

MULTIPLE FAULTS DETECTION USING ARTIFICIAL NEURAL NETWORK

MOHD. KAMARUDDIN BIN ABD. HAMID

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Master of Engineering (Chemical)

Faculty of Chemical and Natural Resources Engineering
Universiti Teknologi Malaysia

JULY 2004

UNIVERSITI TEKNOLOGI MALAYSIA

BORANG PENGESAHAN STATUS TESIS ♦

JUDUL: MULTIPLE FAULT DETECTION USING ARTIFICIAL NEURAL NETWORKS

SESI PENGAJIAN: 2003/2004 II

Saya MOHD. KAMARUDDIN BIN ABD. HAMID
(HURUF BESAR)

mengaku membenarkan tesis (~~PSM/Sarjana/Doktor-Falsafah~~)* ini disimpan di Perpustakaan Universiti Teknologi Malaysia dengan syarat-syarat kegunaan seperti berikut :

1. Tesis adalah hakmilik Universiti Teknologi Malaysia.
2. Perpustakaan Universiti Teknologi Malaysia dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (✓)

SULIT

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

TERHAD

(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

TIDAK TERHAD

Disahkan oleh

(TANDATANGAN PENULIS)

(TANDATANGAN PENYELIA)

Alamat Tetap:

Peti Surat 65,
Kampung Pimping,
89728, Membakut Sabah.

PROF. DR. ARSHAD AHMAD

Nama Penyelia

Tarikh: _____

Tarikh: _____

- CATATAN: * Potong yang tidak berkenaan.
 ** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali sebab dan tempoh tesis ini perlu dikelaskan sebagai SULIT atau TERHAD.
 ♦ Tesis dimaksudkan sebagai tesis bagi Ijazah Doktor Falsafah dan Sarjana secara penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

“I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of the degree of Master of Engineering (Chemical).”

Signature :

Supervisor : PROF. DR. ARSHAD AHMAD

Date :

BAHAGIAN A – Pengesahan Kerjasama*

Adalah disahkan bahawa projek penyelidikan tesis ini telah dilaksanakan melalui kerjasama antara _____ dengan _____

Disahkan oleh:

Tandatangan : Tarikh:
Nama :
Jawatan :
(Cop rasmi)

** Jika penyelidikan tesis/projek melibatkan kerjasama.*

BAHAGIAN B – Untuk Kegunaan Pejabat Sekolah Pengajian Siswazah

Tesis ini telah diperiksa dan diakui oleh:

Nama dan Alamat : **Prof. Dr. G. P. Rangiah**
Pemeriksa Luar : **Dept. of Chemical and Biomolecular Engineering**
National University of Singapore

Nama dan Alamat : **Prof. Madya Dr. Khairiyah Mohd. Yusof**
Pemeriksa Dalam I : **Department of Chemical Engineering**
Universiti Teknologi Malaysia
81310 UTM Skudai, Johor

Pemeriksa Dalam II : **Prof. Madya Dr. M. Wijayanuddin Ali**
Department of Chemical Engineering
Universiti Teknologi Malaysia
81310 UTM Skudai, Johor

Nama Penyelia Lain :
(jika ada)

Disahkan oleh Timbalan Pendaftar di SPS:

Tandatangan : Tarikh:
Nama :

I declare that this thesis entitled “MULTIPLE FAULTS DETECTION USING ARTIFICIAL NEURAL NETWORK” is the result of my own research except as cited in references. The thesis has not been accepted for any degree and is not concurrently submitted in candidate of any other degree.

Signature :
Name : MOHD. KAMARUDDIN BIN ABD. HAMID
Date :

In the Name of Allah, Most Gracious, Most Merciful.

All praise and thanks are due to Allah Almighty and peace and
blessings be upon His Messenger.

ACKNOWLEDGEMENT

First and foremost, I thank God Almighty for giving me the strength to complete my research. I would also like to thank the following people who assisted me in my research:

- Prof. Dr. Arshad Ahmad, my supervisor for his technical guidance support. His encouragement has, without doubt, helped me overcome challenges and meet my goal.
- Assoc. Prof. Dr. Kamarul Asri Ibrahim, Assoc. Prof. Dr. Mohd. Wyanuddin Ali and Assoc. Prof. Dr. Kairiyah Mohd. Yusoff for their advice and encouragement.
- Prof. Dr. CP. Rangiah from National University of Singapore, for his opinions and comments.
- Engineers and plant personnel at Pan Century Biochemicals Sdn. Bhd. Pasir Gudang for their help in providing all the data for the process.

I had received numerous helps and support from my fellow coursemates and friends, I would like to thank them. Finally, I would like to thank my parents for their love, prayers, sacrifices and supports. Thanks to my wife, Norazana Ibrahim for her help, support and motivation in completing my thesis. May God protect and guide all of you.

ABSTRACT

This thesis investigated issues on the development of efficient fault detection scheme for detection of single and multiple faults due to sensor failure and leakage in the process stream. The proposed scheme consisted of two stage mechanism constructed using artificial neural network (ANN). The first stage was a process estimator that was designed to estimate the normal and unfaulty behaviour of the plant. In order to produce reasonably accurate estimation without including the history data of the output, two types of model have been studied. A group of multi input single output (MISO) Elman network and a multi input multi output (MIMO) Feedforward network have been used, and results revealed that MISO model had better generalisation ability compared to MIMO model. The difference between the actual plant signal and this estimated normal plant behaviour, termed as residual was fed to the second stage for fault classification. In the development of fault classifiers, the MISO models had been proven to be better than MIMO model. The effect of adding input with time delayed signals to the network had also been studied. In both cases, successful implementations were obtained. Finally, the proposed fault detection scheme was applied for detection of sensor faults and stream leakage in the Precut column of a fatty acid fractionation plant. The proposed scheme was successful in detecting both single and multiple faults cases imposed to the process. The strategy was also successful in detecting leakage in the process stream even when the percentage of the leakage was as little as 0.1%. The results obtained in this work proved the potential of neural network in detecting multiple faults and leakage in chemical process plant.

ABSTRAK

Tesis ini telah mengkaji isu-isu berkaitan dengan pembangunan skema pengesanan kesilapan yang cekap untuk mengesan kesilapan tunggal dan berbilang ekoran dari kegagalan penerima and kebocoran dalam aliran proses. Skema yang dicadangkan ini terdiri daripada dua peringkat mekanisme yang dibina menggunakan rangkaian saraf buatan (ANN). Peringkat pertama ialah peramal proses yang direkabentuk untuk meramal keadaan loji yang normal dan tidak mempunyai kesilapan. Untuk menghasilkan ramalan yang betul-betul tepat tanpa menggunakan data keluaran masa lalu, dua jenis model telah dikaji. Rangkaian berbagai masukan keluaran tunggal (MISO) Elman dan rangkaian berbagai masukan berbagai keluaran (MIMO) telah digunakan, dan keputusan-keputusan yang diperolehi menunjukkan yang model MISO mempunyai keupayaan meramal yang baik berbanding dengan model MIMO. Perbezaan di antara isyarat loji yang sebenar dengan ramalan keadaan normal ini, disebut baki dimasukkan ke dalam peringkat yang kedua untuk pengkelasan kesilapan. Dalam pembangunan pengesanan kesilapan, model-model MISO telah terbukti lebih baik keupayaannya berbanding model MIMO. Kesan penambahan masukan yang terdiri dari isyarat masa lampau terhadap kebolehan rangkaian juga telah diselidik. Dalam kedua-dua kes ini, pelaksanaannya telah mencapai kejayaan. Akhir sekali, skema cadangan pengesanan kesilapan ini telah dilaksanakan dalam mengesan kesilapan sensor dan kebocoran dalam loji penyulingan minyak asid lemak. Untuk mengesan kesilapan penerima, skema yang dicadangkan ini telah berjaya mengesan kesilapan tunggal dan berganda yang berlaku dalam proses. Strategi yang dicadangkan ini juga telah berjaya mengesan kebocoran di dalam saluran proses walaupun peratusan kebocoran itu terlalu kecil sehingga 0.1%. Keputusan-keputusan yang didapati dari kerja selidik ini telah membuktikan keupayaan rangkaian saraf buatan dalam mengesan kesilapan berganda dan kebocoran yang berlaku dalam loji proses kimia.

TABLE OF CONTENT

Declaration	ii
Dedication	iii
Acknowledgement	iv
Abstract	v
Abstrak	vi
Table of Content	vii
List of Tables	xii
List of Figures	xiii
Nomenclature	xvii
List of Appendices	xx

CHAPTER	TITLE	PAGE
I	INTRODUCTION	
	1.1 Motivation	1
	1.2 Problem Statement and Importance of Study	4
	1.3 Objective and Scopes of Work	5
	1.4 Thesis Organisation	6

II LITERATURE REVIEW

2.1	Overview of Fault Detection	8
2.1.1	State Estimation Approaches	13
2.1.2	Statistical Process Control Approach	14
2.1.3	Knowledge-Based Approaches	15
2.2	Artificial Neural Networks (ANNs)	16
2.2.1	Neuron (<i>Node</i>) and Neural Networks	18
2.2.2	Feedforward Neural Networks	21
2.2.3	Recurrent Neural Networks	25
2.2.4	Elman Networks	26
2.3	Issues in Neural Network Applications to Fault Detection	28
2.3.1	Inputs Patterns	29
2.3.2	Generalisation and Improving Generalisation Ability of Neural Networks	31
2.3.3	Process Fault Detection and Diagnosis Using Neural Network	33
2.4	Summary	36

III PLANT SIMULATION

3.1	Process Description	38
3.1.1	Precut Column	39
3.1.2	Lights Cut Column	40
3.1.3	Middle Cut Column	40
3.1.4	Still and Residue Still Column	41
3.1.5	A Typical Packed Column	42
3.2	Simulation Methodology	44
3.2.1	Physical Properties of the Pure Component	45

3.2.2	Thermodynamic Property	47
3.2.3	Integration Algorithm	52
3.2.4	Mathematical Modelling of the Distillation Operation	53
3.2.5	Modelling and Simulation Assumption	56
3.2.6	Degree of Freedom Analysis	57
3.3	Comparison of Simulation Results	58
3.4	Summary	60

IV PROCESS ESTIMATION FOR FAULT DETECTION PURPOSES

4.1	Introduction	61
4.2	Design of Process Estimator	61
4.3	Selection of Process Estimator's Output and Input Variables	62
4.3.1	Output Variables	63
4.3.2	Input Variables	63
4.4	Design of Excitation Signal	64
4.4.1	Excitation Signal for Training	65
4.4.2	Excitation Signal for Validation	67
4.4.3	Excitation Signal for Testing	68
4.5	Selection of Neural Network Structure, Training and Cross-Validation	69
4.5.1	Multi-Input and Single-Output (MISO) Model	69
4.5.2	Multi-Input and Multi-Output (MIMO) Model	77
4.6	Summary	80

V NEURAL NETWORK FAULT CLASSIFIER

5.1	Introduction	81
5.2	Sensitivity Analysis	83
5.3	Preparation of Training Data	85
5.3.1	Top Column Pressure Sensor Positive Biases (F1)	87
5.3.2	Top Column Pressure Sensor Negative Biases (F2)	88
5.3.3	Bottom Column Temperature Sensor Positive Bias (F3)	89
5.3.4	Bottom Column Temperature Sensor Negative Bias (F4)	90
5.3.5	Leakage to Condenser Stream Fault (F5)	91
5.4	Preparation of Validation Data	92
5.5	Structure Selection, Training and Cross-Validation	96
5.5.1	Multi-Input Single Output (MISO) Networks	96
5.5.2	Multi-Input Multi-Output (MIMO) Network	101
5.5.3	Structure Selection	102
5.6	Performance of Fault Classifier With Non Noise-Corrupted Measurements	103
5.6.1	Single Fault Detection	103
5.6.2	Multiple Faults Detection	107
5.6.3	Leakage Fault Detection	111
5.6.1	Conclusion	113
5.7	Performance of Fault Classifier With Noise-Corrupted Measurements	113
5.7.1	Single Fault Detection	114
5.7.2	Multiple Faults Detection	117
5.7.3	Leakage Fault Detection	120
5.7.4	Conclusion	122
5.8	Summary	123

VI	CONCLUDING REMARKS AND FUTURE WORK	
6.1	Introduction	124
6.2	Summary	125
6.3	Concluding Remarks	126
6.4	Recommendations for Future Work	128
VII	REFERENCES	129
VIII	APPENDICES	139

LIST OF TABLES

TABLE	TITLE	PAGE
3.1	Controlled and manipulated variables of the control loops	43
3.2	Physical property of the fatty acid component	47
3.3	Thermodynamic models for the Precut column	52
3.4	Simulation and plant data comparison for Precut column	59
4.1	Neural network input	64
5.1	List of sensor faults	82
5.2	Precut column operating constraints	83
5.3	Sensor faults simulated for network training	87
5.4	Sensor faults simulated for network training	93
5.5	Comparison of MISO networks and MIMO network performance	102
5.6	Faults simulated for fault classifier performance testing	106
5.7	Simulation of faults for multiple faults classifiers performance testing	108
5.8	Simulation of leakage for leak classifier performance testing	111

LIST OF FIGURES

FIGURE	TITLE	PAGE
1.1	The Model-Based Fault Detection Scheme	3
2.1	A general diagnostic framework	11
2.2	Structure of a single processing node	19
2.3	Structure of a layered neural network	19
2.4	Graph of the information flow in a feedforward neural network	22
2.5	Representation of internally/externally recurrent neural networks	25
2.6	Block diagram of Elman network	26
3.1	Schematic diagram of a fractionation process	39
3.2	A typical packed column	42
3.3	A Precut column with its control system	43
3.4	The main flowsheet in HYSYS	45
3.5	The sub-flowsheet of a Packed column	46
3.6	Distillation tray model	54
4.1	The neural network-based fault detection scheme	62
4.2	Training data set	66
4.3	Cross-validation data set	67
4.4	Testing data set	68
4.5	Multi-Input Single-Output Elman network	70
4.6	Training result of MISO network for Top Stage Pressure (without delay) for different training algorithm	71

4.7	Performance of MISO network for Top Stage Pressure with (a) <i>traingdm</i> (topology 7/24/1 nodes), (b) <i>traingdx</i> (topology 7/6/1 nodes) and (c) <i>trainlm</i> (topology 7/12/1 nodes)	72
4.8	Training result of MISO network for Top Stage Pressure with <i>trainlm</i>	73
4.9	Performance of MISO network for Top Stage Pressure with <i>trainlm</i> , (a) without delayed term (topology 7/12/1 nodes), (b) with 1 delayed term (topology 14/6/1 nodes) and (c) with 2 delayed terms (topology 21/18/1 nodes)	74
4.10	Training result of MISO networks with 2 delayed terms and <i>trainlm</i>	75
4.11	Performance of MISO network with <i>trainlm</i> for (a) Top Stage Pressure (topology 21/18/1 nodes), (b) Bottom Stage Temperature (topology 21/7/1 nodes) and (c) C8 Flowrate (topology 21/11/1 nodes)	76
4.12	Network structure for MIMO model	77
4.13	Training result of MIMO networks with <i>trainlm</i>	78
4.14	Performance of MIMO network with <i>trainlm</i> , without time delay (topology 7/8/1 nodes)	79
5.1	Leakage simulation in HYSYS simulator	82
5.2	Sensitivity Analysis for Precut column	84
5.3	Residual patterns for Precut column	86
5.4	Training data for top column pressure sensor positive bias (F1)	88
5.5	Training data for top column pressure sensor negative bias (F2)	89
5.6	Training data for bottom column temperature sensor positive bias (F3)	90
5.7	Training data for bottom column temperature sensor negative bias (F4)	91
5.8	Training data for leakage <i>To Condenser</i> stream (F5)	92
5.9	Validation data for top column pressure sensor positive bias (F1)	93

5.10	Validation data for top column pressure sensor negative bias (F2)	94
5.11	Validation data for bottom column temperature sensor positive bias (F3)	94
5.12	Validation data for bottom column temperature sensor negative bias (F4)	95
5.13	Validation data for leakage <i>To Condenser</i> stream (F5)	95
5.14	Artificial Neural Network MISO Fault Classifier	96
5.15	Training results of MISO networks fault classifier	97
5.16	Training and validation results for sensor fault F1	98
5.17	Training and validation results for sensor fault F2	98
5.18	Training and validation results for sensor fault F3	99
5.19	Training and validation results for sensor fault F4	100
5.20	Training and validation results for leakage fault F5	100
5.21	Artificial neural network MIMO fault classifier	101
5.22	Cross-validation error of MIMO network	102
5.23	F1 (6%) happened after step change 12% of feed	104
5.24	F2 (5%) happened after step change -10% of feed	105
5.25	F3 (6%) happened after step change 4% of reboiler	106
5.26	F4 (5%) happened after step change -5% of reboiler	106
5.27	F1 (5%) and F3 (6%) happened after step change 4% of reboiler	108
5.28	F1 (5.5%) and F3 (6%) happened after step change 5.5% of reboiler	109
5.29	F2 (4.5%) and F4 (5%) happened after step change -2.5% of reboiler	110
5.30	F2 (5%) and F4 (5.5%) happened after step change -4% of reboiler	110
5.31	F5 (0.1%) happened after 0.1% valve opening	112
5.32	F5 (0.03%) happened after 0.03% valve opening	112
5.33	(6%) happened after step change 12% of feed with 1% noise	114
5.34	F1 (6%) happened after step change 12% of feed with 10% noise	115

5.35	F3 (6%) happened after step change 10% of feed with 1% noise	116
5.36	F3 (6%) happened after step change 10% of feed with 10% noise	116
5.37	F1 (5%) and F3 (6%) happened after step change 5% of reboiler with 1% noise	118
5.38	F1 (5%) and F3 (6%) happened after step change 5% of reboiler with 10% noise	119
5.39	F2 (4.5%) and F4 (5%) happened after step change -2.5% of reboiler with 1% noise	119
5.40	F2 (4.5%) and F4 (5%) happened after step change -2.5% of reboiler with 10% noise	120
5.41	F5 (0.3%) happened after 0.3% valve opening with 1% noise	121
5.42	F5 (0.3%) happened after 0.3% valve opening with 10% noise	122

NOMENCLATURE

A_w	- van der Waals area
a_{ij}	- non-temperature dependent energy parameter between components i and j (cal/gmol)
B	- connection matrix from the input layer to the hidden layer
b_{ij}	- temperature dependent energy parameter between components i and j (cal/gmol-K)
b_h	- bias vector for the hidden layer
b_o	- bias vector for the output layer
C	- the unit conversion constant
C	- <i>connection matrix</i> (matrix of weights)) from the hidden layer to the output layer
F	- tray feed
$f(\cdot)$	- a nonlinear mapping
$g(\cdot)$	- a nonlinear mapping
h	- the step size
h	- the specific enthalpy (J/mol)
h	- the height of liquid above the weir
h_n^L	- the liquid enthalpy for each tray
h_n^V	- the vapour enthalpy for each tray
k	- the conductance
L_n	- the liquid flowrate leaving tray n
L_{n+1}	- liquid enters through the downcomer of the tray from the above
M_n	- the material on the n th tray
l_w	- the weir length
MW	- the molecular weight
N_c	- the number of components in the system
N_{c-1}	- differential material balances
N_{om}	- number of manipulated (input) variables with no steady-state effect
N_{oy}	- number of output variables that need to be controlled

N_m	- dynamic degrees of freedom
N_{ss}	- steady-state degrees of freedom
n	- total number of components
q_i	- van der Waals area parameter
r_i	- van der Waals volume parameter
$s(k)$	- an intermediate variable at a discrete time k
T	- temperature (K)
T_b	- the boiling point temperature
T_n	- the temperature for each tray
u_i	- input vector at a discrete time i
$u(k)$	- input vector at a discrete time k
V_n	- the vapour flowrate leaving tray n
V_{n-1}	- vapour enters the tray from the tray below
V_w	- van der Waals volume
W^u	- the interconnection matrices for the input-hidden layer
W^y	- the interconnection matrices hidden-output layer
X	- the actual input before scaling
X_{\max}	- the maximum value of the inputs
X_{\min}	- the minimum value of the inputs
X_s	- scaled input
$x(k)$	- state vector at a discrete time k
x_i	- mole fraction of component i
$x_{i,n}$	- liquid compositions for each tray
$x^h(k)$	- a hidden unit at a discrete time k
y_i	- output vector at a discrete time i
y_i	- vapour mole fraction
$y_{i,n}$	- the vapour compositions for each tray
$y(k)$	- output vector at a discrete time k
Z	- 10.0 co-ordination number
z^{-1}	- time delay
z_i	- the feed composition (mole fraction)

Greek Symbols

- δ - surface tension
- γ_i - activity coefficient of component i
- η - viscosity
- ρ - density
- $\varphi_h(\cdot)$ - the vector valued functions corresponding to the *activation* (transfer) *functions* of the nodes in the hidden layer
- $\varphi_o(\cdot)$ - the vector valued functions corresponding to the *activation* (transfer) *functions* of the nodes in the output layer
- $\Delta P_{friction}$ - the dry hole pressure drop

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A1	Main Program for Training of Process Predictor: Top Pressure MISO Model	139
A2	Main Program for Training of Process Predictor: Bottom Temperature MISO Model	140
A3	Main Program for Training of Process Predictor: C8 Flowrate MISO Model	141
A4	Main Program for Training of Process Predictor: MIMO Model	142
A5	Subfunction for Data Scaling	144
A6	Subfunction for Data Preparation (No Delayed)	146
A7	Subfunction for Data Preparation (1 Delayed Term)	147
A8	Subfunction for Data Preparation (2 Delayed Terms)	148
A9	Main Program for Training of F1 Classifier	149
A10	Main Program for Training of F2 Classifier	151
A11	Main Program for Training of F3 Classifier	153
A12	Main Program for Training of F4 Classifier	155
A13	Main Program for Training of F5 Classifier	157
B1	Description of Tansig	159
B2	Description of Purelin	161
B3	Batch Gradient Descent with Momentum (traingdm)	162

B4	Gradient Descent with Momentum and Adaptive Learning Backpropagation (<i>traingdx</i>)	163
B5	Levenberg-Marquardt Backpropagation (<i>trainlm</i>)	165
B6	UNIQUAC	167
C1	Training Result of MISO network for Top Stage Pressure (without delay) for different training algorithm	169
C2	Training Result of MISO network for Top Stage Pressure with <i>trainlm</i>	170
C3	Training Result of MISO networks with 2 delayed terms and <i>trainlm</i>	171
C4	Training Result of MIMO networks with <i>trainlm</i>	172
C5	Training results of MISO networks fault classifier	173
C6	Training results of MIMO networks as fault classifier	175
D		176

CHAPTER I

INTRODUCTION

1.1 Motivation

Continuous and batch processes are two important modes of operations in the chemical industry and play an important role in the production and processing of high quality, specialty materials. Monitoring these processes is very important to ensure their safe operation and consistent production of high quality products. Since the manufacturing environment requires higher degrees of automation, fault diagnosis of automated manufacturing systems is becoming a critical issue in production control. Success in detecting and diagnosing faults will decrease system downtime, production delay, and overall production cost.

Research on fault detection has received increased attention in recent years as a result of fatal accidents such as those at Chernobyl and Bhopal, some of which have been traced to sensor failure. Sensors reading provide controllers and operators with a view of the process status and impact the plant operations significantly. Failures in providing correct measurement at the desired frequency can cause process disturbances, loss of control, profit loss, or even catastrophic accidents. In process control, up to 60% of the perceived malfunctions in a plant are found to stem from the lack of credibility of sensor data (Yang and Clarke, 1999). In the interest of maintaining safe and profitable operations, process plant must therefore be equipped with all the required features to provide or maintain reliable measurement.

Major or catastrophic changes are often easy to detect. Unfortunately when such events occur, irreversible damages due to the magnitude of the impact are often incurred. On the contrary, although smaller or non-catastrophic failures may not result in serious immediate damage, detection of these failures is often difficult. Sensor and measurement system failures are examples of the latter. Some of these failure events can result in undesirable process performances. For example, off-calibration instruments provide wrong measurement that will, in turn, result in wrong control action. To alleviate such problems, reliable fault detection mechanisms should be established.

Fault detection is essentially a pattern recognition problem, in which a functional mapping from the measurement space to a fault space is calculated. A wide variety of techniques have been proposed to detect and diagnose faults. Generally speaking, there are three different options available to approach a fault diagnosis problem: state estimation methods, statistical process control methods, and knowledge-based methods. A neural network, a type of knowledge-based system, possesses many desirable and preferred properties for chemical process fault diagnosis. These properties include its abilities to learn from example, extract salient features from data, reason in the presence of novel, imprecise or incomplete information, tolerate noisy and random data, and degrade gracefully in performance when encountering data beyond its range of training (Venkatasubramanian and Chan, 1989). Reviewing the development of neural network fault detection and diagnosis systems, the general trend in research is to increase the robustness of the system to unmodelled patterns, realise fast and reliable diagnosis in dynamic processes, and dynamically filter noisy data used for detection.

In this study, a model-based fault detection system proposed by Ahmad and Leong (2001) will be further developed. Figure 1.1 displays the overall structure of the system. Here, a model-based fault detection system consisting of a process predictor and a fault classifier is proposed. The process predictor is used to predict the normal fault-free operating condition of a Precut column in the Fatty Acid Plant. The deviation of the actual condition from the output of this predictor, termed the residual, is then fed to the classifier, which identifies the residual signal from the process predictor and classifies

the cause of faults. The development of both models utilises the nonlinear mapping capability of neural networks.

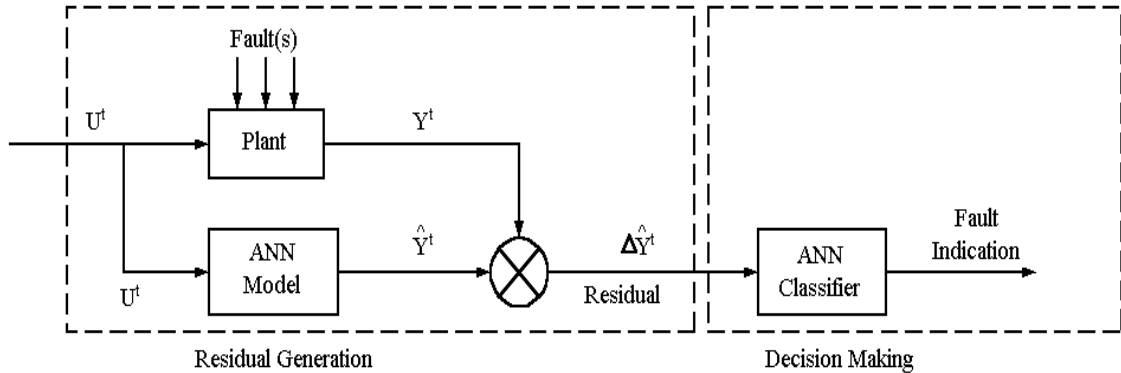


Figure 1.1 The model-based fault detection system

The residual signal plays a central role in the proposed fault detection. It is a measure of process departure from the expected normal operating condition. The idea of utilising the residual signal in fault detection originated from the concept of analytical redundancy. According to Patton et al. (1994), analytical redundancy is a procedure of using model information to generate additional signals to be compared with the original measured quantities.

In their work, Ahmad and Leong (2001) concentrated on the detection of sensor failure in the Tennessee Eastman (TE) plant. Their work revealed the associated problems in implementing the detection scheme. One major difficulty is the training of recurrent network used in the predictor. The proposed system was only capable of detecting single fault. Since various faults can in practice occur simultaneously, there is a need for multiple faults detection scheme.

1.2 Problem Statement and Importance of Study

Artificial neural networks (ANNs) have made rapid developments in the fault diagnosis of chemical processes and related fields. Process fault detection by artificial intelligence techniques have been studied by Venkatasubramanian et al. (1990), Rengaswamy and Venkatasubramanian (2000), Himmelblau (2000), Ungar et al. (1990), Watanabe et al. (1989), Wang et al. (1998), Scenna (2000), Tarifa and Scenna (1999), Pareek et al.(2002), Ferentinos (2003), Sharma et al. (1999, 2004) and others. However, most of the research has been limited to rather ‘simple’ systems that are systems which can be ably simulated by mathematical models. These models are derived from the various functions that relate the system process variables. Further, most of the work is directed towards identifying operating faults with the focus on safety/reliability aspects rather than process faults; perhaps, except for Sharma et al. (2004).

In this work, the potential of the ANNs to diagnose process faults in a commercially important Precut column in the fatty acid fractionation plant has been explored. These process faults only cause fluctuations in the product quality and yields; they do not lead to failures or operational hazards that might lead to equipment damage and/or plant shutdown. The system investigated operates at dynamic. In addition to the single fault detection and diagnosis, the ability of ANNs to extrapolate and detect leakage and multiple faults is also shown. Relative importance of the various input variables on the output variables plays a vital role in selecting the input/output nodes of ANN architecture.

The main contributions of this work to the fault detection and diagnosis techniques, which also represent the new developments in this field, are the following:

1. The proposed scheme can be applied in detection of single fault or simultaneously multiple faults.
2. The proposed scheme can also be applied to detect leakage in the stream.

3. In the development of process estimator and fault classifier, the MISO model has better generalisation ability compared to MIMO model.
4. The proposed scheme capable to detect single and multiple faults as well as leakage in the light noisy environment.

1.3 Objective and Scope of Work

The main aim of this study is to develop a multiple faults detection scheme using artificial neural networks. The proposed fault detection scheme will able to detect single or multiple faults and also leakage in the selected case study. The selected case study is a Precut Column in the Fatty Acid Plant (FAP). In order to achieve the above objective, the following scopes have been drawn.

1. Simulation of case study – Precut Column.
Simulation of the plant was carried out within HYSYS.Plant version 2.4.1 process simulator using a flowsheet provided by a local industry. In doing so, a number of modifications were implemented to successfully represent the behaviour of the FAP process especially in the Precut Column.
2. Development of process predictor.
Process predictor is used to predict the normal behaviour of the process. The development of the process predictor involves selecting an appropriate ANN architecture to differentiate the abnormal behaviour of the process and the normal condition to enable the generation of residual signal. Here, Matlab 6.1 software is used.
3. Development of fault classifier.
Fault classifier is a decision making system used to detect process faults. Residual signals generated from the process predictor serve as an input to the classifier. Structure selection and training method are the criteria that must be taken into consideration. Matlab 6.1 was used to develop the fault classifier.

4. Implementation of the proposed fault detection strategy.

Here, the proposed model-based fault detection strategy is implemented to detect faults in the Precut column. The work illustrates the detection of sensor failures and leakage in pipelines.

1.4 Thesis Organisation

In general the thesis was organised as follows:

Chapter II presents the literature overview about different approaches to the fault detection problem. Early diagnosis of the process faults while the plant is still operating in a controllable region can help avoid event progression and reduce the amount of productivity loss during an abnormal event. Due to the broad scope of the process fault diagnosis problem and the difficulties in its real time solution, various computer-aided approaches have been developed over the years. They cover a wide variety of techniques such as the early attempts using fault trees and diagraphs, analytical approaches, and knowledge-based systems and neural networks in more recent studies. The chapter also discusses neural networks that have been used for the fault detection purposes.

Chapter III elaborates the Fatty Acid Plant (FAP) in detail especially Precut column. Description of typical packed column with pumparound system was also mentioned. Simulation methodology of this case study using HYSYS. Plant dynamic simulator was described and validation of the flowsheet with the real data was presented. The mathematical modelling of the distillation operation also presented in this chapter.

Chapter IV presents in details the development of process estimation for fault detection purposes. The process estimator was used to predict the 'fault free' operating condition of the process. The model was constructed using a class of recurrent neural

network known as Elman network. Procedures for development of process estimator were explained. In this chapter, network structure and training algorithm were studied. Issues of residual generation and prediction accuracy are central to the chapter. The outputs from the process estimator are used as inputs for fault classifier.

Chapter V elaborates the development of fault classifier. This fault classifier is also based on an artificial neural network. In this chapter the study of two types of fault was discussed in details, they are corrupted sensor measurement and leakage in the pipeline. At the end of this chapter, description of the performance of the proposed fault detection scheme tested in the presence of the noise-corrupted measurements and without noise-corrupted measurements was discussed.

Chapter VI is the conclusions and recommendation section. For conclusion, the researcher proposed an effective artificial neural networks fault detection scheme that was able to detect single and multiple faults simultaneously as well as leakage in the light noisy environment. Recommendations were proposed to enhance its efficiency, application and robustness.

CHAPTER II

LITERATURE REVIEW

2.1 Overview of Fault Detection and Diagnosis

The discipline of process control has made tremendous advances in the last three decades with the advent of computer control of complex processes. Low-level control actions such as opening and closing valves, called regulatory control, which used to be performed by human operators are now routinely performed in an automated manner with the aid of computers with considerable success. With progress in distributed control and model predictive control systems, the benefits to various industrial segments such as chemical, petrochemical, cement, steel, power, and desalination industries have been enormous. However, a very important control task in managing process plants still remains largely a manual activity, performed by human operators. This is the task of responding to abnormal events in a process. This involves the timely detection of an abnormal event, diagnosing its causal origins and then taking appropriate supervisory control decisions and action to bring the process back to a normal, safe, operating state. This entire activity has come to be called Abnormal Event Management (AEM), a key component of supervisory control.

However, this complete reliance on human operators to cope with such abnormal events and emergencies has become increasingly difficult due to several factors. It is difficult due to the broad scope of the diagnostic activity that encompasses a variety of

malfunctions such as process unit failures, process unit degradation, parameter drifts and so on. It is further complicated by the size and complexity of modern process plants. For example, in a large process plant there may be as many as 1500 process variables observed every few seconds leading to information overload. In addition, often the emphasis is on quick diagnosis which poses certain constraints and demands on the diagnostic activity. Furthermore, the task of fault diagnosis is made difficult by the fact that the process measurements may often be insufficient, incomplete and/or unreliable due to a variety of causes such as sensor biases or failures.

Given such difficult conditions, it should come as no surprise that human operators tend to make erroneous decisions and take actions which make matters even worse, as reported in the literature. Industrial statistics show that about 70% of the industrial accidents are caused by human errors. These abnormal events have significant economic, safety and environmental impact. Despite of advances in computer-based control of chemical plants, the fact that two of the worst ever chemical plant accidents, namely Union Carbide's Bhopal, India, accident and Occidental Petroleum's Piper Alpha accident (Lees, 1996), happened in recent times is troubling development. Another major recent incident is the explosion at the Kuwait Petrochemical's Mina Al-Ahmedi refinery in June of 2000, which resulted in about 100 million dollars in damages.

Further, industrial statistics have shown that even though major catastrophes and disasters from chemical plant failures may be infrequent, minor accidents are very common, occurring on a day to day basis, resulting in many occupational injuries, illnesses, and costing the society billions of dollars every year (McGraw-Hill Economics, 1985; National Safety Council, 1999). It is estimated that the petrochemical industry alone in the US incurs approximately 20 billion dollars in annual losses due to poor AEM (Nimmo, 1995). The cost is much more when one includes similar situations in other industries such as pharmaceutical, specialty chemicals, power and so on. Similar accidents cost the British economy up to 27 billion dollars every year (Laser, 2000).

Thus, here is the next grand challenge for control engineers. In the past, the control community showed how regulatory control could be automated using computers and thereby removing it from the hands of human operators. This has led to great progress in product quality and consistency, process safety and process efficiency. The current challenge is the automation of AEM using intelligent control systems, thereby providing human operators the assistance in this most pressing area of need. People in the process industries view this as the next major milestone in control systems research and application.

The automation of process fault detection and diagnosis forms the first step in AEM. Due to the broad scope of the process fault diagnosis problem and the difficulties in its real time solution, various computer-aided approaches have been developed over the years. They cover a wide variety of techniques such as the early attempts using fault trees and diagraphs, analytical approaches, and knowledge-based systems and neural networks in more recent studies. From a modelling perspective, there are methods that require accurate process models, semi-quantitative models, or qualitative model. At the other end of the spectrum, there are methods that do not assume any form of model information and rely only on process history information. In addition, given the process knowledge, there are different search techniques that can be applied to perform diagnosis. Such as collection of bewildering array of methodologies and alternatives often pose a difficult challenge to any aspirant who is not a specialist in these techniques. Some of these ideas seem so far apart from one another that a non-expert researcher or practitioner is often left wondering about the suitability of a method for his or her diagnostic situation. While there have been some excellent reviews in this field in the past, they often focused on a particular branch, such as analytical models, of this broad discipline.

By way of introduction, we first address the definitions and nomenclature used in the area of process fault diagnosis. The term *fault* is generally defined as a departure from an acceptable range of an observed variable or a calculated parameter associated with a process (Himmelblau, 2000). This defines a fault as a process abnormality or

symptom, such as high temperature in a reactor or low product quality and so on. The underlying cause(s) of this abnormality, such as a failed coolant pump or a controller, is(are) called the *basic event(s)* or the *root cause(s)*. The basic event is also referred to as a *malfunction* or a *failure*. Since one can view the task of diagnosis as a classification problem, the diagnostic system is also referred to as a diagnostic classifier. Figure 2.1 depicts the components of a general fault diagnosis framework. The figure shows a controlled process system and indicates the different sources of failures in it. In general, one has to deal with three classes of failures or malfunctions as described below:

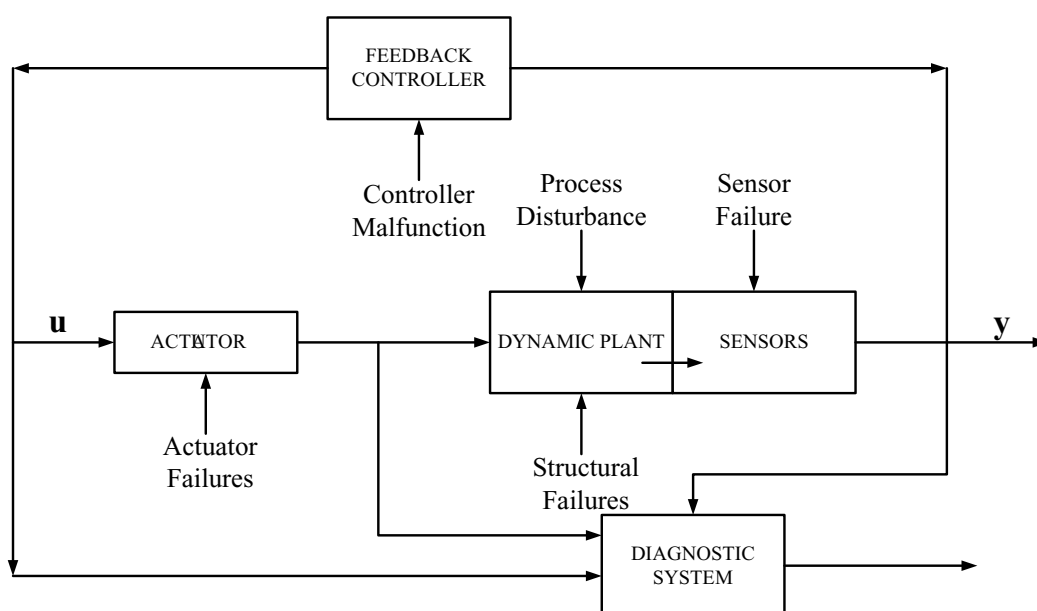


Figure 2.1 A general diagnostic framework

a) *Gross parameter changes in a model*

In any modelling, there are processes occurring below the selected level of detail of the model. These processes which are not modelled are typically lumped as parameters and these include interactions across the system boundary. Parameter failures arise when there is a disturbance entering the process from the environment through one or more exogenous (independent) variables. An example of such a malfunction is a change in the concentration of the reactant from its normal or steady-state value in a

reactor feed. Here, the concentration is an exogenous variable, a variable whose dynamics is not provided with that of the process. Another example is the change in the heat transfer coefficient due to fouling in a heat exchanger.

b) Structural changes

Structural changes refer to changes in the process itself. They occur due to hard failures in equipment. Structural malfunctions result in a change in the information flow between various variables. To handle such a failure in a diagnostic system would require the removal of the appropriate model equations and restructuring the other equations in order to describe the current situation of the process. An example of a structural failure would be failure of a controller. Other examples include a stuck valve, a broken or leaking pipe.

c) Malfunctioning sensors and actuators

Gross errors usually occur with actuators and sensors. These could be due to a fixed failure, a constant bias (positive and negative) or an out-of range failure. Some of the instruments provide feedback signals which are essential for the control of the plant. A failure in one of the instruments could cause the plant state variables to deviate beyond acceptable limits unless the failure is detected promptly and corrective actions are accomplished in time. It is the purpose of diagnosis to quickly detect any instrument fault which could seriously degrade the performance of the control system.

Outside the scope of fault detection and diagnosis are unstructured uncertainties, process noise and measurement noise. Unstructured uncertainties are mainly faults that are not modelled a priori. Process noise refers to the mismatch between the actual process and the predictions from model equations, whereas, measurement noise refers to high frequency additive component in the sensor measurements.

In this literature review, the researcher will provide a review of the various techniques that have been proposed to solve the problem of fault detection and diagnosis. Current approaches for fault detection and diagnostic have focused on the use of state estimation approaches, statistical process control (SPC) approaches, or detailed knowledge-based models.

2.1.1 State Estimation Approaches

The approaches using fundamental models are usually based on state estimation methods, which combine a fundamental model of the process with on-line measurements to provide on-line, recursive estimates of the underlying theoretical states of the process. The simplest approach for monitoring is to formulate state estimators, such as a Kalman filter, based on the stochastic models that describe the system. An alternative approach suited to chemical and biochemical CSTR and batch reactors which are subject to stochastic disturbance such as impurities and parameter variations, is to incorporate one's knowledge about these stochastic states into the state model and estimator. Monitoring the progress of chemical processes then consists of tracking the development of important deterministic state variables (such as conversion, composition, and molecular weight) with time to check whether they are following satisfactory trajectories. If unacceptable deviations in any of these states are detected, then not only an alarm can be set, but also an assignable cause can usually be found in the behaviour of the stochastic states (such as an increase in impurity concentration or decrease of a heat-transfer coefficient due to fouling). All these state estimation approaches build into their models the possible faults or reasons for deviations from normal behaviour (Nomikos and MacGregor, 1994).

The advantage of the model-based method is that it can estimate unmeasurable parameters, if they are observable. However, they require an exact process model, which

is not always available or economical to obtain in an industrial environment. In practice, the heavy computation load involved in this approach can also be a problem.

2.1.2 Statistical Process Control Approach

Nomikos and MacGregor (1994) proposed an approach based on the idea of statistical process control to monitor batch processes with empirical models developed from a multi-way principal component analysis (MPCA) of the existing batch data. In the batch-monitoring problem, the data takes the form of three-way arrays (batch run, measurement variable, and time evolution). MPCA is used to handle data. It is equivalent to performing ordinary PCA on a large two-dimensional matrix formed by unfolding the three-way array in one possible ways. The objective of this version of MPCA is to decompose the three-way array into a series of principal components consisting of score vectors and loading matrices plus a residual matrix. MPCA explains the variance-covariance structure of a multivariate dataset through a few linear combinations of the original variables whose properties of variance reveal the underlying model. The principal components, each characterized by a loading vector and a score vector, represent the selection of a new coordinate system, and are ordered from the largest variation to the smallest one. Compared with non-directional Statistical Process Control methods, such as the Ordinary Least Squares, the proposed multivariate statistical procedures provide more diagnostic information from the underlying MPCA model by effectively filtering the noise.

MPCA provides a tool for investigating very large databases of batch data and obtain valuable knowledge about the process. However, MPCA and MPLS (Multi-way Partial Least Squares) model are not cause and effects models, but rather only models of the correlation structure of the process variables under routine operating conditions. They cannot be used to predict the effects that independent changes in some of the measurement variables will have on the quality of the final product. To suggest a

corrective action when a fault is detected, MPCA has to use some on-line diagnostic tools to interrogate the underlying projection model for possible reasons for the fault, and respond accordingly using one's process knowledge, a cause and effect model, or an expert system. Moreover, Principal Component Analysis, Principal Component Regression, and Least Squares all rely on linear models, while most chemical systems are highly non-linear processes. There are some non-linear versions for those algorithms. Nevertheless, their capability to model any nonlinear relationship is still limited by the assumed basis functions used in the regression.

2.1.3 Knowledge-Based Approaches

Knowledge-based approaches use expert systems or artificial intelligence methods to process data. In rule-based expert systems, the process model is represented by a set of qualitative and quantitative governing rules, based on the knowledge about the process variable from operators and engineers. These behavioural and causal descriptions are arranged in a hierarchical structure, and diagnostic rules for each node in the hierarchy are generated from those descriptions.

Fault diagnosis using rule-based expert systems needs an extensive database of rules and the accuracy of the diagnosis depends on the rules. Creating a rich and detailed database of rules is usually a time-consuming task and many process experts are needed. Also, the inability of the system to learn or dynamically improve its performance, and unpredictability of the system outside its domain of expertise are obvious problems when large industrial plants are considered.

Neural networks, a pattern-recognition method, identify process faults based on the operating conditions of a given process. In contrast to expert systems where the analysis is based on deductive or inductive logic, pattern recognition involves reasoning through generalisation from a set of examples. This approach is particularly useful when

expert knowledge or theoretical first principal models are lacking. Artificial neural networks can easily handle non-linear processes and no mechanistic process model is needed; they learn the diagnosis by means of the information included in the training data set. Neural networks are noise tolerant and have some ability to generalise knowledge from a set of examples, as well as to adapt during use.

Neural network computations are collectively performed by the entire network with knowledge represented in the connection weights between processing elements. Consequently, the collective operations result in a high degree of parallel computation, which enables the network to solve complex problems rapidly. In addition, the distributed representations lead to greater fault tolerance and to graceful degradation when problems are encountered beyond its range of training. Other beneficial attributes include the ability to perform useful generalisations from specific examples, and adjust dynamically to environmental changes, e.g., a Radial Basis Function Network can theoretically be easily adapted in the occurrence of a new input pattern simply by locally adjusting some centre vectors or adding new ones. Problems that are common to most estimation methods, such as the sufficient richness of the training data and the computational effort required to train the models, are the main disadvantages of this approach (Sorsa and Koivo, 1993; Lou et al., 2003).

The remaining part of this section introduces the algorithms of two neural network models which have been studied in this project. The two neural networks are Feedforward Neural Network and Recurrent Neural Network.

2.2 Artificial Neural Networks (ANNs)

Early works in the field of neural networks centred on modelling the behaviour of neurons found in the human brain. Engineering systems are considerably less complex than the brain, hence from an engineering viewpoint ANN can be viewed as

nonlinear empirical models that are especially useful in representing input-output data, making predictions in time, classifying data, and recognising patterns.

Despite the above mentioned capabilities, ANNs are not a solution for all modelling problems. Therefore, it is necessary to understand the advantages and disadvantages of ANN in contrast with first principle models or other empirical models to determine their applicability for a particular problem. ANN has several advantages as described by Baughman and Liu (1995):

1. *Distribution of information over a field of nodes.* This feature allows greater flexibility and robustness of ANN because a slight error or failure in certain sections of the network will not affect the entire system.
2. *Learning ability of ANN.* ANN is able to adjust its parameters in order to adapt itself to changes in the surrounding systems by using an error-correction training algorithm.
3. *Extensive knowledge indexing.* ANN is also able to store a large amount of information and access it easily when needed. Knowledge is kept in the network through the connection between nodes and the weights of every connection.
4. *Imitation of the human learning process.* The network can be trained iteratively, and by tuning the strengths of the parameters based on observed results. The network can develop its own knowledge base and determine cause and effect relations after repeated training and adjustments.
5. *Potential for on-line use.* Once trained, ANN can yield results from a given input relatively quickly, which is a desired feature for the on-line use.

Some of the limitation of ANN summarised by Baughman and Liu (1995):

1. *Long training time.* Training time for ANN can take too much time especially for large networks.

2. *Requires large amount of data.* ANN needs large amount of input-output data for a better generalisation. Therefore, if there is only a small amount of input-output data available, ANN may not be suitable for modelling the system.
3. *No guarantee to optimal results and reliability.* Although the network contains parameters that can be tuned by the training algorithm, there is no guarantee that the resulting model is perfect for the system. The tuned model may be accurate in one region but inaccurate in another
4. *Difficulty in selecting good sets of input variables.* Selection of input variables is difficult because too many input variables or wrongly selected input variables will cause overfitting and poor generalization. Too little or inappropriate input variables will lead to poor mapping of the system.

2.2.1 Neuron (*Node*) and Neural Networks

Figure 2.2 shows the basic structure of a single processing unit in an ANN which will be referred to as a *node* in this work and is a simplified mimic of a single neuron in the human brain. A node receives one or more input signals, u_i , which may come from other nodes or from some other source. Each input is weighted according to the value $w_{i,j}$ that is called *weight*. These weights are similar to the synaptic strength between two connected neurons in the human brain. The weighted signals to the node are summed and the resulting signal, called the *activation*, h , is sent to the *transfer function*, g , which can be any type of mathematical function, but is usually taken to be a simple bounded differentiable function such as the sigmoid. The resulting output of the node y_i , may then be sent to one or more nodes as an input or taken as the output of a ANN model.

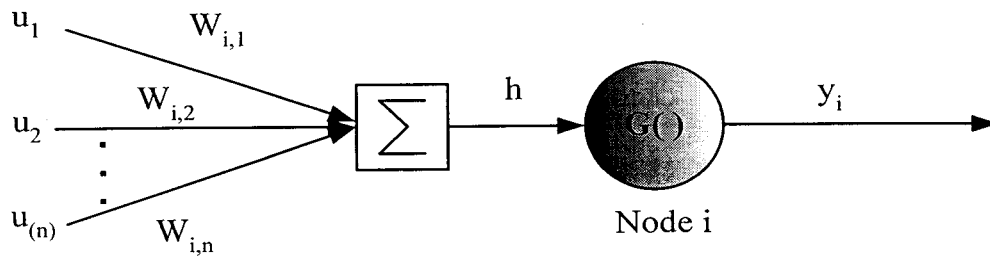


Figure 2.2 Structure of a single processing node.

Neural network consists of interconnected neurons arranged into several layers. A group of nodes called the *input layer* receives a signal from some external source. In general, this input layer does not process signal unless it needs scaling. Another group of nodes, called the *output layer*, return signals to the external environment. The remaining nodes in the network, are called *hidden nodes* because they do not receive any signals from or send a signal to an external source or location. The hidden nodes may be grouped into one or more *hidden layers* depending on the architecture of the network. Each of the connection between two nodes has a weight associated with it. Figure 2.3 shows a network with fully connected layers. The connections are in forward direction and are known as a multilayer feedforward neural network.

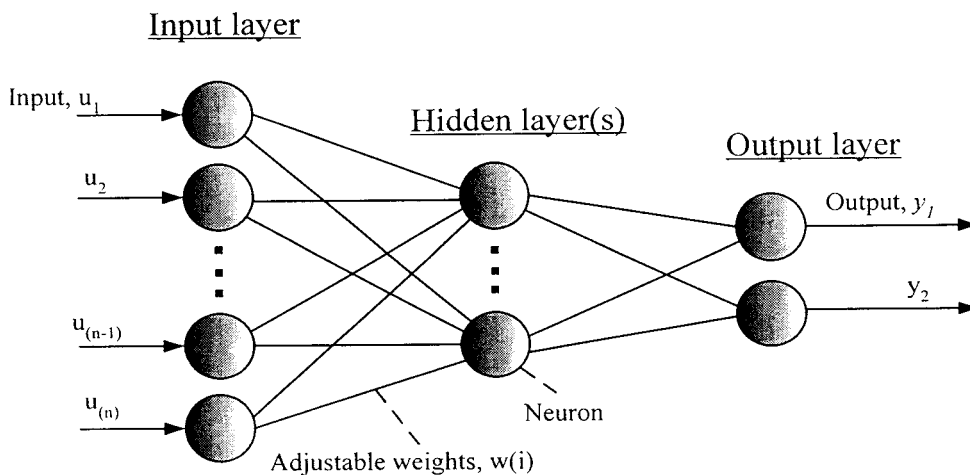


Figure 2.3 Structure of a layered neural network.

In principle, connections between nodes can be in any direction for nodes in nonadjacent layers or within the same layer. Feedback connections from a node in one

layer to a node in a previous layer can also be made. When feedback connections are involved, the network is referred to as recurrent networks.

Due to the complexity of the network, analytical method of calculating the values of the weights for a particular network to represent process behaviour is not discouraging. Instead the network must be *trained* on a set of data (called the *training set*) collected from the process to be modelled. Training is a procedure of estimating the values of the weights and establishing the network structure, and the algorithm used to do this is called a 'learning' algorithm. The learning algorithm is essentially an optimisation algorithm. Once a network is trained, it can be conveniently used as a model to represent the system for various different purposes.

A key difficulty with optimisation for neural network problems is that multiple minima occur especially in large networks. Since most training procedures used for neural networks typically find local minima starting from randomly selected guesses for the parameters, it should be expected that local minima of varying quality will be found. While use of a global optimisation procedure, such as genetic algorithms, branch and bound, or simulated annealing might thus appear to be called for, the training time for such algorithms oftentimes expands to an unacceptable degree. This practically limits their application.

Regardless of what training algorithm is used to calculate the values of the weights, all of the training methods go through the same general steps. First, the available data is divided into a training and test set(s). The following procedure is then used to determine the values of weights of the network:

1. for a given ANN architecture, the values of the weights in the network are initialized as small random numbers;
2. the inputs of the training set are sent to the network and the resulting outputs are calculated;

3. some measure (an objective function) of the error between the outputs of the network and the known correct (target) values are calculated;
4. the gradient of the objective function with respect to each of the individual weights are calculated;
5. the weights are changed according to the optimisation search direction and step length determined by the optimisation code;
6. the procedure returns to step 2;
7. the iteration terminates when the value of the objective function calculated using the data in the test set starts to increase.

As mentioned above, the available data is divided into the training and test set. The purpose of partitioning the available data into the training and test set is to evaluate how well the network *generalises* (predicts) to domains that were not included in the training set. For non-trivial problems it is often difficult to collect all of the possible input-output patterns needed to span the input-output space for a particular behaviour or process to be modelled. Therefore, the network should be trained with some subset of all the possible input-output patterns. In addition, the training set must also be representative of the domain of interest. If not, the net may not predict well for similar data, and may predict poorly for completely novel data (extrapolate). The test set is used to evaluate how well the neural network generalises using the weights computed during the training exercise. Since it is the ability of the network to predict 'unseen' data serves as a measure of model capability, the performance on the test set is used to select the optimal set of weights as well as network topology.

2.2.2 Feedforward Neural Networks

Two layer (sometimes called three layers) feedforward ANN are commonly encountered models in the literature. Computation nodes are arranged in layers and information feeds forward from layer to layer via weighted connections as illustrated in

the Figure 2.4. Here, circles represent computation nodes (transfer functions), and lines represent weighted connections. The bias thresholding nodes are represented by squares.

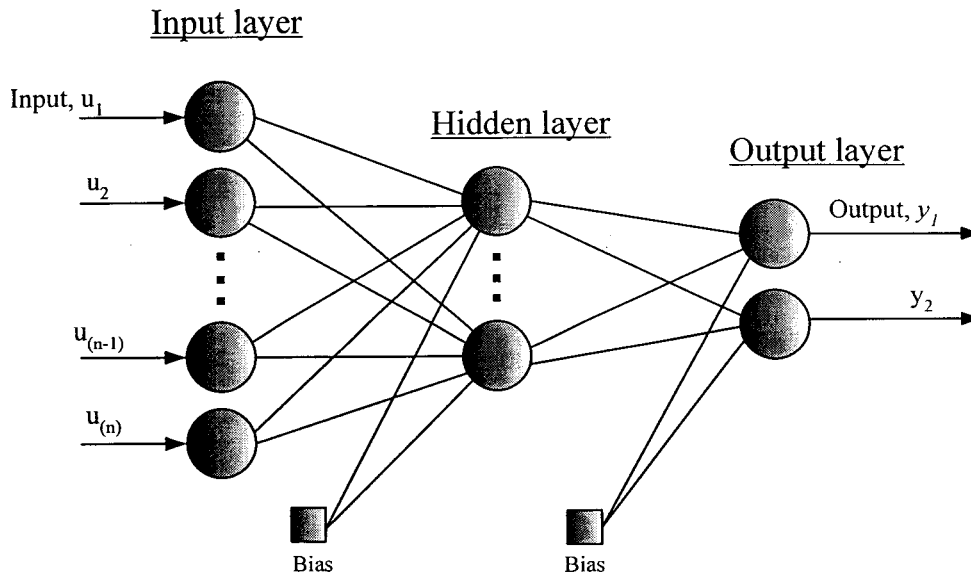


Figure 2.4 Graph of the information flow in a feedforward neural network.

Mathematically, the typical feedforward network can be expressed as

$$y_i = \varphi_o [C \varphi_h (B u_i + b_h) + b_o] \quad (2.1)$$

where y_i is the output vector corresponding to input vector u_i , C is the *connection matrix* (matrix of weights) represented by arcs (the lines between two nodes) from the hidden layer to the output layer, B is the connection matrix from the input layer to the hidden layer, and b_h and b_o are the bias vector for the hidden and output layer, respectively. $\varphi_h(\cdot)$ and $\varphi_o(\cdot)$ are the vector valued functions corresponding to the *activation* (transfer) *functions* of the nodes in the hidden and output layers, respectively. Thus, feedforward neural network models have the general structure of

$$y_i = f(u) \quad (2.2)$$

where $f(\cdot)$ is a nonlinear mapping. Hence feedforward neural networks are structurally similar to nonlinear regression models, and Eq.(2.2) represents a steady state process.

To use models for identification of dynamic systems or prediction of time series, a vector comprised of a moving window of past input values (*delayed coordinates*) must be introduced as inputs to the net. This procedure yields a model analogous to a nonlinear finite impulse response model where

$$y_i = y_t \text{ and } u_i = [u_t, u_{t-1}, \dots, u_{t-m}] \text{ or } y_t = f([u_t, u_{t-1}, \dots, u_{t-m}]) \quad (2.3)$$

The lengths of the moving window must be long enough to capture the system dynamics for each variable in practice. In practice, the duration of the data windows are determined by trial and error (cross validation) and each individual input and output variable might have a separate data window for optimal performance.

Backpropagation learning algorithm is one of the earliest and the most common method for training multilayer feedforward neural networks. Development of this learning algorithm was one of the main reasons for renewed interest in this area and this learning rule has become central to many current works on learning in ANN. It is used to train nonlinear, multilayered networks to successfully solve difficult and diverse problems.

Standard backpropagation is a gradient descent algorithm in which the network weights are moved along the negative of the gradient of the performance function. The term backpropagation refers to the manner in which the gradient is computed for nonlinear multilayer networks. There are a number of variations on the basic algorithm that are based on other standard optimisation techniques. These techniques will be discussed below (Demuth and Beale, 2000):

1. *Gradient Descent with Momentum Backpropagation*
Known as *traingdm* in MATLAB Toolbox. Details of this algorithm can be found in Appendix B3. This batch algorithm for feedforward networks provides faster convergence. Momentum allows a network to respond not only to the local gradient, but also to recent trends in the error surface. Acting like a low-pass filter, momentum allows the network to ignore small features in the error surface. Without momentum a network may get stuck in a shallow local minimum. With momentum a network can slide through such a minimum.
2. *Gradient Descent with Momentum and Adaptive Learning Backpropagation*
This algorithm is also known as *traingdx* in MATLAB Toolbox. The performance of the steepest descent algorithm can be improved if we allow the learning rate to change during the training process. An adaptive learning rate will attempt to keep the learning step sizes as large as possible while keeping learning stable. The learning rate is made responsive to the complexity of the local error surface. Details of this algorithm can be found in Appendix B4.
3. *Levenberg-Marquardt Backpropagation*
Details of this algorithm can be referred in Appendix B5. This algorithm is known as *trainlm* in MATLAB Toolbox, actually a hybrid of the Gauss-Newton Nonlinear Regression method and Steepest Gradient Descent method. Gauss-Newton method is slow converging while Steepest Gradient Descent method suffered local minimum problem. Lavenberg-Marquardt method was designed in such a way that it can choose wisely the direction by crossing between these two methods. This algorithm appears to be the fastest method for training moderate-sized feedforward neural networks (up to several hundred weights).

These algorithms are used in this research to train feedforward neural networks and their performances in term of mean squared error will be compared.

2.2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNN) have architectures similar to standard feedforward neural networks with layers of nodes connected via weighted feedforward connections, but also include time delayed feedback or recurrent connections in the network architecture. Examine Figure 2.5.

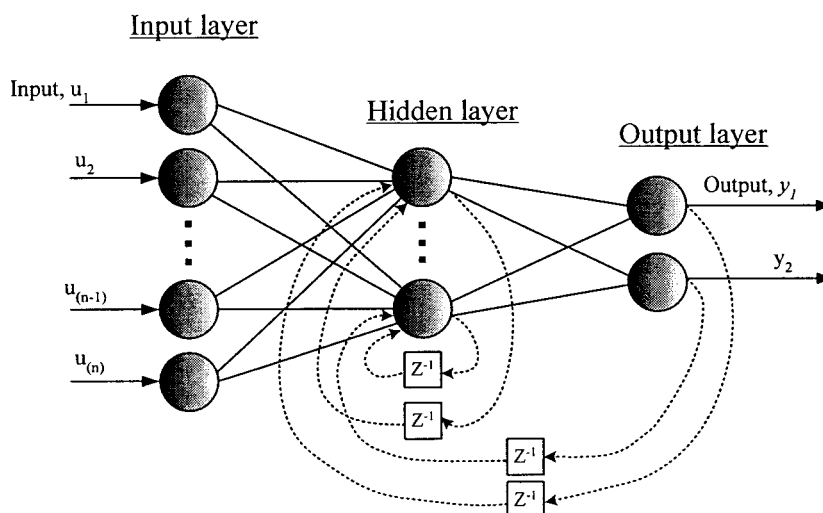


Figure 2.5 Representation of internally/externally recurrent neural networks.

Figure 2.4 shows a representation of internally/externally recurrent neural networks. Circles represent computation nodes, lines represent weighted connections, z^{-1} indicates time delay. For clarity not all recurrent connections are shown and bias nodes are omitted. Two individual variations of RNN architectures are commonly employed. The first is called Internally Recurrent Network (IRN) that is characterized by time delayed feedback connections from the output of hidden nodes to its input. This feedback path allows IRN to learn to recognise and generate temporal patterns, as well as spatial patterns. The remainder of the network comprises standard feedforward architecture. This structure is also known as an Elman network (Elman, 1990). Externally Recurrent Networks (ERN), on the other hand, contain time delayed feedback connections from the output layer to the hidden layer.

Although the ERN and IRN can exhibit comparable modelling performance, they have different features that make one more desirable than the other for a particular process. Just like the conventional linear state space model, the IRN does not have any structural limit on the number of model states because the number of hidden nodes can be freely varied. The ERN, however, can only have the same number of states as model outputs because the outputs are the states. The IRN thus tends to be more flexible in modelling.

2.2.4 Elman Networks

The Elman network is commonly a two-layer network with feedback from the hidden layer output to the hidden layer input. This recurrent connection allows the Elman network to both detect and generate time-varying patterns. A block diagram of two-layer Elman network is shown below.

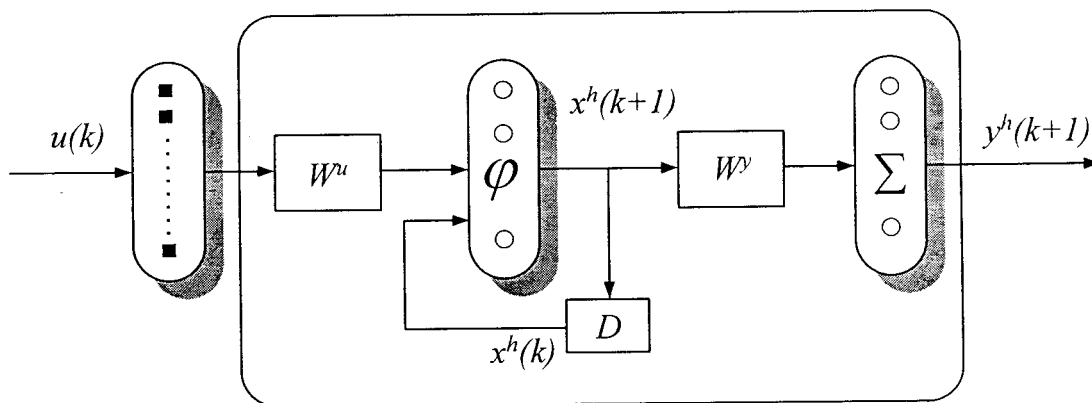


Figure 2.6 Block diagram of Elman network.

To understand the feature offered by Elman network, consider a multivariable plant with m inputs and q outputs, described by a general nonlinear input-output discrete time state space model (Elman, 1990):

$$x(k+1) = f\{x(k), u(k)\} \quad (2.4)$$

$$y(k) = g\{x(k)\} \quad (2.5)$$

where $f: \mathfrak{R}^{n+p} \rightarrow \mathfrak{R}^n$ and $g: \mathfrak{R}^n \rightarrow \mathfrak{R}^q$ are non-linear functions; $u(k) \in \mathfrak{R}^m$, $y(k) \in \mathfrak{R}^q$ and $x(k) \in \mathfrak{R}^n$ are, respectively, the input vector, the output vector and the state vector, at a discrete time k . In addition to the input and the output units, the Elman network has a hidden unit, $x^h(k) \in \mathfrak{R}^n$. $W^u \in \mathfrak{R}^{n \times p}$ and $W^y \in \mathfrak{R}^{q \times n}$ are the interconnection matrices, respectively, for the input-hidden layer and hidden-output layer. Theoretically, an Elman network with n hidden units is able to represent an n^{th} order dynamic system. The recurrent and output of the Elman network is described by the difference equations (2.6) – (2.8).

$$s(k+1) = x^h(k) + W^u u(k) \quad (2.6)$$

$$x^h(k+1) = \varphi\{s(k+1)\} \quad (2.7)$$

$$y^h(k+1) = W^y x^h(k+1) \quad (2.8)$$

where $s(k) \in \mathfrak{R}^n$ is an intermediate variable and $\varphi(\cdot)$ is an hyperbolic tangent function.

Elman network has hyperbolic tangent sigmoid transfer function (known as *tansig* in MATLAB Toolbox) neurons in its hidden (recurrent) layer, and linear transfer function (known as *purelin* in MATLAB Toolbox) neurons in its output layer. Details of these two transfer functions can be referred in Appendices B1 and B2. This combination is special in that two-layer networks with these transfer functions can approximate any function (with a finite number of discontinuities) with arbitrary accuracy. The only requirement is that the hidden layer must have enough neurons. More hidden neurons are needed as the function being fit increases in complexity.

Note that the Elman network differs from conventional two-layer feedforward networks since the first layer has a recurrent connection. The delay in this connection stores values from previous time step, to be applied in the current time step. Because the network can store information for future reference, it is able to learn temporal patterns as well as spatial patterns. The Elman network can be trained to respond to, and to generate, both kinds of patterns.

When training Elman network the following occurs at each epoch:

1. The entire input sequence is presented to the network, and its outputs are calculated and compared with the target sequence to generate an error sequence.
2. For each time step, the error is backpropagated to find gradients of errors for each weight and bias. This gradient is actually an approximation since the contributions of weights and biases to errors via the delayed recurrent connection are ignored.
3. This gradient is then used to update the weights with the backpropagation training function.

For an Elman network to have the best chance at learning a problem it requires a relatively larger number of neurons in its hidden layer. The Elman network is less able to find the most appropriate weights for hidden neurons since the error gradient is approximated. Therefore, having a fair number of neurons to begin with makes it more likely that the hidden neurons will start out dividing up the input space in useful ways.

2.3 Issues in Neural Network Applications to Fault Detection

In the preliminary stage of this research, the literature has been reviewed, regarding the following topics, with an objective to minimising misclassification in fault

detection and diagnosis problems: 1) inputs patterns;2) generalisation and improving generalisation ability of neural networks;and 3) fault diagnosis systems.

2.3.1 Inputs Patterns

Training patterns has been a rich topic for discussion due to at least two difficulties faced quite often in neural network application to fault detection: 1) The neural networks have difficulty in distinguishing fault events that share common symptom patterns;2) Dynamic detection as opposed to steady state behaviour is difficult.

1) Common Symptom Pattern

Misclassification of faults often occurs due to different faults having similar inputs patterns referred to as common symptoms. In 1990, Venkatasubramanian et al. used a neural network with steady state patterns for fault diagnosis for a fluidised catalytic cracking process. The authors realised that the faults, which are similar to the training symptoms, tend to get weighted more in the corresponding outputs generated by the neural network, as compared to other faults. This is because neural networks perform cooperative computation, that is, the output from any node depends on the contributions from all other nodes in the network it is connected to.

Later, when people turned their interest to dynamic fault detection, the input pattern became an even more significant problem. The classification of individual measurement patterns is not always unique in dynamic situations. More precisely, the symptoms caused by two different faults could be very similar to each other during some period of the operation, which may initiate a false alarm (Tsai et al., 1996;Maki et al., 1997). Great effort has been made on determining the structure and developing

algorithms for neural networks in order to solve the problem, but little has been achieved.

2) *Input Patterns for Dynamic Fault Detection*

Neural Networks were initially applied to fault detection in chemical engineering at the end of 1980's. Dynamic fault detection and diagnosis was soon identified as a big challenge and has been actively studied in the past decade. Here, the term "dynamic" implies that the network monitors the process on line and provides its diagnosis while the system is operating during transients.

It is well known that in batch processes the system is continuously changing with time. Moreover, for processes like a CSTR, it is also necessary to tackle dynamic fault diagnosis problem, because whenever a fault enters a system, the process will go through a transient until it reaches a new steady state. Since some state variables have large time constants, the transient may last a long time. If the network is to be trained with steady state data, there are only two patterns, normal and faulty. It is very unlikely that the network trained by the steady state patterns will be able to predict the fault correctly right after the corresponding transient begins.

Although different state variables evolve in a different way during the transient stage, Li et al. (1990) have suggested approximating the transient by one of two simple forms: ramp upwards and ramp downwards. The authors noticed that neural networks fail tracking faults in ramp form, if they are trained only with step changes. The network can follow the ramp change quite well, after it is trained with such a pattern. Since the network is trained with dynamic data, instead of steady state data, the network was more sensitive to noise, because of poor signal to noise ratio at the beginning of a fault, even though the noise level was still acceptable. It should be remembered that for networks trained with steady state data, Boolean variables (i.e., fault or no-fault) can be used to train the output layer. This is less feasible for dynamic situations since the variables change continuously with time from a fault to a no-fault situation or vice versa. The

trade-off between fault sensitivity and noise sensitivity in selecting the training data set is one of the features of dynamic fault detection by neural networks.

2.3.2 Generalisation and Improving Generalisation Ability of Neural Networks

Generalisation is usually a desirable property in the application of Neural Networks, especially when undersized training sets are used, when parent distributions of fault classes undergo shifts subsequent to training, or when the input data is corrupted by missing or biased sensors. These situations cause relatively high error rates in Neural Network classification. Applications of Back Propagation Networks (BPNs) to failure state recognition in chemical plants has been studied by Hoskins and Himmelblau (1988), Watanabe et al. (1989), Hoskins et al. (1989), Ferrada et al. (1990), Hagar et al. (1990), Venkatasubramanian et al. (1990) and Sharma et al. (2004). These papers have shown that the BPNs have the ability to learn the classifications of a set of training examples, and can often successfully generalise this knowledge to classify new cases of known failure types.

Kramer and Leonard (1990) pointed out that, while BPN can usually produce decision surfaces that correctly classify the training examples, regions of the input space not occupied by training data may not be classified correctly. As a result, the networks may not accurately extrapolate from the training data. Kramer and Leonard believed that distance-based classifiers should be preferred to BPN in diagnosis applications. Classifiers based on distance metrics assign regions of the input space according to their proximity to the training data, and thus extrapolation is not arbitrary but based on the most relevant data. Later, Kramer and Leonard (1991) proposed Recurrent Neural Network (RNN) that uses radial basis functions instead of sigmoid threshold units, as BPN does. The radial basis function provides an estimate of how close a test case to the original training data. It allows the classifier to determine that a test case potentially

represents a novel class. For applications where this type of behaviour is important, such as fault diagnosis, the RNN offers clear advantages over the BPN.

Improving Generalisation

One of the problems that occur during neural network training is network overfitting. In this case, the error on the training set is driven to a very small value, but when new data is presented to the network the error is large. This is because the network has memorised the training examples, but is has not learned to generalise to new situations.

A way to improve network generalisation is to use a network that is just large enough to provide an adequate fit. The larger a network you use, the more complex the functions the network can create. If we use a small enough network, it will not have enough power to over fit the data. Unfortunately, it is difficult to know beforehand how large a network should be for a specific application. There are two other methods for improving generalisation that are regularisation and early stopping (Demuth and Beale, 2000). For this research, early stopping technique will be implemented.

In early stopping technique the available data is divided into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set. The error on the validation is monitored during the training process. The validation error will normally decrease during the initial phase of training, as does the training set error. However, when the network begins to overfit the data, the error on the validation set will typically begin to rise. When the validation error increases for a specified number of iterations, the training is stopped, and the weights and biases at the minimum of the validation error are returned.

The third subset which is the test set is not used during the training, but it is used to compare different models. It is also useful to plot the test set error during the training

process. If the error in the test set reaches a minimum at a significantly different iteration number than the validation set error, this may indicate a poor division of the data set.

2.3.3 Process Fault Detection and Diagnosis Using Neural Network

This section is reviewing some representative modelling methodologies using neural networks. Although a single neural network is most frequently used, sometimes it is also integrated with other techniques or other neural networks into a more powerful modelling system. The main purpose is, of course, to overcome some inherent shortcoming of single neural network model and meet the requirement of complicated real applications. The review here will focus on the construction, and the merits of those modelling systems.

1) *Nonlinear Partial Least Squares Modelling using Neural Network*

Qin and McAvoy (1992) propose a Neural Network Partial Least Square (NNPLS) method, which has the potential of modelling any continuous nonlinear relation and also attaining the robust properties of PLS regression. This model structure integrates the PLS regression and neural networks to formulate an approach which can handle nonlinearity, the occurrence of correlated inputs and limited observations. The PLS method first projects the data down to a number of principal factors, and then neural networks are used to model the factors by 1-D linear regression. In other words, the outer relation is treated with PLS, whereas the neural networks are used as the inner regressors. A direct benefit of doing so is that only a SISO network is trained at each time, i.e., one neural network per latent variable. Thus, a multi-input-multi-output nonlinear modelling task is decomposed into linear outer relations and simple nonlinear inner relations, which are handled by a number of single-input-single-output networks. Since only a small network is trained at one time, over-parameterisation is circumvented

even when the training data are very sparse. Furthermore, the quality of modelling can be visualised by plotting the inner relations, which can be used for further analysis such as outlier detection (Qin and Li, 2001; Seongkyu and MacGregor, 2001).

2) *Hybrid Intelligent System*

Insufficient amount of data for training may result in performance deficiency of a neural network model. A hybrid intelligent system (HIS) is developed by Ye and Xoa (1996) to overcome this problem. It integrates neural networks with a procedural decision making algorithm to implement hypothesis-test cycles of system fault diagnosis. In their system, two separate neural networks were developed to support deficiency detection. One neural network implements hypothesis generation, and is trained with symptom-cause pairs. The other neural network implements hypothesis testing, and is trained with cause-symptom pairs. Thus, trained on the same cause-effect structure of the manufacturing system but in reverse directions, the two neural networks represent different sets of knowledge about the manufacturing system. Differences in their knowledge structure are utilised to prevent the two neural networks from exhibiting the same type of irrational behaviour at the same time.

Ye and Xoa (1996) found that a problem with single neural network for system fault diagnosis is the misclassification of faults with common or similar symptoms. They believed that the use of an on-site verification is the only way to solve this problem if no other information, except the symptom pattern, is available. The HIS decomposes a complex, large diagnostic problem involving multiple faults into a series of simple, small diagnostic problem involving single faults. Specifically, the HIS first searches for a fault cause, then verifies the fault cause, and finally corrects the fault cause to reduce the total number of potential fault causes.

Samanta et al. (2003) compared the performance of bearing fault detection using two different classifiers, namely, artificial neural networks and support vector machines (SMVs). The time-domain vibration signals of a rotating machine with normal and

defective bearings are processed for feature extraction. The extracted features from original and preprocessed signals are used as inputs to the classifiers for two-class (normal or fault) recognition. The classifier parameters, e.g., the number of nodes in the hidden layer in case of ANNs and the radial basis function kernel parameter (width) in case of SVMs along with the selection of input features are optimised using genetic algorithms. The classifiers are trained with a subset of the experimental data for known machine conditions and are tested using the remaining set of data. The procedure is illustrated using the experimental vibration data of a rotating machine. The role of different vibration signals and signal preprocessing techniques are investigated.

3) *Hierarchical Artificial Neural Network*

Watanabe et al. (1994) proposed a micro-architecture of neural networks called Hierarchical Artificial Neural Networks (HANN), which in its simplest form would comprise two connected stages. In the first stage, all measurements are used as inputs to a neural network. In the second stage, not only the measurements but also the Boolean outputs of the neural network in the first stage are fed to a group of neural networks. The output of the HANN is the result of a logical OR' operation on the outputs of all the neural networks in the second stage. The same concepts have been used by several researchers. (Patton et al., 1994; Benkhedda and Patton, 1996; Rengaswamy et al., 2001, Ahmad and Leong, 2001, Chen and Lee, 2002).

The concept underlying the HANN is to develop architecture for neural network that can not only diagnose faults correctly or at least with a minimum misclassification rate, but also can be trained easily. Such architecture can be an optimal task-specific macro-architecture for the problem of diagnosing multiple faults. The HANN divides a large number of patterns into many smaller subsets so the classification can be carried out more effectively via a neural network. The main objective to study this concept and use to detect multiple faults happened in Precut column.

In this study, the neural network-based fault detection and diagnosis scheme is implemented in detecting sensor failure as well as stream leakage in the Precut column. For residual generation, the neural network replaces the analytical model that describes the system under normal operation. First, the neural network has to be trained for this task. For this purpose, an input data base and a corresponding output data have to be applied. These data can either be collected directly at the system, if possible, or with the help of a simulation model that is realistic as possible. The latter possibility is of special interest for collecting data on the different faulty situations in order to test the residual generator, since generally those data are not available for the real system. The training is then performed and after finishing the training, the neural network is ready for residual generation. In order to perform the fault diagnosis, another neural network has to be fed with residuals, which generated from the first neural network. The residual data base and a corresponding signature data base are employed to train the second neural network. After finishing the training, the second neural network can be used for diagnosis to decide whether a fault has occurred, and indicate its possible cause.

2.4 Summary

Process Fault Diagnosis (PFD) involves interpreting the current status of the plant given sensor readings and process knowledge. Early diagnosis of the process faults while the plant is still operating in a controllable region can help avoid event progression and reduce the amount of productivity loss during an abnormal event. Modern industrial plants are extremely complex, and there is a growing demand for fault detection and diagnosis scheme. Various approaches have been suggested and developed for fault detection and diagnosis. One of the methods is using the ability of artificial intelligence method such as neural networks.

Almost all the methods that have been developed consider the detection and diagnosis of a single fault only, where as in practice, numerous faults can happen at the

same time. Therefore, the development of multiple faults detection and diagnosis is very important.

The potential of neural networks for fault detection and diagnosis in chemical engineering has been demonstrated in recent years. The neural network approach is especially applicable to systems for which first principle mathematical models are difficult to formulate. Neural networks approximate nonlinear and undetermined processes, and learn diagnosis tasks by means of empirical information referred to as training data. They are noise-tolerant, and their ability to generalise from the learned model, as well as to adapt during their use, are extremely useful properties.

The next chapter describes the industrial process modelled in this research. The industrial process, Precut column, is a very challenging process where fatty acids are split from the feed materials. A dynamic simulation of a Precut column is done using HYSYS.Plant software.

CHAPTER III

PLANT SIMULATION

3.1 Process Description

In this work, a Precut column in a Fatty Acid Plant is used as the case study. The Fatty Acid fractionation unit consists of a Precut column, Light Cut column, Middle Cut column, Still and Residue Still connected in series. The feed to the plant is either Palm Kernel Oil (PKO) or Palm Stearine. All the distillation columns operate at highly vacuum condition generated using steam ejectors. The columns are packed with structured packing to provide the desired separation properties. The schematic diagram of the plant is presented in the Figure 3.1.

The Dehydrated Crude Fatty Acid (DCFA) from PKO contains all fatty acid fractions from C6 to C18, plus a residue. The various fractions are separated into the following components:

- C10 and lighter cut, recovered as distillate in the Precut Column
- C12 cut, recovered as distillate in the Light Cut Column
- C14 cut, recovered as distillate in the Middle Cut Column
- Mixed C16 and C18 cut, recovered as distillate in the Still
- Pitch, recovered as bottoms in the Residue Still (the Residue Still Distillate being recycled to the Still).

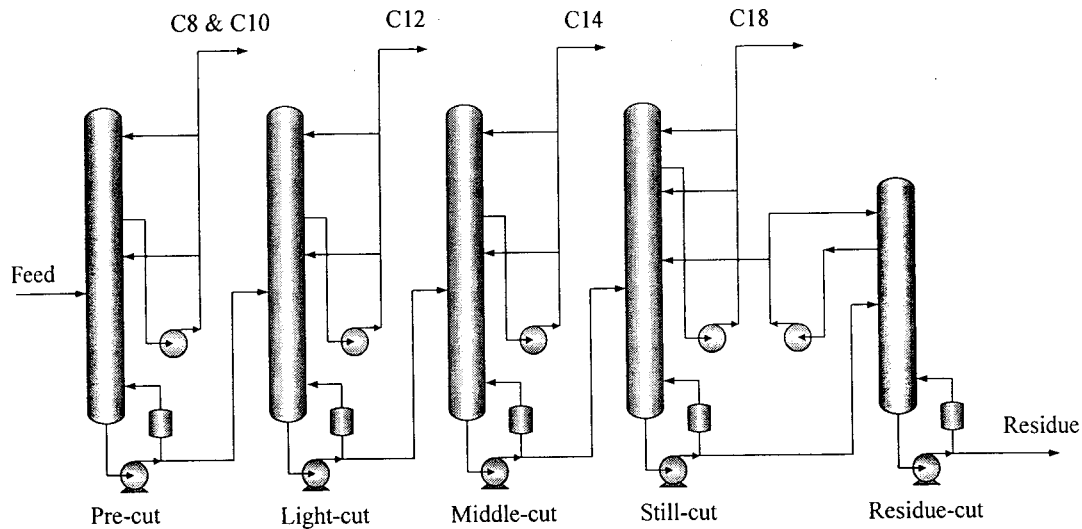


Figure 3.1 Schematic diagram of a fractionation process

3.1.1 Precut Column

The hydrogenated DCFA feed from the Hydrogenated Unit or from storage is filtered and heated successively by DCFA in a preheater and by hot oil in a feed heater and put through a dryer-deaerator where air and water are removed by vacuum. It is then flashed into the Precut column. The vapours, consisting of the C6, C8 and C10 fractions, are condensed by direct contact in the pumparound section of the column. The condensed distillate is pumped by the reflux pump and cooled by cooling water in the condenser before being returned to the top of the pumparound section. Part of the hot distillate is sent as reflux to the rectification section of the column. The net distillate is pumped to storage under level control. The bottoms are pumped by the bottoms pump and heated by hot oil, under pressure to suppress vaporisation, in the reboiler. The net bottoms product is pumped, to the Lights Cut column. Vacuum is maintained at the top of the column by the second stage of the Precut column ejector.

3.1.2 Light Cut Column

The feed which is the bottom product from the Precut column, flashes inside the column upon exiting the feed distributor. The flashed vapours, consisting of the C12 fraction, are condensed by direct contact in the condenser section of the column. The condensed distillate is pumped by the reflux pump and cooled by generating low pressure steam in the lights cut condenser, before being returned to the top of the pumparound section. Part of the hot distillate is sent as reflux to the rectification section of the column. The net distillate is cooled by cooling water in the lights cut product cooler and sent to storage. In order to minimise fatty acid losses, the column vent is contacted by cooled distillate in a small packed section before exiting the column. The bottoms are pumped and heated by hot oil, under pressure to suppress vaporisation, in the lights cut reboiler. The net bottoms product is pumped to the Middle Cut column. Vacuum is maintained at 19 mm Hg absolute at the top of the column by the Lights Cut ejector, which discharges after the first stage of the Middle Cut ejector system.

3.1.3 Middle Cut Column

The pressurised products from the Lights Cut bottom pump flashes inside the column upon exiting the feed distributor of the middle cut column. The flashed vapours, consisting of the C14 fraction, are condensed by direct contact in the pumparound section of the column. The condensed distillate is pumped by the middle cut reflux pump and cooled by generating low pressure steam in the middle cut condenser before being returned to the top of the pumparound section. Part of the hot distillate is sent as reflux to the rectification section of the column. The net distillate is cooled by cooling water in the middle cut product cooler and sent to storage. The bottoms are pumped by the middle cut bottoms pump and heated by hot oil, under pressure to suppress vaporisation, in the Middle Cut reboiler. The net bottoms product is pumped to the Still.

Vacuum is maintained at 12 mm Hg at the top of the column by the Middle Cut ejector system.

3.1.4 Still and Residue Still Column

The feed is the bottom product from the Middle Cut column. The stream is pressurised by the middle cut bottoms pump and flashes inside the column into C16 and C18 fractions. The vapours are condensed by direct contact in a pumparound section of the column. The condensed distillate is pumped by the still reflux pump and cooled by generating low pressure steam in the still condenser, before being returned to the top of the pumparound section. Part of the hot distillate is sent as reflux to the two vapour wash trays located above the flash zone of the column. The net distillate is cooled by cooling water in the distillate cooler and sent to the storage. In order to minimise fatty acid losses, the column vent is contacted by cooled distillate in a small packed section before exiting the column. The bottom products are pumped by the still bottoms pump and heated by hot oil in pressurised reboiler to suppress vaporisation. The net bottoms product is pumped to the Residue Still. Vacuum is maintained at 9 mm Hg at the top of the column by the Still ejector discharging after the second stage the Residue Still ejector system.

The purpose of the Residue Still is to limit the losses of fatty acid in the Residue, or pitch, to 20 wt% of the residue flow. The feed to the Residue Still, pressurised by the Still bottoms pump, flashes inside the column upon exiting the feed distributor. The flashed vapours, consisting of C16 and C18, are condensed by direct contact in the pumparound section of the column. The condensed distillate is pumped by the Residue Still condenser before being returned to the top of the pumparound section. The net distillate is heated by the pitch in the Still recycle heater and recycled to the Still. The bottoms stream is pumped by the Residue pump and cooled by the Residue cooler, and

sent to storage under level control. Vacuum is maintained at 3 mm Hg at the top of the column by the Residue Still ejector system.

3.1.5 A Typical Packed Column

The column is divided into three sections, i.e. stripping, rectifying and condensing sections. Each section is essentially a set of packing. In addition, the unit also include a pumparound system. Here, a product stream is taken off from the chimney tray at the top of the rectifying section, cooled in an external cooler before being split into the distillate product and pumparound streams. The cooled pumparound product returning to the top of the column serves as a medium of condensation through direct contact with the overhead vapour. The pumparound system acts as internally recycled and heat-integrated stream to the distillation column. This system is used to support the operation of the direct contact condenser. This configuration is preferred for most oleochemical processes.

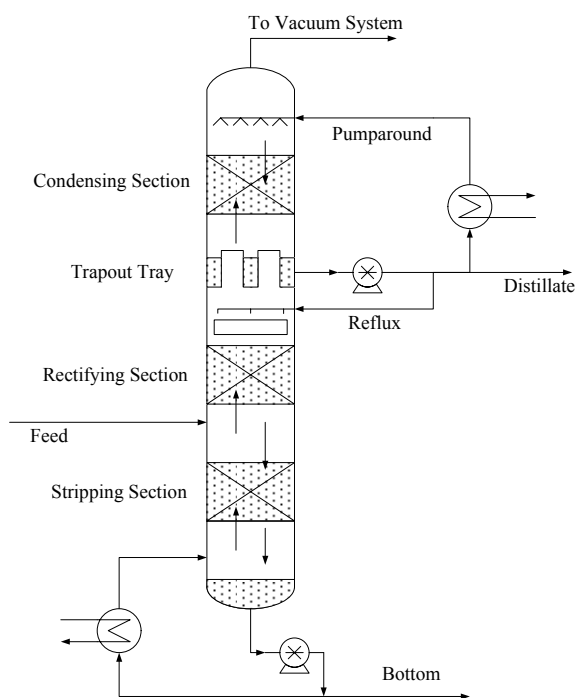


Figure 3.2 A typical packed column

Figure 3.3 shows the control system employed by the Precut Column. The column has six single-input-single-output (SISO) control loops. The control loops are labelled with number 1 to number 6. Table 3.1 depicts the controlled and manipulated variables of each of the control loops.

Table 3.1: Controlled and manipulated variables of the control loops

Control Loop	Controlled Variable	Manipulated Variable
Loop 1	Bottom Temperature	Reboiler Duty
Loop 2	Reflux Flow	Reflux Valve
Loop 3	Cooler Temperature	Cooler Duty
Loop 4	Pumparound Flow	Pumparound Flowrate
Loop 5	Bottom level	Bottom Flowrate
Loop 6	Trap-out Tray Level	Distillate Flowrate

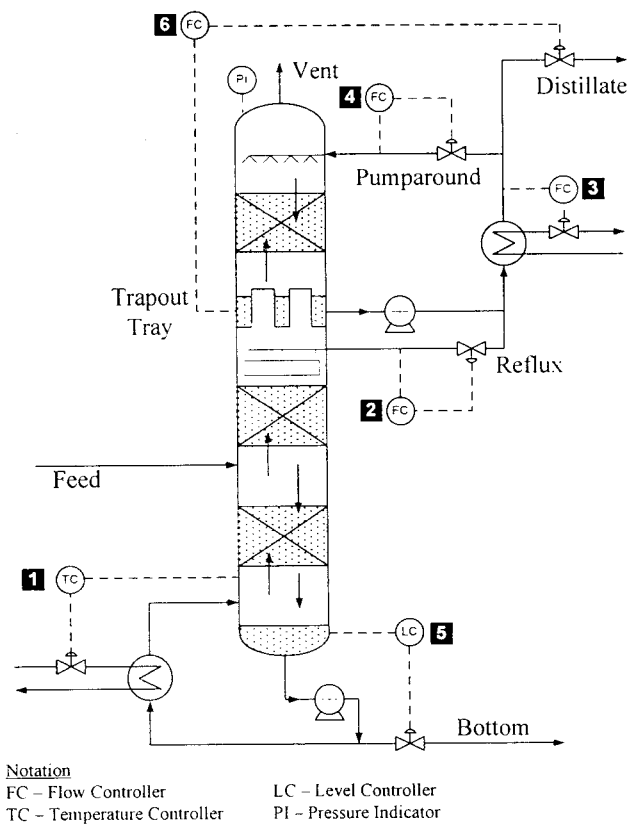


Figure 3.3 A Precut column with its control system

3.2 Simulation Methodology

The plant simulation was carried out using HYSYS.Plant, a commercial simulator having both the steady state and dynamic simulation capabilities. The flowsheet was generated based on a standard P&I Diagram of a Fatty Acid Plant and validated against typical plant data obtained from a local plant.

Typically, the simulation process takes the following stages:

- (i) Preparation Stage
 - (a) Selecting the thermodynamic model
 - (b) Defining chemical components
- (ii) Building Stage
 - (a) Adding and defining streams
 - (b) Adding and defining unit operations
 - (c) Connecting streams to unit operations
 - (d) Installing valves and controllers
- (iii) Execution Stage
 - (a) Starting integration

The programming environment within HYSYS.Plant can be divided into two main parts: the main and the column environments. The column operation in HYSYS is a unique style of Sub-Flowsheet that contains equipment such as separator and streams, and exchanges information with the Main Flowsheet Environment through the connected streams. From the Main Environment, the column flowsheet appears as a single multi-feed multi-product operation. The Main Flowsheet and Sub-Flowsheet are shown in the Figure 3.4 and Figure 3.5.

HYSYS.Plant simulator is made up of four major parts to form a rigorous modelling and simulation environment.

- A component library consisting of pure component physical properties

- Thermodynamic packages for transport and physical properties prediction
- Integrator for dynamic simulation and/or solver for steady state simulation
- Mathematical modelling of unit operation

Each of the above components is described in section below.

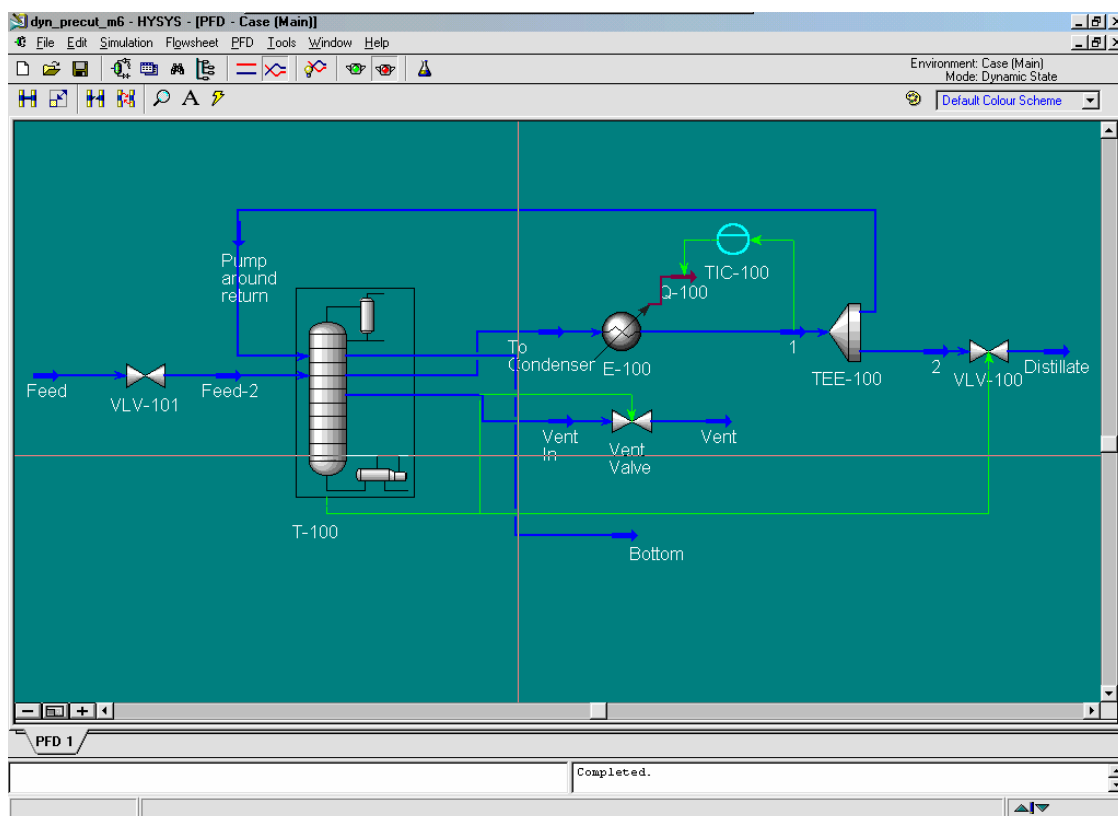


Figure 3.4 The main flowsheet in HYSYS

3.2.1 Physical Properties of the Pure Component

Feedstock to the process consists of caproic acid (C6), caprylic acid (C8), capric acid (C10), lauric acid (C12), myristic acid (C14), palmitic acid (C16), stearic acid (C18), oleic acid (C18-1), linoleic acid (C18-2) and residue. Due to the non-conventional nature of the fatty acid distillation system, the development of the required

flowsheet for plant simulation requires some modifications. For example, the long-chained fatty acid, caprylic acid (C8) is not available within the HYSYS Component Properties Library. As such, the properties of C8 have to be estimated using HYSYS Hypothetical Component Manager. In such cases, one has to provide as much data as possible from the industry so that the estimation will be realistic. The residue of the feedstock is ignored in this simulation and considered as heavy component by replacing it by oleic and linoleic acid. Moreover, only a little amount of residue is contained in the feedstock. Some of the pure component properties of the fatty acids are listed in the Table 3.2.

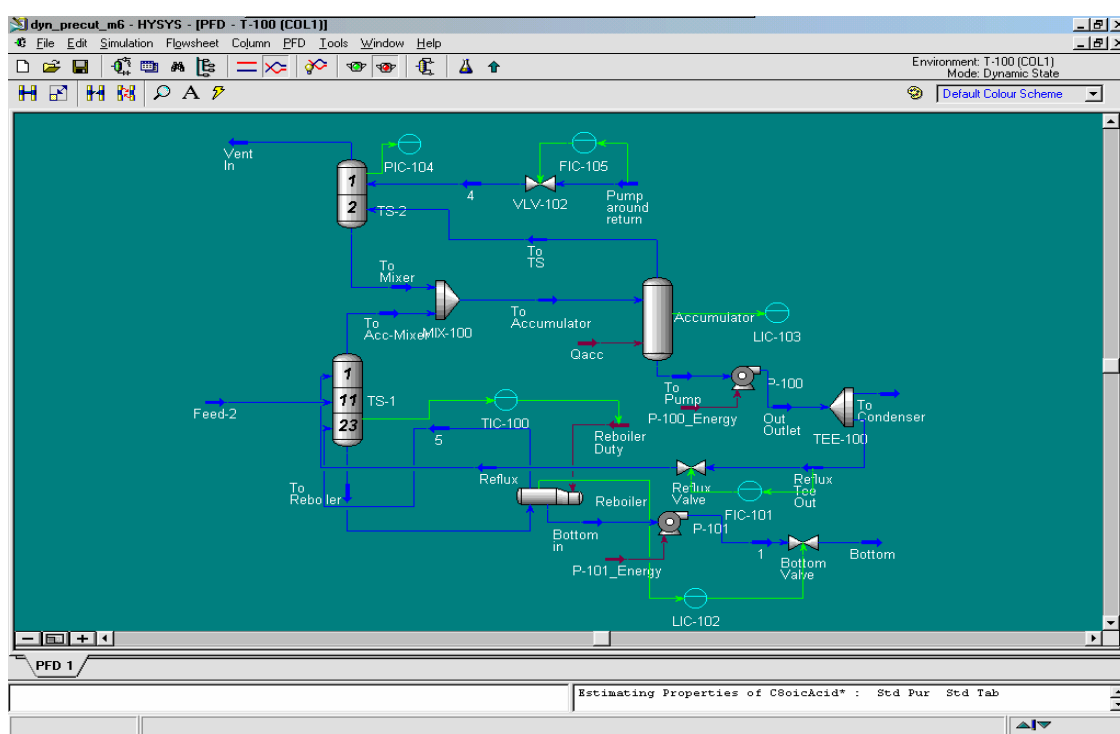


Figure 3.5 The sub-flowsheet of a Packed column

Table 3.2: Physical property of the fatty acid component

Component	MW	T _b (°C)	ρ(kg/m ³)	δ(mN/m)	η(cP)
Caproic Acid, C6	116.16	202.00	923.80	26.60	2.53
Caprylic Acid, C8	144.22	237.50	906.40	27.70	5.16
Capric Acid, C10	172.27	268.40	1017.60	28.40	5.37
Lauric Acid, C12	200.32	299.20	1009.90	28.10	7.30
Myristic Acid, C14	228.38	318.00	1002.00	28.40	7.48
Palmitic Acid, C16	256.43	353.80	1025.00	28.20	7.80
Stearic Acid, C18	284.49	370.00	1025.00	28.90	9.04
Oleic Acid, C18-1	282.47	360.00	887.00	35.20	9.41
Linoleic Acid, C18-2	280.45	355.00	902.50	-	-

Here, MW is the molecular weight, T_b is the boiling point temperature, ρ is density, δ is surface tension, and η is viscosity.

3.2.2 Thermodynamic Property

In order to define the process, the thermodynamic property models used to characterise the liquid and vapour behaviours of the Precut column must be specified. The Precut column involves separating a mixture of fatty acids which are compounds of the carboxylic acid group and exhibit polar characteristic. The usual approach of representing both vapour and liquid behaviours of this system through an equation of state (EOS) model is therefore not feasible since the EOS is limited to systems with primarily nonpolar hydrocarbons. Consequently, a dual model approach was used to fully characterise the system. In this approach, an EOS is used for predicting the vapour fugacity coefficients and an activity coefficient model is used for the liquid phase. For this simulation, the Virial EOS was used to represent the vapour behaviour and the UNIQUAC (Universal QUAsi Chemical) activity model was selected to represent the liquid behaviour.

The UNIQUAC equation proposed by Abrams and Prausnitz in year 1975 uses statistical mechanics and the quasi-chemical theory of Guggenheim to represent the liquid structure. The equation is capable of representing Liquid/Liquid Equilibria (LLE), Vapour/Liquid Equilibria (VLE) and Vapour/Liquid/Liquid Equilibria (VLLE) with accuracy comparable to the NRTL (Non-Random-Two-Liquid) equation, but without the need for a non-randomness factor. The UNIQUAC equation is significantly more detailed and sophisticated than any of the other activity models. Its main advantage is that a good representation of both VLE and LLE can be obtained for a large range of non-electrolyte mixtures using only two adjustable parameters per binary. The fitted parameters usually exhibit a smaller temperature dependence which makes them more valid for extrapolation purposes (HYSYS Documentation, 2002).

The UNIQUAC equation utilises the concept of local composition as proposed by Wilson. Since the primary concentration variable is a surface fraction as opposed to a mole fraction, it is applicable to systems containing molecules of vary different sizes and shape, such as polymer solutions. The UNIQUAC equation can be applied to a wide range of mixtures containing H₂O, alcohols, nitriles, amines, esters, ketones, aldehydes, halogenated hydrocarbon, and hydrocarbons.

HYSYS contains the following four-parameter extended form of the UNIQUAC equation. The four adjustable parameters for the UNIQUAC equation in HYSYS are the a_{ij} and a_{ji} terms (temperature independent), and the b_{ij} and b_{ji} terms (temperature dependent). The equation uses parameter values stored in HYSYS or any user supplied value for further fitting the equation to a given set of data.

$$\ln \gamma_i = \ln \left(\frac{\Phi_i}{x_i} \right) + 0.5Zq_i \ln \left(\frac{\theta_i}{\Phi_i} \right) + L_i - \left(\frac{\Phi_i}{x_i} \right) \sum_{j=1}^n L_j x_j \cdots$$

$$\cdots + q_i \left(1.0 - \ln \sum_{j=1}^n \theta_j \tau_{ji} \right) - q_i \sum_{j=1}^n \left(\frac{\theta_j \tau_{ij}}{\sum_{k=1}^n \theta_k \tau_{kj}} \right) \quad (3.1)$$

where: γ_i = activity coefficient of component i

x_i = mole fraction of component i

T = temperature (K)

n = total number of components

$$L_j = 0.5Z(r_j - q_j) - r_j + 1$$

$$\theta_i = \frac{q_i x_i}{\sum_j q_j x_j}$$

$$\tau_{ij} = \exp \left[- \frac{a_{ij} + b_{ij} T}{RT} \right]$$

$$\Phi_i = \frac{r_i x_i}{\sum_j r_j x_j}$$

Z = 10.0 co-ordination number

a_{ij} = non-temperature dependent energy parameter between components

i and j (cal/gmol)

b_{ij} = temperature dependent energy parameter between components i and

j (cal/gmol-K)

q_i = van der Waals area parameter - $Aw_i / (2.5e9)$

A_w = van der Waals area

r_i = van der Waals volume parameter - $Vw_i / (15.17)$

V_w = van der Waals volume

The Virial EOS was chosen for the Precut Column because its ability to model the vapour phase fugacities of systems displaying strong vapour phase interactions. Typically occurs in systems containing carboxylic acids, or compounds that have the tendency to form stable H₂ bonds in the vapour phase. In these cases, the fugacity coefficient shows large deviation from ideality, even at low or moderate pressures (HYSYS Documentation, 2002). If the virial coefficients need to be calculated, HYSYS contains correlations using the following pure component properties:

- critical temperature
- critical pressure
- dipole moment
- mean radius of gyration
- association parameter
- association parameter for each binary pair

This option is restricted to systems where the density is moderate, typically less than one-half the critical density. The Virial equation used is valid for the following range:

$$P \leq \frac{T \sum_{i=1}^m y_i P c_i}{2 \sum_{i=1}^m y_i T c_i} \quad (3.2)$$

To fully specify the UNIQUAC activity model, the UNIFAC estimation temperature was defined at 150 °C. This is the average operating temperature for the process. The specification is necessary for estimating interaction parameters using the UNIFAC method. Additionally, the Poynting correction factor was also enabled. The correction factor uses each component's molar volume (liquid phase) in the calculation of the overall compressibility factor.

The liquid enthalpy and entropy for the UNIQUAC activity model is based on the Cavett Correlation as shown below:

for $T_{ri} < 1$:

$$\frac{H^L - H^{ID}}{Tc_i} = \max\left(\frac{\Delta H_i^{oL}(sb)}{Tc_i}, \frac{\Delta H_i^{oL}(sb)}{Tc_i}\right) \quad (3.3)$$

for $T_{ri} \geq 1$:

$$\frac{H^L - H^{ID}}{Tc_i} = \max\left(\frac{\Delta H_i^{oL}(sb)}{Tc_i}, \frac{\Delta H_i^{oL}(sp)}{Tc_i}\right) \quad (3.4)$$

where:

$$\frac{\Delta H_i^{oL}(sb)}{Tc_i} = a_1 + a_2(1 - Tr_i)^{1-a_3(Tr_i-0.1)}$$

$$\frac{\Delta H_i^{oL}(sp)}{Tc_i} = \max(0, b_1 + b_2 Tr_i^2 + b_3 Tr_i^3 + b_4 Tr_i^4 + b_5 Tr_i^5)$$

where a_1 , a_2 , and a_3 are functions of the Cavett parameter, fitted to match one known heat of vapourisation. The Gas enthalpies and entropies are dependent on the model chosen to represent the vapour phase behaviour. For the Virial equation:

$$\frac{H - H^{ID}}{RT} = -\frac{T}{V - B} \frac{dB}{dt} + (Z - 1) \quad (3.5)$$

$$\frac{S - S_o^{ID}}{R} = -\frac{RT}{V - B} \frac{dB}{dt} - R \ln \frac{V}{V - B} + R \ln \frac{V}{V^o} \quad (3.6)$$

where: B = second virial coefficient of the mixture

A summary of the thermodynamic models used in the simulation of the Precut Column is given in the Table 3.3.

Table 3.3: Thermodynamic models for the Precut column

Phase	Property Method	VLE Calculation	Enthalpy/Entropy Calculation
Liquid	UNIQUAC	UNIQUAC	Cavett
Vapour	Virial	Virial	Virial

3.2.3 Integration Algorithm

A dynamic model is represented by a set of ordinary differential equations (ODEs) in HYSYS.Plant. In order to solve the model, an implicit Euler method is used to integrate the ODEs. The Implicit Euler method is simply an approximation of Y_{n+1} using rectangular integration. Graphically, a line of slope zero and length h (the step size) is extended from t_n to t_{n+1} on a $f(Y)$ vs. time plot. The area under the curve is approximated by a rectangle of length h and height $f_{n+1}(Y_{n+1})$:

$$Y_{n+1} = Y_n + hf_{n+1}(Y_{n+1}) \quad (3.7)$$

The Implicit Euler method handles stiff systems well. This is an implicit method because information is required at time t_{n+1} . Integration parameters such as the integration time step can be specified in the Integrator view from the Simulation menu in HYSYS. The integration time step can be adjusted to increase the speed or stability of the system (HYSYS Documentation, 2002).

In HYSYS, dynamic calculations are performed at three different frequencies:

- Volume (pressure-flow)
- Energy
- Composition

These relations are not solved simultaneously at every time step. This would be computationally expensive. The compromise is to solve the balances at different time step frequencies. The default solution frequencies, which are multiples of the integration time step, are one, two, and then for the pressure flow equations, energy, and composition balances.

Meaning pressure flow equations are solved at every time step while composition balances are solved at every 10th time step. Since composition tends to change much more gradually than the pressure, flow, or energy in a system, the equations associated with composition can be solved less frequently. An approximate flash is used for each pressure flow integration time step. A rigorous flash is performed at every composition integration time step.

3.2.4 Mathematical Modelling of the Distillation Operation

The schematic diagram of the dynamic model of a single distillation tray is shown in the Figure 3.6. Liquid enters through the downcomer of the tray from the above (L_{n+1}). Vapour enters the tray from the tray below (V_{n-1}). The vapour and liquid are completely mixed on the tray and the vapour (V_n) that is in equilibrium with the tray liquid compositions leaves through to the tray above. The liquid (L_n) flows over the weir into the downcomer and to the tray below. The tray may also contain a feed (F) depending on its location in the column. In dynamic simulation, the pressure drop across a stage is determined by summing the static head and the frictional losses.

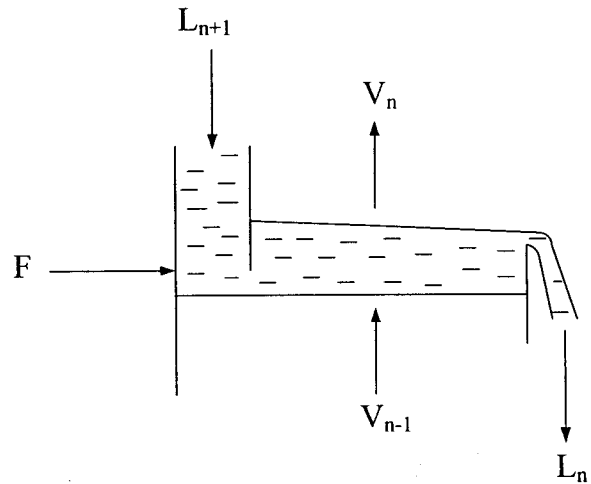


Figure 3.6 Distillation tray model

The dynamic model of a single tray contains N_{C-1} differential material balances, where N_C is the number of components in the system as equation 3.8.

$$\frac{dx_i M_n}{dt} = z_{i,f} F + x_{i,n+1} L_{n+1} + y_{i,n-1} V_{n-1} - x_{i,n} L_n - y_{i,n} V_n \quad (3.8)$$

where M_n denotes the material on the n th tray, x_i and y_i are the liquid and vapour mole fraction respectively, and z_i is the feed composition (mole fraction).

The overall material balance for the n th tray:

$$\frac{dM_n}{dt} = F + L_{n+1} + V_{n-1} - L_n - V_n \quad (3.9)$$

The overall energy balance for the n th tray:

$$\frac{dh_n^L M_n}{dt} = h_f F + h_{n+1}^L L_{n+1} + h_{n-1}^V V_{n-1} - h_n^L L_n - h_n^V V_n \quad (3.10)$$

Here, h is the specific enthalpy (J/mol) and assuming $\frac{dh_n^L}{dt} \approx 0$ (the change in specific enthalpy is very small compared to the total tray enthalpy).

The hydraulics in the tray is accounted for using the Francis Weir equation:

$$L_n = C\rho \cdot l_w h^{1.5} \quad (3.11)$$

where L_n is the liquid flowrate leaving tray n , C is the unit conversion constant, ρ is the density of liquid on tray, l_w is the weir length and h denotes the height of liquid above the weir.

The vapour flowrate leaving a tray was determined by the resistance equation:

$$V_n = k\sqrt{\Delta P_{friction}} \quad (3.12)$$

where V_n is the vapour flowrate leaving tray n , k is the conductance, which is a constant representing the reciprocal of resistance to flow and is proportional to the square of the column diameter and $\Delta P_{friction}$ denotes the dry hole pressure drop, which is associated with vapour flowing through the tray section.

The solution of the dynamic stage model is developed using the following procedure (HYSYS Documentation, 2002):

1. Liquid compositions for each tray, $x_{i,n}$, throughout the column are initialised from the steady-state solution, which is based on a product specification in the distillate.

2. The vapour compositions for each tray, $y_{i,n}$, are calculated using the UNIQUAC and Virial models and the temperature for each tray, T_n , is calculated using a bubble point iterative flash calculation.
3. The liquid and vapour enthalpies for each tray, h_n^L and h_n^V are calculated from the Cavett equation and the Virial equation respectively.
4. The liquid flowrate leaving the tray, L_n , is calculated using Francis Weir equation.
5. The vapour leaving each tray, V_n , is calculated from the resistance equation (Equation 3.12).
6. The derivatives for the component and total mass balance are calculated.
7. The equations are then integrated using the Implicit Euler Method.

3.2.5 Modelling and Simulation Assumption

As stated earlier, the residue of the feedstock was ignored because of the difficulties in obtaining the physical properties directly from the industry. Some of the modification of the plant model was done to speed out the simulation. The modifications are listed below:

1. The packing in the section where the flow is leaving through vent was ignored.
2. HYSYS.Plant does not support packed column and as such, the Height-Equivalent Theoretical Plate (HETP) calculation was used. The HETP value was computed from the Fatty Acid Plant based on their design information, which was obtained directly from the plant.
3. The model of a column with internal direct contact condenser is not available within the HYSYS model library, therefore modifications of the standard column template have been carried out to produce an equivalent configuration.

4. The characteristics of tray sections for the columns were calculated within the HYSYS Tray Sizing utility. It is important to note that in simulating the distillation column for dynamic behaviour, some parameters that are not required for steady-state simulations must also be specified. These include, the tray diameter, weir length, weir height, and the tray spacing that are crucial for accurate dynamic simulation.
5. For better dynamic simulation results, a basic control structure is required. The control structure is the same as in the actual plant but the modification of controller tuning was performed.
6. The residue component in feedstock was ignored.

3.2.6 Degree of Freedom Analysis

There are two type of degree of freedom. The first one is dynamic degrees of freedom, N_m (here m denotes manipulated). N_m is usually easily obtained by process insight as the number of independent variables that can be manipulated by external means. In general, this is the number of adjustable valves plus other adjustable electrical and mechanical devices. Next, steady-state degrees of freedom, N_{ss} , which is the number of variables needed to be specified in order for a simulation to converge. To obtain the number of steady-state degrees of freedom we need to subtract from N_m .

- N_{om} is the number of manipulated (input) variables with no steady-state effect (or more generally, with no effect on the operation cost). Typically, these are extra manipulated variables used to improve the dynamic response, e.g. an extra bypass on a heat exchanger.
- N_{oy} is then number of output variables that need to be controlled, but which has no steady-state effect (or more generally, no effect on the operation cost). Typically, these are liquid levels in holdup tanks.

As a result, Equation 3.13 is obtained.

$$N_{ss} = N_m - (N_{om} + N_{oy}) \quad (3.13)$$

In any process simulation work, it is essential that degrees of freedom analysis be carried out to determine the number of variables to be specified. Based on the analysis performed, 4 degree of freedoms was needed for Precut column. In this case, the pressure of the distillate product, bottom product, overhead vent and feed streams should be specified.

A close match between the simulation and the industrial data has been obtained despite the unavailability of an exact column template.

3.3 Comparison of Simulation Results

In order to validate either this simulation data can represent the actual process, the comparison between simulation and industrial data was carried out. Table 3.4 shows the comparison between simulation and plant data.

Table 3.4: Simulation and plant data comparison for Precut column

Fatty Acid	Feed Flow (kg/h)		Top Product Flow (kg/h)		Bottom Product Flow (kg/h)	
	Plant	Simulation	Plant	Simulation	Plant	Simulation
C6	11.0	11.0	11.0	7.7		
C8	306.0	306.0	306.0	306.4		
C10	309.0	309.0	302.0	304.2	7.0	10.8
C12	4343.0	4343.0	5.0	3.7	4338.0	4339.7
C14	1489.0	1489.0			1489.0	1485.4
C16	724.0	724.0			724.0	729.5
C18	172.0	172.0			172.0	176.6
C18:1	1431.0	1431.0			1431.0	1438.3
C18:2	239.0	239.0			239.0	240.9
			Plant		Simulation	
Feed Temperature (°C)			210.0		210.0	
Bottom Stage Temperature (°C)			240 - 250		243.5	
Pump-around Flow (kg/h)			8500 - 9000		8601.5	
Reflux Flow (kg/h)			3092.0		3092.8	

The results obtained suggested that the model adequately represent the selected distillation column. Minor discrepancies detected in the simulation results were due to:

- i) The process flow diagram had been simplified which combination of other processes such as pre-treatment, light-cut, middle-cut, still-cut and residue-cut were not considered.
- ii) The recycle stream in the still-cut had been ignored since still-cut column was not considered.
- iii) The fractionation column that is actually a packed column was sized using HETP concept.

However, these errors were not significant and we can therefore conclude that HYSYS.Plant model constructed here is a reasonably accurate match of actual process unit in the industry.

3.4 Summary

The Fatty Acid Plant (FAP) studied to develop and test different types of ANN models have been described in detail in this chapter. Method of modelling and simulation of this process using HYSYS.Plant have also been explained. Some modifications have been made to satisfy the modelling requirement so that the output obtained from the simulation successfully mimics the actual process. The result shows a close match between simulation and the industrial data.

In Chapter 4, development of process estimation using ANN are presented. There are two types of ANN models and three training algorithms investigated. The main focus of the next chapter is the selection of architecture, training algorithm, and network topology of ANN for estimating the sensor signals and generating consistent residuals from normal data.

CHAPTER IV

PROCESS ESTIMATION FOR FAULT DETECTION PURPOSES

4.1 Introduction

As discussed earlier, a model-based technique as proposed by Ahmad and Leong (2001) is further investigated in this study. The proposed fault detection and diagnosis scheme for the neural network implemented for Precut Column in a Fatty Acid Plant (FAP) is shown in the Figure 4.1 below. The scheme is hierarchical in structure and involves two types of model i.e. an estimator and a classifier. Both models are based on neural network. The first is the process estimator that estimates the normal' or fault-free process behaviour. The estimated output is then compared to the actual measurement. The differences between the two termed as residuals are sent to the second model that serves as a classifier. The role of the classifier is to classify the residuals into a number of distinguishable patterns corresponding to different faulty situations. On occasion of fault, the residuals will have some finite values. A value of zero is expected when the plant is in normal' operation condition.

4.2 Design of Process Estimator

The process estimator is used to predict the fault free' operating condition of the process. The model is constructed using a class of recurrent neural network known as

Elman network. Similar to other data-based modelling, the development of a neural network model follows the standard procedure of system identification.

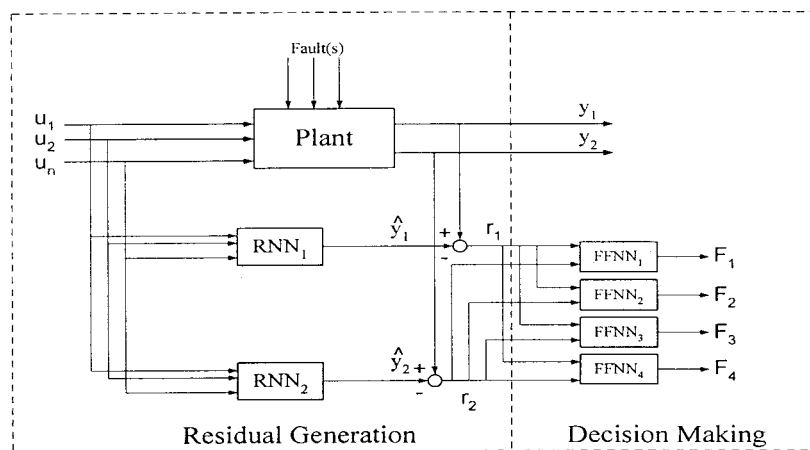


Figure 4.1 The neural network-based fault detection scheme

In general, the system identification of a process involves the following procedures. Every step is important to make sure a proper process model is developed.

- (i) Selection of output and input variables
- (ii) Selecting a process excitation signal
- (iii) Selecting the model structure
- (iv) Parameter estimation
- (v) Validating the resulting model

4.3 Selection of Process Estimator's Output and Input Variables

Selection of process estimator's output and input variables must be carried out carefully in order to capture interaction between those variables so that the resulting model can be applied to represent the process successfully. Input variables to be selected must be variables that have significant effect on the process output.

4.3.1 Output Variables

In this study, the fault detection scheme developed is used to detect the bottom temperature and top pressure sensor faults as well as leakage in the Precut column. Thus, bottom temperature and top pressure measurements will be used as estimator's outputs and also C8 flowrate to monitor leakage. Bottom temperature and top pressure measurements are crucial to the column as they give direct effect to the process yield, efficiency and also plant safety.

In this case, the bottom column temperature is selected as the indicator as it showed more significant response to product purity. The bottom column temperature is maintained at 237.8 °C by controlling temperature of the reboiler. Top column pressure is selected to monitor pressure of the column. Vacuum is maintained at 7.4 kPa at the top of the column by the Precut column ejector system. To monitor leakage in the column, C8 flowrate is monitored and must be not less than 100 kg/h to maintain the operating condition.

4.3.2 Input Variables

In order to develop a good representative model for bottom column temperature, top column pressure and C8 flowrate in the product, input variables for neural network must be carefully selected. Selection of the input is made based on the effect of the input variables on the selected outputs. In order to simplify the model structure of neural network, only input variables that have significant impact to the process outputs are selected, others are neglected. For the Precut column, manipulated variable in the closed control loops are neglected from input variables selection. Table 4.1 shows the selected inputs for process predictor.

Table 4.1: Neural network inputs

Input	Variable
1	Reflux flowrate
2	Condenser flowrate
3	Pumparound return flowrate
4	Top stage temperature
5	Distillate flowrate
6	Bottom flowrate
7	Feed flowrate

Additional inputs added are the delayed measurements for each of the selected inputs. Addition of one delayed term or two delayed terms may be used to provide dynamic information.

4.4 Design of Excitation Signal

Similar to a typical identification procedure, a significantly 'excited' data that represent the system is required to fit the required neural network model. To fulfil this requirement, process inputs for the Precut column are perturbed to generate a periodic process pattern comprising dynamic process information. This is done within the HYSYS simulator environment described in the previous chapter.

Process inputs for neural network models include sensors reading that are inconsistent in term of their magnitudes. These input data are scaled within a consistent range (e.g. 0 to 1) before being introduced to the input layer of the network to ensure that each data is given fair contribution in determining the network output. The data scaling method employed in this research is

$$X_s = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (4.1)$$

Here, X_s is scaled input and X is the actual input before scaling whereas X_{\min} and X_{\max} are the maximum and minimum values of the inputs respectively. The maximum and minimum values of the input are selected based on data set available for training.

In order to obtain a representative excitation signal, the feeds are perturbed within $\pm 10\%$ of the feed base case steady state value. A mixture of short and long input step change intervals will provide sufficient dynamic information for training purpose.

4.4.1 Excitation Signal for Training

The results of a sensitivity analysis showed that step input for feed with less than 10% from its steady state value affected the product purity. The purity of product C8-C10 must have maximum 1% of C12. From this finding, the training data is designed with perturbation of feed with $\pm 9\%$ of the feed base case steady state value. The training data sets are shown in the Figure 4.2. The sampling period used is 10 second and the data sets consists of 2000 sampling points for a duration of 333 minutes.

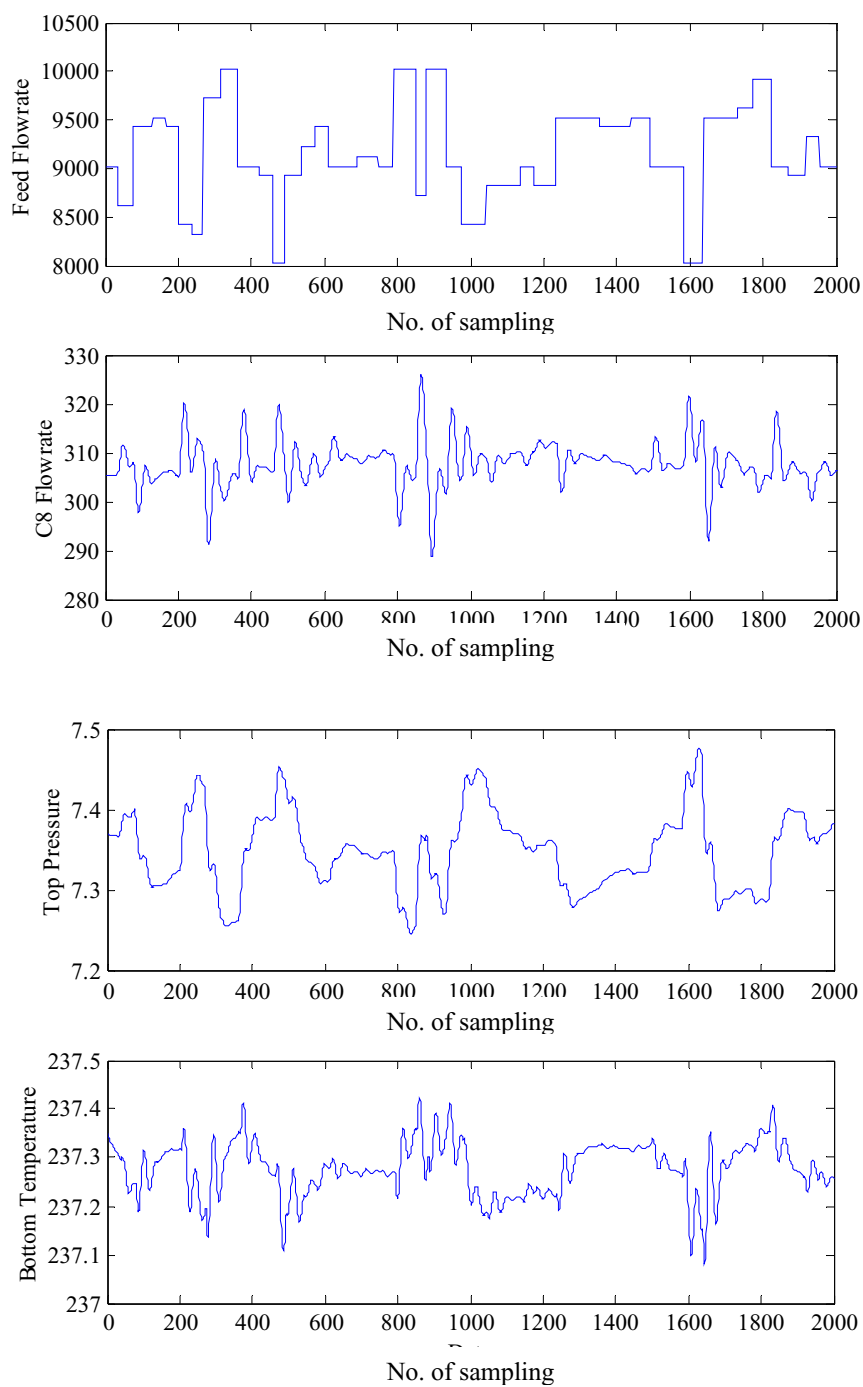


Figure 4.2 Training data set

4.4.2 Excitation Signal for Validation

The trained neural networks model is validated with a different set of data in order to know the performance of the model with the unseen/unknown data. The process feed is perturbed randomly with different seeds within 5% from its steady state value. A total of 2000 data is collected for 333 minutes with sampling period of 10 sec. The validation data sets are shown in the Figure 4.3.

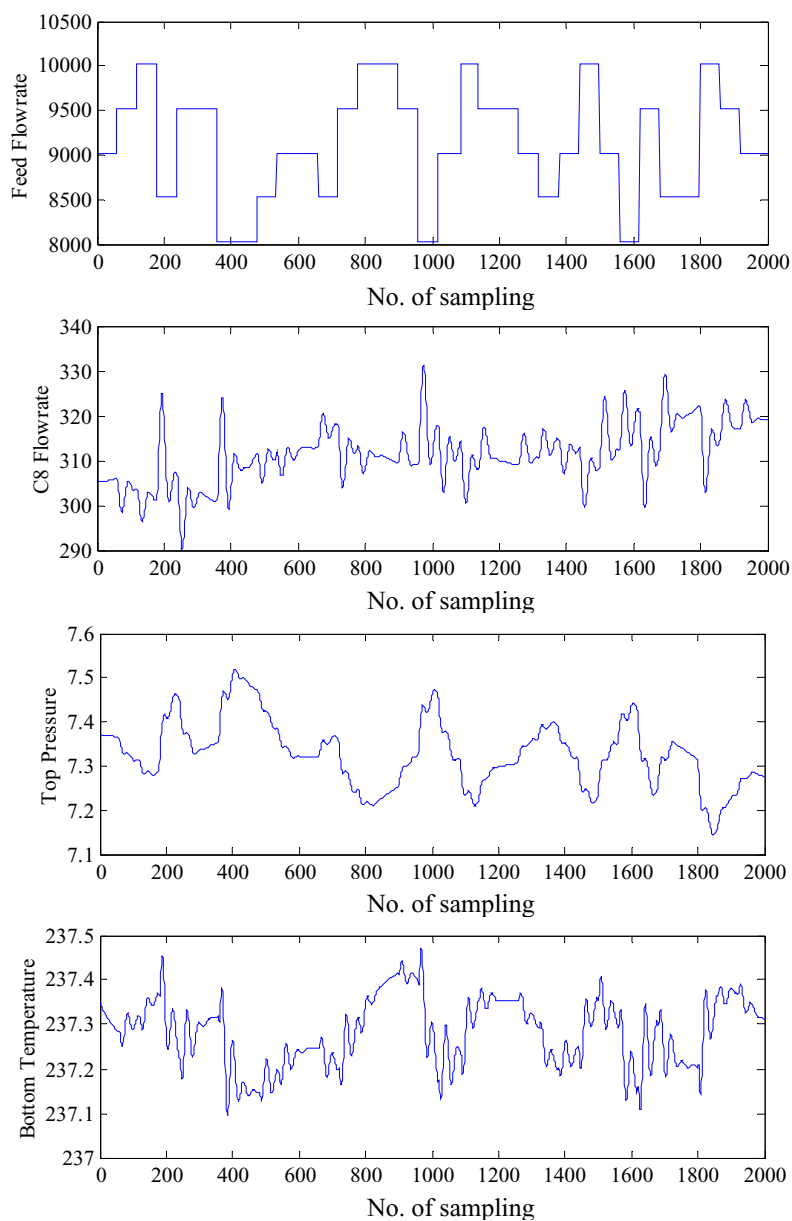


Figure 4.3 Cross-validation data set

4.4.3 Excitation Signal for Testing

The trained model is also tested with another set of data for comparison purpose. The process feed is perturbed randomly within $\pm 7\%$ of the feed base case steady state value. The testing data sets are shown in the Figure 4.4 below. The data is also collected for duration of 333 minutes with 10 sec sampling period and consists 2000 sampling points.

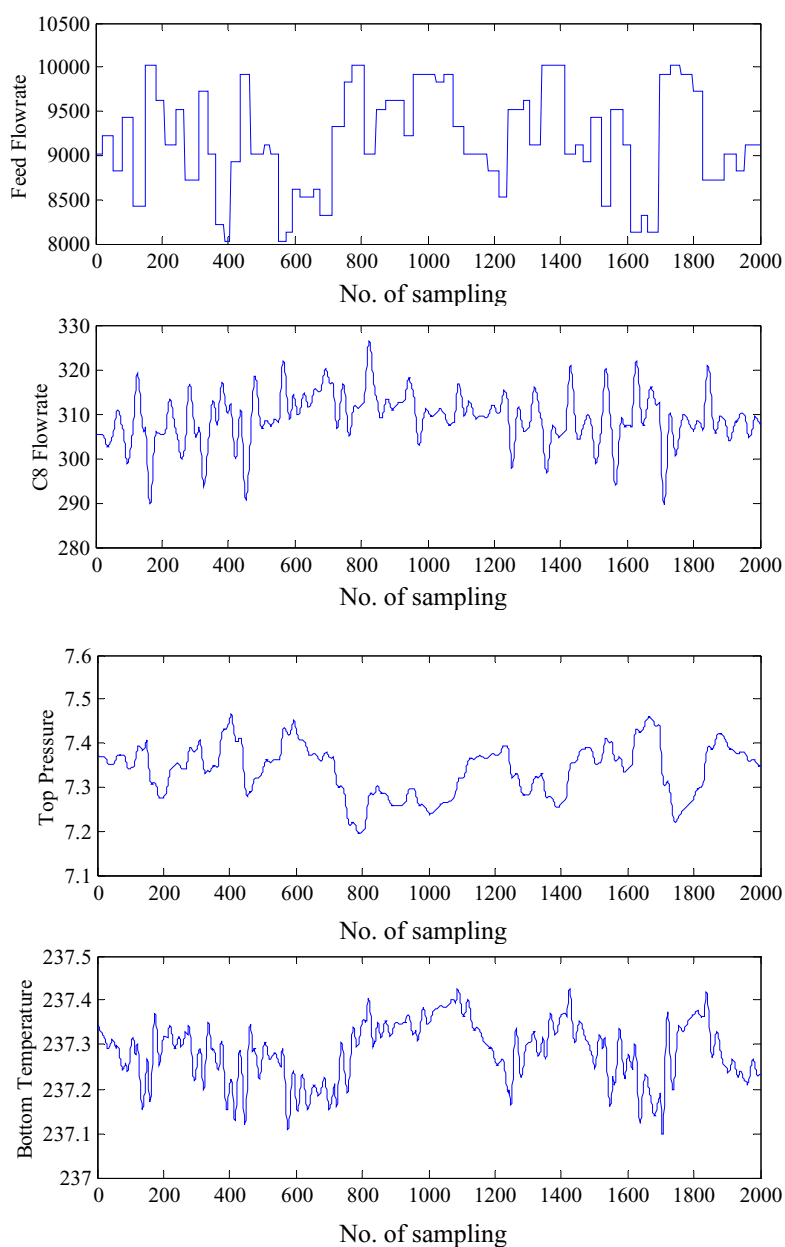


Figure 4.4 Testing data set

4.5 Selection of Neural Network Structure, Training and Cross-Validation

There are two types of structure for artificial neural network that are used in this study. The structure of artificial neural network process estimator can be either a set of Multi-Input Single-Output (MISO) networks or single Multi-Input Multi-Output (MIMO) network. The selection criteria depend on how efficient the two structures in predicting the process. Therefore, the performance of the both network structures are tested and compared.

4.5.1 Multi-Input and Single-Output (MISO) Model

The process estimator developed in this research is an Elman network. Elman network is constructed to estimate the normal or fault-free process condition. Thus, actual process outputs cannot be used as inputs because they are affected directly by process faults. The network should be independent of the actual process outputs to enable the generation of residual as a measure of actual process departure from normal operating condition.

Three MISO networks are required to model every single selected process outputs. The schematic diagram of MISO network is shown in the Figure 4.5.

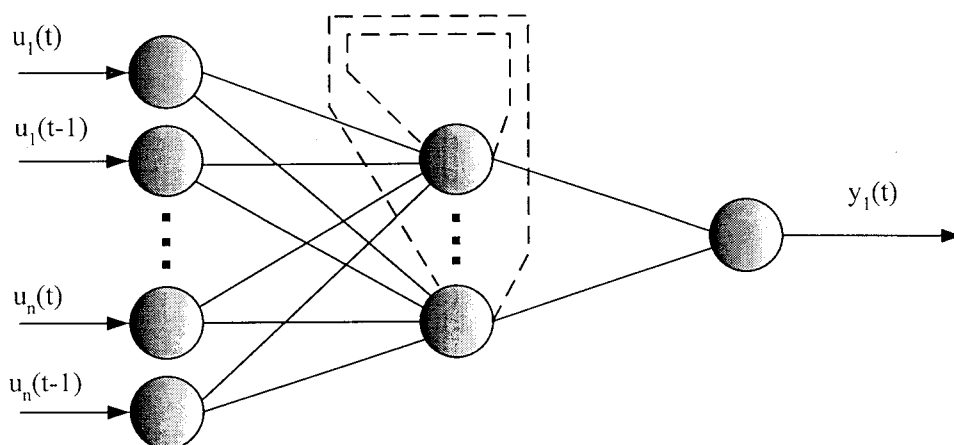


Figure 4.5 Multi-Input Single-Output Elman network

Here, $y_1(t)$ is referred to as network output which is either the Top Column Pressure, Bottom Column Temperature or C8 Flowrate, while $u_n(t)$ and $u_n(t-1)$ are referred to as process inputs without and with time delay, respectively.

The network parameters of the Elman MISO were initialised by training the network with three different backpropagation algorithms, they are Gradient Descent with Momentum Backpropagation (*traingdm*), Gradient Descent with Momentum and Adaptive Learning Backpropagation (*traingdx*), and Levenberg-Marquardt Backpropagation (*trainlm*): The training errors converged at the rate of 1.0×10^{-8} . Network's topologies having 5 to 25 hidden nodes were trained accordingly and the results of the training are shown in the Figure 4.6 below and in the Table C1 in the Appendix C1.

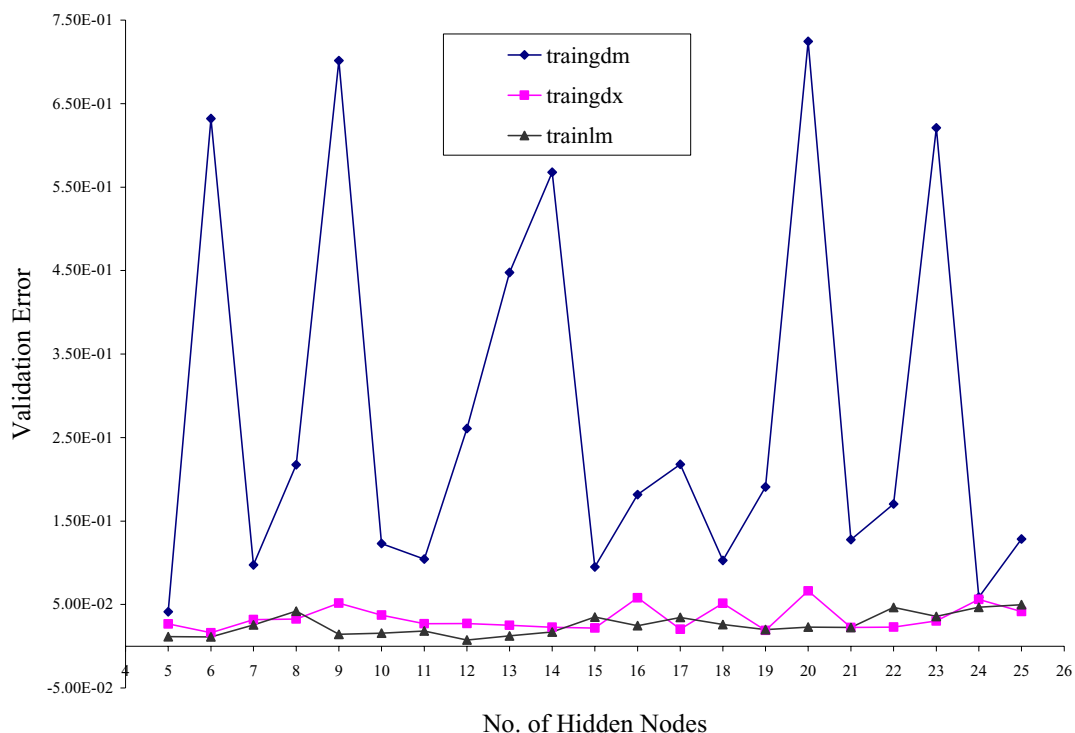


Figure 4.6 Training result of MISO network for Top Stage Pressure (without delay) for different training algorithm

Optimum network is selected among the networks tested based on the smallest cross-validation errors produced. Thus, from the Figure 4.6 above, the optimum MISO network for top stage pressure is selected from topologies with 24, 6 and 12 hidden nodes for *traingdm*, *traingdx* and *trainlm*, respectively. The selection of the network is based on the smallest cross-validation error among the three algorithms. From the three algorithms, *trainlm* gives the smallest cross-validation error at 12 hidden nodes. The performances of these algorithms are depicted in the Figure 4.7. From the graph, it is obvious that the MISO network with *trainlm* algorithm shows a better performance. The predicted output from the network with *trainlm* algorithm followed the pattern of actual output with smaller error compared with other two algorithms.

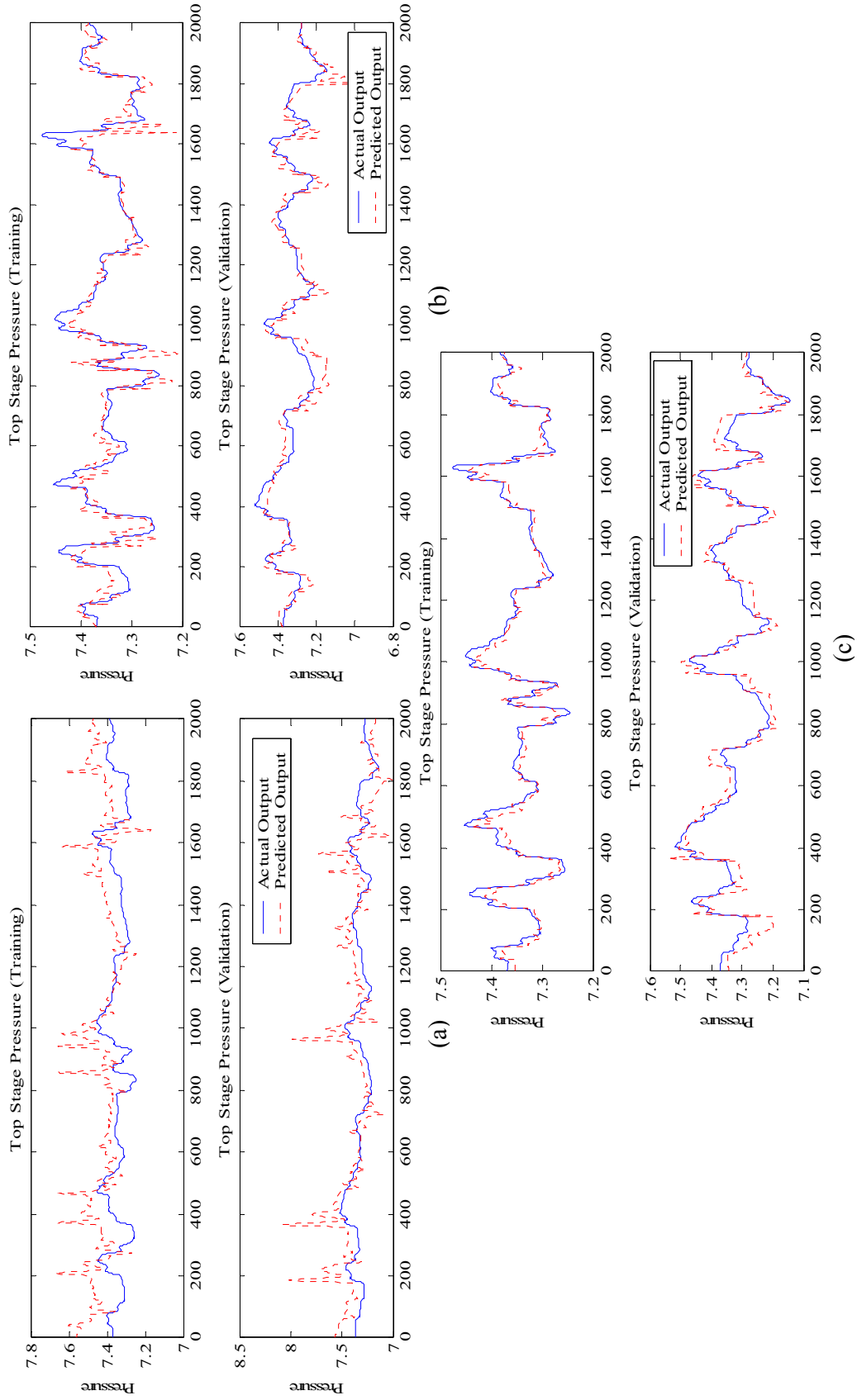


Figure 4.7 Performance of MISO network for Top Stage Pressure with (a) *traingdm* (topology 7/24/1 nodes), (b) *traingdx* (topology 7/6/1 nodes) and (c) *trainlm* (topology 7/12/1 nodes)

The effect of adding time delay was also investigated. The optimum networks are selected based on the smallest cross-validation error. Since *trainlm* algorithm shown better performance than others (refer Figure 4.6), study of the effect of adding time delay will be performed on network with *trainlm* algorithm only. The result of the training is shown in the Figure 4.8 and in the Table C2 in the Appendix C2.

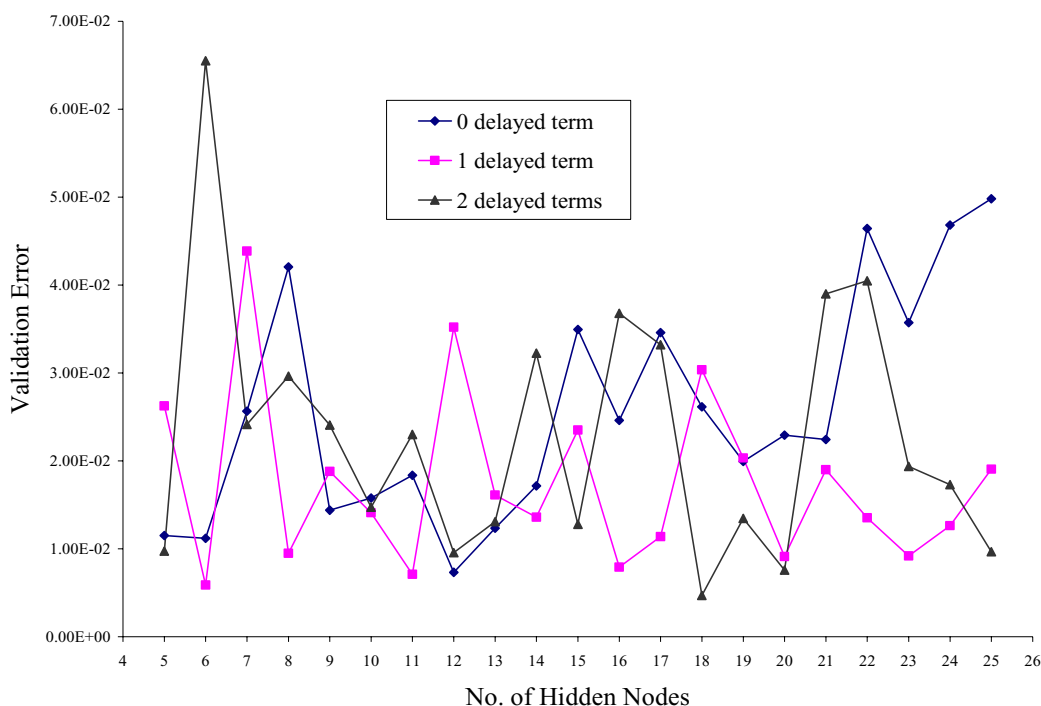


Figure 4.8 Training result of MISO network for Top Stage Pressure with *trainlm*

The optimum MISO network for top stage pressure is selected from topologies with 12, 6 and 18 hidden nodes for network with no delay, with 1 delay and with 2 delays, respectively. From the results, it is obvious that by adding time delay to the networks' inputs, performance of the networks will improve. The MISO network with 2 delayed terms shows better performance compared to the network without delay and with 1 delayed term since it gives the smallest cross-validation error. The performances of these networks are shown in the Figure 4.9. From the graph, it is obvious that the MISO network with 2 delayed terms shows a better performance. The predicted output from the network with *trainlm* algorithm followed the pattern of actual output with smaller error.

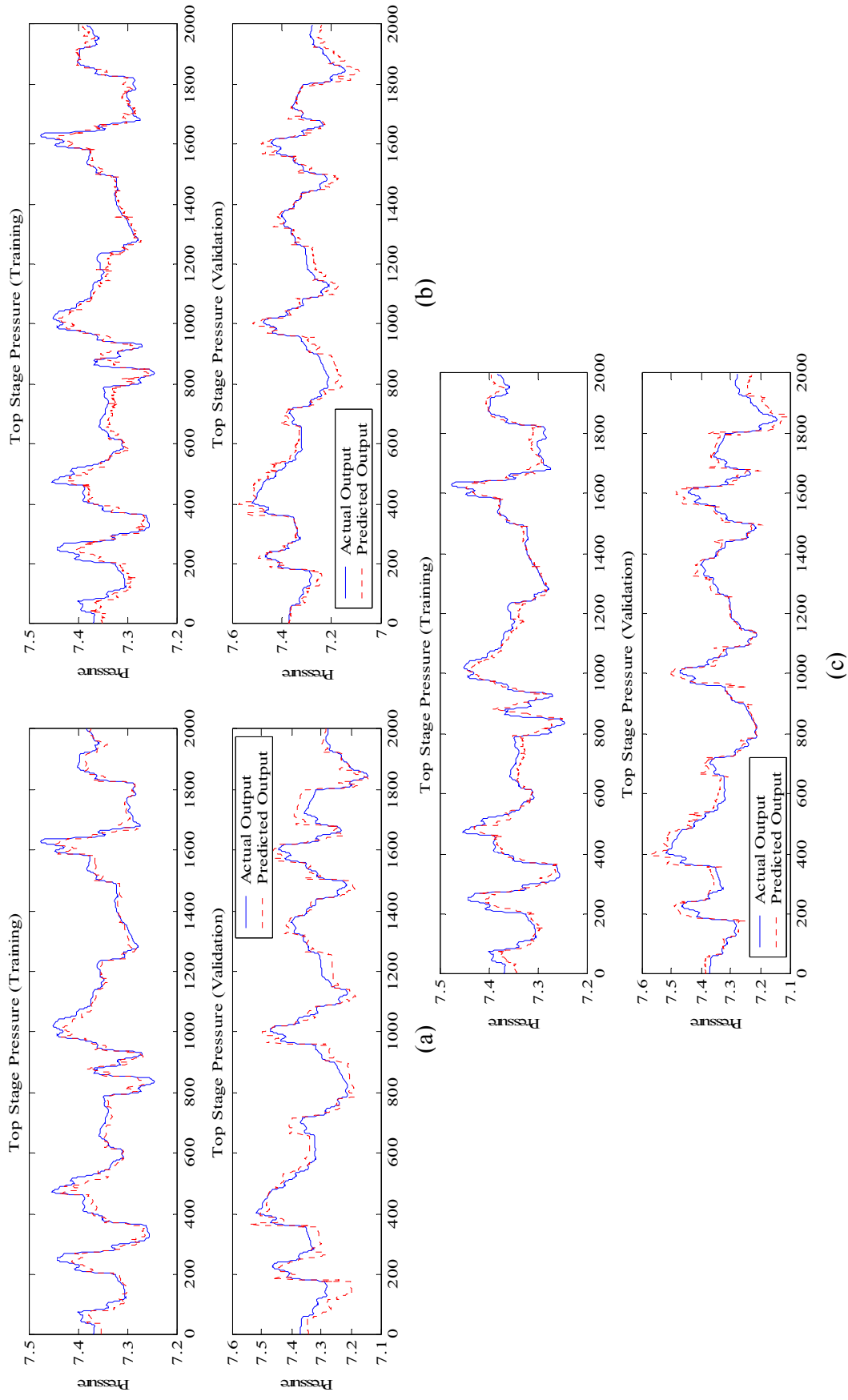


Figure 4.9 Performance of MISO network for Top Stage Pressure with *trainlm*, (a) without delayed term (topology 7/12/1 nodes), (b) with 1 delayed term (topology 14/6/1 nodes) and (c) with 2 delayed terms (topology 21/18/1 nodes)

MISO networks for bottom stage temperature and C8 flowrate are also trained and cross-validated with *trainlm* algorithm and the results are shown in the Figure 4.10 and in the Table C3 in the appendix C3. The networks were used 2 delayed terms and showed an excellent prediction. Thus, from the results, the optimum MISO networks for top stage pressure, bottom stage temperature and C12 composition are selected from topologies with 18, 7 and 11 hidden nodes respectively.

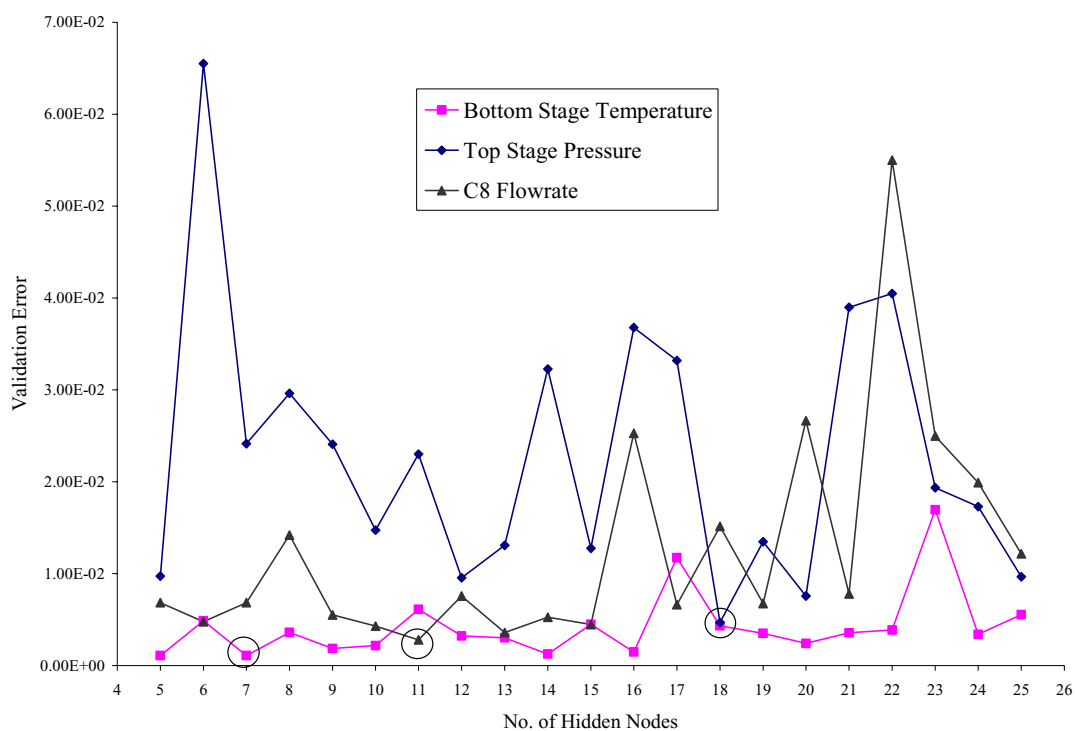


Figure 4.10 Training result of MISO networks with 2 delayed terms and *trainlm*

The graphical presentations of the optimum MISO networks prediction are depicted in the Figure 4.11. The results show that Elman network with 2 delayed terms that using *trainlm* algorithm produced excellent predictions.

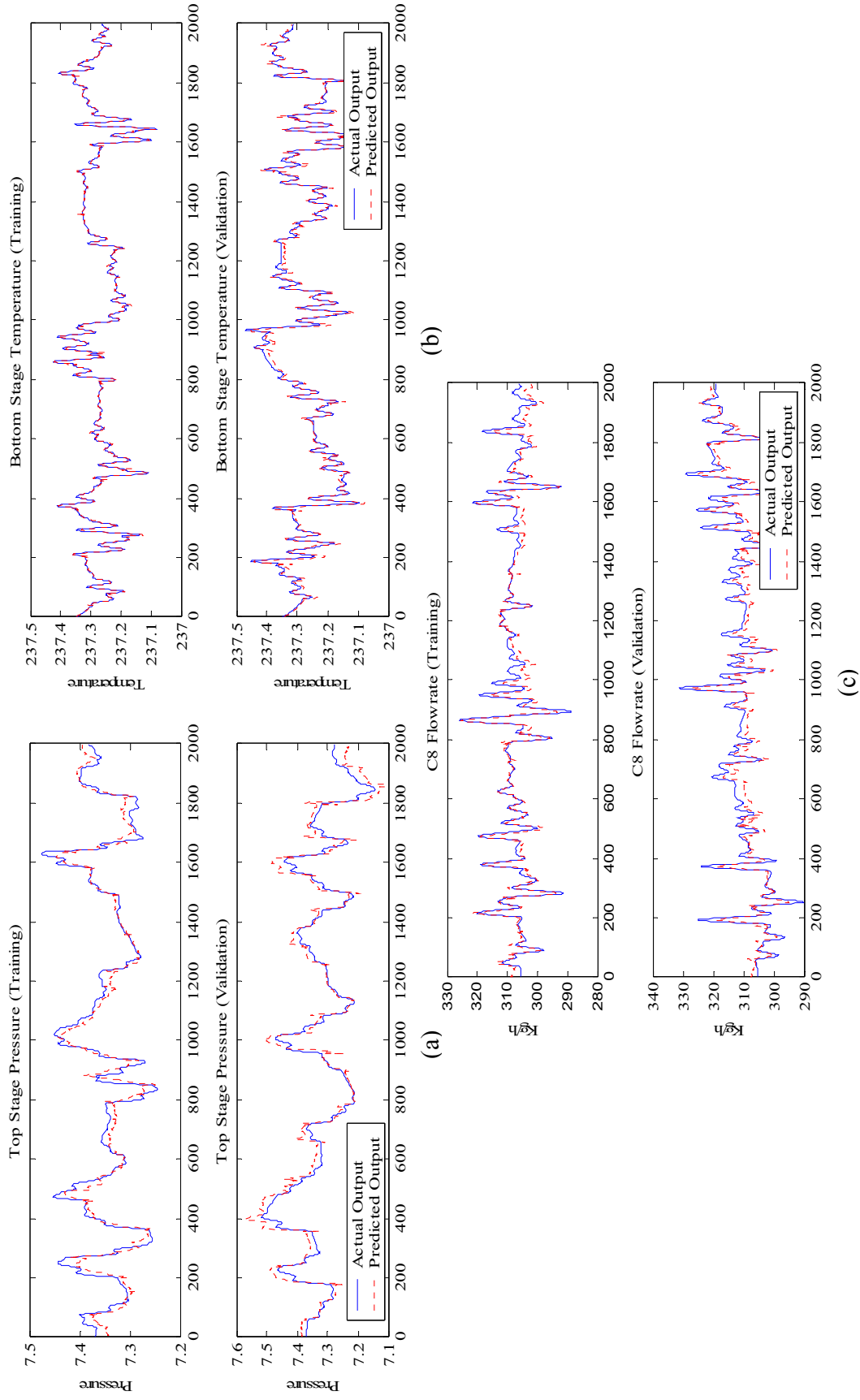


Figure 4.11 Performance of MISO network with *trainIm* for (a) Top Stage Pressure (topology 21/18/1 nodes), (b) Bottom Stage Temperature (topology 21/18/1 nodes) and (c) C8 Flowrate (topology 21/7/1 nodes) and (d) Bottom Stage Temperature (topology 21/7/1 nodes)

4.5.2 Multi-Input and Multi-Output (MIMO) Model

The MIMO network structure for the process estimator is shown in the Figure 4.12. Here, $y_1(t)$, $y_2(t)$ and $y_3(t)$ are referred as network outputs, in this case Top Column Pressure, Bottom Column Temperature and C8 Flowrate, respectively. While $u_n(t)$ and $u_n(t-1)$ are referred as process inputs without and with time delay, respectively. The performance of the network will be tested without and with time delay.

Similar with MISO networks training, the MIMO network parameters are initialised by training network using Levenberg-Marquardt method (*trainlm*). Network topologies with 5 hidden nodes to 25 hidden nodes were tested and the results of the training are shown in the Figure 4.13.

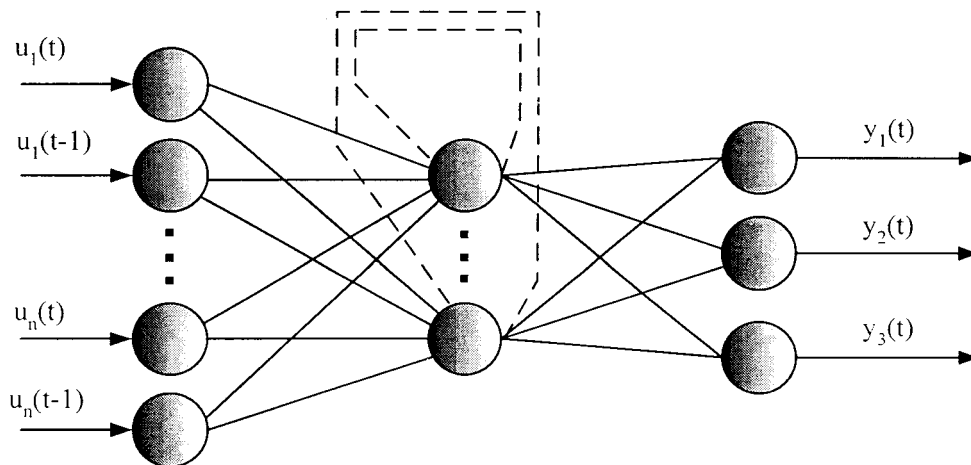


Figure 4.12 Network structure for MIMO model

The optimum topologies are selected based on the smallest cross-validation errors ever produced among the networks tested. The validation error here is the average validation error from every output in the network. Based on the results obtained, the smallest cross-validation error is produced by MIMO network with 9, 8 and 6 hidden nodes without delay, with 1 delayed term and with 2 delayed terms, respectively.

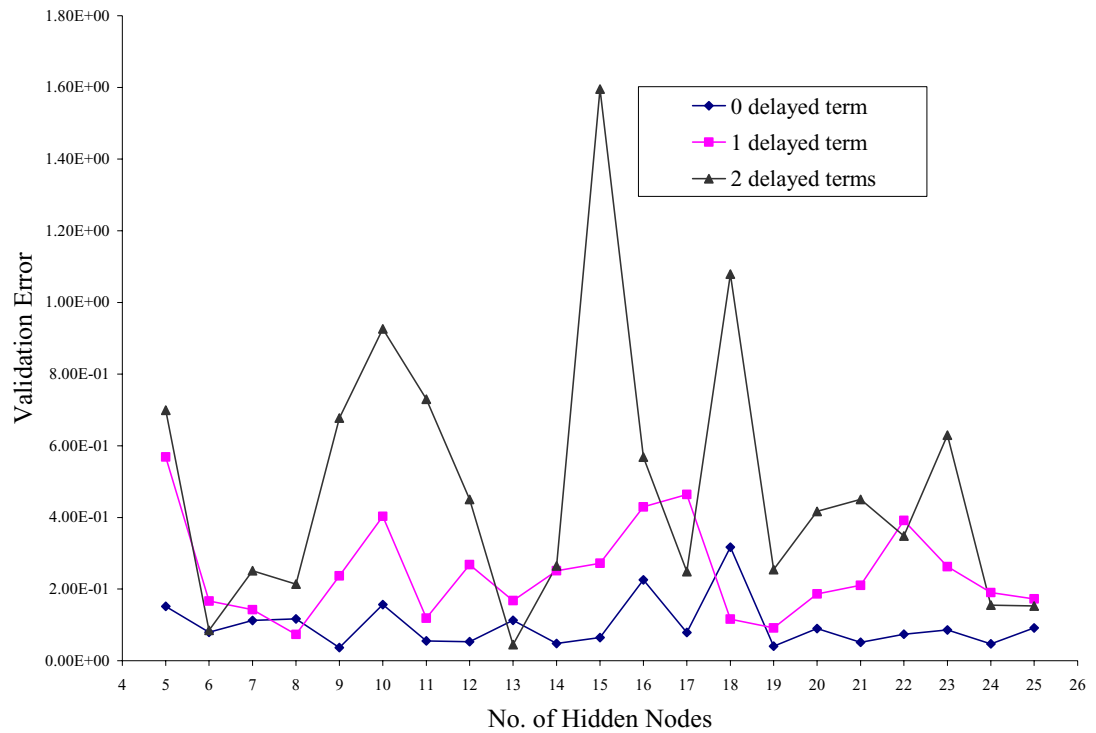


Figure 4.13 Training result of MIMO networks with *trainlm*

By analysing the results in the Figure 4.13 above and Table C4 in the Appendix C4, it is found that by adding a time delay to the networks input, the performance of the networks will decrease. The validation error is increased when the number of delayed term is increased. The performance of MIMO network without time delay is depicted in the Figure 4.14.

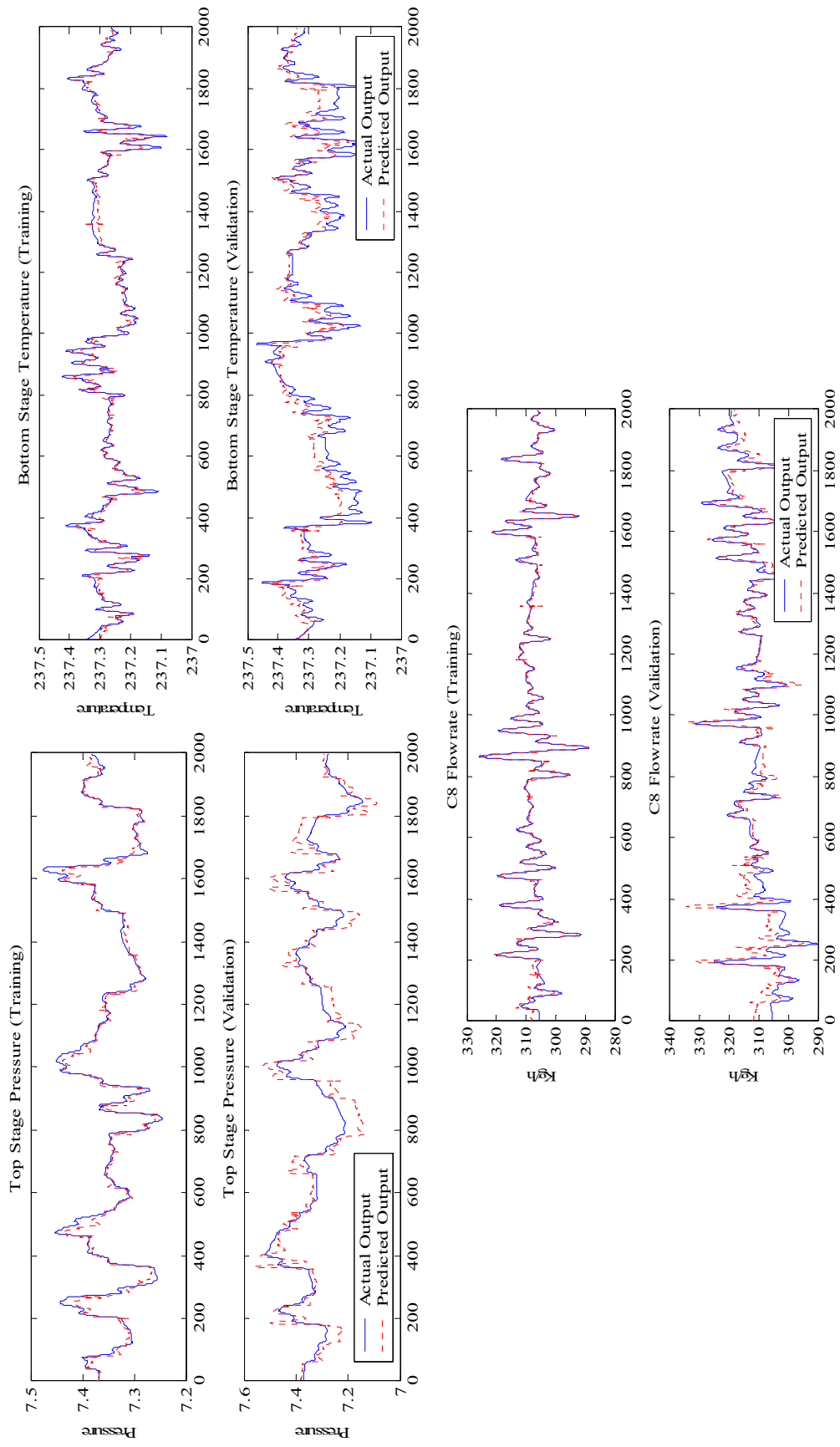


Figure 4.14 Performance of MIMO network with *trainlm*, without time delay (topology 7/8/1 nodes)

4.6 Summary

From the results of development for process estimator, it was found that the MISO Elman networks have successfully estimated the process accurately while the MIMO Elman network show results with lesser accuracy. Addition of time delay to the input network variables improved the performance of MISO networks while it deteriorated the performance of MIMO network. This is shown in the Figure 4.8 and Figure 4.13.

In the study of training algorithm, Levenberg-Marquardt Backpropagation (*trainlm*) produced the smallest network error compared to Gradient Descent with Momentum Backpropagation (*traingdm*) and Gradient Descent with Momentum and Adaptive Learning Backpropagation (*traingdx*). This statement is proved in the Figure 4.7.

The main focus of this study is estimating the sensor signals and generating consistent residuals from normal data. Good estimations facilitate the performance of the detector accurately carry out fault detection without making false classifications. The outputs from the process estimator are used as inputs for fault classifier. The development of the fault classifier will be discussed in the next chapter. This fault classifier is also based on artificial neural network.

CHAPTER V

NEURAL NETWORK FAULT CLASSIFIER

5.1 Introduction

The second stage of the neural network-based fault detection approach is decision making using another neural network model as a classifier. The residual vector has different structures for different faults. This feature can be used to detect these faults, and has been used in the neural network method by Patton et al. (1994) and Yu et al. (1996) using Multilayer Feedforward (MLFF) network. In this work, the neural network classifier is realised by a MLFF network.

In this study two types of fault are studied, they are corrupted sensor measurement and leakage in the pipeline. Sensor fault will be simulated as if the sensor is encountered with a sudden mechanical failure and caused a sustained bias to its measurement. The degree of bias will be used as a measurement for severity of the sensor fault. Bottom column temperature sensor and top column pressure sensor are two main elements in control system of the Precut column. Both positive and negative bias measurement of the sensors will be simulated for fault detection. For leakage fault, leak is simulated by creating a two output tee-junction in the *To Condenser* pipeline. For the leak line, the percentage of leakage is controlled by a control valve as shown in the Figure 5.1. The performance of the proposed fault detection scheme will be tested in the presence of the noise-corrupted measurements and without noise-corrupted measurements.

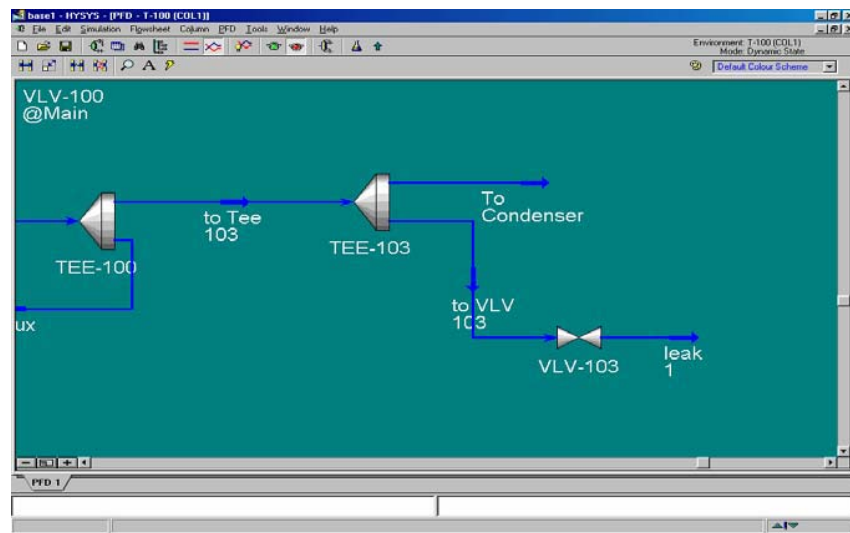


Figure 5.1 Leakage simulation in HYSYS simulator

Table 5.1 shows the list of process faults considered in this study. The target for this decision making stage is to develop fault classifiers that are able to closely monitor the sensor faults and at the mean time accurately classify the fault.

Table 5.1: List of sensor faults

1	Top column pressure sensor +ve bias	F1
2	Top column pressure sensor -ve bias	F2
3	Bottom column temperature sensor +ve bias	F3
4	Bottom column temperature sensor -ve bias	F4
5	Leakage To Condenser stream	F5

Artificial neural network will be used to develop the classifier. Design considerations of the neural network fault classifier involved are stated as follows:

- i) Sensitivity analysis
- ii) Preparation of training data and cross-validation data
- iii) Network training and cross-validation
- iv) Implementation with and without noise-corrupted measurements

5.2 Sensitivity Analysis

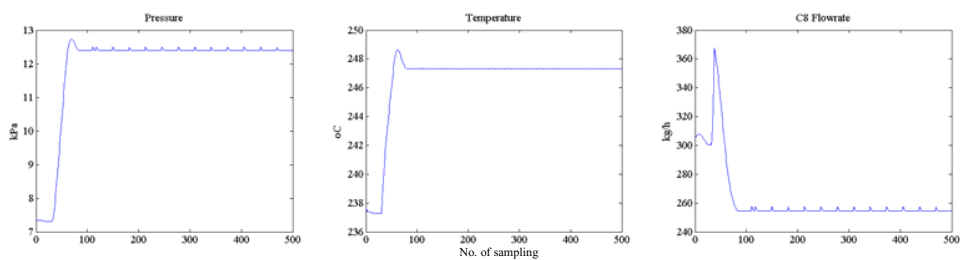
A sensitivity analysis is conducted to determine the degree of sensor bias which will cause violation of the operation limits. The analysis is done on trial and error basis with various sensor biases simulated at steady state base case condition. As defined by Downs and Vogel (1993), bias means deviation from the base case steady state condition. The relatively small sensor bias will cause the process to change to another steady state condition. But if the bias is relatively large, the process will not be able to absorb this disturbance and will eventually end up with out of control and violation of the operation limit.

There are several process operating limits for the Precut column with has to be observed and studied to avoid the accidents or catastrophic events. The following table shows the operating limits of the Precut column.

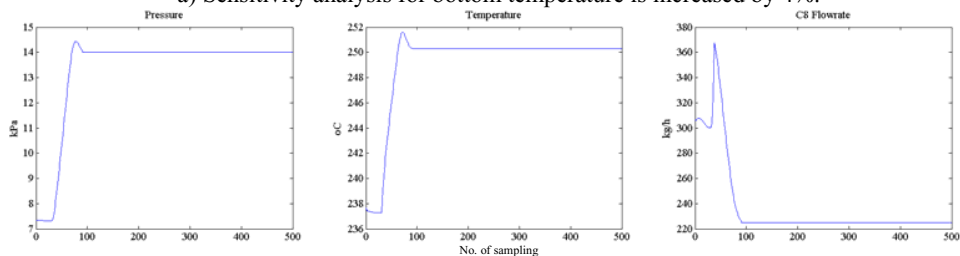
Table 5.2: Precut column operating constraints

Process Variables	Normal Operating Limits		Shut Down Limits	
	Low Limit	High Limit	Low Limit	High Limit
Top Column Pressure	3 Kpa	13 Kpa	1 kPa	20 Kpa
Bottom Column Temperature	230 °C	247 °C	220 °C	255 °C
Distillate (C8 Flowrate)	100 kg/h	None	75 kg/h	None
Pump Around Return Flowrate	2000 kg/h	15000 kg/h	0 kg/h	18000 kg/h
Reflux Flowrate	1000 kg/h	5000 kg/h	0 kg/h	6000 kg/h

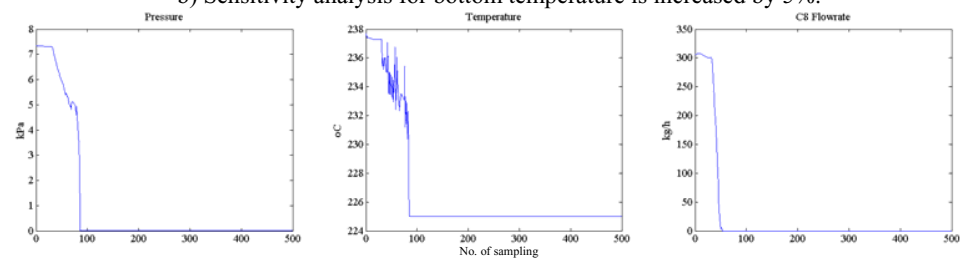
Analysis of the results showed that top column pressure, bottom column temperature and C8 flowrate operating limits can be easily violated while other variables are still well under controlled. Therefore, these three variables are considered as main indicator in the sensitivity analysis. The results of the analysis are summarised and depicted in the Figure 5.2.



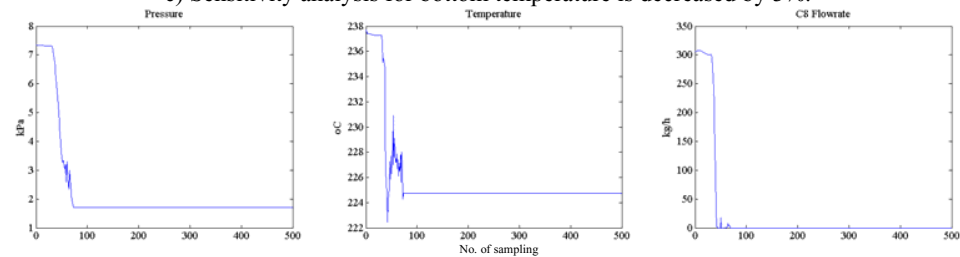
a) Sensitivity analysis for bottom temperature is increased by 4%.



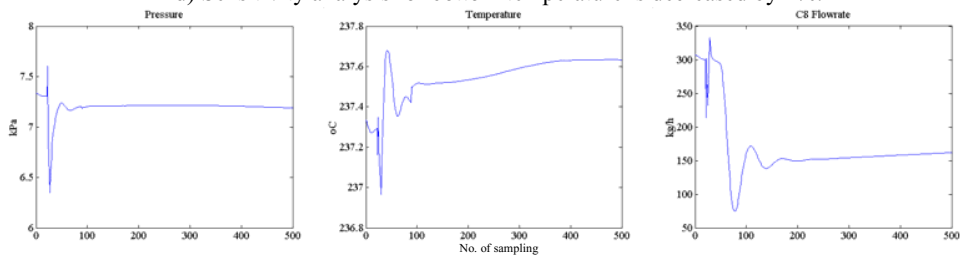
b) Sensitivity analysis for bottom temperature is increased by 5%.



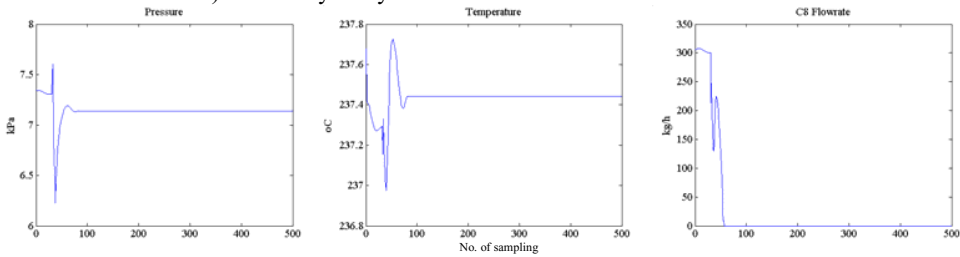
c) Sensitivity analysis for bottom temperature is decreased by 3%.



d) Sensitivity analysis for bottom temperature is decreased by 4%.



e) Sensitivity analysis for *To Condenser* stream leak is 0.1%.



f) Sensitivity analysis for *To Condenser* stream leak is 0.3%.

Figure 5.2 Sensitivity analysis for the Precut column

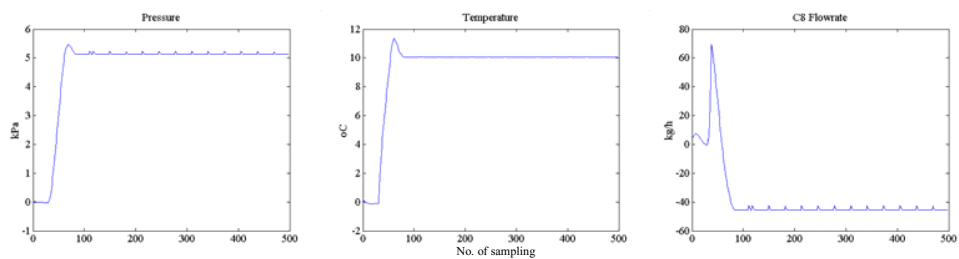
As observed from the Figure 5.2, the violation of top column pressure and bottom column temperature operating limits happen when the bias value of F1 is 4% or greater. In the case of F2, F3 and F4, the violation of operating limits are set at 3%, 4% and 3%, respectively. The violation of operating limit for C8 flowrate at distillate happens when the bias value of F5 is 0.1% or greater. Therefore, the fault classifier will be developed to monitor the violation of top column pressure, bottom column temperature and C8 flowrate at distillate.

5.3 Preparation of Training Data

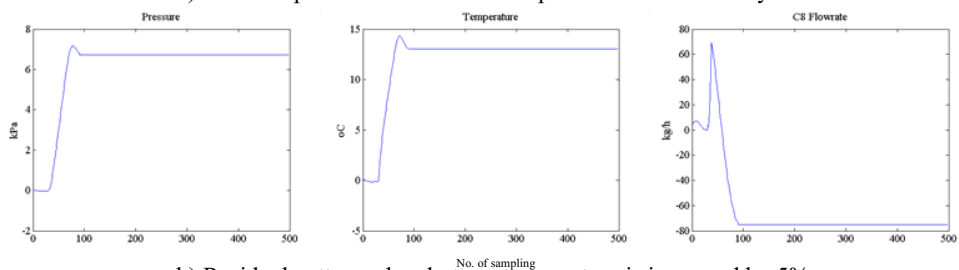
Input data for the artificial neural network (ANN) classifier are in the form of residual signal generated by ANN predictor. The faults data as stated in the Figure 5.2 are fed to the ANN predictor and the responses of the process obtained are shown in the Figure 5.3. However, prior to presentation to the neural network classifier, input data is scaled between 0 and 1. Scaling is necessary in order to get the same order of magnitude variables and to avoid numerical instability problems. The same input data scaling method used for process predictor is employed here. The scaling was performed using equation (5.1):

$$X_s = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (5.1)$$

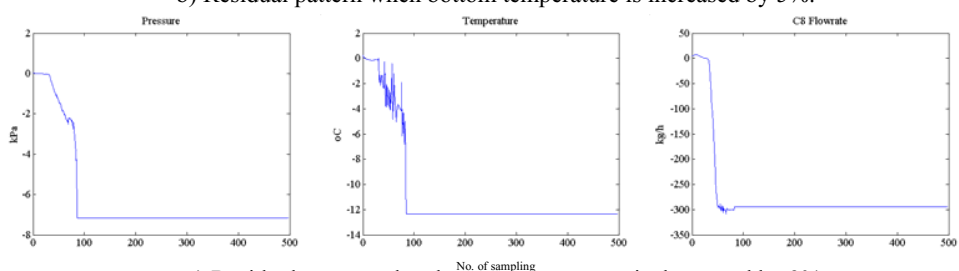
Here, X_s is scaled input and X is the actual input before scaling whereas X_{\min} and X_{\max} are actual inputs at their maximum and minimum respectively. X_s is the scaled residual signal for input variables i.e. top column pressure, bottom column temperature and C8 flowrate.



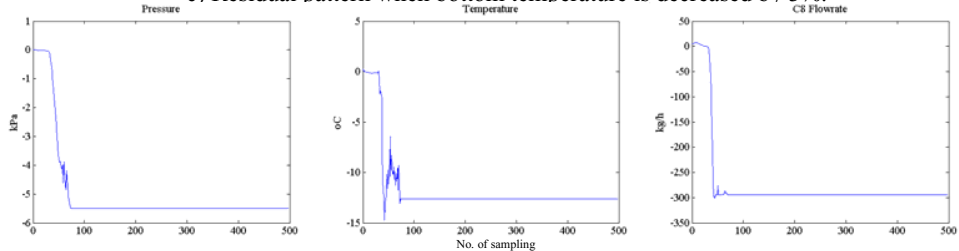
a) Residual pattern when bottom temperature is increased by 4%.



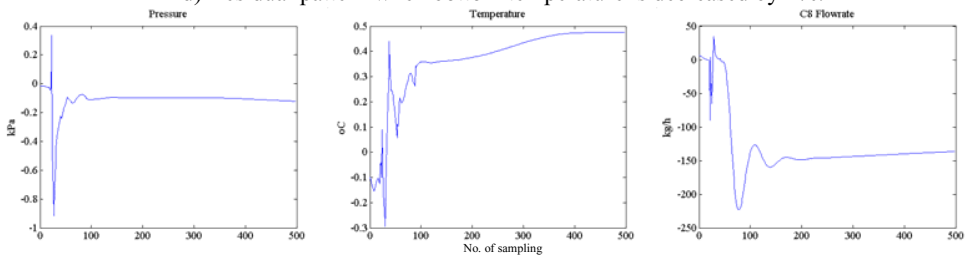
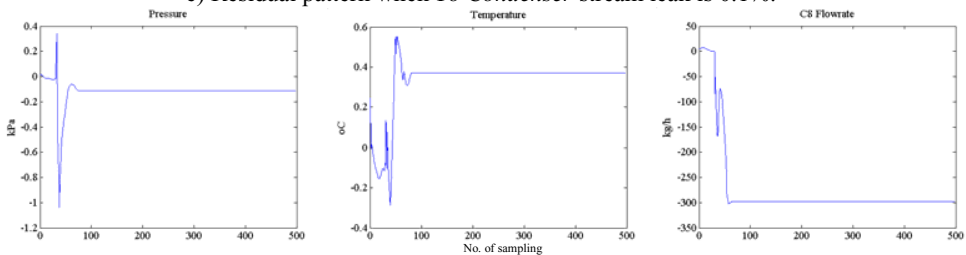
b) Residual pattern when bottom temperature is increased by 5%.



c) Residual pattern when bottom temperature is decreased by 3%.



d) Residual pattern when bottom temperature is decreased by 4%.

e) Residual pattern when *To Condenser* stream leak is 0.1%.f) Residual pattern when *To Condenser* stream leak is 0.3%.**Figure 5.3** Residual patterns for the Precut column

The ANN classifier receives the output from the ANN predictor and then analyses the residuals for fault classification. Before the ANN classifier can be applied for residual evaluation, training is required. At the output layer of the network, output nodes indicate the classes of sensor fault occurred. The classifier that will be developed in this study is used to detect multiple faults. Therefore, one or two output nodes are expected to respond when faults occurred. Training of ANN classifier is different from training of ANN predictor. In training for process predictor, input and output data for the network originated from the plant while the ANN classifier only required process variables as input to the model. The output is determined by the user.

The output data for the fault is designed to span between 0 and 1 by a linear relationship. The target is set as all network outputs being zero for the normal data or being 1 to indicate the violation of process limits. Two training data sets will be simulated for each sensor faults as shown in the Table 5.3.

Table 5.3: Sensor faults simulated for network training

Types of Faults	Magnitude of Biases
Top column pressure sensor +ve bias, F1	4.0 %
	5.0 %
Top column pressure sensor -ve bias, F2	3.0 %
	4.0 %
Bottom column temperature sensor +ve bias, F3	4.0 %
	5.0 %
Bottom column temperature sensor –ve bias, F4	3.0 %
	4.0 %
Leakage To Condenser stream, F5	0.3 %
	0.5 %

5.3.1 Top Column Pressure Sensor Positive Biases (F1)

From the sensitivity analysis that has been done, it is found that the top column pressure sensor bias of magnitude 4% or greater will cause violation to the

top column pressure high operating limit. Behaviours of the sensor resulting from 4% and 5% biases are shown in the Figure 5.4. The fault patterns from both cases are similar but the bigger bias resulted in faster violation of operating limits.

For monitoring fault efficiently, the fault classifier will be designed to give fault indication signal when the residual of top column pressure reach 6 kPa. At this point, the actual top column pressure has deviated by 6 kPa from the normal operating condition. The output data corresponds to 6 kPa residual's pressure was assigned a value of 1.0. The following figures show the training data and corresponding output indexes.

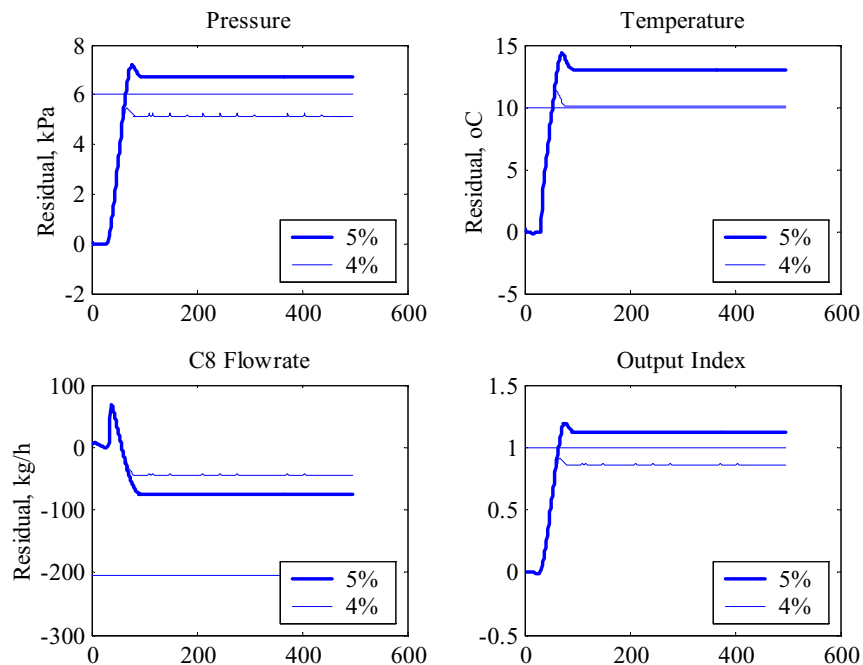


Figure 5.4 Training data for top column pressure sensor positive bias (F1)

5.3.2 Top Column Pressure Sensor Negative Biases (F2)

Figure 5.5 shows the process response of the top column pressure sensor negative bias of 3% or greater. The F2 fault classifier will be designed differently from F1 fault classifier. The F2 fault classifier will be designed to give fault

indication signal when top column pressure reaches residual -4 kPa from its normal operating limit. This signal can be used to acknowledge the operator or trigger the alarm system. The output data corresponds to -4 kPa residual pressure will be assigned a value of 1.0 similar way with fault classifier F1. In order to generate the input data for training, process fault F2 of 3% and 4% are simulated. The output indices for both faults are shown in the following figures.

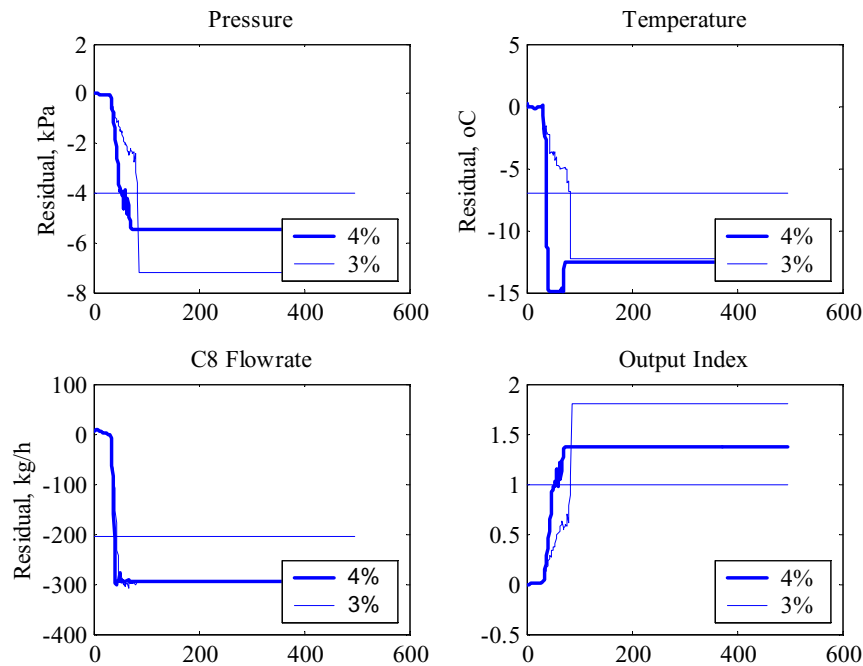


Figure 5.5 Training data for top column pressure sensor negative bias (F2)

5.3.3 Bottom Column Temperature Sensor Positive Bias (F3)

Based on the sensitivity analysis, bottom column temperature positive biases of 4% or greater will cause the violation of process high operating limits. The behaviour of the process when bottom column temperature sensor positive biases F3 are 4% or greater are depicted in the Figure 5.6. For this kind of classifier, it will be designed in order to give fault indication signal when the residual of bottom column temperature reach 10 °C. When bottom column temperature reaches residual of 10

$^{\circ}\text{C}$, the linear output index is 1.0 similar with the previous classifiers. Graphical illustration of bias 4% and greater are shown in Figure 5.6.

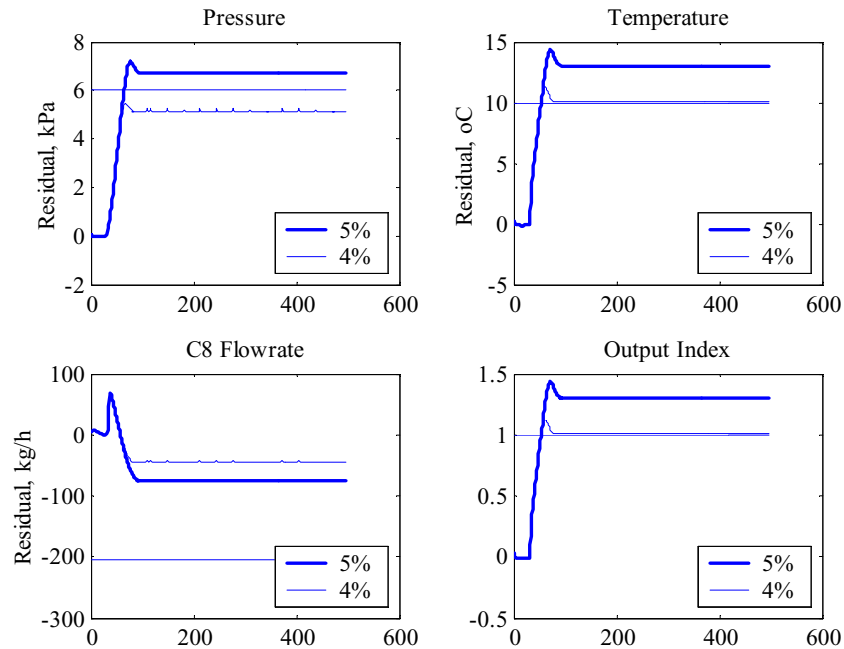


Figure 5.6 Training data for bottom column temperature sensor positive bias (F3)

5.3.4 Bottom Column Temperature Sensor Negative Bias (F4)

Behaviour of the sensor failure resulting from 3% and 4% biases are shown in the Figure 5.7. The fault sensitivity analysis suggested that the bottom column temperature sensor negative biases of magnitude of 3% or greater will cause violation of the column temperature low operating limit.

In order to efficiently monitor the fault, the classifier will be designed to give fault indication signal when the residual column temperature reaches -7°C . At this point, the actual column temperature has deviated by -7°C from its normal operating condition. The output data corresponds to -7°C residual's temperature will be assigned a value of 1.0. The following figures show the training data and its corresponding output indexes.

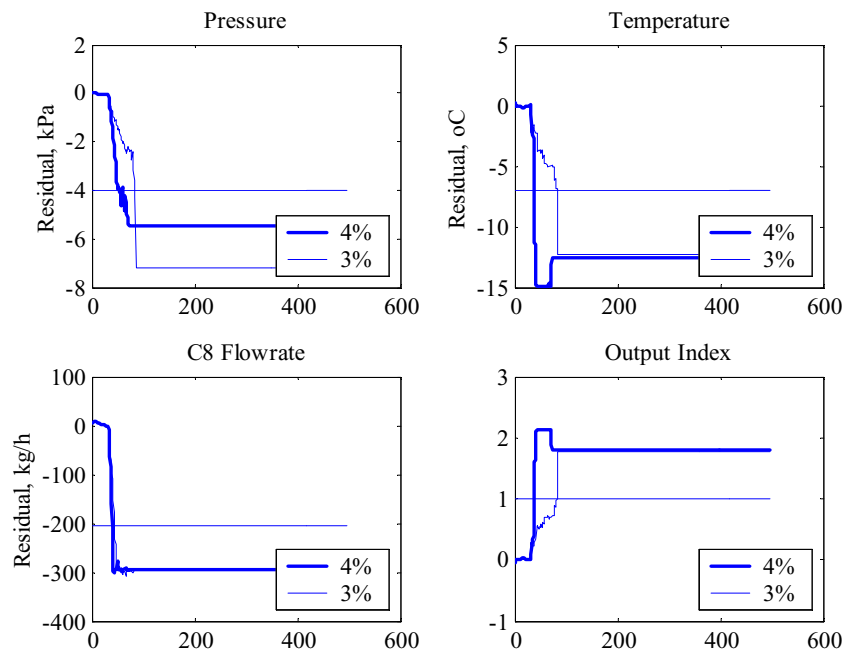


Figure 5.7 Training data for bottom column temperature sensor negative bias (F4)

5.3.5 Leakage to Condenser Stream Fault (F5)

Leakage fault classifier will be designed differently from the sensor fault classifiers. This classifier will design to give fault indication signal when the residual C8 flowrate reaches -206 kg/h. Based on the sensitivity analysis that has been done, leakage with magnitude of 0.3% or greater will cause violation to the C8 flowrate low operating limit. At this point, the actual C8 flowrate has deviated by -206 kg/h from its normal condition. Though the fault patterns from the both cases are quite similar but the bigger magnitude of leakage causes the faster violation of operating limits.

When the residual of C8 flowrate reaches -206 kg/h, the linear output index is set to 1.0. In order to generate the input data for training, leakage fault F5 of 0.3% and 0.5% are simulated. The output indexes for both faults are shown in the Figure 5.8.

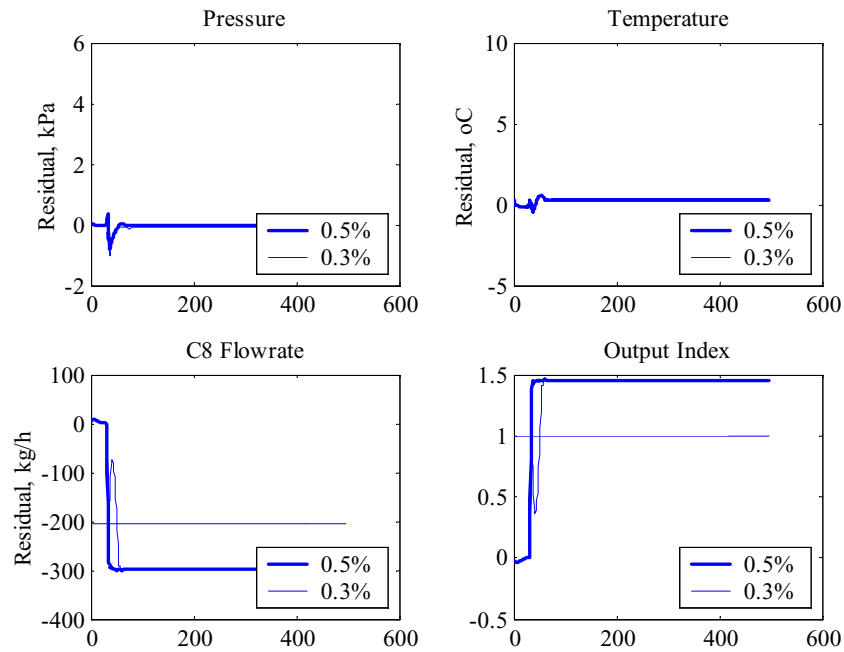


Figure 5.8 Training data for leakage *To Condenser* stream (F5)

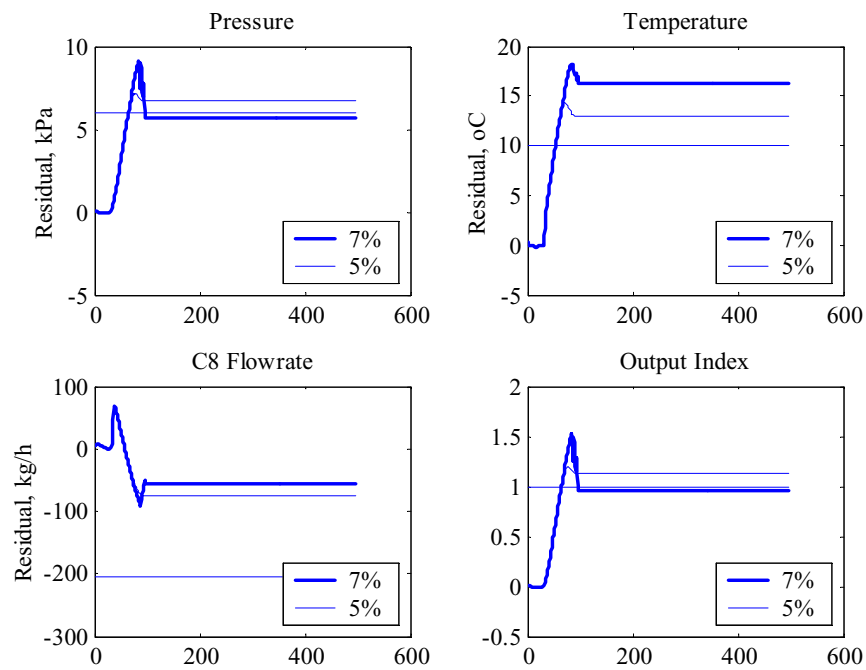
5.4 Preparation of Validation Data

Cross-validation of the neural network classifier is done parallel with the iterative training of the network. The cross-validation of the ANN classifier is important to ensure the classifier has a good generalisation capability. The concept of cross-validation is that after the ANN training using a given sample of data, the quality of the ANN mapping is evaluated using a different set of data (validation data) which is not trained. The best mapping of ANN is defined as the one which minimises the prediction error on validation data.

The data for the validation set will be designed in the same manner as in the case of the training data. Two cross-validation data sets will be simulated for each sensor fault and leakage as shown in the Table 5.4. The graphical presentations of the data are shown in the Figure 5.9 until Figure 5.13.

Table 5.4: Sensor faults simulated for network validation

Types of Faults	Magnitude of Biases
Top column pressure sensor +ve bias, F1	5.0 %
	7.0 %
Top column pressure sensor -ve bias, F2	4.0 %
	6.0 %
Bottom column temperature sensor +ve bias, F3	5.0 %
	7.0 %
Bottom column temperature sensor -ve bias, F4	4.0 %
	6.0 %
Leakage To Condenser stream, F5	0.5 %
	1.0 %

**Figure 5.9** Validation data for top column pressure sensor positive bias (F1)

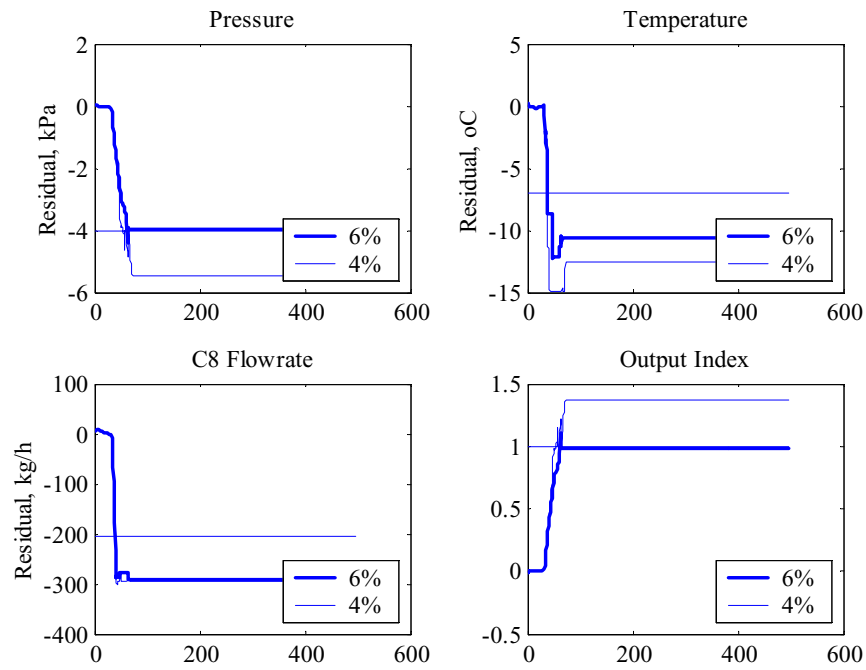


Figure 5.10 Validation data for top column pressure sensor negative bias (F2)

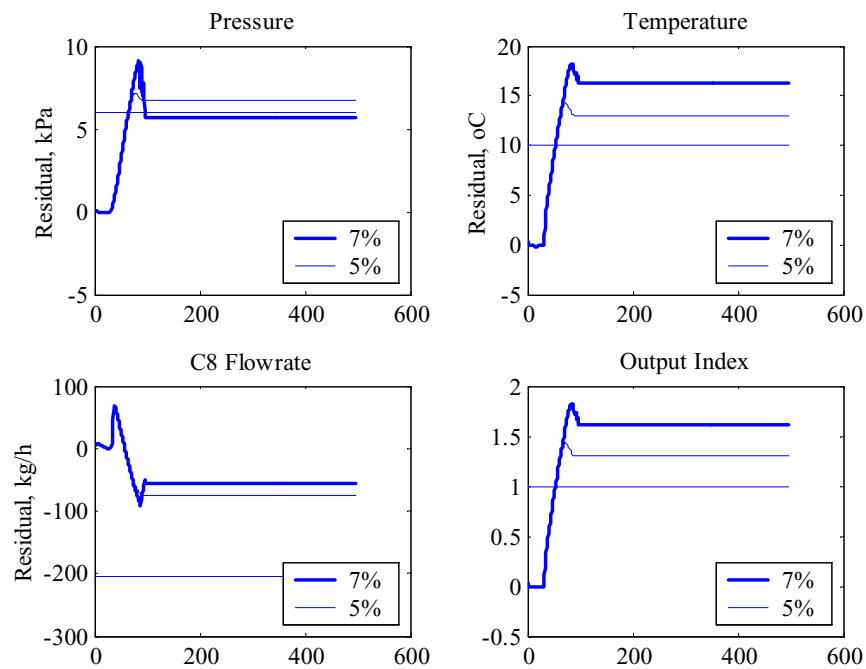


Figure 5.11 Validation data for bottom column temperature sensor positive bias (F3)

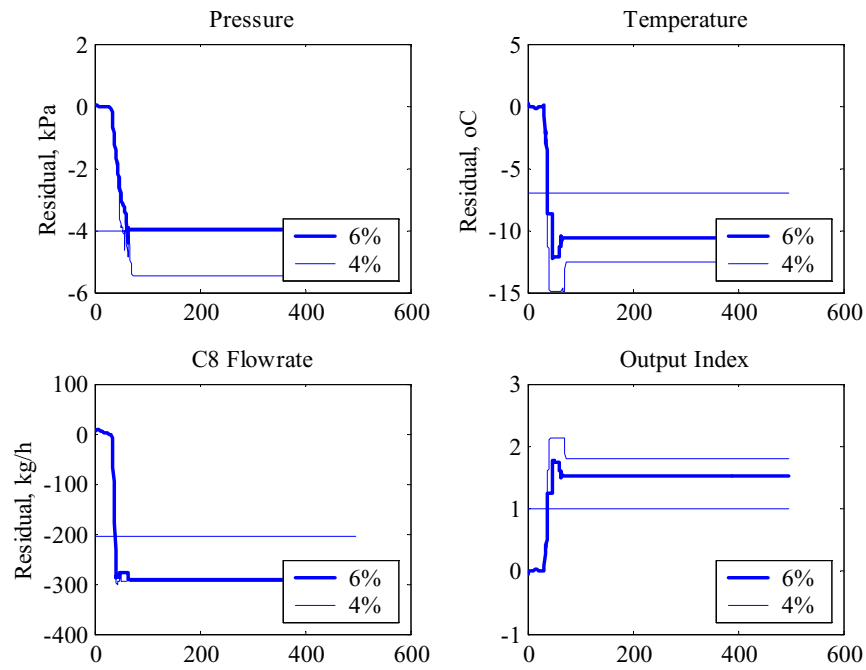


Figure 5.12 Validation data for bottom column temperature sensor negative bias (F4)

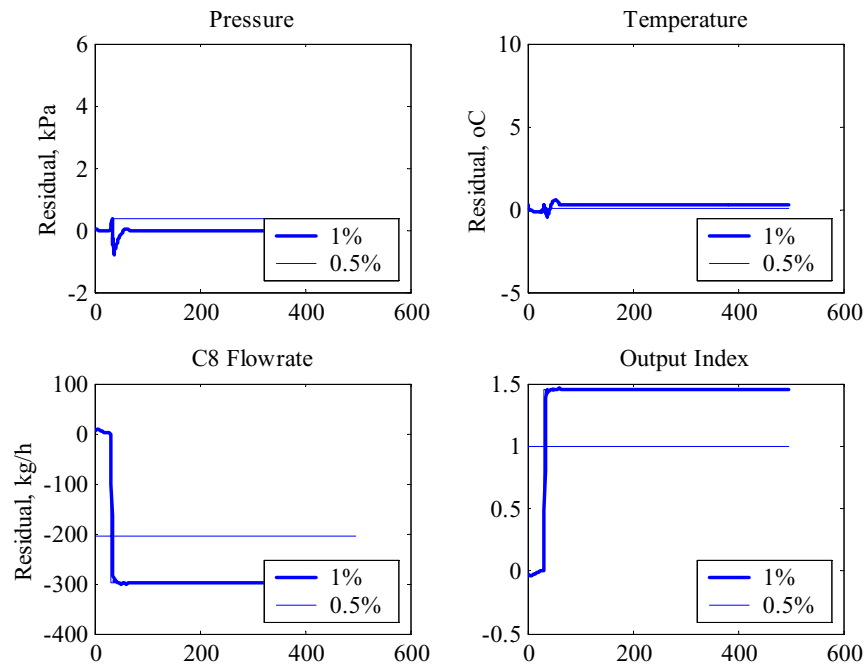


Figure 5.13 Validation data for leakage To Condenser stream (F5)

5.5 Structure Selection, Training and Cross-Validation

The fault classifier that will be used in this study is constructed using Multilayer Feedforward (MLFF) network with a single hidden layer. The MLFF network can either be a set of MISO networks or a single MIMO network. In this study, both type of network will be used and investigated.

For the training of MLFF network, Levenberg-Marquardt (*trainlm*) learning algorithm will be implemented. The network will be cross-validated at every batch of the training and the network weights and biases will be selected based on the minimum cross-validation error achieved in the training.

5.5.1 Multi-Input Single Output (MISO) Networks

The structure of MISO networks for fault classification is illustrated in the Figure 5.14. The network consists of five classifiers, each of them responsible for a single fault class. Levenberg-Marquardt learning algorithm will be used for training of each MISO network. The results of network training and cross-validation are shown in the figure 5.15 and Table C5 in the Appendix C5.

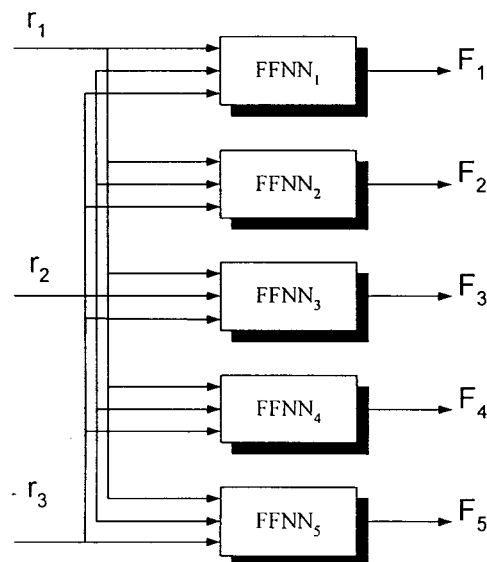


Figure 5.14 Artificial neural network MISO fault classifier

The smallest cross-validation error conceded by the network determined the optimum networks selected. From the Figure 5.15 and Table C5 in the Appendix C5, the minimum cross-validation error is conceded by 5, 8, 6, 5 and 5 hidden nodes for classifier F1, F2, F3, F4 and F5, respectively. To simplify the topology description, optimum topologies for the fault classifiers are (3/5/1) nodes, (3/8/1) nodes, (3/6/1) nodes, (3/5/1) nodes and (3/5/1) nodes. The training and cross-validation results are depicted in the following figures.

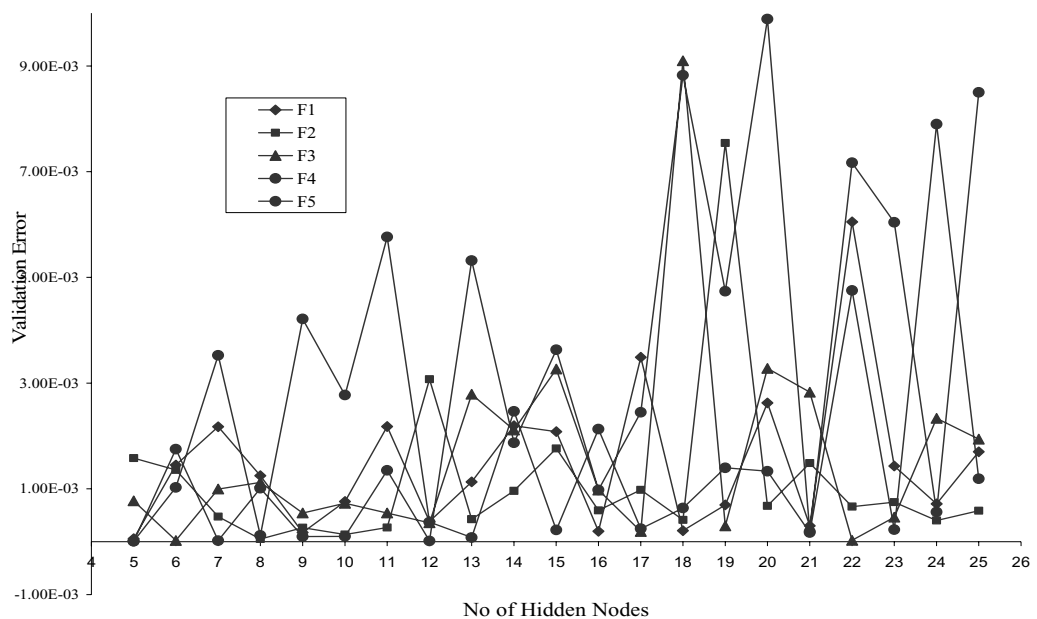


Figure 5.15 Training results of MISO networks fault classifier

Figure 5.16 shows training and cross-validation results for sensor fault F1. The figure clearly shows that the output of F1 node for training and cross-validation exceed the limit index of 1.0 while outputs of the other nodes are zero. This indicates that only sensor fault F1 happened during the training and cross-validation.

In the figure 5.17 training and cross-validation results for sensor fault F2 are shown. It is clearly shown that the output of F2 node for both training and cross-validation exceed the limit index of 1.0 while outputs of the other nodes are zero. This is because only sensor fault F2 occurred.

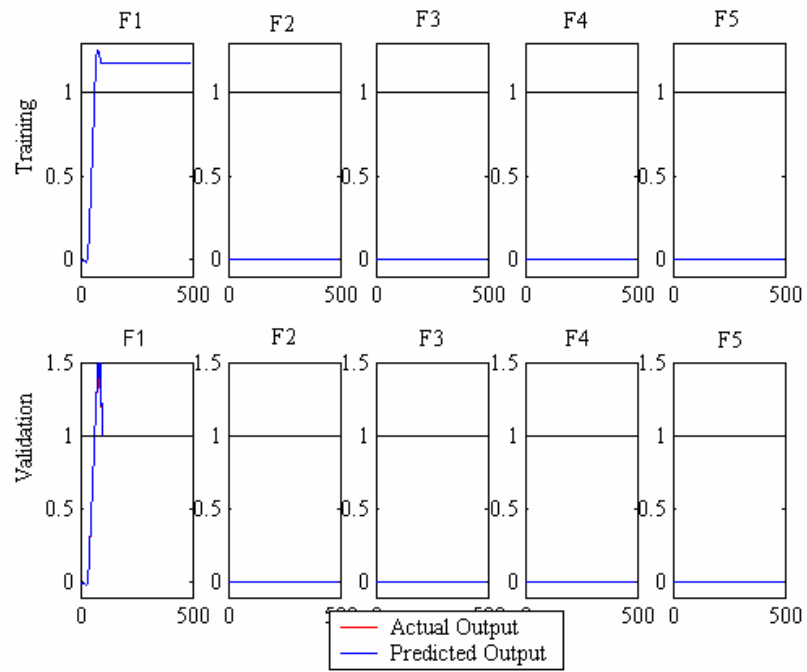


Figure 5.16 Training and validation results for sensor fault F1

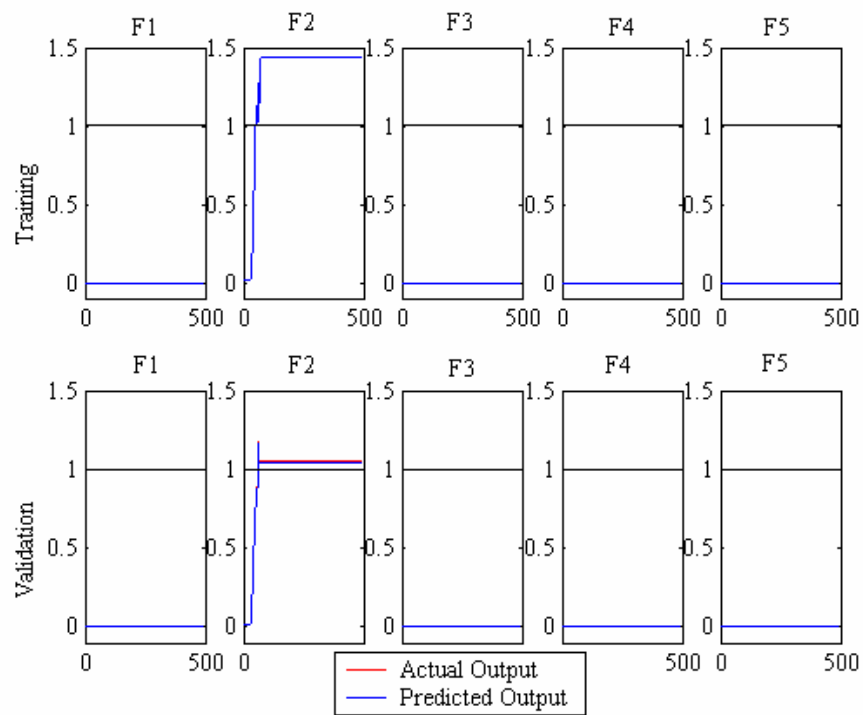


Figure 5.17 Training and validation results for sensor fault F2

Training and cross-validation results for sensor fault F3 are shown in the Figure 5.18. Again from the figure, the output of F3 node for both training and cross-validation exceed the limit index of 1.0 while outputs of the other nodes are zero indicated that only sensor fault F3 is happened.

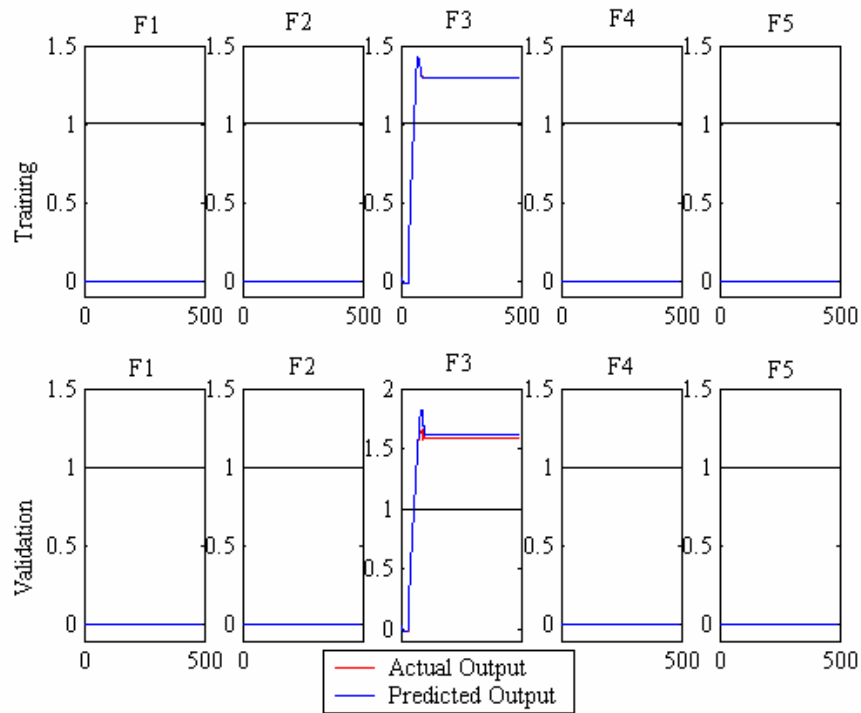


Figure 5.18 Training and validation results for sensor fault F3

Figure 5.19 shows training and cross-validation results for sensor fault F4. The good training and cross-validation results was achieved during the training and cross-validation process. The outputs of F4 node exceed the limit index of 1.0 while outputs of the other nodes are zero. This informs that only sensor fault F4 was occurred.

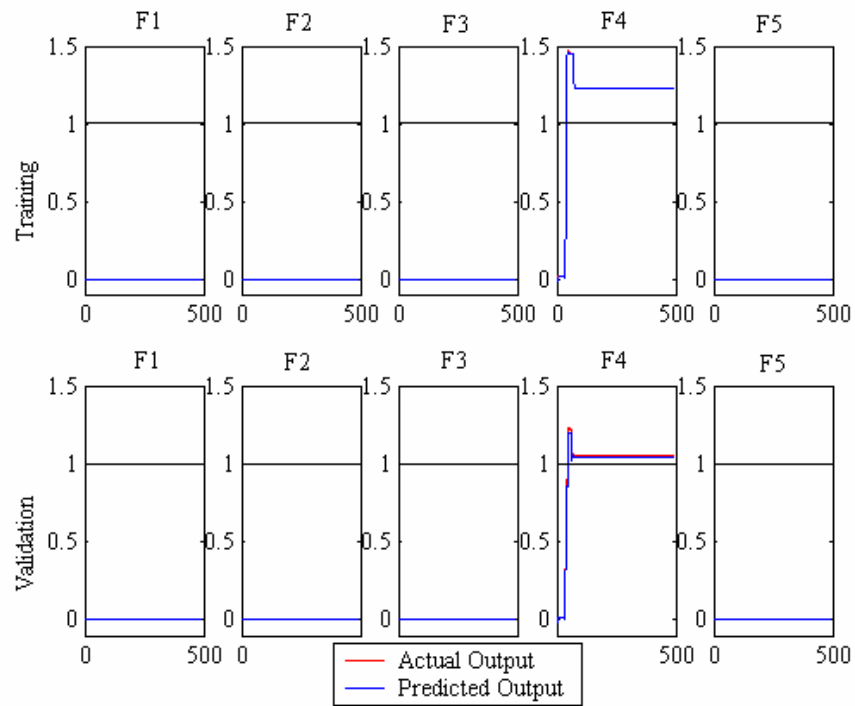


Figure 5.19 Training and validation results for sensor fault F4

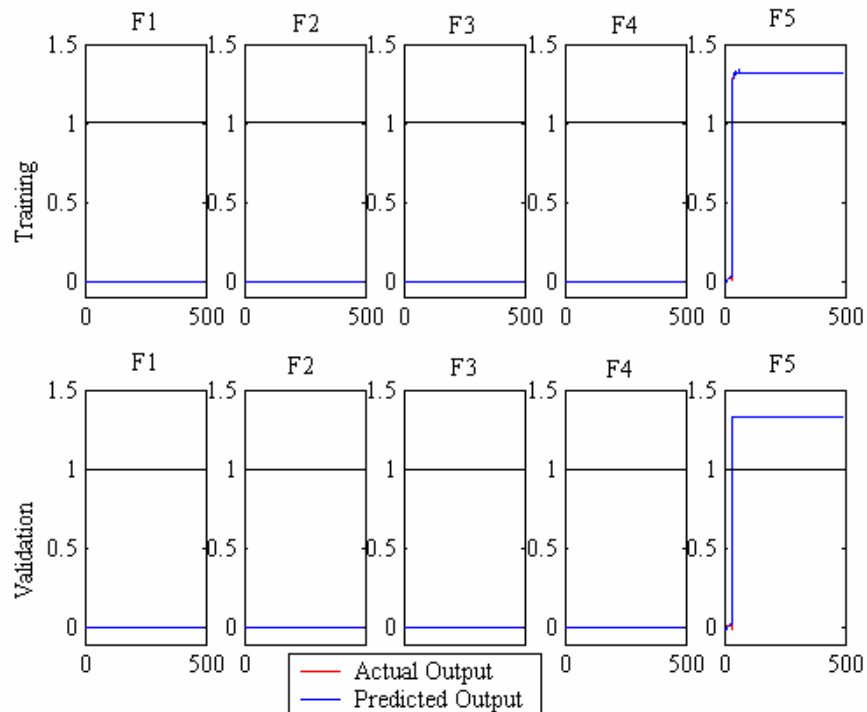


Figure 5.20 Training and validation results for leakage fault F5

In the Figure 5.20 training and cross-validation results for leakage fault F5 are shown. Again, the good training and cross-validation results was achieved. The output of F5 node for both training and cross-validation processes exceed the limit index of 1.0 while outputs of the other nodes are zero. This indicates that only sensor fault F5 was happened.

5.5.2 Multi-Input Multi-Output (MIMO) Network

MIMO network structure that will be used for fault classifier is illustrated in Figure 5.21. The network has three inputs in term of residual generated from the ANN process classifier. There are five outputs for this network, each of them representing a single fault class. The network will be trained using Levenberg-Marquardt training algorithm.

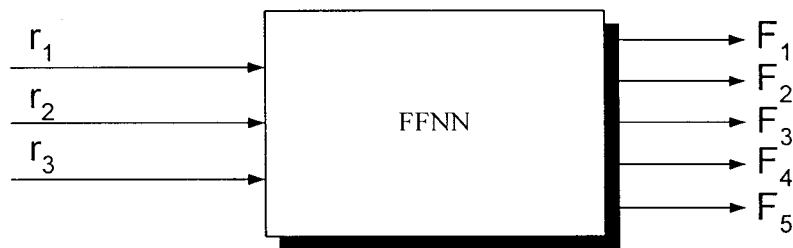


Figure 5.21 Artificial neural network MIMO fault classifier

Topologies of 5 to 25 hidden nodes were trained. The optimum topology was selected based on the smallest cross-validation error conceded among the networks trained. The results of training are shown in the Figure 5.22 and in the Table C6 in the Appendix C6. The error in this network is the average error for all outputs.

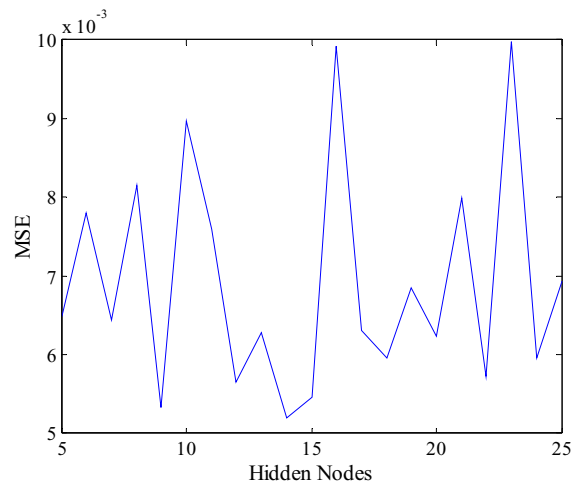


Figure 5.22 Cross-validation error of MIMO network

Based on the results obtained, network with 14 nodes in the hidden layer is selected as optimum topology for MIMO network. Network topology is represented by (3/14/5) nodes.

5.5.3 Structure Selection

The performance of the selected MISO networks and MIMO network for the fault classifier are compared and displayed in Table 5.5. For MISO networks, training and cross-validation errors are calculated by adding every training and cross-validation errors for every F1, F2, F3, F4, and F5 networks.

Table 5.5: Comparison of MISO networks and MIMO network performance

	MISO NETWORKS	MIMO NETWORK
Training Error	2.53785e-5	1.588596e-4
Cross-validation error	1.49405e-4	5.200906e-3

From the results obtained, MISO networks shows better generalisation ability compared to the MIMO network. This is because the training of MISO networks is relatively easier and faster because the number of the weight connections involved is smaller. Thus, the MISO networks are selected as the fault classifier for the decision making the fault detection scheme.

5.6 Performance of Fault Classifier With Non Noise-Corrupted Measurements

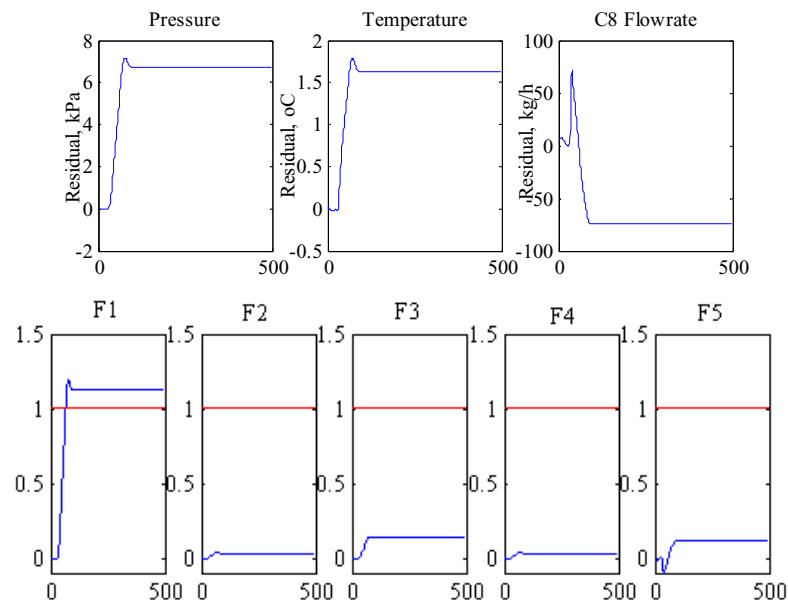
In this section, the fault detection scheme is implemented in the Precut column in order to know the scheme will detect fault happened successfully. There are three types of application for the testing of this detection scheme. The first is application for detecting single fault. This will be either top column pressure sensor biases or bottom column temperature sensor biases. The second application is for detecting multiple faults happened to the column and detecting the faults simultaneously. Lastly, is the application for detecting of leak in the *To Condenser* stream. All the performance of fault classifier will be tested with the non noise-corrupted measurements. After that, the performance of the proposed fault detection scheme will be tested with the noise-corrupted measurements.

5.6.1 Single Fault Detection

Detection of single fault occurred in the Precut column was applied to the proposed fault detection scheme. Here, sensor faults were simulated in order to create fault to the column. Descriptions of the simulated faults are given in the Table 5.6. The fault patterns and outputs of each classifier are depicted in the following figures.

Table 5.6: Faults simulated for fault classifier performance testing

Fault Pattern	Sensor	Fault	Simulated Condition	Violation of Operating Limit
1	Top Column Pressure	F1 (6%)	Step change 12% of feed	Yes
2	Top Column Pressure	F2 (5%)	Step change -10% of feed	Yes
3	Bottom Column Temperature	F3 (6%)	Step change 4% of reboiler	Yes
4	Bottom Column Temperature	F4 (5%)	Step change -5% of reboiler	Yes

**Figure 5.23** F1 (6%) happened after step change 12% of feed

In the Figure 5.23, fault F1 occurs and clearly detected by the proposed fault detection scheme. A 6% sensor fault F1 occurs after 12% step change is applied to the feed. This is shown by the output of F1 node exceeds the limit index of 1.0 which indicates the violation of sensor top column pressure. The other outputs do not exceed the limit.

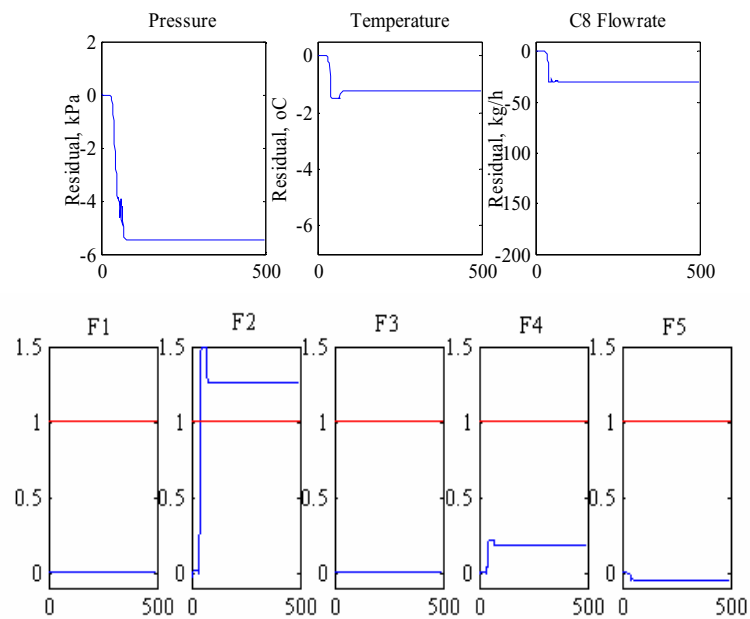


Figure 5.24 F2 (5%) happened after step change -10% of feed

Figure 5.24 shows 5% sensor fault F2 is occurred after -10% step change of feed. This fault is clearly detected as the output of F2 node exceeds the limit of 1.0 which indicates the violation of sensor top column pressure.

Performance of the proposed fault detection scheme in detecting F3 fault is shown in the Figure 5.25. A 6% sensor fault F3 occurs after 4% step change is applied to the reboiler temperature. This is shown by the output of F3 node exceeds the limit of 1.0 which indicates the violation of sensor bottom column temperature. The other outputs do not exceed the limit.

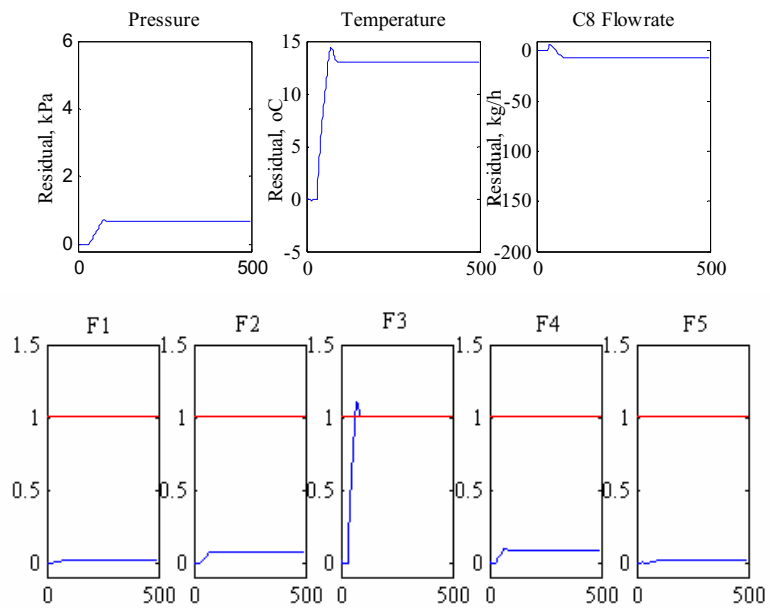


Figure 5.25 F3 (6%) happened after step change 4% of reboiler

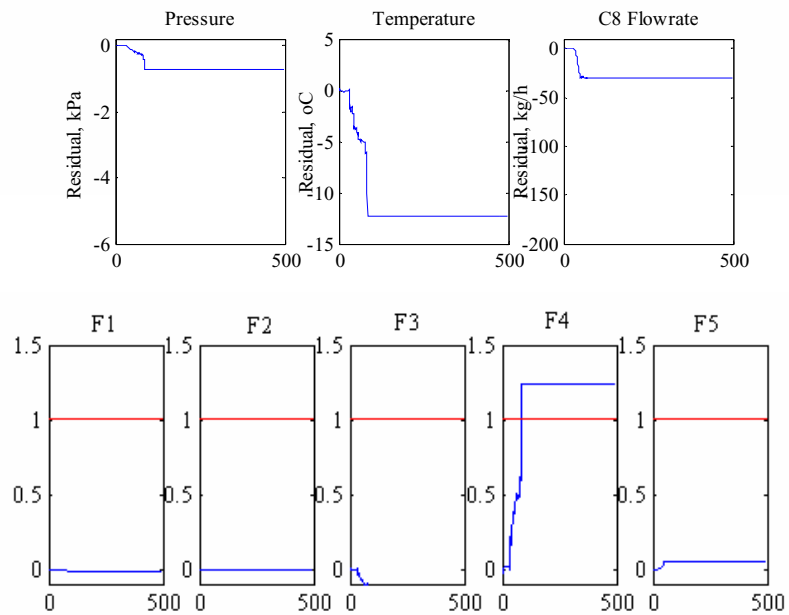


Figure 5.26 F4 (5%) happened after step change -5% of reboiler

In detecting a F4 fault, the proposed fault detection scheme also gives the same result as the other three faults. In the Figure 5.26, a 5% of sensor fault F4 is detected after -5% step change of reboiler temperature is applied. The fault is clearly detected as the output of F4 node exceeds the limit index of 1.0 which indicates the violation of sensor bottom column temperature.

The results reveal the success of the classifiers in detecting a single process fault. The output of the classifier reached the index of 1.0 which indicated the violation of operating limit and the cause of the violation. Note that the outputs of the other nodes were not zero when there is no faults occurred. According to Ruiz (2001), this is happened a result of system uncertainty and modelling errors.

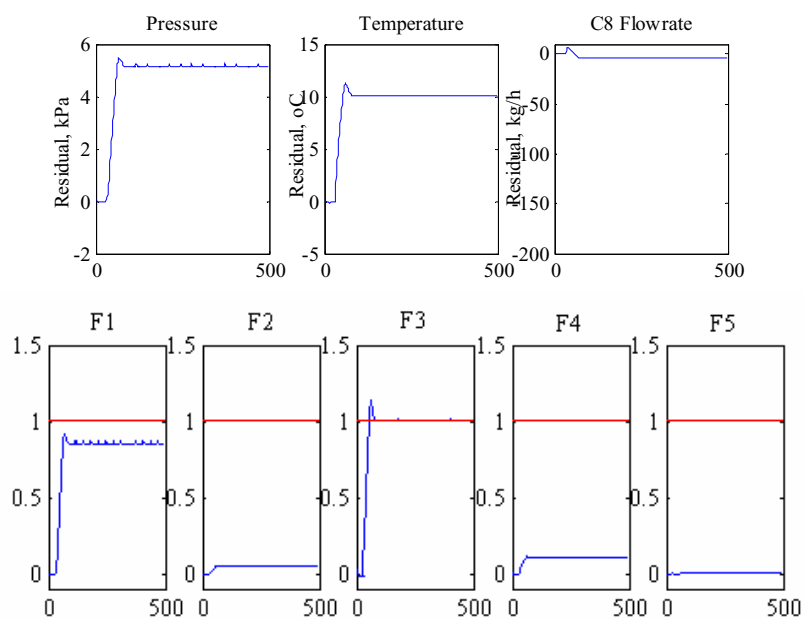
5.6.2 Multiple Faults Detection

From the previous section, the results proved that the fault detection scheme can detect single faults successfully. However, the performance of this detection scheme upon simultaneous multiple faults must be tested. Simulated multiple faults are given in the Table 5.9. The fault detection scheme is tested using these simulation data and the fault patterns and outputs of each classifier are depicted in the following figures.

Figure 5.27 shows a 5% of sensor F1 and a 6% of sensor F3 occur after 4% step change is applied to the reboiler temperature. The output of the F1 node does not exceed the limit index of 1.0 and therefore does not violate the sensor of top column pressure. But, precaution must be taken since it is nearly exceeding the limit index. The violation of sensor bottom column temperature is detected as the output node of sensor fault F3 exceeds the limit index of 1.0. The others node do not exceed the limit index. For this case, the violation of multiple operating limits is not happened.

Table 5.7: Simulation of faults for multiple faults classifiers performance testing

Fault Pattern	Sensor	Fault	Simulated Condition	Violation of Multiple Operating Limit
1	Top Column Pressure and Bottom Column Temperature	F1 (5%) F3 (6%)	Step change 4% of reboiler	No
2	Top Column Pressure and Bottom Column Temperature	F1 (6%) F3 (6.5%)	Step change 5.5% of reboiler	Yes
3	Top Column Pressure and Bottom Column Temperature	F2 (4.5%) F4 (5%)	Step change -2.5% of reboiler	Yes
4	Top Column Pressure and Bottom Column Temperature	F2 (5%) F4 (5.5%)	Step change -4% of reboiler	Yes

**Figure 5.27** F1 (5%) and F3 (6%) happened after step change 4% of reboiler

Since a 4% step change of reboiler did not cause a multiple faults happen, the researcher increased the step change to 5.5% in order to create a multiple faults. Figure 5.28 shows 5.5% of sensor F1 and 6% of sensor F3 happened after 5.5% step change of reboiler temperature. From the figure, it is clear that the output of F1 and F3 nodes exceed the limit index of 1.0. For this case, the violation of multiple operating limits is happened and clearly detected by the proposed fault detection scheme.

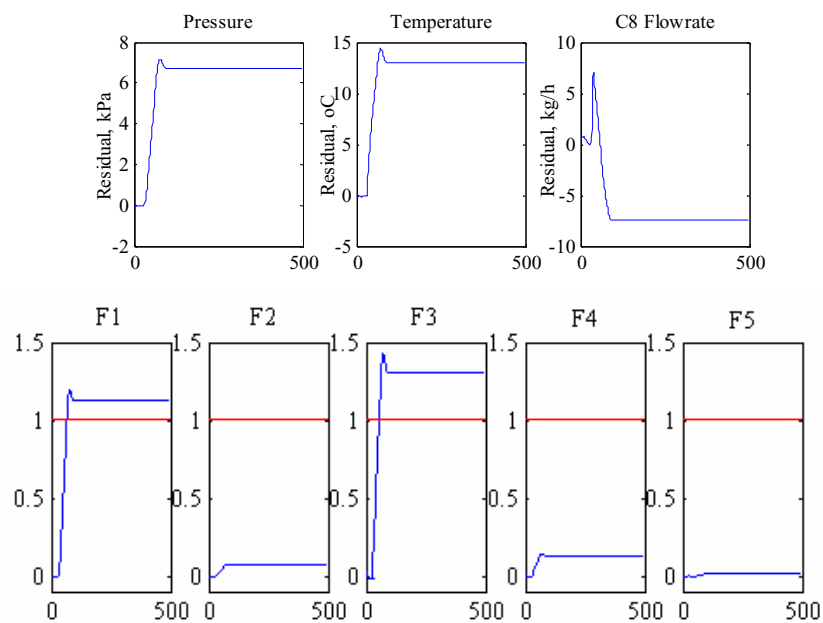


Figure 5.28 F1 (5.5%) and F3 (6%) happened after step change 5.5% of reboiler

In the Figure 5.29, 4.5% of sensor F2 and 5% of sensor F4 happened after -2.5% step change of reboiler temperature are shown. The output of F2 and F4 nodes exceed the limit index of 1.0. The others node do not exceed the limit index. For this case, the violation of multiple operating limits is happened and clearly detected.

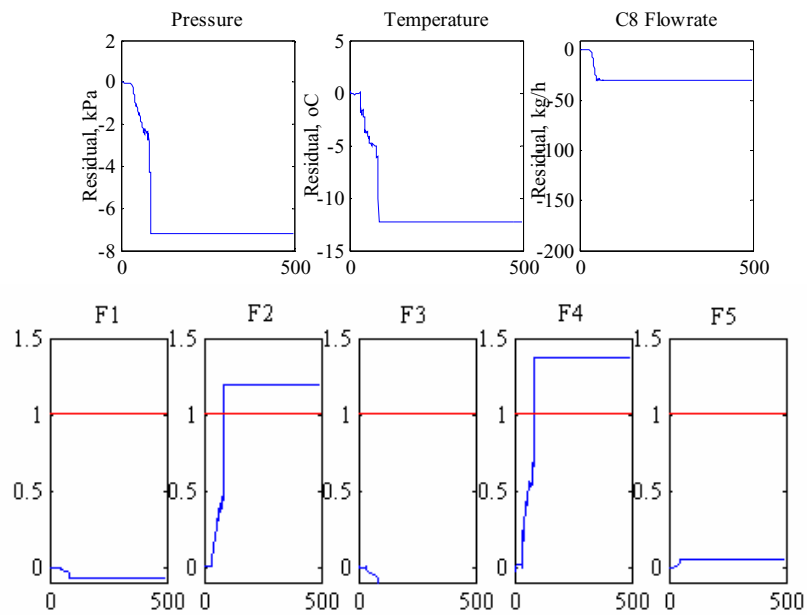


Figure 5.29 F2 (4.5%) and F4 (5%) happened after step change -2.5% of reboiler

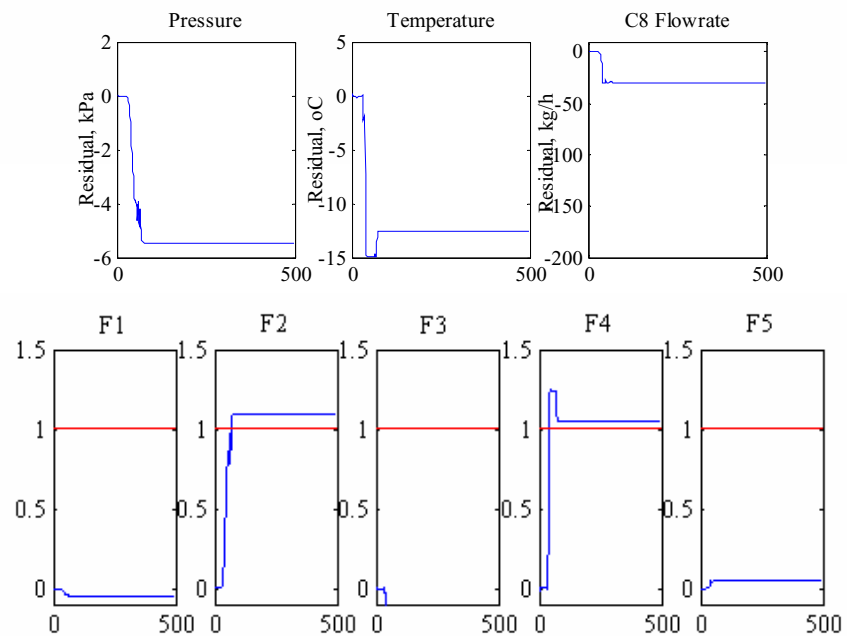


Figure 5.30 F2 (5%) and F4 (5.5%) happened after step change -4% of reboiler

Figure 5.30 above shows 5% of sensor F2 and 5.5% of sensor F4 happened after -4% step change of reboiler temperature. The output of F2 and F4 nodes exceed the limit index of 1.0 and cause the violation of sensor top column pressure and bottom column temperature. For this case, the violation of multiple operating limits is happened and clearly detected.

From the figures shown above, the classifiers are successful in detecting multiple faults in the Precut column. When the output of the classifiers reach index of 1.0, faults are acknowledged. Therefore, this fault detection scheme can be used in detecting single or multiple faults and the cause of the violation.

5.6.3 Leakage Fault Detection

Leakage in the *To Condenser* stream is simulated by creating a Tee-junction with two outputs in that stream. Leakage is simulated by opening the control valve at the one of the two outputs. By opening the valve with certain percentage, we assume that the real leakage happened to the stream. Descriptions of the simulated leakage are given in the Table 5.8. The fault patterns and outputs of each classifier are depicted in following figures.

Table 5.8: Simulation of leakage for leak classifier performance testing

Fault Pattern	Leakage	Fault	Simulated Condition	Violation of Operating Limit
1	To Condenser stream	F5 (0.1%)	0.1% valve opening	Yes
2	To Condenser stream	F5 (0.2%)	0.2% valve opening	Yes

Figure 5.31 below shows 0.1% of leakage fault F5 after 0.1% of *To Condenser* valve is opened. It is clearly shown that the output of F5 node exceeds the limit index of 1.0 and violating the operating limit.

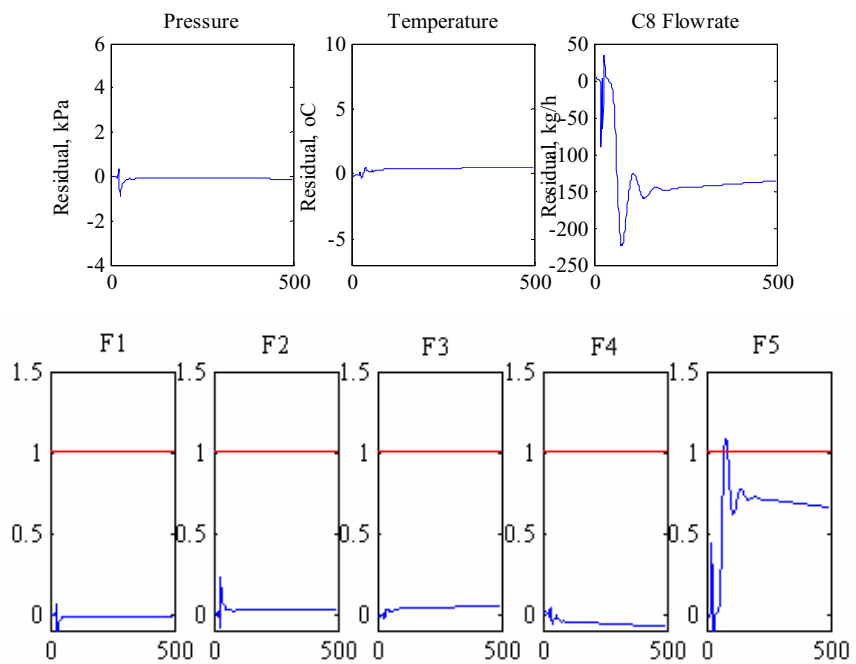


Figure 5.31 F5 (0.1%) happened after 0.1% valve opening

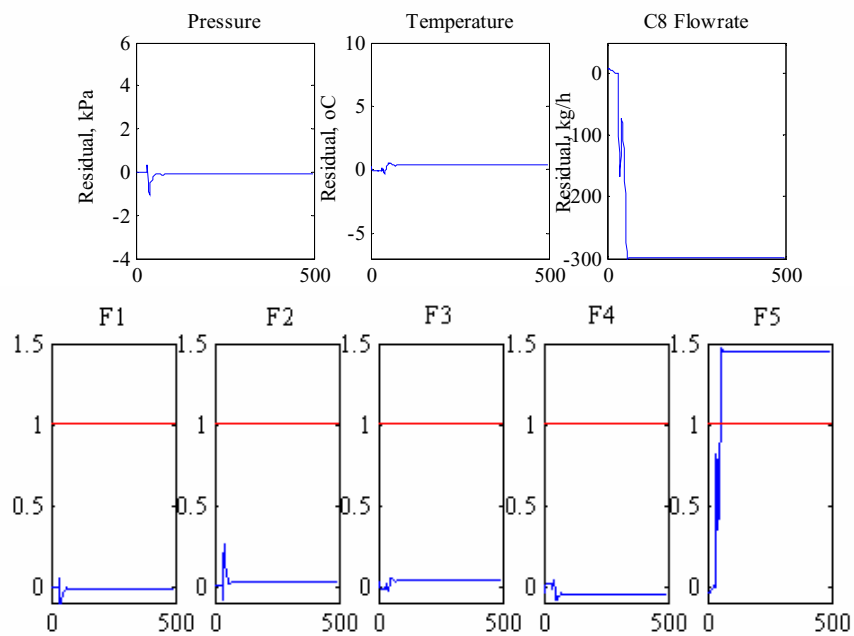


Figure 5.32 F5 (0.3%) happened after 0.3% valve opening

In the Figure 5.32, 0.2% of leakage fault F5 after 0.2% of *To Condenser* valve opened is shown. It is clearly shown that the output of F5 node exceeds the limit index of 1.0 and violating the operating limit.

The results show the success of the classifier in detecting leakage in the process stream. Even though the percentage of the leakage is very small, the classifier can detect the fault successfully. The classifier's output reaches the index of 1.0 to indicate the violation of the operating limit and the cause of the violation.

5.6.4 Conclusion

From all the results collected, the researcher concluded that the proposed fault detection scheme was successfully applied in detecting single or multiple faults and as well as leakage. The proposed scheme produced an accurate detection of all faults in the non noise-corrupted environment.

5.7 Performance of Fault Classifier With Noise-Corrupted Measurements

In this section, the proposed fault detection scheme will be tested its performance using noise-corrupted measurements. The same previous method will be applied here but with some additional of noise introduced to the measurement data. There are three types of application for the testing of the proposed detection scheme. The first is application for detecting single fault. This will be either top column pressure sensor biases or bottom column temperature sensor biases. The second application is for detecting multiple faults happened to the column and detecting the faults simultaneously. Lastly, is the application for detecting of leak in the *To Condenser* stream.

The performance of the proposed detection scheme will be tested using noise-corrupted measurements data. The noise-corrupted measurements data are simulated

in order to create process noise to the Precut column. There are two types of noise that have been simulated, *light* and *mild noises*. The *light noise* is simulated by introducing 1% random noise to the measurements data while *mild noise* is simulated by developing about 10% of random noise to the measurements data.

5.7.1 Single Fault Detection

Detection of single fault with noise-corrupted measurements that occurred in the Precut column was applied to the proposed fault detection scheme. Here, sensor faults with light (1%) noise and mild (10%) noise were simulated in order to create the process noise to the column. The fault patterns and outputs of each classifier are depicted in the following figures.

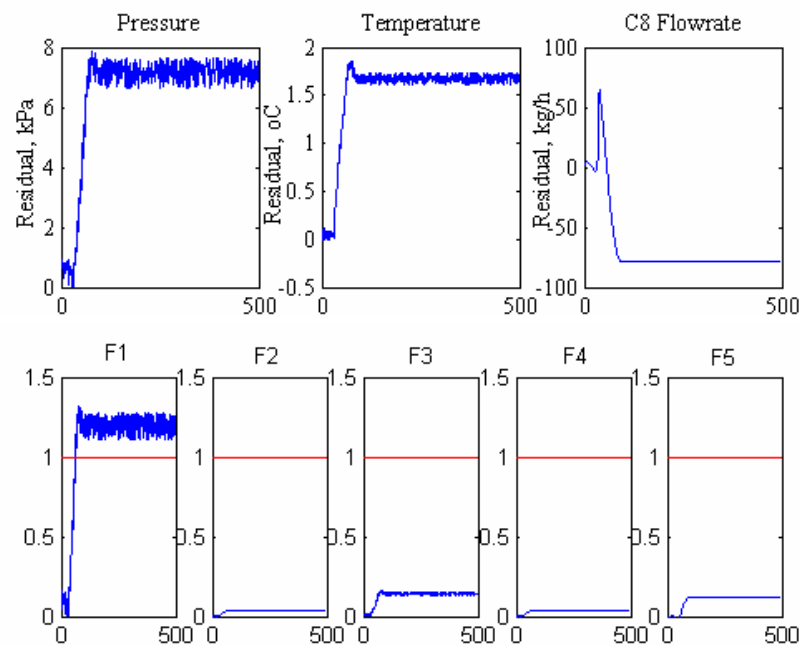


Figure 5.33 F1 (6%) happened after step change 12% of feed with 1% noise

In the Figure 5.33, fault F1 occurs and clearly detected by the proposed fault detection scheme even with 1% noise. The figure clearly shows that the process data is oscillating with some slight noise. With this magnitude of noise, the proposed fault

detection scheme successfully detected the F1 fault. This is shown by the output of F1 node exceeds the limit index of 1.0 which indicates the violation of sensor top column pressure. The other outputs do not exceed the limit.

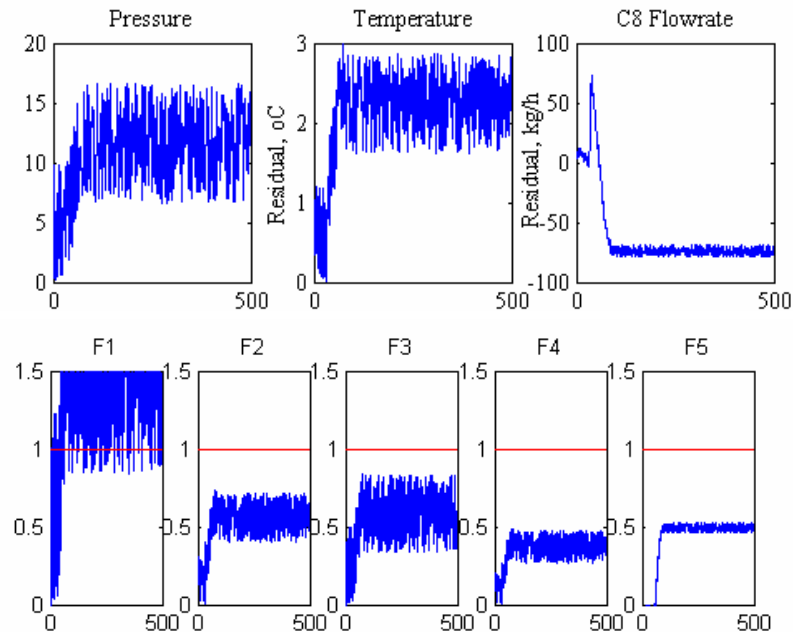


Figure 5.34 F1 (6%) happened after step change 12% of feed with 10% noise

Figure 5.34 shows 6% sensor fault F1 is occurred after 12% step change of feed with some additional of 10% process noise. Since the magnitude of noise is bigger compared to the previous case (Figure 5.32), the process is oscillating with a bigger range. This fault is clearly detected as the output of F2 node exceeds the limit of 1.0 which indicates the violation of sensor top column pressure. But some precautions have to be taken since with this magnitude of noise, the performance of the proposed fault detection scheme is becoming worst.

In the Figure 5.35, 6% of fault F3 happened after step change of 10% applied to the feed with addition of 1% of process noise. Again with this light noise, the proposed fault detection scheme able to detect fault F3. This is shown by the output of F3 exceeds the index of 1.0 indicating violation of sensor.

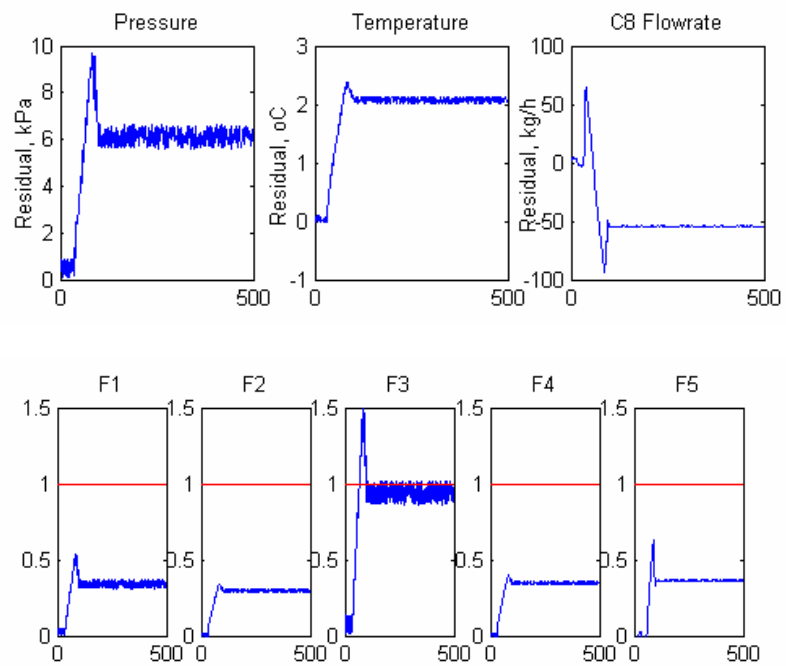


Figure 5.35 F3 (6%) happened after step change 10% of feed with 1% noise

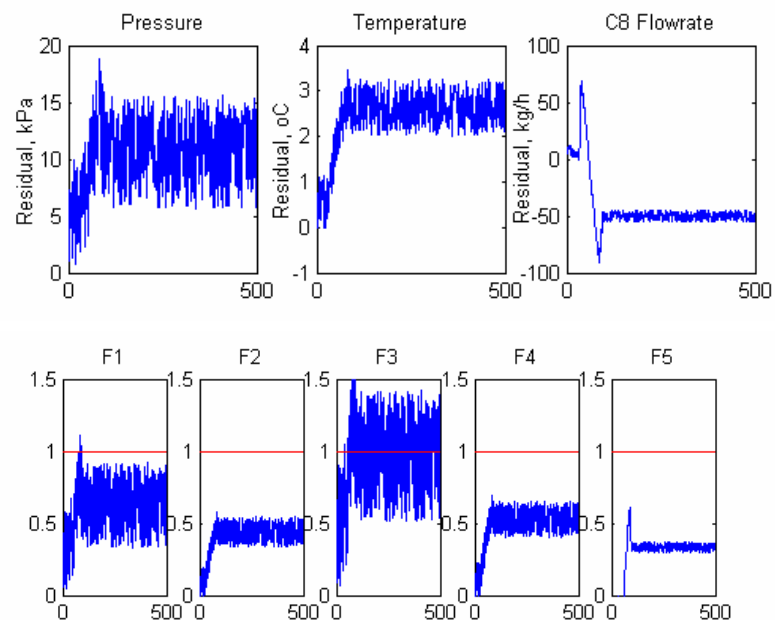


Figure 5.36 F3 (6%) happened after step change 10% of feed with 10% noise

But, in the Figure 5.36, different result is shown. In this case fault F3 happened after 10% step change applied to the feed. In addition, there is 10% of process noise introduced to the process. With this mild noise, the output of the proposed detection scheme shows violation of two sensors, F1 and F3. Technically, only sensor F3 will be violated as shown by Figure 5.35. By introducing 10% of process noise, the proposed fault detection scheme can not detect the fault accurately.

The same results also happened for a F2 and F4 sensor (Refer Appendix D). The results reveal the success of the classifiers in detecting a single process fault with 1% of process noise introduced to the measurements data. The output of the classifier reached the index of 1.0 which indicated the violation of operating limit and the cause of the violation. When 10% of noise introduced to the process, the performance of proposed fault detection scheme became worst. This indicates that the scheme can not deal with the process where significant magnitude of noise occurred.

5.7.2 Multiple Faults Detection

From the previous section, the results proved that the fault detection scheme can detect single faults successfully even though there was a small noise with magnitude of 1%. However, the performance of this proposed detection scheme became worst when 10% of process noise was introduced. In this section, the proposed fault detection scheme will be tested using the simulation data and the fault patterns with the existing of noise. The performance of detecting multiple faults simultaneously is the characteristic that will be tested in this section. The outputs of each classifier are depicted in the following figures.

Figure 5.37 shows a 5% of sensor F1 and a 6% of sensor F3 occur after a 5% step change is applied to the reboiler temperature. The figure clearly shows that there was some oscillation happened in the process because of noise. With 1% of noise in the process, the proposed fault detection scheme can successfully detect multiple

faults in the process. In the other word, the performance of the proposed detection scheme is not affected by the presence of 1% noise.

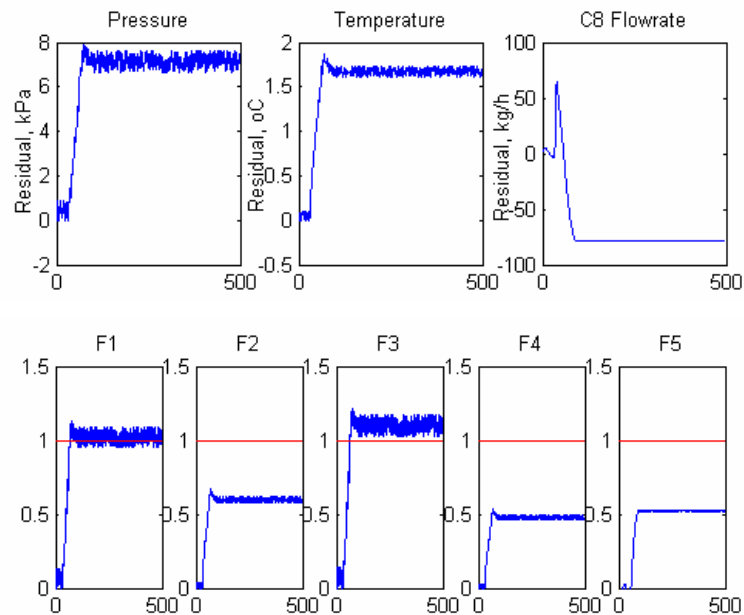


Figure 5.37 F1 (5%) and F3 (6%) happened after step change 5% of reboiler with 1% noise

Since a 1% noise did not affect the performance of the proposed fault detection scheme, therefore in this section the researcher introduced a 10% of noise to study either the scheme is able to detect multiple faults in the noisy environment. Figure 5.38 shows 5% of sensor F1 and 6% of sensor F3 happened after 5% step change of reboiler temperature. From the figure, it is clear that there is some mild oscillation happened in the process. In this case, the output of sensors F1, F2, and F3 exceeded the operating limit. Technically, only the sensors of F1 and F3 will show violation of multiple operating limits as shown in the Figure 5.37. Because of the presence of 10% of process noise, the proposed fault detection scheme can not detect the multiple faults precisely or give the wrong detection.

In the next figures, Figure 5.39 and Figure 5.40, show the same finding to verify that with the presence of light noise (1%), the proposed scheme is able to detect multiple faults without gives some significant error in the detection. On the

other hand, with the presence of mild noise (10%), the scheme gives the wrong detection and can not detect multiple faults that actually happened in the process.

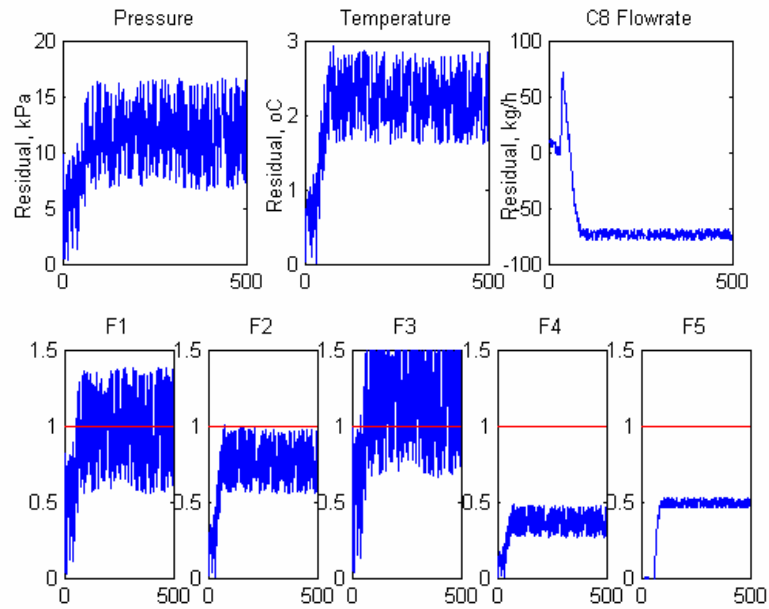


Figure 5.38 F1 (5%) and F3 (6%) happened after step change 5% of reboiler with 10% noise

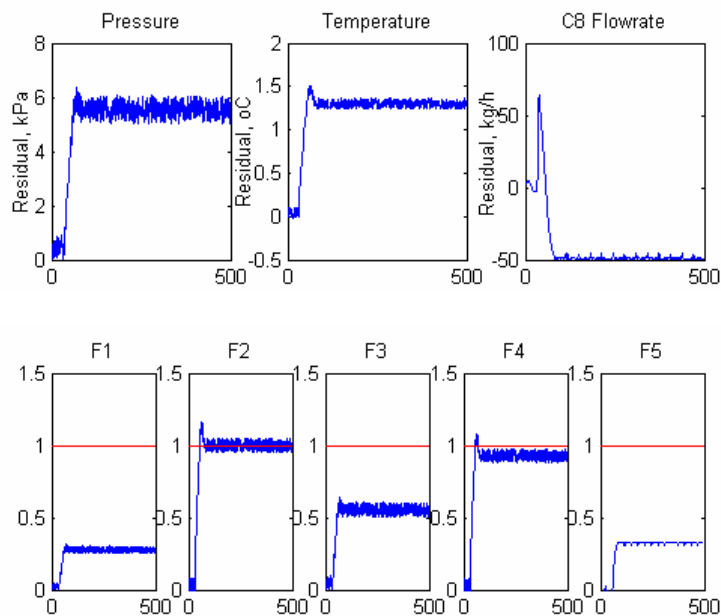


Figure 5.39 F2 (4.5%) and F4 (5%) happened after step change -2.5% of reboiler with 1% noise

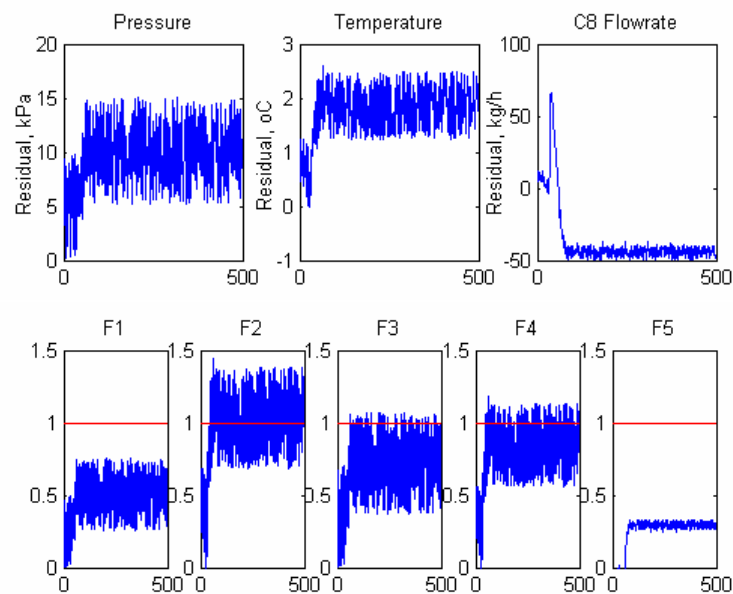


Figure 5.40 F2 (4.5%) and F4 (5%) happened after step change -2.5% of reboiler with 10% noise

In summary, in detecting multiple faults that happened in the process the proposed fault detection scheme was able to detect the fault although there was some noise in the process. There are some percentages of magnitude of noise that will affect the performance of the proposed scheme. The proposed fault detection scheme can manage to detect the multiple faults with presence of 1% of process noise. When the magnitude of noise increased to 10%, some error in detection of faults occurred. The proposed scheme was not able to give an accurate detection.

5.7.3 Leakage Fault Detection

Lastly, the researcher will test the performance of the proposed fault detection scheme in detecting leakage with the presence of process noise. The same method as previous (Section 5.6.3) will be applied in order to simulate the leakage but with some addition of process noise. There are two types of process noise applied

here, light (1%) and mild (10%) noises. The results are illustrated in the following figures.

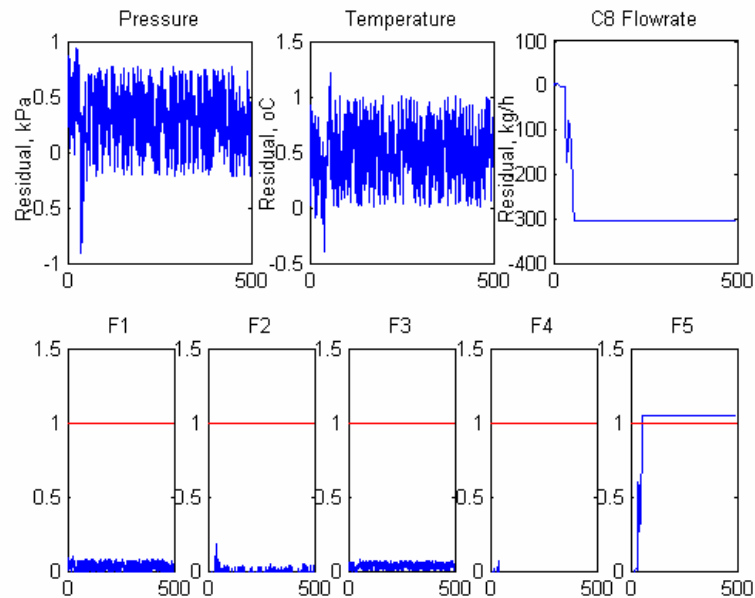


Figure 5.41 F5 (0.3%) happened after 0.3% valve opening with 1% noise

In the Figure 5.41, 0.3% of leakage fault F5 after 0.3% of *To Condenser* valve opened is shown. It is clearly shown that the output of F5 node exceeds the limit index of 1.0 and violating the operating limit although there was 1% noise introduced to the process.

As shown in the Figure 5.42, the proposed scheme is analysed with 10% of noise to detect leakage. It is clearly shown that with the presence of 10% noise in the process, some significant oscillation occurred and the proposed scheme can not detect leakage fault effectively. This is because with the presence of 10% noise, the output of F1 almost exceeded the index of 1.0 which technically will not happened as shown in the Figure 5.41. Therefore, the performance of the proposed fault detection scheme will deteriorate as the magnitude of the process noise increasing.

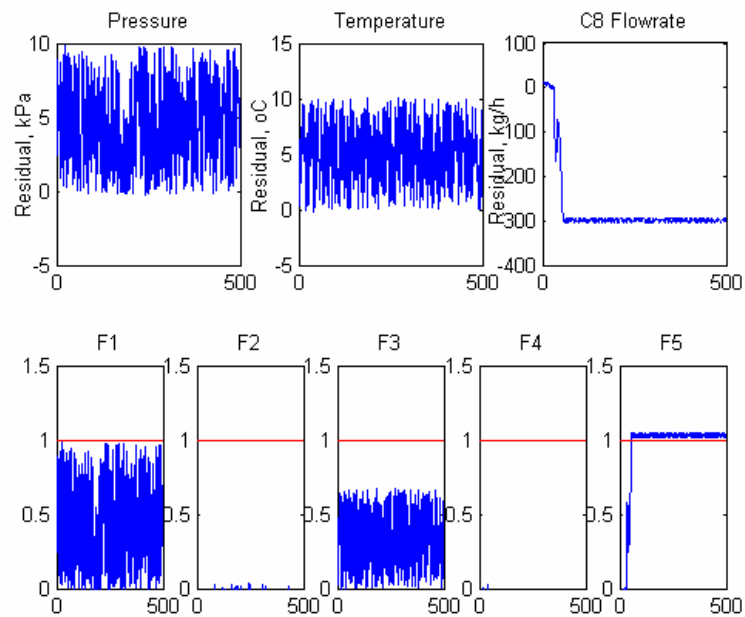


Figure 5.42 F5 (0.3%) happened after 0.3% valve opening with 10% noise

5.7.4 Conclusion

After implemented the proposed fault detection scheme in the noisy environment, the researcher came with the conclusion that the performance of the proposed scheme in detecting single or multiple faults and also leakage with the presence of process noise deteriorated when the magnitude of process noise increasing. The proposed scheme can detect all the faults in the light (1%) noisy environment and produced significant detection error when 10% of the process noise was introduced.

5.8 Summary

From the development of fault classifier, it was found that the MISO Multilayer Feedforward (MLFF) networks have successfully classified the faults accurately compared to the MIMO MLFF network. MISO MLFF produced the smallest validation errors in the every training executed. Lavenberg-Marquardt learning algorithm was the excellent training algorithm for the classifiers. The selected structure for the classifiers was determined by the topology which gave minimum validation error during training.

The performance of the proposed fault detection scheme was analysed in the non noise-corrupted measurements and also in the noise-corrupted measurements. After implementation of the proposed scheme to both environments, some findings were established. In the non noisy environment, the proposed fault detection scheme successfully detected single sensor fault, multiple sensor faults and also stream leakage. The output of the classifier reached the index of 1.0 that indicated the violation of operating limit and the cause of the violation. Note that, the outputs are not zero when no faults occur, as a result of the effects system uncertainty and modelling errors. In the noisy environment, the researcher concluded that the performance of the proposed scheme deteriorated when the magnitude of process noise increasing. The proposed scheme can detect all the faults in the light (1%) noisy environment and produced significant detection error when 10% of the process noise was introduced.

CHAPTER VI

CONCLUDING REMARKS AND FUTURE WORK

6.1 Introduction

It is well acknowledged that fault detection and diagnosis play important roles in industry, i.e., improving the safety and reducing the cost of operation. The applications of fault detection and diagnosis techniques are extensively used in some industrial areas, such as, petrochemical plants, chemical process plants, and fatty acid fractionation plant. On the other hand, there are still many areas where fault detection techniques are not yet utilised and there exist exciting opportunities for their applications. One obvious reason is that fault detection and diagnosis models are process dependent, and not applicable to all processes. For example, a fault detection model used in a chemical process plant might not be suitable for the application to a fatty acid fractionation plant. The sparseness of commercial products is also an indication that the fault detection and diagnosis technique is still far from perfected. There is still much work that needs to be done to fully develop this technique. The following is a summary of the contribution of this work to fault detection and diagnosis, and some recommendations for the future work.

6.2 Summary

The main objective of this research was to develop an efficient fault detection scheme that is able to detect single fault or multiple faults as well as leakage. The proposed artificial neural network-based fault detection scheme consists of two stages. The first stage of the neural network-based fault detection scheme was to design a process estimator. The role of the estimator was to generate residual signal which is a deviation from the desirable “normal” operating condition, which served as inputs for fault classifier. The second stage of the neural network-based fault detection approach was decision making or fault classification. This was accomplished using a neural network classifier. Since the residual vector had different structures for different faults, the classifiers were able to correctly classify the faults.

In this study, the network parameters were initialised by training the network using three different backpropagation algorithms. They were Gradient Descent with Momentum Backpropagation (*traingdm*), Gradient Descent with Momentum and Adaptive Learning Backpropagation (*traingdx*) and Levenberg-Marquardt Backpropagation (*trainlm*). *Trainlm* algorithm was found more efficient and therefore used to train Elman network as process predictor and multilayer feedforward neural network as fault classifier. The effect of addition of inputs with time delay had also been studied.

The proposed scheme was implemented in a Precut column in the fatty acid fractionation plant. The simulation of the column was carried out using HYSYS.Plant, a commercial simulator, having both the steady state and dynamic simulation capabilities. A good match to actual industrial data was obtained.

The process estimator was developed to generate residual signals to be fed to the fault classifier. The process estimator was used to predict the normal operating condition of the process. Both MISO and MIMO models were investigated. The results showed that MISO models were preferable in term of training accuracy and efficiency. In the

case of the fault classifier, multilayer feedforward neural network was used. Both MISO and MIMO models were applied and their performances were evaluated. Again, the results showed that MISO models were also preferable.

The proposed fault detection scheme was applied to detect single and multiple faults as well as stream leakage. Implementation of the proposed scheme was prepared using non noise-corrupted and noise-corrupted measurements data. The promising results were obtained when the proposed scheme implemented with non noise-corrupted measurements. However, different results were achieved when the proposed scheme was implemented in the noisy environment. In the light noisy environment, the proposed scheme was proficient to detect single and multiple faults as well as stream leakage. But, dissimilar outcomes were obtained when the proposed fault detection scheme applied in the mild noisy environment.

6.3 Concluding Remarks

A number of important observations were noted based on the analyses of results as presented in the previous chapter. The main contributions of this research to the fault detection and diagnosis techniques, which also represent the new developments in this field, are the following:

1. The proposed fault detection scheme using artificial neural network had been successfully applied in detection of single fault or simultaneous multiple faults. The scheme was able to detect sensor faults in the top column pressure and bottom column temperature.
2. The proposed neural network fault detection scheme was also able to detect leakage in the stream even if the percentage of the leakage is very small.

3. In the development of the process estimator, the Elman network trained using Levenberg-Marquardt Backpropagation (*trainlm*) algorithm had successfully predicted product compositions in the Precut column in normal condition. The results revealed that MISO model had better generalisation ability compared to MIMO model. Addition of input with time delayed also improved the performance of MISO model.
4. In the development of fault classifiers, the MISO models had been proven to be better than MIMO model. In this case, MISO model was easier and faster to train compared to MIMO model due to the substantially less amount of weight connection involved.
5. Implementation of the fault detection scheme in the noise-corrupted data. The fault detection scheme capable to detect single and multiple faults as well as leakage in the light noisy environment.

6.4 Recommendations for Future Work

The study undertaken here is preliminary towards providing a complete solution to the fault detection and fault diagnosis problem in complex plants. Although this study proved the capability of proposed fault detection scheme, there are still more works required to study various aspects of the approach. To enhance its efficiency, application and robustness, the following works are recommended:

1. Online implementation was not applied in this study. Through online implementation, robustness and efficiencies of the scheme can be evaluated.
2. Capability of the proposed scheme to work effectively in the very noisy environment. In actual process condition, significant process noise can affect the performance of fault detection.
3. Integration of the “fault detection and diagnosis” and the process control system would be most beneficial to the process industries. Investigations should include the efficiency of the fault detection scheme in assisting process control and safety issues. The aim is towards providing a complete package for fault-tolerant control system.

REFERENCES

- Ahmad, A., and Leong, W.H. (2001). Model-based Fault Detection Using Hierarchical Artificial Neural Network. *Regional Symposium on Chemical Engineering*, Bandung, 29-31 Oct.
- Bankhedda, H., and Patton, R.J. (1996). Fault Diagnosis Using Quantitative and Qualitative Knowledge Integration. *Proceeding of UKACC International Conference of Control'96*, 2-5 Sept, Exeter, UK. Vol. 2: 849-854.
- Baughman, D., and Liu, Y. (1995). *Neural Networks in Bioprocessing and Chemical Engineering*, Academic Press, San Diego, CA.
- Changkyu Lee , Sang Wook Choi and In-Beum Lee (2004). Sensor Fault Identification Based on Time-Lagged PCA in Dynamic Processes. *Chemometrics and Intelligent Laboratory Systems*, 70(2): 165-178.
- Chen, J., and Liao, C.M. (2002). Dynamic Process Monitoring Based on Neural Network and PCA. *Journal of Process Control* 12: 277-289.
- Chen, Y.M. and Lee, M.L. (2002). Neural Networks-Based Scheme for System Failure Detection and Diagnosis. *Mathematics and Computers in Simulation*, 58: 101-109.
- Demuth, H., and Beale, M. (2000). *Neural Network Toolbox: For Use with MATLAB User's Guide Version 4*, The MathWorks, Inc.

- Downs, J.J., and Vogel, E.F. (1993). A Plant-wide Industrial Process Control Problem. *Computer and Chemical Engineering*, vol.17, no.3: 245-255.
- Elman, J., (1990). Finding Structure in Time. *Cognitive Science*, 14: 179-211.
- Ferentinos, K.P., and Albright, L.D., (2003). Fault Detection and Diagnosis in Deep-trough Hydroponics using Intelligent Computational Tools. *Biosystems Engineering*. 84 (1): 13-30.
- Ferrada, J.J., M.D. Gordon and I.W. Osborne-Lee (1990). Application of Neural Networks for Fault Diagnosis in Plant Production, AIChE Mtg, San Francisco.
- Frank, P.M. (1990). Fault Diagnosis in Dynamic System Using Analytical and Knowledge-based Redundancy: A Survey and Some New Results. *Automatica*, 26: 459-474.
- Frank, P. M., Ding, S. X., & Marcu, T. (2000). Model-based Fault Diagnosis in Technical Processes. *Transactions of the Institution of Measurement and Control*, 22 (1): 57-101.
- Fuat Doymaz, James Chen, Jose A. Romagnoli and Ahmet Palazoglu (2001). A Robust Strategy for Real-Time Process Monitoring. *Journal of Process Control*, 11(4): 343-359.
- Fuente, M. J. and P. Vega (1999). Neural Networks Applied to Fault Detection of a Biotechnological Process. *Engineering Applications of Artificial Intelligence*, 12(5): 569-584.
- Gomm, J.B. (1996). On-line Learning for Fault Classification Using an Adaptive Neuro-fuzzy Network. *Proceeding of 13th IFAC World Congress*, 30 June-5 July, San Francisco, USA Vol. N: 175-180.

- Himmelblau, D.M. (2000). Applications of Artificial Neural Networks in Chemical Engineering. *Korean Journal of Chemical Engineering*, 17(4): 373-392.
- Himmelblau, D.M. (1978). *Fault Detection and Diagnosis in Chemical Engineering and Petrochemical Processes*, NY: Elsevier.
- Himmelblau, D.M. (1992). Use of Artificial Neural Networks to Monitor Faults and for Troubleshooting in the process Industries. Delaware, USA: *IFAC On-line Fault Detection and Supervision in the Chemical Process Industries*,; 201-206
- Hoskins, J.C, and Himmelblau, D.M. (1988). Artificial Neural Network Models of Knowledge Representation in Chemical Engineering. *Computer and Chemical Engineering*, 12: 881-890.
- Hoskins, J.C., K. M. Kaliyur and D.M. Himmelblau (1989). The Application of Artificial Neural Networks of Fault Diagnosis in Chemical Processing, *AICHE Spring Mtg*, Houston.
- HYSYS.Plant User's Manual*. (2002), Hyprotech Ltd., Calgary, Canada.
- Isermann, R. (1984). Process Fault Detection Based on Modelling and Estimation Methods: A Survey. *Automatica*, 20: 387-404.
- Kramer, M.A. and Leonard, J.A. (1990). Diagnosis using Backpropagation Neural Networks – Analysis and Criticism, *Computers and Chemical Engineering*, 14(12): 1323-1338.
- Kramer, M.A., and Leonard, J.A. (1991). Radial Basis Function Networks for Classifying Process Faults. *IEEE Control System Magazine*, 11(3): 31-38.

- Laser, M. (2000). Recent Safety and Environmental Legislation. *Trans IchemE* 78 (B): 419-422.
- Lennox, B., P. Rutherford, G. A. Montague and C. Haughin (1998). Case Study Investigating the Application of Neural Networks for Process Modelling and Condition Monitoring. *Computers & Chemical Engineering*, 22(11): 1573-1579.
- Lees, F. P. (1996). *Loss Prevention in Process Industries: Hazard Identification, Assessment and Control*. London: Butterworth-Heinemann.
- Leger, R. P., Wm. J. Garland and W. F. S. Poehlman (1998). Fault Detection and Diagnosis using Statistical Control Charts and Artificial Neural Networks. *Artificial Intelligence in Engineering*, 12(1-2): 35-47.
- Li, R., Olson, J.H., and Chester, D.L. (1990). Dynamic Fault Detection and Diagnosis Using Neural Networks, *Computer and Chemical Engineering*, 14(10): 957-971.
- Lou, S.J., Budman, H., and Duever, T.A. (2003). Comparison of Fault Detection Techniques, *Journal of Process Control*, 13: 451-464.
- Luyben, W.L., B.D. Tyreus and M. Luyben. (1998). *Plantwide Process Control*, McGraw-Hill.
- Maki, Y., and Laparo, K.A. (1997). A Neural Network Approach to Fault Detection and Diagnosis in Industrial Processes, *IEEE Transactions on Control System Technology*, 5(6): 529-540.
- Manish Misra, H. Henry Yue, S. Joe Qin and Cheng Ling (2002). Multivariate Process Monitoring and Fault Diagnosis by Multi-Scale PCA. *Computers & Chemical Engineering*, 26(9): 1281-1293.

- McAvoy, T. (2002). Intelligent "Control" Applications in the Process Industries. *Annual Reviews in Control*, 26(1): 75-86.
- McDurff, R.J., and Simpson, P.K. (1990). An Adaptive Resonance Diagnostic System, *Journal of Neural Network Computing*, 2: 19-29.
- McGraw-Hill Economics (1985). *Survey of Investment in Employee Safety and Health*. NY: McGraw-Hill Publishing Co.
- Naidu, S.R., Zafiriou, E., and McAvoy, T.J. (1990). Use of Neural Networks for Sensor Failure Detection in Control System. *IEEE Control System Magazine*, 10: 49-55.
- National Safety Council (1999). *Injury Facts 1999 Edition*, Chicago: National Safety Council.
- Nimmo, I. (1995). Adequately Address Abnormal Situation Operations. *Chemical Engineering Progress* 91 (9): 36-45.
- Nomikos, P., and MacGregor, J.F. (1994). Monitoring Batch Processes Using Multiway Principal Component Analysis. *AIChE Journal*, 40(8): 1361-1373.
- Pareek, V.K., M.P. Brungs, A.A. Adesina, and R. Sharma (2002). Artificial Neural Network Modelling of a Multiphase Photodegradation System, *J. Photochem. Photobiol. A: Chem.* 149: 139-146.
- Parthasarathy Kesavan and Jay H. Lee (2001). A Set Based Approach To Detection and Isolation of Faults in mMultivariable Systems. *Computers & Chemical Engineering*, 25(7-8): 925-940.
- Patton, R.J., Frank, P.M., and Clark, R.N. (1989). *Fault Diagnosis in Dynamic Systems: Theory and Applications*. Prentice Hall, London.

- Patton, R.J., Chen, J., and Siew, T.M. (1994). Fault Diagnosis in Nonlinear Dynamic Systems via Neural Networks. *Proceeding of IEEE International Conference of Control'94*, 21-24 March, Coventry, UK 2: 1346-1351.
- Qin, S.J., and McAvoy, T.J. (1992). Nonlinear PLS Modelling using Neural Networks, *Computers and Chemical Engineering*, 16(4): 379-391.
- Qin, S.J., and Li, W. (2001). Detection and Identification of Faulty Sensors in Dynamic Processes. *AIChE Journal* 47(7): 1581-1593.
- Ralston, P., DePuy, G., and Graham, J.H. (2001). Computer-based Monitoring and Fault Diagnosis: A Chemical Process Case Study. *ISA Transaction* 40: 85-98.
- Rengaswamy, R., and Venkatasubramanian, V. (2000). A Fast Training Neural Network and Its Updation for Incipient Fault Detection and Diagnosis. *Computers and Chemical Engineering* 24(2-7): 431-437.
- Rengaswamy, R., Dinkar Mylaraswamy, K. -E. Årzén and V. Venkatasubramanian (2001). A Comparison of Model-Based and Neural Network-Based Diagnostic Methods. *Engineering Applications of Artificial Intelligence*, 14(6): 805-818.
- Rich, S.H., and Venkatasubramanian, V. (1987). Model-Based Reasoning in Diagnostic Expert System for Chemical Process Plants. *Computers and Chemical Engineering*, 11: 11-22.
- Rozier, D. (2001). A Strategy for Diagnosis Complex Multiple-fault Situations with a Higher Accuracy/Cost Ratio. *Engineering Applications of Artificial Intelligence* 14: 217-227.

- Ruiz, D., Nougues, J.M., and Puigjaner, L. (2001). Fault Diagnosis Support System for Complex Chemical Plant, *Computers and Chemical Engineering*, 25: 151-160.
- Russell, E. L., Chiang, L. H., and Braatz, R. D. (2000). *Data-driven techniques for fault detection and diagnosis in chemical processes*. London: Springer.
- Samanta, B., K. R. Al-Balushi and S. A. Al-Araimi (2003). Artificial Neural Networks and Support Vector Machines with Genetic Algorithm for Bearing Fault Detection. *Engineering Applications of Artificial Intelligence*, 16(7-8): 657-665.
- Scenna, N., S. Benz, B. Drozdowicz, and E. Lamas (2000). A Diagnosis System for Fault Diagnosis in Batch Distillation Columns, ESCAPE10, *Computed Aided Process Engineering*. 8: 805-810.
- Scenna, N. J. (2000). Some Aspects of Fault Diagnosis in Batch Processes. *Reliability Engineering and System Safety*, 70(1): 95-110.
- Seongkyu, Y., and J.F. MacGregor (2001). Fault Diagnosis with Multivariate Statistical Models Part I: Using Steady State Fault Signatures, *Journal of Process Control*, 11(4): 387-400.
- Sharma, R., D. Singhal, R. Ghosh, and A. Dwivedi (1999). Potential Applications of Artificial Neural Networks to Thermodynamics: Vapor-Liquid Equilibrium Predictions. *Computers and Chemical Engineering*, 23: 385-390.
- Sharma, R., Kailash Singh, Diwakar Singhal, and Ranjana Ghosh (2004). Neural Network Application for Detecting Process Faults in Packed Towers, *Chemical Engineering and Processing*, 43(7): 841-847.

- Skoundrianos, E. N., and S. G. Tzafestas (2002). Fault Diagnosis via Local Neural Networks. *Mathematics and Computers in Simulation*, 60(3-5): 69-180.
- Sorsa, T., and Koivo, H.N. (1993). Application of Artificial Neural Network in Process Fault Diagnosis. *Automatica*, 29(4): 843-849.
- Sorsa, T., Koivisto, H., and Koivo, H.N. (1991). Neural Networks in Process Fault Diagnosis. *IEEE Trans. System, Man & Cybernetics.*, 21: 815-825.
- Sourabh Dash and Venkat Venkatasubramanian (2000). Challenges in the Industrial Applications of Fault Diagnostic Systems. *Computers & Chemical Engineering*, 24(2-7): 785-791.
- Tarifa, E., and N. Scenna (1999). A Methodology for Fault Diagnosis in Batch Reactor. *Computers and Chemical Engineering*, 6: 223-226.
- Thomas, P., and Lefebvre, D. (2002). Fault Detection and Isolation in Non-linear System by Using Oversized Neural Networks, *Mathematics and Computers in Simulation* 60: 181-192.
- Tsai, C.S., C.T. Chang, and C.S. Chen (1996). Fault Detection and Diagnosis in Batch and Semi Batch Processes using Artificial Neural Networks. *Chemical Engineering Comm.*, 143: 39-71.
- Tsuge, Y., Hiratsuka, K., Takeda, K., and Matsuyama, H. (2000). A Fault Detection and Diagnosis for the Continuous Process with Load-fluctuations using Orthogonal Wavelets. *Computer and Chemical Engineering* 24: 761-767.
- Tzu-Ming Yeh, Ming-Chieh Huang, and Chi-Tsung Huang (2002). Estimate of Process Compositions and Plant-wide Control from Multiple Secondary Measurements using Artificial Neural Networks. *Computer and Chemical Engineering* 00: 1-18.

- Ungar L.H., B.A. Powell and S.N. Kamens (1990). Adaptive Networks for Fault Diagnosis and Process Control, *Computers and Chemical Engineering*, 14: 561-573.
- Vachhani, P., Rengaswamy, R., and Venkatasubramanian, V. (2001). A Framework for Integrating Diagnostic Knowledge with Nonlinear Optimization for Data Reconciliation and Parameter Estimation in Dynamic Systems. *Chemical Engineering Science*, 56(6): 2133-2148.
- Venkatasubramanian, V., and Chan, K. (1989). A Neural Network Methodology for Process Fault Diagnosis. *AIChE Journal*, 35(12): 1993-2002.
- Venkatasubramanian, V., Vaidynathan, and R., Yamamoto, Y. (1990). Process Fault Detection and Diagnosis Using Neural Networks-I: Steady-state Processes. *Computers and Chemical Engineering*, 14: 699-712.
- Venkatasubramanian, V., R. Rengaswamy, K. Yin and S. N. Kavuri (2003). A Review of Process Fault Detection and Diagnosis: Part I: Quantitative Model-Based Methods. *Computers & Chemical Engineering*, 27(3): 293-311.
- Venkatasubramanian, V., R. Rengaswamy, K. Yin and S. N. Kavuri (2003). A Review of Process Fault Detection and Diagnosis: Part III: Process History Based Methods. *Computers & Chemical Engineering*, 27(3): 327-346.
- Wang, H., Y. Oh, and E. Yoon (1998). Strategies for Modelling and Control of Nonlinear Chemical Processes using Neural Networks. *Computers and Chemical Engineering*, 22: 823-836.
- Watanabe, K., I. Matsuura, M. Abe, M. Kubota and D.M. Himmelblau (1989). Incipient Fault Diagnosis of Chemical Processes via Artificial Neural Networks, *AIChE Journal*, 35: 1803-1812.

- Watanabe, K., Hirota, S., and Hou, L. (1994). Diagnosis of Multiple Simultaneous Fault via Hierarchical Artificial Neural Networks. *AIChE Journal*, 40(5): 301-308.
- Yang, J. C. and David W. Clarke (1999). The Self-Validating Actuator. *Control Engineering Practice*, 7(2): 249-260.
- Ye, N., and Zhao, B. (1996). A Hybrid Intelligent System for Fault Diagnosis of Advanced Manufacturing System. *Int. J. Prod. Res.*, 34(2): 555-576.
- Yu, D.L., Shields, D.N., and Daley, S. (1996). A Hybrid Fault Diagnosis Approach using Neural Networks. *Neural Computing and Application*, 3(4): 21-26.
- Yunbing Huang, Janos Gertler and Thomas J. McAvoy (2000). Sensor and Actuator Fault Isolation by Structured Partial PCA with Nonlinear Extensions. *Journal of Process Control*, 10(5): 459-469.
- Zhang, J., A. J. Morris, E. B. Martin and C. Kiparissides (1999). Estimation of Impurity and Fouling in Batch Polymerisation Reactors through the Application of Neural Networks. *Computers & Chemical Engineering*, 23(3): 301-314.

APPENDIX A1

MAIN PROGRAM FOR TRAINING OF PROCESS PREDICTOR: TOP PRESSURE MISO MODEL

```

clc;
clear;

[input,output,X,min,max]=dprep;
[Vinput,Voutput,X,min,max]=dprepV;
[Tinput,Toutput,X,min,max]=dprepT;

ptr=data; ttr=data(1,:);      % Training
v.P=Vdata; v.T=Vdata(1,:);   % Validation
t.P=Tdata; t.T=Tdata(1,:);   % Testing

S1=5; % Number of nodes
net1=newelm(minmax(data),[S1 1],{'tansig' 'purelin'},'trainlm');
net1.trainparam.epochs=500;          % Max epoch number
net1.trainParam.goal=1e-8;
net1.trainParam.max_fail=10;
net1.trainParam.show=5;
net1=init(net1);
[net1,tr]=train(net1,ptr,ttr,[],[],v,t);

an=sim(net1,data);
error=an-data(1,:);
trainmse=sumsq(error)/X;

Van=sim(net1,Vdata);
valmse=sumsq(Van-Vdata(1,:))/X;

Tan=sim(net1,Tdata);
testmse=sumsq(Tan-Tdata(1,:))/X;

fprintf('TrainMSE=%e, ValMSE=%e, TestMSE=%e\n',trainmse,valmse,testmse);

time=1:X;
figure(1)
subplot(2,1,1),plot(time,an,'r',time,data(1,:),'b');
ylabel('Pressure, kPa');
title('Top Column Pressure Predictor (Training)')

subplot(2,1,2),plot(time,Van,'r',time,Vdata(1,:),'b');
ylabel('Pressure, kPa');
title('Top Column Pressure Predictor (Validation)')
legend('Actual','Predicted',4)

```

APPENDIX A2

MAIN PROGRAM FOR TRAINING OF PROCESS PREDICTOR: BOTTOM TEMPERATURE MISO MODEL

```

clc;
clear;

[input,output,X,min,max]=dprep;
[Vinput,Voutput,X,min,max]=dprepV;
[Tinput,Toutput,X,min,max]=dprepT;

ptr=data; ttr=data(2,:);      % Training
v.P=Vdata; v.T=Vdata(2,:);   % Validation
t.P=Tdata; t.T=Tdata(2,:);   % Testing

S1=5; % Number of nodes
net2=newelm(minmax(data),[S1 1],{'tansig' 'purelin'},'trainlm');
net2.trainparam.epochs=500;      % Max epoch number
net2.trainParam.goal=1e-8;
net2.trainParam.max_fail=10;
net2.trainParam.show=5;
net2=init(net2);
[net2,tr]=train(net2,ptr,ttr,[],[],v,t);

an2=sim(net2,data);
error2=an2-data(2,:);
trainmse=sumsq(error2)/X;

Van2=sim(net2,Vdata);
valmse=sumsq(Van2-Vdata(2,:))/X;

Tan2=sim(net2,Tdata);
testmse=sumsq(Tan2-Tdata(2,:))/X;

fprintf('TrainMSE=%e, ValMSE=%e, TestMSE=%e\n',trainmse,valmse,testmse);

time=1:X;
figure(1)
subplot(2,1,1),plot(time,an2,'r',time,data(2,:),'b');
ylabel('Temperature, oC');
title('Bottom Column Temperature Predictor (Training)')

subplot(2,1,2),plot(time,Van2,'r',time,Vdata(2,:),'b');
ylabel('Temperature, oC');
title('Bottom Column Temperature Predictor (Validation)')
legend('Actual','Predicted',4)

```

APPENDIX A3

MAIN PROGRAM FOR TRAINING OF PROCESS PREDICTOR: C8 FLOWRATE MISO MODEL

```

clc;
clear;

[input,output,X,min,max]=dprep;
[Vinput,Voutput,X,min,max]=dprepV;
[Tinput,Toutput,X,min,max]=dprepT;

ptr=data; ttr=data(3,:); % Training
v.P=Vdata; v.T=Vdata(3,:); % Validation
t.P=Tdata; t.T=Tdata(3,:); % Testing

S1=5; % Number of nodes
net3=newelm(minmax(data),[S1 1],{'tansig' 'purelin'},'trainlm');
net3.trainparam.epochs=500; % Max epoch number
net3.trainParam.goal=1e-8;
net3.trainParam.max_fail=10;
net3.trainParam.show=5;
net3=init(net3);
[net3,tr]=train(net3,ptr,ttr,[],[],v,t);

an3=sim(net3,data);
error3=an3-data(3,:);
trainmse=sumsq(error3)/X;

Van3=sim(net3,Vdata);
valmse=sumsq(Van3-Vdata(3,:))/X;

Tan3=sim(net3,Tdata);
testmse=sumsq(Tan3-Tdata(3,:))/X;

fprintf('TrainMSE=%e, ValMSE=%e, TestMSE=%e\n',trainmse,valmse,testmse);

time=1:X;
figure(1)
subplot(2,1,1),plot(time,an3,'r',time,data(3,:),'b');
ylabel('Flowrate, kg/h');
title('C8 Flowrate Predictor (Training)')

subplot(2,1,2),plot(time,Van3,'r',time,Vdata(3,:),'b');
ylabel('Flowrate, kg/h');
title('C8 Flowrate Predictor (Validation)')
legend('Actual','Predicted',4)

```

APPENDIX A4

MAIN PROGRAM FOR TRAINING OF PROCESS PREDICTOR: MIMO MODEL

```

clc;
clear;

[input,output,X,min,max]=dprep;
[Vinput,Voutput,X,min,max]=dprepV;
[Tinput,Toutput,X,min,max]=dprepT;

ptr=data; ttr=data;          % Training
v.P=Vdata; v.T=Vdata;      % Validation
t.P=Tdata; t.T=Tdata;      % Testing

S1=5; % Number of nodes
net4=newelm(minmax(data),[S1 3],{'tansig' 'purelin'},'trainlm');
net4.trainparam.epochs=500;          % Max epoch number
net4.trainParam.goal=1e-8;
net4.trainParam.max_fail=10;
net4.trainParam.show=5;
net4=init(net4);
[net4,tr]=train(net4,ptr,ttr,[],[],v,t);

an4=sim(net4,data);
error4=an4-data(4,:);
trainmse=sumsq(error4)/X;

Van4=sim(net4,Vdata);
valmse=sumsq(Van4-Vdata(4,:))/X;

Tan4=sim(net4,Tdata);
testmse=sumsq(Tan4-Tdata(4,:))/X;

fprintf('TrainMSE=%e, ValMSE=%e, TestMSE=%e\n',trainmse,valmse,testmse);

time=1:X;
figure(1)
subplot(2,1,1),plot(time,an4,'r',time,data(1,:),'b');
ylabel('Pressure, kPa');
title('Top Column Pressure Predictor (Training)')

subplot(2,1,2),plot(time,Van4,'r',time,Vdata(1,:),'b');
ylabel('Pressure, kPa');
title('Top Column Pressure Predictor (Validation)')
legend('Actual','Predicted',4)

figure(2)
subplot(2,1,1),plot(time,an4,'r',time,data(2,:),'b');
ylabel('Temperature, oC');
title('Bottom Column Temperature Predictor (Training)')

```

```
subplot(2,1,2),plot(time,Van4,'r',time,Vdata(2,:),'b');  
ylabel('Temperature, oC');  
title('Bottom Column Temperature Predictor (Validation)')  
legend ('Actual','Predicted',4)
```

```
figure(3)  
subplot(2,1,1),plot(time,an4,'r',time,data(3,:),'b');  
ylabel('Flowrate, kg/h');  
title('C8 Flowrate Predictor (Training)')
```

```
subplot(2,1,2),plot(time,Van4,'r',time,Vdata(3,:),'b');  
ylabel('Flowrate, kg/h');  
title('C8 Flowrate Predictor (Validation)')  
legend ('Actual','Predicted',4)
```

APPENDIX A5

SUBFUNCTION FOR DATA SCALING

```

function [datas,datad,datad1,p,min,max]=dscale
%DSCALE
%-----
% This subfunction scales data to value between 0 and 1
%
%  datas = scaled data
%  data  = actual data before scaling
%  min   = actual data at their minimum
%  max   = actual data at their maximum

load leak1_5.txt;
input=leak1_5;

[r,m]=size(input);
  refl=input(:,3);      % Reflux flowrate
  cond=input(:,13);    % Condenser flowrate
  pump=input(:,8);     % Pumpharound return flowrate
  toptemp=input(:,6);  % Top Stage Temperature
  dist=input(:,4);     % Distillate flowrate
  bott=input(:,5);     % Bottom flowrate
  feed=input(:,15);    % Feed flowrate
  toppres=input(:,7);  % Top stage pressure
  bottemp=input(:,2);  % Bottom stage temperature
  C8=input(:,11);     % C8 flowrate

  j=r;
for i=1:r
  j=r;
  data(1,i)=refl(j);
  data(2,i)=cond(j);
  data(3,i)=pump(j);
  data(4,i)=toptemp(j);
  data(5,i)=dist(j);
  data(6,i)=bott(j);
  data(7,i)=feed(j);
  data(8,i)=toppres(j);
  data(9,i)=bottemp(j);
  data(10,i)=C8(j);
  r=r-1;
end

```

```
[n,p]=size(data);
for i=1:n
    max(i)=data(i,1);
    min(i)=data(i,1);
    for j=1:p
        if data(i,j)>max(i)
            max(i)=data(i,j);
        end
        if data(i,j)<min(i)
            min(i)=data(i,j);
        end
    end
    datas(i,:)=(data(i,:)-min(i))/(max(i)-min(i));
    datad(i,1:p-1)=datas(i,2:p); % 1 delayed term
    datad1(i,1:p-2)=datas(i,3:p); % 2 delayed term
end
```

APPENDIX A6

SUBFUNCTION FOR DATA PREPARATION (NO DELAYED)

```

function [input,output,X,min,max]=dprep
%DPREP
%-----
% This subfunction creates data for training and cross-validation
%
% input,output = training data
% vinput,voutput = cross-validation data

[datas,datad,datad1,p,min,max]=dscale;
X=p;
% Training data
input(1,1:X)=datas(1,1:X);      % Reflux flowrate
input(2,1:X)=datas(2,1:X);      % Condenser flowrate
input(3,1:X)=datas(3,1:X);      % Pumpharound return flowrate
input(4,1:X)=datas(4,1:X);      % Top stage temperature
input(5,1:X)=datas(5,1:X);      % Distillate flowrate
input(6,1:X)=datas(6,1:X);      % Bottom flowrate
input(7,1:X)=datas(7,1:X);      % Feed flowrate
output(1,1:X)=datas(8,1:X);     % Top stage pressure
output(2,1:X)=datas(9,1:X);     % Bottom stage temperature
output(3,1:X)=datas(10,1:X);    % C8 flowrate

```

APPENDIX A7

SUBFUNCTION FOR DATA PREPARATION (1 DELAYED TERM)

```

function [input,output,X,min,max]=dprep
%DPREP
%-----
% This subfunction creates data for training and cross-validation
%
% input,output = training data
% vinput,voutput = cross-validation data

[datas,datad,datad1,p,min,max]=dscale;
X=p-1;
% Training data
input(1,1:X)=datas(1,1:X);      % Reflux flowrate
input(2,1:X)=datas(2,1:X);      % Condenser flowrate
input(3,1:X)=datas(3,1:X);      % Pumpharound return flowrate
input(4,1:X)=datas(4,1:X);      % Top stage temperature
input(5,1:X)=datas(5,1:X);      % Distillate flowrate
input(6,1:X)=datas(6,1:X);      % Bottom flowrate
input(7,1:X)=datas(7,1:X);      % Feed flowrate
input(8,1:X)=datad(1,1:X);       % Reflux flowrate with 1 delayed term
input(9,1:X)=datad(2,1:X);       % Condenser flowrate with 1 delayed term
input(10,1:X)=datad(3,1:X);      % Pumpharound return flowrate with 1 delayed term
input(11,1:X)=datad(4,1:X);      % Top stage temperature with 1 delayed term
input(12,1:X)=datad(5,1:X);      % Distillate flowrate with 1 delayed term
input(13,1:X)=datad(6,1:X);      % Bottom flowrate with 1 delayed term
input(14,1:X)=datad(7,1:X);      % Feed flowrate with 1 delayed term
output(1,1:X)=datas(8,1:X);      % Top stage pressure
output(2,1:X)=datas(9,1:X);      % Bottom stage temperature
output(3,1:X)=datas(10,1:X);     % C8 flowrate

```

APPENDIX A8

SUBFUNCTION FOR DATA PREPARATION (2 DELAYED TERMS)

```

function [input,output,X,min,max]=dprep
%DPREP
%-----
% This subfunction creates data for training and cross-validation
%
% input,output = training data
% vinput,voutput = cross-validation data

[datas,datad,datad1,p,min,max]=dscale;
X=p-2;
% Training data
input(1,1:X)=datas(1,1:X);      % Reflux flowrate
input(2,1:X)=datas(2,1:X);      % Condenser flowrate
input(3,1:X)=datas(3,1:X);      % Pumpharound return flowrate
input(4,1:X)=datas(4,1:X);      % Top stage temperature
input(5,1:X)=datas(5,1:X);      % Distillate flowrate
input(6,1:X)=datas(6,1:X);      % Bottom flowrate
input(7,1:X)=datas(7,1:X);      % Feed flowrate
input(8,1:X)=datad(1,1:X);      % Reflux flowrate with 1 delayed term
input(9,1:X)=datad(2,1:X);      % Condenser flowrate with 1 delayed term
input(10,1:X)=datad(3,1:X);     % Pumpharound return flowrate with 1 delayed term
input(11,1:X)=datad(4,1:X);     % Top stage temperature with 1 delayed term
input(12,1:X)=datad(5,1:X);     % Distillate flowrate with 1 delayed term
input(13,1:X)=datad(6,1:X);     % Bottom flowrate with 1 delayed term
input(14,1:X)=datad(7,1:X);     % Feed flowrate with 1 delayed term
input(15,1:X)=datad1(1,1:X);    % Reflux flowrate with 2 delayed terms
input(16,1:X)=datad1(2,1:X);    % Condenser flowrate with 2 delayed terms
input(17,1:X)=datad1(3,1:X);    % Pumpharound return flowrate with 2 delayed terms
input(18,1:X)=datad1(4,1:X);    % Top stage temperature with 2 delayed terms
input(19,1:X)=datad1(5,1:X);    % Distillate flowrate with 2 delayed terms
input(20,1:X)=datad1(6,1:X);    % Bottom flowrate with 2 delayed terms
input(21,1:X)=datad1(7,1:X);    % Feed flowrate with 2 delayed terms
output(1,1:X)=datas(8,1:X);     % Top stage pressure
output(2,1:X)=datas(9,1:X);     % Bottom stage temperature
output(3,1:X)=datas(10,1:X);    % C8 flowrate

```

APPENDIX A9

MAIN PROGRAM FOR TRAINING OF F1 CLASSIFIER

```

clc;
clear;

load net;
[input,output,X,min,max]=dprep;
[Vinput,Voutput,X,min,max]=dprepV;

an1=sim(net1,input);
error1=output(1,)-an1;

an2=sim(net2,input);
error2=output(2,)-an2;

an3=sim(net3,input);
error3=output(3,)-an3;

data(1,:)=(error1*(max(8)-min(8))+min(8));
data(2,:)=(error2*(max(9)-min(9))+min(9));
data(3,:)=(error3*(max(10)-min(10))+min(10));

Van1=sim(net1,Vinput);
Verror1=Voutput(1,)-Van1;

Van2=sim(net2,Vinput);
Verror2=Voutput(2,)-Van2;

Van3=sim(net3,Vinput);
Verror3=Voutput(3,)-Van3;

Vdata(1,:)=(Verror1*(max(8)-min(8))+min(8));
Vdata(2,:)=(Verror2*(max(9)-min(9))+min(9));
Vdata(3,:)=(Verror3*(max(10)-min(10))+min(10));

[n,p]=size(data);
min(1)=7.34; max(1)=2;
min(2)=237.3; max(2)=229.3;
min(3)=306; max(3)=75;
for i=1:n
    data(i,:)=(data(i,)-min(i))/(max(i)-min(i));
    Vdata(i,:)=(Vdata(i,)-min(i))/(max(i)-min(i));
end

ptr=data; ttr=data(1,:); % Training
v.P=Vdata; v.T=Vdata(1,:); % Validation

S1=5; % Number of nodes
net7=newff(minmax(data),[S1 1],{'tansig' 'purelin'},'trainlm');
net7.trainparam.epochs=500; % Max epoch number

```

```
net7.trainParam.goal=1e-8;
net7.trainParam.max_fail=10;
net7.trainParam.show=5;
net7=init(net7);
[net7,tr]=train(net7,ptr,ttr,[],[],v);

Fan1=sim(net7,data);
Error1=Fan1-data(1,:);
trainmse1=sumsq(Error1)/X;

Fan2=sim(net7,Vdata);
valmse1=sumsq(Fan2-Vdata(1,:))/X;
fprintf('TrainMSE=%e, ValMSE=%e\n',trainmse1,valmse1);

time=1:X;
upper(1,:)=1;
figure(1)
subplot(2,1,1),plot(time,Fan1,'r',time,data(1,:),'b',time,upper,'k');
% ylim ([0 1]);
ylabel('Pressure Positive Bias');
title('Classifier (Training)')

subplot(2,1,2),plot(time,Fan2,'r',time,Vdata(1,:),'b',time,upper,'k');
ylabel('Pressure Positive Bias');
title('Classifier (Validation)')
```

APEENDIX A10

MAIN PROGRAM FOR TRAINING OF F2 CLASSIFIER

```

clc;
clear;

load net;
[input,output,X,min,max]=dprep;
[Vinput,Voutput,X,min,max]=dprepV;

an1=sim(net1,input);
error1=output(1,)-an1;

an2=sim(net2,input);
error2=output(2,)-an2;

an3=sim(net3,input);
error3=output(3,)-an3;

data(1,:)=(error1*(max(8)-min(8))+min(8));
data(2,:)=(error2*(max(9)-min(9))+min(9));
data(3,:)=(error3*(max(10)-min(10))+min(10));

Van1=sim(net1,Vinput);
Verror1=Voutput(1,)-Van1;

Van2=sim(net2,Vinput);
Verror2=Voutput(2,)-Van2;

Van3=sim(net3,Vinput);
Verror3=Voutput(3,)-Van3;

Vdata(1,:)=(Verror1*(max(8)-min(8))+min(8));
Vdata(2,:)=(Verror2*(max(9)-min(9))+min(9));
Vdata(3,:)=(Verror3*(max(10)-min(10))+min(10));

[n,p]=size(data);
min(1)=7.34; max(1)=2;
min(2)=237.3; max(2)=229.3;
min(3)=306; max(3)=75;
for i=1:n
    data(i,:)=(data(i,)-min(i))/(max(i)-min(i));
    Vdata(i,:)=(Vdata(i,)-min(i))/(max(i)-min(i));
end

ptr=data; ttr=data(1,:); % Training
v.P=Vdata; v.T=Vdata(1,:); % Validation

S1=25; % Number of nodes
net9=newff(minmax(data),[S1 1],{'tansig' 'purelin'},'trainlm');
net9.trainparam.epochs=500; % Max epoch number

```

```

net9.trainParam.goal=1e-8;
net9.trainParam.max_fail=10;
net9.trainParam.show=5;
net9=init(net9);
[net9,tr]=train(net9,ptr,ttr,[],[],v);

Fan1=sim(net9,data);
Error1=Fan1-data(1,:);
trainmse1=sumsq(Error1)/X;

Fan2=sim(net9,Vdata);
valmse1=sumsq(Fan2-Vdata(1,:))/X;
fprintf('TrainMSE=%e, ValMSE=%e\n',trainmse1,valmse1);

time=1:X;
upper(1,:)=1;
figure(1)
subplot(2,1,1),plot(time,Fan1,'r',time,data(1,:),'b',time,upper,'k');
% ylim ([0 1]);
ylabel('Pressure Negative Bias');
title('Classifier (Training)')

subplot(2,1,2),plot(time,Fan2,'r',time,Vdata(1,:),'b',time,upper,'k');
ylabel('Pressure Negative Bias');
title('Classifier (Validation)')

```

APPENDIX A11

MAIN PROGRAM FOR TRAINING OF F3 CLASSIFIER

```

clc;
clear;

load net;
[input,output,X,min,max]=dprep;
[Vinput,Voutput,X,min,max]=dprepV;

an1=sim(net1,input);
error1=output(1,)-an1;

an2=sim(net2,input);
error2=output(2,)-an2;

an3=sim(net3,input);
error3=output(3,)-an3;

data(1,:)=(error1*(max(8)-min(8))+min(8));
data(2,:)=(error2*(max(9)-min(9))+min(9));
data(3,:)=(error3*(max(10)-min(10))+min(10));

Van1=sim(net1,Vinput);
Verror1=Voutput(1,)-Van1;

Van2=sim(net2,Vinput);
Verror2=Voutput(2,)-Van2;

Van3=sim(net3,Vinput);
Verror3=Voutput(3,)-Van3;

Vdata(1,:)=(Verror1*(max(8)-min(8))+min(8));
Vdata(2,:)=(Verror2*(max(9)-min(9))+min(9));
Vdata(3,:)=(Verror3*(max(10)-min(10))+min(10));

[n,p]=size(data);
min(1)=7.34; max(1)=14;
min(2)=237.3; max(2)=247.3;
min(3)=306; max(3)=75;
for i=1:n
    data(i,:)=(data(i,)-min(i))/(max(i)-min(i));
    Vdata(i,:)=(Vdata(i,)-min(i))/(max(i)-min(i));
end

ptr=data; ttr=data(2,:); % Training
v.P=Vdata; v.T=Vdata(2,:); % Validation

S1=25; % Number of nodes
net6=newff(minmax(data),[S1 1],{'tansig' 'purelin'},'trainlm');
net6.trainparam.epochs=500; % Max epoch number

```

```
net6.trainParam.goal=1e-10;
net6.trainParam.max_fail=10;
net6.trainParam.show=5;
net6=init(net6);
[net6,tr]=train(net6,ptr,ttr,[],[],v);

Fan1=sim(net6,data);
Error1=Fan1-data(2,:);
trainmse1=sumsq(Error1)/X;

Fan2=sim(net6,Vdata);
valmse1=sumsq(Fan2-Vdata(2,:))/X;
fprintf('TrainMSE=%e, ValMSE=%e\n',trainmse1,valmse1);

time=1:X;
upper(1,:)=1;
figure(1)
subplot(2,1,1),plot(time,Fan1,'r',time,data(2,:),'b',time,upper,'k');
% ylim ([0 1]);
ylabel('Temperature Positif Bias');
title('Classifier (Training)')

subplot(2,1,2),plot(time,Fan2,'r',time,Vdata(2,:),'b',time,upper,'k');
ylabel('Temperature Positif Bias');
title('Classifier (Validation)')
```

APPENDIX A12

MAIN PROGRAM FOR TRAINING OF F4 CLASSIFIER

```

clc;
clear;

load net;
[input,output,X,min,max]=dprep;
[Vinput,Voutput,X,min,max]=dprepV;

an1=sim(net1,input);
error1=output(1,)-an1;

an2=sim(net2,input);
error2=output(2,)-an2;

an3=sim(net3,input);
error3=output(3,)-an3;

data(1,:)=(error1*(max(8)-min(8))+min(8));
data(2,:)=(error2*(max(9)-min(9))+min(9));
data(3,:)=(error3*(max(10)-min(10))+min(10));

Van1=sim(net1,Vinput);
Verror1=Voutput(1,)-Van1;

Van2=sim(net2,Vinput);
Verror2=Voutput(2,)-Van2;

Van3=sim(net3,Vinput);
Verror3=Voutput(3,)-Van3;

Vdata(1,:)=(Verror1*(max(8)-min(8))+min(8));
Vdata(2,:)=(Verror2*(max(9)-min(9))+min(9));
Vdata(3,:)=(Verror3*(max(10)-min(10))+min(10));

[n,p]=size(data);
min(1)=7.34; max(1)=2;
min(2)=237.3; max(2)=229.3;
min(3)=306; max(3)=75;
for i=1:n
    data(i,:)=(data(i,)-min(i))/(max(i)-min(i));
    Vdata(i,:)=(Vdata(i,)-min(i))/(max(i)-min(i));
end

ptr=data; ttr=data(2,:); % Training
v.P=Vdata; v.T=Vdata(2,:); % Validation

S1=25; % Number of nodes
net8=newff(minmax(data),[S1 1],{'tansig' 'purelin'},'trainlm');
net8.trainparam.epochs=500; % Max epoch number

```

```
net8.trainParam.goal=1e-8;
net8.trainParam.max_fail=10;
net8.trainParam.show=5;
net8=init(net8);
[net8,tr]=train(net8,ptr,ttr,[],[],v);

Fan1=sim(net8,data);
Error1=Fan1-data(2,:);
trainmse1=sumsq(Error1)/X;

Fan2=sim(net8,Vdata);
valmse1=sumsq(Fan2-Vdata(2,:))/X;
fprintf('TrainMSE=%e, ValMSE=%e\n',trainmse1,valmse1);

time=1:X;
upper(1,:)=1;
figure(1)
subplot(2,1,1),plot(time,Fan1,'r',time,data(2,:),'b',time,upper,'k');
% ylim ([0 1]);
ylabel('Temperature Negative Bias');
title('Classifier (Training)')

subplot(2,1,2),plot(time,Fan2,'r',time,Vdata(2,:),'b',time,upper,'k');
ylabel('Temperature Negative Bias');
title('Classifier (Validation)')
```

APPENDIX A13

MAIN PROGRAM FOR TRAINING OF F5 CLASSIFIER

```

clc;
clear;

load net;
[input,output,X,min,max]=dprep;
[Vinput,Voutput,X,min,max]=dprepV;

an1=sim(net1,input);
error1=output(1,)-an1;

an2=sim(net2,input);
error2=output(2,)-an2;

an3=sim(net3,input);
error3=output(3,)-an3;

data(1,:)=(error1*(max(8)-min(8))+min(8));
data(2,:)=(error2*(max(9)-min(9))+min(9));
data(3,:)=(error3*(max(10)-min(10))+min(10));

Van1=sim(net1,Vinput);
Verror1=Voutput(1,)-Van1;

Van2=sim(net2,Vinput);
Verror2=Voutput(2,)-Van2;

Van3=sim(net3,Vinput);
Verror3=Voutput(3,)-Van3;

Vdata(1,:)=(Verror1*(max(8)-min(8))+min(8));
Vdata(2,:)=(Verror2*(max(9)-min(9))+min(9));
Vdata(3,:)=(Verror3*(max(10)-min(10))+min(10));

[n,p]=size(data);
min(1)=7.34; max(1)=14;
min(2)=237.3; max(2)=247.3;
min(3)=306; max(3)=75;
for i=1:n
    data(i,:)=(data(i,)-min(i))/(max(i)-min(i));
    Vdata(i,:)=(Vdata(i,)-min(i))/(max(i)-min(i));
end

ptr=data; ttr=data(3,:); % Training
v.P=Vdata; v.T=Vdata(3,:); % Validation

S1=5; % Number of nodes
net4=newff(minmax(data),[S1 1],{'tansig' 'purelin'},'trainlm');
net4.trainparam.epochs=300; % Max epoch number

```

```
net4.trainParam.goal=1e-6;
net4.trainParam.max_fail=10;
net4.trainParam.show=5;
net4=init(net4);
[net4,tr]=train(net4,ptr,ttr,[],[],v);

Fan1=sim(net4,data);
Error1=Fan1-data(3,:);
trainmse1=sumsq(Error1)/X;

Fan2=sim(net4,Vdata);
valmse1=sumsq(Fan2-Vdata(3,:))/X;
fprintf('TrainMSE=%e, ValMSE=%e\n',trainmse1,valmse1);

time=1:X;
upper(1,:)=1;
figure(1)
subplot(2,1,1),plot(time,Fan1,'r',time,data(3,:),'b',time,upper,'k');
% ylim ([0 1]);
ylabel('Leak'); xlabel('data set')
title('Classifier (Training)')

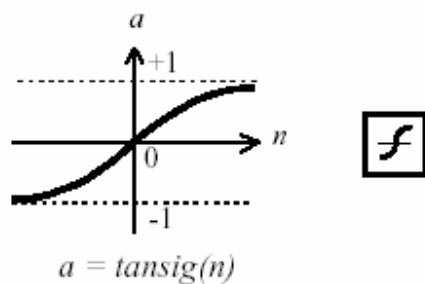
subplot(2,1,2),plot(time,Fan2,'r',time,Vdata(3,:),'b',time,upper,'k');
ylabel('Leak'); xlabel('data set')
title('Classifier (Validation)')
```

APPENDIX B1

Description of Tansig

Hyperbolic tangent sigmoid transfer function

Graph and Symbol



Tan-Sigmoid Transfer Function

Syntax

$A = \text{tansig}(N)$

$\text{info} = \text{tansig}(\text{code})$

Description

`tansig` is a transfer function. Transfer functions calculate a layer's output from its net input.

`tansig(N)` takes one input,

N - $S \times Q$ matrix of net input (column) vectors.

and returns each element of N squashed between -1 and 1.

`tansig(code)` return useful information for each code string:

- 'deriv' - Name of derivative function.
- 'name' - Full name.
- 'output' - Output range.

- 'active' - Active input range.

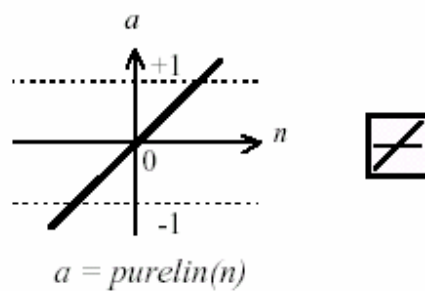
tansig is named after the hyperbolic tangent, which has the same shape. However, tanh may be more accurate and is recommended for applications that require the hyperbolic tangent.

APPENDIX B2

Description of Purelin

Linear transfer function

Graph and Symbol



Linear Transfer Function

Syntax

$A = \text{purelin}(N)$

$\text{info} = \text{purelin}(\text{code})$

Description

`purelin` is a transfer function. Transfer functions calculate a layer's output from its net input.

`purelin(N)` takes one input,

N - $S \times Q$ matrix of net input (column) vectors.

and returns N .

`purelin(code)` returns useful information for each code string:

- 'deriv' - Name of derivative function.
- 'name' - Full name.
- 'output' - Output range.
- 'active' - Active input range.

APPENDIX B3

Batch Gradient Descent with Momentum (traingdm)

This is batch algorithm for neural networks that often provides faster convergence - traingdm, steepest descent with momentum. Momentum allows a network to respond not only to the local gradient, but also to recent trends in the error surface. Acting like a low-pass filter, momentum allows the network to ignore small features in the error surface. Without momentum a network may get stuck in a shallow local minimum. With momentum a network can slide through such a minimum.

Momentum can be added to backpropagation learning by making weight changes equal to the sum of a fraction of the last weight change and the new change suggested by the backpropagation rule. The magnitude of the effect that the last weight change is allowed to have is mediated by a momentum constant, mc , which can be any number between 0 and 1. When the momentum constant is 0, a weight change is based solely on the gradient. When the momentum constant is 1, the new weight change is set to equal the last weight change and the gradient is simply ignored. The gradient is computed by summing the gradients calculated at each training example, and the weights and biases are only updated after all training examples have been presented.

If the new performance function on a given iteration exceeds the performance function on a previous iteration by more than a predefined ratio `max_perf_inc` (typically 1.04), the new weights and biases are discarded, and the momentum coefficient mc is set to zero.

The batch form of gradient descent with momentum is invoked using the training function `traingdm`.

APPENDIX B4

Gradient Descent with Momentum and Adaptive Learning Backpropagation (traingdx)

With standard steepest descent, the learning rate is held constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate. If the learning rate is set too high, the algorithm may oscillate and become unstable. If the learning rate is too small, the algorithm will take too long to converge. It is not practical to determine the optimal setting for the learning rate before training, and, in fact, the optimal learning rate changes during the training process, as the algorithm moves across the performance surface.

The performance of the steepest descent algorithm can be improved if we allow the learning rate to change during the training process. An adaptive learning rate will attempt to keep the learning step size as large as possible while keeping learning stable. The learning rate is made responsive to the complexity of the local error surface.

An adaptive learning rate requires some changes in the training procedure used by traingdm. First, the initial network output and error are calculated. At each epoch new weights and biases are calculated using the current learning rate. New outputs and errors are then calculated.

As with momentum, if the new error exceeds the old error by more than a predefined ratio max_perf_inc (typically 1.04), the new weights and biases are discarded. In addition, the learning rate is decreased (typically by multiplying by $\text{lr_dec} = 0.7$). Otherwise, the new weights, etc., are kept. If the new error is less than the old error, the learning rate is increased (typically by multiplying by $\text{lr_inc} = 1.05$).

This procedure increases the learning rate, but only to the extent that the network can learn without large error increases. Thus, a near-optimal learning rate is obtained for the local terrain. When a larger learning rate could result in stable learning, the learning rate is increased. When the learning rate is too high to guarantee a decrease in error, it gets decreased until stable learning resumes.

Backpropagation training with an adaptive learning rate is implemented with the function `traingdx`, which is called just like `traingdm`, except for the additional training parameters `max_perf_inc`, `lr_dec`, and `lr_inc`. The function `traingdx` combines adaptive learning rate with momentum training.

APPENDIX B5

Levenberg-Marquardt Backpropagation (trainlm)

The Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feedforward networks), then the Hessian matrix can be approximated as

$$H = J^T J$$

and the gradient can be computed as

$$g = J^T e$$

where J is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and e is a vector of network errors. The Jacobian matrix can be computed through a standard backpropagation technique that is much less complex than computing the Hessian matrix.

The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e$$

When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift towards Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm.

The training parameters for trainlm are epochs, show, goal, time, min_grad, max_fail, mu, mu_dec, mu_inc, mu_max, mem_reduc. The parameter mu is the initial

value for μ . This value is multiplied by `mu_dec` whenever the performance function is reduced by a step. It is multiplied by `mu_inc` whenever a step would increase the performance function. If `mu` becomes larger than `mu_max`, the algorithm is stopped. The parameter `mem_reduc` is used to control the amount of memory used by the algorithm. This algorithm appears to be the fastest method for training moderate-sized feedforward neural networks (up to several hundred weights). It also has a very efficient MATLAB implementation, since the solution of the matrix equation is a built-in function, so its attributes become even more pronounced in a MATLAB setting.

APPENDIX B6

UNIQUAC

The **UNIQUAC** (Universal Quasi Chemical) equation proposed by Abrams and Prausnitz in 1975 uses statistical mechanics and the quasichemical theory of Guggenheim to represent the liquid structure. The equation is capable of representing LLE, VLE and VLLE with accuracy, but without the need for a nonrandomness factor. The **UNIQUAC** equation is significantly more detailed and sophisticated than any of the other activity models. Its main advantage is that a good representation of both VLE and LLE can be obtained for a large range of non-electrolyte mixtures using only two adjustable parameters per binary. The fitted parameters usually exhibit a smaller temperature dependence which makes them more valid for extrapolation purposes.

The **UNIQUAC** equation utilizes the concept of local composition as proposed by Wilson. Since the primary concentration variable is a surface fraction as opposed to a mole fraction, it is applicable to systems containing molecules of very different sizes and shape, such as polymer solutions. The **UNIQUAC** equation can be applied to a wide range of mixtures containing H₂O, alcohols, nitriles, amines, esters, ketones, aldehydes, halogenated hydrocarbons and hydrocarbons. HYSYS contains the following four-parameter extended form of the **UNIQUAC** equation. The four adjustable parameters for the **UNIQUAC** equation in HYSYS are the a_{ij} and a_{ji} terms (temperature independent), and the b_{ij} and b_{ji} terms (temperature dependent). The equation will use parameter values stored in HYSYS or any user supplied value for further fitting the equation to a given set of data.

$$\ln \gamma_i = \ln \left(\frac{\phi_i}{x_i} \right) + 0.5Zq_i \ln \left(\frac{\theta_i}{\phi_i} \right) + L_i - \left(\frac{\theta_i}{\phi_i} \right) \sum_{j=1}^n L_j x_j + q_i \left(1.0 - \ln \sum_{j=1}^n \theta_j \tau_{ji} \right) - q_i \sum_{j=1}^n \left(\frac{\theta_j \tau_{ij}}{\sum_{k=1}^n \theta_k \tau_{kj}} \right)$$

where: γ_i = activity coefficient of component i

x_i = mole fraction of component i

T = temperature (K)

n = total number of components

$$L_j = 0.5Z(r_j - q_j) - r_j + 1$$

$$\theta_i = \frac{q_i x_i}{\sum q_j x_j}$$

$$\tau_{ij} = \exp\left[-\frac{a_{ij} + b_{ij}T}{RT}\right]$$

$Z = 10.0$ co-ordination number

a_{ij} = non-temperature dependent energy parameter between components

i and j (cal/gmol)

b_{ij} = temperature dependent energy parameter between components i

and j (cal/gmol-K)

q_i = van der Waals area parameter - $A_{w_i}/(2.5e9)$

A_w = van der Waals area

r_i = van der Waals volume parameter - $V_{w_i}/(15.17)$

V_w = van der Waals volume

APPENDIX C

No. of Hidden Nodes	TRAINGDM		TRAINGDX		TRAINLM	
	Training Error	Validation Error	Training Error	Validation Error	Training Error	Validation Error
5	4.077936e-2	4.146371e-2	2.714509e-2	2.668403e-2	2.033593e-2	1.150226e-2
6	1.960384e-1	6.320533e-1	1.885380e-2	1.619954e-2	3.831442e-3	1.117903e-2
7	5.994495e-2	9.738402e-2	1.087403e-2	3.197743e-2	5.139687e-3	2.565639e-2
8	8.738046e-2	2.173743e-1	1.334813e-2	3.259475e-2	3.034932e-3	4.205496e-2
9	6.099058e-1	7.015554e-1	1.361482e-2	5.173155e-2	3.209633e-3	1.437706e-2
10	1.097615e-1	1.231374e-1	2.335807e-2	3.729337e-2	2.147424e-3	1.576162e-2
11	4.445715e-2	1.043836e-1	1.368673e-2	2.687253e-2	5.328705e-3	1.835738e-2
12	1.409678e-1	2.608809e-1	9.963393e-3	2.714422e-2	4.800347e-3	7.332069e-3
13	3.191961e-1	4.477899e-1	1.023495e-2	2.503431e-2	6.155818e-3	1.235041e-2
14	2.276566e-1	5.677477e-1	9.490088e-3	2.264961e-2	6.923769e-3	1.715382e-2
15	1.485612e-1	9.500044e-2	8.055003e-3	2.187585e-2	2.123856e-3	3.493085e-2
16	9.226687e-2	1.816993e-1	6.431652e-2	5.813904e-2	2.589559e-3	2.459820e-2
17	4.086714e-1	2.179816e-1	1.259948e-2	2.044586e-2	2.105830e-3	3.457715e-2
18	1.977091e-1	1.028668e-1	7.971603e-2	5.145028e-2	4.491554e-3	2.614507e-2
19	1.152908e-1	1.908080e-1	9.541068e-3	1.907367e-2	2.620040e-3	1.993369e-2
20	8.873883e-1	7.244756e-1	4.377993e-2	6.621480e-2	4.800747e-3	2.292373e-2
21	1.243773e-1	1.278531e-1	7.515338e-3	2.259664e-2	3.795192e-3	2.242983e-2
22	1.092900e-1	1.703446e-1	8.133094e-3	2.301014e-2	5.815528e-3	4.643484e-2
23	3.194726e-1	6.210042e-1	6.896200e-3	3.019960e-2	2.534504e-3	3.572799e-2
24	1.137607e-1	5.884579e-2	5.200857e-2	5.623890e-2	1.711280e-3	4.682936e-2
25	1.111413e-1	1.283433e-1	1.536788e-2	4.163787e-2	3.211102e-3	4.981050e-2

APPENDIX C

trainlm

No. of Hidden Nodes	NO DELAYED TERM		1 DELAYED TERM		2 DELAYED TERMS	
	Training Error	Validation Error	Training Error	Validation Error	Training Error	Validation Error
5	2.033593e-2	1.150226e-2	2.846111e-2	2.625636e-2	2.696728e-3	9.741325e-3
6	3.831442e-3	1.117903e-2	5.730432e-3	5.870323e-3	6.864018e-4	6.550493e-2
7	5.139687e-3	2.565639e-2	1.175781e-2	4.386275e-2	9.568673e-4	2.414696e-2
8	3.034932e-3	4.205496e-2	1.060145e-3	9.500084e-3	6.825227e-3	2.963047e-2
9	3.209633e-3	1.437706e-2	8.974037e-4	1.878614e-2	1.358239e-3	2.407328e-2
10	2.147424e-3	1.576162e-2	1.187160e-3	1.409885e-2	3.619941e-2	1.472941e-2
11	5.328705e-3	1.835738e-2	2.759800e-3	7.103157e-3	2.342017e-2	2.301196e-2
12	4.800347e-3	7.332069e-3	2.115252e-3	3.519536e-2	1.140377e-3	9.562185e-3
13	6.155818e-3	1.235041e-2	9.890669e-4	1.611668e-2	1.314896e-2	1.309807e-2
14	6.923769e-3	1.715382e-2	4.214622e-3	1.358827e-2	2.029408e-4	3.226269e-2
15	2.123856e-3	3.493085e-2	2.179702e-3	2.350473e-2	1.357324e-2	1.277784e-2
16	2.589559e-3	2.459820e-2	2.535576e-3	7.909181e-3	1.064376e-3	3.677664e-2
17	2.105830e-3	3.457715e-2	3.119867e-3	1.138822e-2	1.311161e-3	3.319353e-2
18	4.491554e-3	2.614507e-2	1.665088e-3	3.036091e-2	5.901860e-3	4.683696e-3
19	2.620040e-3	1.993369e-2	5.598165e-4	2.032893e-2	1.415379e-2	1.347251e-2
20	4.800747e-3	2.292373e-2	8.198696e-3	9.121664e-3	6.561103e-3	7.580858e-3
21	3.795192e-3	2.242983e-2	1.436512e-3	1.898330e-2	4.459425e-4	3.899606e-2
22	5.815528e-3	4.643484e-2	2.912506e-3	1.352630e-2	1.113746e-2	4.049055e-2
23	2.534504e-3	3.572799e-2	1.061734e-2	9.186592e-3	1.431521e-2	1.936342e-2
24	1.711280e-3	4.682936e-2	1.249624e-3	1.263992e-2	2.817406e-2	1.730887e-2
25	3.211102e-3	4.981050e-2	3.886777e-3	1.906357e-2	2.399016e-3	9.664076e-3

APPENDIX E

trainm**trainm***trainlm*

No. of Hidden Nodes	TOP STAGE PRESSURE		BOTTOM STAGE TEMPERATURE		C8 FLOWRATE	
	Training Error	Validation Error	Training Error	Validation Error	Training Error	Validation Error
5	2.696728e-3	9.741325e-3	1.965789e-4	1.098152e-3	5.630093e-3	6.850242e-3
6	6.864018e-4	6.550493e-2	8.316765e-5	4.886944e-3	2.176698e-3	4.759817e-3
7	9.568673e-4	2.414696e-2	1.875507e-4	1.097067e-3	6.022385e-4	6.841415e-3
8	6.825227e-3	2.963047e-2	4.891176e-4	3.619773e-3	5.118487e-3	1.419495e-2
9	1.358239e-3	2.407328e-2	2.836350e-4	1.870291e-3	5.088225e-4	5.527025e-3
10	3.619941e-2	1.472941e-2	1.186632e-4	2.175429e-3	1.369624e-3	4.302835e-3
11	2.342017e-2	2.301196e-2	8.301194e-4	6.113405e-3	2.721438e-3	2.791515e-3
12	1.140377e-3	9.562185e-3	1.327573e-4	3.221621e-3	2.707649e-3	7.579502e-3
13	1.314896e-2	1.309807e-2	2.418482e-4	3.022892e-3	2.668948e-3	3.587602e-3
14	2.029408e-4	3.226269e-2	3.541857e-4	1.268764e-3	1.267853e-3	5.266927e-3
15	1.357324e-2	1.277784e-2	1.040967e-4	4.482436e-3	2.458897e-3	4.459251e-3
16	1.064376e-3	3.677664e-2	1.672519e-4	1.481734e-3	5.001075e-3	2.526958e-2
17	1.311161e-3	3.319353e-2	7.660308e-4	1.175150e-2	1.647529e-3	6.635936e-3
18	5.901860e-3	4.683696e-3	1.991235e-4	4.323552e-3	3.439583e-3	1.513190e-2
19	1.415379e-2	1.347251e-2	4.435661e-5	3.501583e-3	1.367297e-3	6.753807e-3
20	6.561103e-3	7.580858e-3	1.193494e-4	2.416865e-3	9.460975e-4	2.664814e-2
21	4.459425e-4	3.899606e-2	4.407482e-5	3.570845e-3	4.500387e-3	7.815148e-3
22	1.113746e-2	4.049055e-2	7.708730e-5	3.862406e-3	5.695127e-3	5.501059e-2
23	1.431521e-2	1.936342e-2	1.477254e-3	1.695642e-2	1.733385e-3	2.497227e-2
24	2.817406e-2	1.730887e-2	1.109466e-3	3.392022e-3	2.296181e-2	1.991131e-2
25	2.399016e-3	9.664076e-3	1.571372e-3	5.529578e-3	1.335754e-4	1.216002e-2

APPENDIX C

trainlm

No. of Hidden Nodes	NO DELAY		1 DELAYED TERM		2 DELAYED TERMS	
	Training Error	Validation Error	Training Error	Validation Error	Training Error	Validation Error
5	2.968019e-2	1.521266e-1	4.806578e-1	5.687381e-1	1.305059e-1	6.994347e-1
6	3.805614e-2	7.927428e-2	8.751078e-2	1.667295e-1	1.783850e-2	8.482157e-2
7	1.205707e-2	1.122480e-1	1.561151e-1	1.424303e-1	4.949681e-1	2.510704e-1
8	3.229455e-2	1.165517e-1	3.847091e-2	7.374770e-2	2.782199e-2	2.140691e-1
9	2.953485e-2	3.678754e-2	6.739390e-2	2.368342e-1	1.175401e-2	6.774574e-1
10	3.567543e-2	1.566688e-1	1.195253e-2	4.032183e-1	9.637837e-2	9.260911e-1
11	2.656183e-2	5.535806e-2	4.644836e-2	1.188381e-1	7.328746e-2	7.301820e-1
12	1.364887e-2	5.319744e-2	8.539785e-3	2.680599e-1	8.151640e-2	4.505416e-1
13	2.309238e-2	1.128832e-1	1.372395e-1	1.677345e-1	9.874390e-2	4.479072e-2
14	2.185592e-2	4.806495e-2	9.777221e-2	2.509824e-1	4.210368e-2	2.647319e-1
15	1.291797e-2	6.451672e-2	2.420675e-1	2.719128e-1	1.197150	1.595480
16	2.354372e-2	2.256938e-1	2.067485e-1	4.293077e-1	7.749002e-2	5.681614e-1
17	3.429155e-2	7.889348e-2	1.949006e-3	4.638980e-1	3.620832e-2	2.487301e-1
18	4.562694e-3	3.170314e-1	1.459105e-2	1.163605e-1	2.251157	1.078655
19	2.793945e-2	4.069720e-2	6.658193e-3	9.157891e-2	1.803804e-1	2.543552e-1
20	8.929397e-3	9.007075e-2	5.190752e-3	1.866692e-1	3.096745e-1	4.167288e-1
21	1.273386e-2	5.162294e-2	7.141213e-2	2.104655e-1	8.798952e-2	4.505040e-1
22	3.351622e-2	7.405724e-2	1.172064e-1	3.915862e-1	6.465915e-2	3.481933e-1
23	7.061361e-3	8.604899e-2	1.435992e-2	2.626484e-1	5.213728e-1	6.293648e-1
24	1.152028e-2	4.692901e-2	1.153342e-3	1.900537e-1	1.846885e-1	1.551082e-1
25	1.023521e-2	9.163373e-2	4.232230e-3	1.728930e-1	1.466460e-2	1.527744e-1

APPENDIX 6

Fig 6

Table

No. of Hidden Nodes	PRESSURE +VE BIAS (F1)		PRESSURE -VE BIAS (F2)		TEMP +VE BIAS (F3)	
	Training Error	Validation Error	Training Error	Validation Error	Training Error	Validation Error
5	9.966888e-7	5.783098e-5	3.435874e-4	1.579746e-3	8.394299e-5	7.703526e-4
6	1.455510e-7	1.450385e-3	1.222341e-4	1.357435e-3	1.375538e-7	2.261510e-5
7	8.446317e-4	2.175329e-3	9.984730e-6	4.747119e-4	2.939014e-3	9.940738e-4
8	5.147036e-4	1.248019e-3	2.754508e-7	5.150159e-5	2.232254e-7	1.123182e-3
9	3.810043e-4	1.730661e-4	3.754778e-5	2.637293e-4	2.217236e-7	5.427224e-4
10	4.051283e-4	7.579846e-4	5.542196e-5	1.343673e-4	4.183906e-5	7.249715e-4
11	5.808353e-5	2.178724e-3	5.956948e-4	2.679779e-4	3.769675e-7	5.430902e-4
12	9.921367e-7	3.830433e-4	1.932581e-3	3.072961e-3	3.230719e-7	3.582025e-4
13	4.092007e-6	1.132508e-3	1.932156e-4	4.256156e-4	3.788977e-7	2.789952e-3
14	5.620355e-6	2.193887e-3	4.465895e-6	9.614703e-4	4.247597e-5	2.113569e-3
15	4.029854e-6	2.081015e-3	1.006866e-5	1.764451e-3	1.388188e-6	3.264229e-3
16	1.374466e-5	1.957395e-4	3.216443e-5	5.924039e-4	2.133248e-4	9.766941e-4
17	7.161799e-6	3.489002e-3	5.660963e-5	9.785437e-4	1.070705e-5	1.918713e-4
18	9.991313e-7	2.084647e-4	1.122170e-7	4.134357e-4	3.643976e-4	9.098080e-3
19	2.548906e-4	6.935795e-4	1.795717e-7	7.542417e-3	5.713351e-6	2.991194e-4
20	2.484196e-6	2.625356e-3	5.052386e-6	6.765520e-4	6.617377e-5	3.278907e-3
21	2.445318e-6	3.008138e-4	4.863260e-6	1.488151e-3	9.803106e-6	2.829310e-3
22	2.306503e-7	6.049951e-3	8.584135e-6	6.642563e-4	9.927655e-6	2.659332e-5
23	7.136626e-5	1.430984e-3	2.499883e-7	7.463891e-4	2.309424e-6	4.618420e-4
24	5.708095e-6	7.172891e-4	4.078760e-4	4.000056e-4	1.259025e-5	2.331668e-3
25	1.211316e-5	1.701261e-3	5.973007e-5	5.854936e-4	1.355976e-6	1.939828e-3

EJAO

EJAO

No. of Hidden Nodes	TEMP -VE BIAS (F4)		LEAKAGE (F5)	
	Training Error	Validation Error	Training Error	Validation Error
5	4.316692e-7	6.367534e-6	2.353716e-5	1.109019e-5
6	5.203206e-4	1.749702e-3	3.839039e-5	1.024365e-3
7	2.522086e-7	1.837451e-5	8.834487e-5	3.527061e-3
8	6.714939e-7	1.009699e-3	4.875869e-5	1.178918e-4
9	9.915633e-7	9.701977e-5	4.784626e-3	4.214639e-3
10	5.557870e-6	1.008873e-4	5.114614e-5	2.771619e-3
11	9.526666e-7	1.349404e-3	1.366973e-4	5.765655e-3
12	8.572062e-7	1.428610e-5	3.005566e-4	3.676785e-4
13	8.106843e-4	5.319330e-3	1.312997e-4	7.731330e-5
14	1.842753e-5	1.871585e-3	8.323252e-3	2.466752e-3
15	2.816762e-5	3.631095e-3	1.227990e-4	2.201055e-4
16	6.502516e-7	9.795131e-4	8.468933e-5	2.129597e-3
17	6.780591e-4	2.446567e-3	4.240486e-6	2.444376e-4
18	1.025415e-4	8.825681e-3	1.207543e-6	6.352933e-4
19	9.063623e-4	4.737836e-3	8.289171e-4	1.399145e-3
20	1.112931e-5	9.888117e-3	7.987803e-3	1.331796e-3
21	1.489123e-6	1.937513e-4	7.206751e-5	1.715392e-4
22	3.071433e-4	7.169766e-3	1.094338e-5	4.752952e-3
23	7.689886e-6	6.041562e-3	1.143766e-3	2.282391e-4
24	1.022670e-7	5.599486e-4	1.520425e-4	7.900378e-3
25	3.144106e-4	8.502013e-3	5.824322e-4	1.190514e-3

APPENDIX C

MIMO

MIMO

No. of Hidden Nodes	MIMO CLASSIFIER	
	Training Error	Validation Error
5	9.098259e-3	6.478957e-3
6	2.552751e-4	7.794624e-3
7	1.391457e-4	6.443764e-3
8	4.844969e-3	8.148053e-3
9	1.807678e-3	5.331252e-3
10	5.394608e-3	8.967411e-3
11	5.109709e-4	7.596552e-3
12	5.338829e-4	5.651899e-3
13	4.122663e-3	6.277964e-3
14	1.588596e-4	5.200906e-3
15	2.296811e-3	5.458320e-3
16	1.789994e-3	9.915375e-3
17	2.530811e-3	6.308742e-3
18	3.762146e-3	5.957535e-3
19	3.411133e-3	6.851991e-3
20	2.035114e-3	6.238214e-3
21	1.415152e-3	7.987124e-3
22	9.873412e-3	5.729677e-3
23	8.014766e-4	9.970874e-3
24	2.977905e-3	5.954816e-3
25	1.475434e-4	6.931723e-3

APPENDIX D

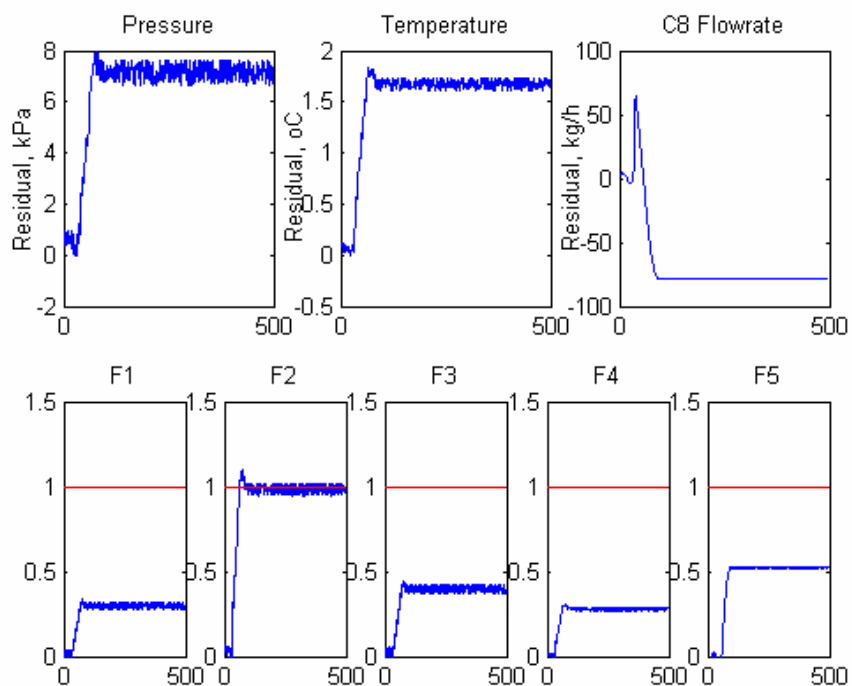


Figure D1 Fault F2 with 1% noise

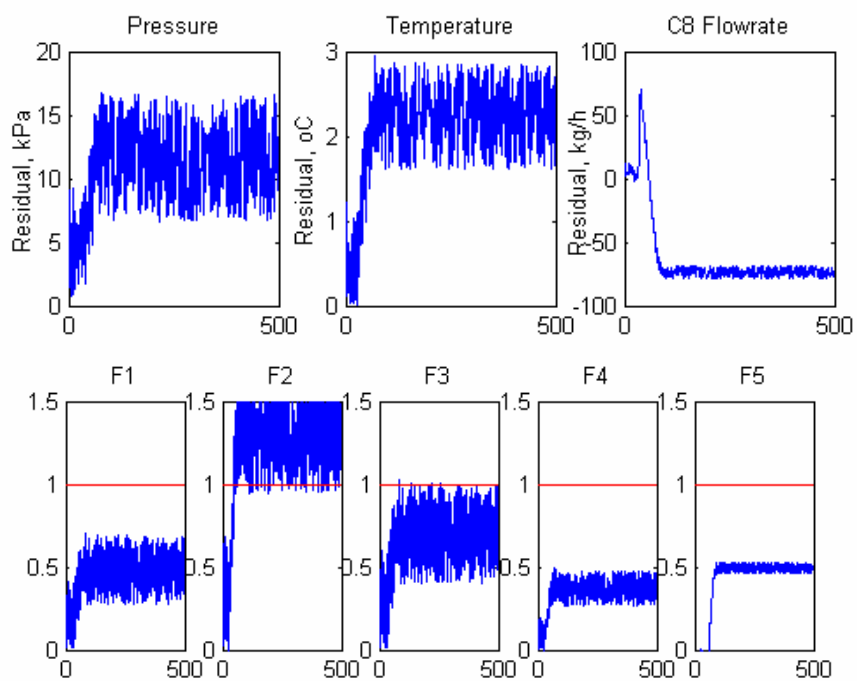


Figure D2 Fault F2 with 10% noise

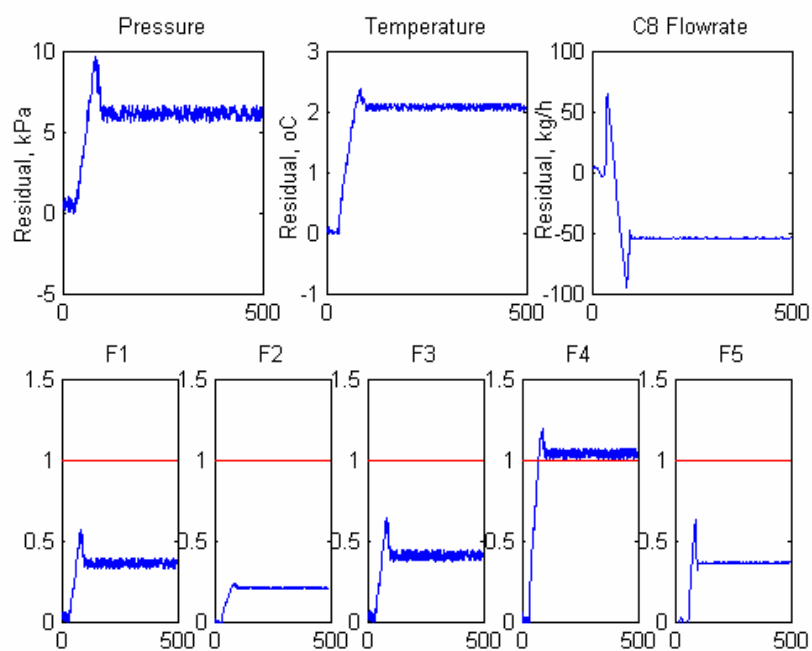


Figure D3 Fault F4 with 1% noise

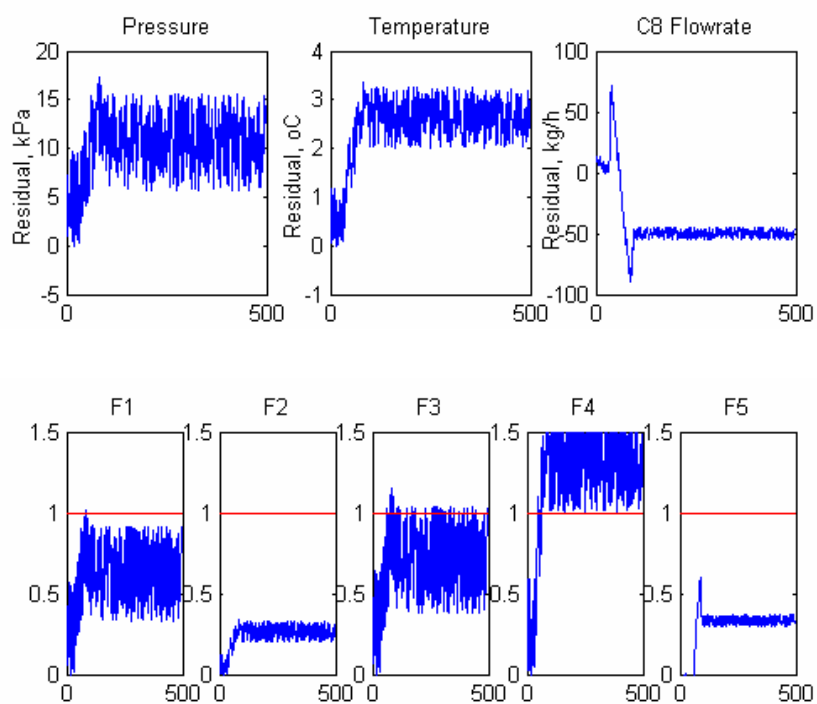


Figure D4 Fault F4 with 10% noise