SimConB2(:,j));
SimCon(6,j+1)=sim(netB3,SimConB3(:,j));
SimCon(7,j+1)=SimCon(7,j);

if j==1
    Delf2=SimCon(1,j)-SimCon(1,j);
else
    Delf2=SimCon(1,j)-SimCon(1,j-1);
end

if j==1
    Delf3=SimCon(2,j)-SimCon(2,j);
else
    Delf3=SimCon(2,j)-SimCon(2,j-1);
end

if j==1
    Delf5=SimCon(3,j)-SimCon(3,j);
else
    Delf5=SimCon(3,j)-SimCon(3,j-1);
end

fin1(j)=(c1spdown-SimCon(4,j))^2+Delf2^2;
fin2(j)=(c2spdown-SimCon(5,j))^2+Delf3^2;
fin3(j)=(c3spdown-SimCon(6,j))^2+Delf5^2;

end

for j=1:1:PP
    fin1sum=fin1sum+fin1(j);
end

for j=1:1:PP
    fin2sum=fin2sum+fin2(j);
end

for j=1:1:PP
    fin3sum=fin3sum+fin3(j);
end

f=fin1sum+fin2sum+fin3sum;

%=====

```

VITA

Miss Wachira Daosud was born in Prachinburi, on December 21, 1976. After completing high school from Assumption College Sriracha in 1994, she graduated her Bachelor Degree of Engineering in Chemical Engineering from Burapha University in 1998. She began her graduate studies when she entered the Graduate School of Chulalongkorn University and completed her Doctoral Degree of Engineering in Chemical Engineering in 2006.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย