

Design and Performance Study of Self-Tuning Fuzzy and Neuro-Fuzzy Control Systems

Thesis Submitted

By

Arabinda Kumar Pal

Doctor of Philosophy (Engineering)

Department of Instrumentation & Electronics Engineering

Faculty Council of Engineering & Technology

Jadavpur University

Kolkata 700032

India

October, 2017

JADAVPUR UNIVERSITY

KOLKATA – 700 032, INDIA

Index no: 83/10/E

Title of the Thesis	Design and Performance Study of Self-Tuning Fuzzy and Neuro-Fuzzy Control Systems
Name, Designation & Institution of the Supervisor	Prof. Rajani Kanta Mudi Professor Department of Instrumentation and Electronics Engineering Jadavpur University Salt Lake Campus Kolkata –700 098 West Bengal, India

List of Publication:

Papers published in National / International Journals:

- [1] A. K. Pal and R. K. Mudi, “An adaptive PD-type FLC and its real time implementation to overhead crane control”, *International Journal of Emerging Technologies in Computational and Applied Sciences*, vol.6(2), pp.178-183, 2013.
- [2] A. K. Pal and I. Naskar, “Design of self-tuning fuzzy PI controller in LABVIEW for control of real time process”, *International Journal of Electronics and Computer Science Engineering*, vol.2(2), pp.538-545, 2013.
- [3] A. K. Pal and R. K. Mudi, “Speed control of DC motor using relay feedback tuned PI, fuzzy PI and self-tuned fuzzy PI controller”, *Control Theory and Informatics*, vol.2(1), pp.24-32, 2012.
- [4] A. K. Pal, R. K. Mudi, and C. Dey, “Rule extraction through self-organizing map for a self-tuning fuzzy logic controller”, *Advanced Materials Research (MEMS, NANO and Smart Systems)*, vols.403-408, pp.4957-4964, 2012.
- [5] A. K. Pal, “Development of neuro-fuzzy controller for application to HVAC system, inverted pendulum and other processes”, *International Journal of Computational Cognition*, vol.6(2), pp.1-6, 2008.
- [6] A. K. Pal and R. K. Mudi, “Self-tuning fuzzy PI controller and its application to HVAC system”, *International Journal of Computational Cognition*, vol.6(1), pp.25-30, 2008.
- [7] A. K. Pal and S. K. Bag, “Characteristics of user defined neuro-fuzzy controller for nonlinear processes”, *Journal of Systems Science and Engineering*, vol.15(2), pp.53-60, 2007.

Papers published as Book Chapter:

- [1] A. K. Pal and J. Chakraborty, “Design a fuzzy logic controller with a non-fuzzy tuning scheme for swing up and stabilization of inverted pendulum”, *Advances in Intelligent Systems and Computing*, Publisher: Springer, vol.308, pp.221-230, 2015.
- [2] A. K. Pal, R. K. Mudi and R. R. De Maity, “A non-fuzzy self-tuning scheme of PD-type FLC for overhead crane control”, *Advances in Intelligent Systems and Computing*, Publisher: Springer-Verlag, vol.199, pp.35-42, 2013.
- [3] A. K. Pal and R. K. Mudi, “Development of a self-tuning fuzzy controller through relay feedback approach”, *Communications in Computer and Information Science*, Publisher: Springer-Verlag, vol.250, pp.424-426, 2011.

Papers published in International Conference Proceedings:

- [1] A. K. Pal and J. Chakraborty, “Adaptive fuzzy control of inverted pendulum with a fuzzy-based set-point weighting scheme”, *Proceedings of IEEE International Conference of Emerging Applications of Information Technology*, pp.46-51, 2014.
- [2] A. K. Pal and R. K. Mudi, “An adaptive fuzzy controller for overhead crane”, *Proceedings of IEEE International Conference on Advanced Communication Control and Computing Technologies*, pp.300-304, 2012.
- [3] R. R. De(Maity), R. K. Mudi, A. K. Pal, “A PD-type self-tuning FLC for second-order systems with dead-time”, *Proceedings of IEEE International Conference on Advanced Communication Control and Computing Technologies*, pp.409-413, 2012.
- [4] A. K. Pal, R. K. Mudi, and C. Dey, “Rule extraction through self-organizing map for a self-tuning fuzzy logic controller”, *International Conference on Control, Robotics and Cybernetics*, pp.156-160, 2011.

List of Patents: Nil

List of Presentations in International Conferences:

- [1] 4th International Conference of Emerging Applications of Information Technology (EAIT 2014), ISI-Kolkata, India, 19-21 Dec.2014.
- [2] International Conference on Intelligent Computing, Communication and Devices (ICCD-2014), SOA University, Odisha, India, 18-19 Apr.2014.
- [3] International Conference on Frontiers in Intelligent Computing Theory and Application (FICTA-2012), Bhubaneswar, Odisha, India, 22-23 Dec.2012.
- [4] IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT 2012), Ramanathapuram, Tamilnadu, India, 23-25 Aug.2012.
- [5] International Conference on Control, Communication and Power Engineering (CCPE 2011), Pune, India, 7-8 Nov.2011.
- [6] International Conference on Control, Robotics and Cybernetics (ICCRC 2011), New Delhi, India, 21-23 Mar.2011.
- [7] International Conference on Modeling and Simulation, Kolkata, India, 3-5 Dec.2007.

Certificate from the Supervisor

Date: October, 2017

*This is to certify that the thesis entitled “**Design and Performance Study of Self-Tuning Fuzzy and Neuro-Fuzzy Control Systems**” submitted by **Mr. Arabinda Kumar Pal**, who got his name registered on 12th October, 2010 for the award of Ph.D. (Engineering) degree of Jadavpur University, is absolutely based upon his own work under supervision of the undersigned; and that neither his thesis nor any part of the thesis has been submitted for any degree/diploma or any other academic award anywhere before.*

Prof. Rajani Kanta Mudi
Professor
Department of Instrumentation
& Electronics Engineering
Jadavpur University
Salt Lake Campus
Kolkata-700 098, India

Acknowledgements

I would like to gratefully acknowledge the enthusiastic supervision and guidance of Prof. Rajani Kanta Mudi during this work. He is my source of inspiration. As my supervisor, his insight, observations and suggestions helped me to establish the overall direction of the research and contributed immensely to the success of the work. I always had the freedom to follow my own ideas, which I am very grateful for. It is his help for which I stand where I am.

I would like to thank Prof D. C. Patranabis, Prof. Rajib Bandhpadhyay, Prof. Bapon Tudu, Prof. Bhaswati Goswami and Prof. Anutosh Chatterjee of Instrumentation and Electronics Engineering Dept., Jadavpur University for extending their valuable suggestions and help whenever I approached. I am grateful to Prof. Chanchal Dey of Applied Physics Dept. of Calcutta University for his help. I would also like to express my earnest thanks to all my co-authors of various publications.

My special thanks to Prof Abhijit Biswas of Calcutta University and Prof. Bibhas Chandra Dhara of Jadavpur University for their constant inspiration and encouragement during my research.

I am also grateful to IEE Dept. of Jadavpur University for providing me adequate infrastructure to carry out the present investigations. I would also like to thank all the staff members of Instrumentation & Electronics Engineering Department, Jadavpur University for their cooperation and assistance towards completion of this work.

I sincerely acknowledge the financial support under Research Promotion Scheme (RPS) from the All India Council of Technical Education (AICTE), Government of India.

I thank to all my friends for being there whenever I needed them. I acknowledge all staff and research scholars IEE Dept., Jadavpur University for helping me.

Finally, I would like to express my deep sense of gratitude to my mother to bless me with unstinted and unconditional support, encouragement and motivation during this research work. I specially thank to my wife, daughter and son for being a constant motivator and for their help in all and every possible way throughout this enlightening journey.

October' 2017

Arabinda Kumar Pal

*Department of Instrumentation & Electronics Engineering
Jadavpur University, Salt Lake Campus
Sector – III, Block – LB, Kolkata – 700 098.*

Contents

Short Bibliography	ii
List of Publications	iii
Certificate from the supervisors	vi
Acknowledgements	vii
List of Figures	xii
List of Tables	xvii
List of Abbreviations	xix
Chapter 1: Introduction and scope of the thesis	1-34
1.1. Introduction	1
1.2. Background	4
1.2.1. Conventional controller	4
1.2.2. Fuzzy logic controller	5
1.2.3. Self-tuning and adaptive fuzzy control	11
1.2.4. Neural network	12
1.2.5 System identification and Neuro-fuzzy systems	17
1.3. Literature survey	19
1.3.1. Self-tuning fuzzy control	19
1.3.2. Rule extraction and fuzzy modeling	26
1.4. Objectives of the thesis	31
1.5. Motivation of the present work	32
1.6. Thesis organisation	32
Chapter 2: Self-tuning fuzzy logic controller and its application to HVAC and inverted pendulum	35-80
2.1. Introduction	35
2.2. Types of fuzzy logic controllers	37
2.2.1. PD type fuzzy controller	37

2.2.2. PI type fuzzy controller	38
2.3. The proposed self-tuning scheme of FLCs	39
2.3.1. Membership functions	40
2.3.2. Scaling factors	41
2.3.3. The rule-bases	42
2.3.4. The self-tuning mechanism	45
2.4. HVAC system	47
2.4.1. HVAC system control	50
2.4.2. Results and discussion	50
2.4.3. Comparative study	59
2.5. Inverted pendulum	60
2.5.1. Inverted pendulum and its mathematical model	61
2.5.2. Inverted pendulum control by PID, FPDC and STFPDC	68
2.5.2.1. PID controller	69
2.5.2.2. FPDC and STFPDC	71
2.5.3. Comparative study	75
2.6. Conclusion	79
Chapter 3: Development of self-tuning fuzzy controller through relay feedback approach - STFPICα	81-102
3.1. Introduction	81
3.2. Relay feedback tuning	82
3.3. Self-tuning scheme with dynamic gain (α)	84
3.4. Simulation experiments with FPIC, STFPIC and STFPIC α	89
3.5. Real time experiments on DC motor	96
3.5.1. Relay feedback test	98
3.5.2. Results and discussion	99
3.6. Conclusion	102
Chapter 4: Development of Adaptive fuzzy PD controller - AFPDC	103-134
4.1. Introduction	103

4.2. Design of the proposed controller - AFPDC	106
4.2.1. Fuzzy membership functions	106
4.2.2. Scaling factors	107
4.2.3. Gain updating factor	108
4.2.4. The rule-bases	109
4.3. Simulation experiments	111
4.4. Real time experiments on overhead crane	119
4.5. Conclusion	134
Chapter 5: Fuzzy rule-based system identification using self-organizing map	135-190
5.1. Introduction	135
5.2. Self-Organizing Map - SOM	137
5.2.1. Introduction to SOM	137
5.2.2. The SOM algorithm	139
5.3. Rule extraction from clusters	144
5.3.1. The proposed rule extraction scheme	152
5.3.2. Fine tuning of extracted fuzzy rule-base model	157
5.4. Study of the gain surface of STFPIC	159
5.5. Study of the control surface of STFPIC	167
5.6. Simulation study with the identified STFPIC	171
5.6.1. With reduced gain rules	171
5.6.2. With reduced control rules	177
5.7. Rule merging	179
5.7.1. Similarity measurement and fuzzy rule minimization	179
5.7.2. Results	183
5.8. Generalization of the proposed scheme	185
5.9. Conclusion	190
Chapter 6: Real time implementation of SOM-based self-tuning fuzzy controller	191-206

6.1. Introduction	191
6.2. Real time systems	192
6.2.1. Demonstration on an overhead crane	192
6.2.2. Demonstration on water pressure control loop	197
6.3 Conclusion	206
Chapter 7: Conclusion	207-210
7.1. Thesis contributions	207
7.2. Future scope	209
Bibliography	211-223

List of Figures

Fig. No.	Captions	Page
1.1	Block diagram of a PID controller	5
1.2	A simple artificial neural net	12
2.1	Block diagram of PD type fuzzy logic controller	38
2.2	Block diagram of PI type fuzzy logic controller	39
2.3	Block diagram of the proposed self-tuning PI-type FLC (STFPIC)	40
2.4	Membership functions of inputs (e , Δe) and output (Δu)	40
2.5	Membership function of gain updating factor, β	40
2.6	Gain surface of $\{e, \Delta e, \beta\}$ with 49 gain rules	46
2.7	A typical HVAC system	50
2.8	A supply air pressure loop of typical HVAC system	51
2.9(a)	Performance of HVAC with FPIC	52
2.9(b)	Performance of HVAC with STFPIC	53
2.10(a)	Performance of HVAC with FPIC	53
2.10(b)	Performance of HVAC with STFPIC	54
2.11(a)	Performance of HVAC with FPIC	54
2.11(b)	Performance of HVAC with STFPIC	55
2.12(a)	Performance of HVAC with FPIC	55
2.12(b)	Performance of HVAC with STFPIC	56
2.13	Performance of HVAC (Case1) with STFPIC for load variation at 15s	57
2.14	Performance of HVAC (Case2) with STFPIC for load variation at 25s	57
2.15	Performance of HVAC (Case3) with STFPIC for load variation at 20s	58
2.16	Performance of HVAC (Case4) with STFPIC for load variation at 20s	58
2.17	Response of HVAC system under PID, FPIC and STFPIC	59
2.18	Digital pendulum mechanical unit	62
2.19	Free body diagrams of inverted pendulum system	63
2.20	The inverted pendulum simulink model	68
2.21	Basic block diagram of the inverted pendulum control scheme	69
2.22	Subsystem blocks for inverted pendulum control	70
2.23	Block diagram of the inverted pendulum control scheme for stabilization	70
2.24	Block diagram of the inverted pendulum control scheme for swing up	71
2.25	Block diagram of FPDC (twin) for inverted pendulum stabilization	72
2.26	MFs of e_x , Δe_x , u_x	72
2.27	MFs of e_θ , Δe_θ , u_θ	72
2.28	Block diagram of FPDC for swing up control	74
2.29	Block diagram of STFPDC for swing up control	74

2.30	MFs of e and Δe	75
2.31	MFs for β	75
2.32	Plot of control voltage against time for PID	75
2.33	Plot of pendulum cart position against time for PID control	76
2.34	Plot of pendulum cart position against time (FPDC; STFPDC)	77
2.35	Plot of inverted pendulum sway angle during swing up for PID	77
2.36	Plot of inverted pendulum sway angle during swing up for FPDC	78
2.37	Plot of inverted pendulum sway angle during swing up for STFPDC	78
3.1	Block Diagram of relay-feedback test	84
3.2	Relay-feedback test response	84
3.3	Block Diagram of STFPIC α	85
3.4	MFs of e , Δe and Δu	85
3.5	MFs of β	85
3.6(a)	Variation of gain updating factor (β) with e and Δe	87
3.6(b)	Variation of β with e and Δe (in reverse direction)	87
3.7	Control surface of FPIC	88
3.8	Control surface of STFPIC α	88
3.9	Responses of process (3.9) with STFPIC α for $L=0$ and $L=0.3$	90
3.10	Responses of process (3.9) with $L=0.2$ and load disturbance at $t=40s$	90
3.11	Responses of process (3.10) with STFPIC α for $L=0$ and $L=0.3$	92
3.12	Responses of process (3.10) with $L=0.2$ and load disturbance at $t=55s$	92
3.13	Responses of process (3.11) with STFPIC α for $L=0.1$ and $L=0.4$	94
3.14	Responses of process (3.11) with $L=0.3$ and load disturbance at $t=30s$	94
3.15	Block diagram for set-up of speed control of DC motor	96
3.16	Experimental set-up for speed control of DC motor	97
3.17	Frequency (proportional to motor speed) vs. voltage curve	97
3.18	Voltage vs. rpm curve	97
3.19	Relay Feedback response of a DC motor in close loop	98
3.20	Step response of the DC motor for ZNPIC	99
3.21	Step response of the DC motor for FPIC	100
3.22	Step response of the DC motor for STFPIC	100
3.23	Step Response of the DC motor for STFPIC α	101
4.1	Block Diagram of FPDC	106
4.2	Block Diagram of AFPDC	107
4.3	MFs of e , Δe and u	107
4.4	Variation of β with e_N and Δe_N	109
4.5	Control surface of FPDC	110

4.6	Control surface of AFPDC	111
4.7	Responses of (4.4) with PID for $L=0.0$ and $L=0.1$	112
4.8	Responses of (4.4) with FPDC and AFPDC for $L=0$ and load at 10s	113
4.9	Responses of (4.4) with FPDC and AFPDC for $L=0.1$ and load at 15s	113
4.10	Relay feedback tuning graph of $d^2y/dt^2+0.3.y.dy/dt=u(t-L)$	115
4.11	Responses of (4.5) with PID for $L=0.0, 0.1$ and 0.2	115
4.12	Responses of (4.5) with PID and AFPDC for $L=0$ and load at 20s	116
4.13	Responses of (4.5) with FPDC and AFPDC for $L=0$ and load at 20s	116
4.14	Responses of (4.5) with FPDC and AFPDC for $L=0.3$ and load at 20s	117
4.15	Responses of (4.6) with PID, FPDC and AFPDC for $L=0$ and load at 30s	118
4.16	Response of (4.6) with AFPDC for $L=0.1$ and load at 30s	119
4.17	Overhead crane set-up	120
4.18	Crane control diagram	121
4.19	Model of the overhead crane system	122
4.20	Dual control structure (position and angle) for overhead crane control	123
4.21	Diagram of FPDC for overhead crane control	124
4.22	Diagram of AFPDC for overhead crane control	125
4.23	MFs for position $[-0.5, +0.5]$ and angle $[-20^\circ, +20^\circ]$ controller	125
4.24	Overhead crane position control using PID for step input (0.3m)	126
4.25	Overhead crane position control using PID for sine input ($\pm 0.3m$)	127
4.26	Overhead crane swing angle control using PID for step input (0.3m)	127
4.27	Overhead crane swing angle control using PID for sine input ($\pm 0.3m$)	127
4.28	Position control for step input (0.3m) using FPDC and AFPDC	128
4.29	Position control for square input using FPDC and AFPDC	128
4.30	Overhead crane swing angle control for step input using AFPDC	129
4.31	Overhead crane swing angle control for square input using AFPDC	130
4.32	Overhead crane position control for different amplitude at constant speed	130
4.33	Position control for sine input using FPDC	131
4.34	Position control for sine input using AFPDC	131
4.35	Position control for saw tooth input using FPDC	131
4.36	Position control for saw tooth input using AFPDC	131
4.37	Overhead crane swing angle control for sine input using FPDC	132
4.38	Overhead crane swing angle control for sine input using AFPDC	132
4.39	Overhead crane swing angle control for saw tooth input using FPDC	132
4.40	Overhead crane swing angle control for saw tooth input using AFPDC	132
4.41	Position control using AFPDC for different amplitude at constant speed	133
4.42	Position control using AFPDC at different speed with constant amplitude	133
5.1	Two-Dimensional 3 by 3 rectangular SOM neural network	139

5.2	Kohonen Self-Organizing Map	139
5.3	Neighborhood scheme for SOM (rectangular)	141
5.4	Gaussian neighborhood function	142
5.5	Outline flow diagram of the proposed scheme	153
5.6(a)	$\mathbf{x}_k^* - d_{ij}(\mathbf{x}_k^*)$ plot	154
5.6(b)	$\mathbf{x}_k^* - \mu_{ij}(\mathbf{x}_k^*)$ plot	154
5.7	Block Diagram of STFPIC	159
5.8(a)	Gain surface of STFPIC with 49 rules	161
5.8(b)	Control surface of STFPIC with 98 rules	161
5.9	MF plots of 25 nodes for e , Δe and β	164
5.10	Representation 25 fuzzy <i>if-then</i> rules for e , Δe and β	166
5.11(a)	Gain surface of STFPIC with 25 identified rules	167
5.11(b)	Error surface (difference between original and identified gain surface)	167
5.12	MF plots of 4 nodes out of 25 for e , Δe and u after 1000 iteration	169
5.13	Control surface with 25 identified rules and comparison	170
5.14	Response of (5.10) with load for STFPIC and SOM-STFPIC	172
5.15	Response of (5.11) for SOM-STFPIC	173
5.16	Response of (5.11) with load for STFPIC and SOM-STFPIC	173
5.17	Response of (5.12) for SOM-STFPIC	175
5.18	Response of (5.12) with load for STFPIC and SOM-STFPIC	176
5.19	Responses of (5.13) for STFPIC (98 rules) and SOM-FLC ₁ (25 rules)	178
5.20	Responses of (5.14) for STFPIC (98 rules) and SOM-FLC ₁ (25 rules)	179
5.21	MF plots of 25 identified fuzzy sets for $\{e, \Delta e, u\}$	180
5.22	Fuzzy sets A, B and C	180
5.23	Fuzzy sets A and B	180
5.24	Two Fuzzy rules in the form of $\{e, \Delta e, u\}$	181
5.25	Conversion of 25 to 14 rules $\{e, \Delta e, u\}$ after similarity measurement	182
5.26	Responses of 5.17 for STFPIC, SOM-FLC ₁ and Reduced SOM-FLC ₁	183
5.27	Responses of 5.18 for STFPIC, SOM-FLC ₁ and Reduced SOM-FLC ₁	184
5.28	Surface plot of the nonlinear <i>equation</i> 5.19	185
5.29	MFs for the identified fuzzy model of the nonlinear <i>equation</i> 5.19	186
5.30	Surface plot of the <i>equation</i> (5.19) with extracted 25 fuzzy rules	186
5.31	Error surface [difference between original surface and model surface]	187
5.32	Comparative plot of <i>equation</i> (5.19) for actual output and model output	187
5.33	Representation 12 fuzzy <i>if-then</i> rules obtained after similarity measure	188
5.34	Surface plot of (a) original (b) with 25 fuzzy rules (c) with 12 fuzzy rules	189
5.35	Output determination (z) for given inputs (x and y) from Model FLC	189
5.36	Comparative plot of actual output and model output	189

6.1	Mechanical unit of overhead crane set-up	192
6.2	Diagram of STFPDC / SOM-STFPDC ₁ for overhead crane control	194
6.3	Position control (step input) using FPDC, STFPDC and SOM-STFPDC ₁	195
6.4	Position control (square i/p) using FPDC, STFPDC and SOM-STFPDC ₁	196
6.5	Swing angle control for step input using STFPDC and SOM-STFPDC ₁	196
6.6	Swing angle control for square input using STFPDC and SOM-STFPDC ₁	197
6.7	Hardware set-up of real time pressure loop	199
6.8	Pneumatic control valve with positioner in real time pressure loop	199
6.9	Real time pressure loop connected with PC for operation	200
6.10	Schematic diagram of pressure loop	200
6.11	Pressure header in pressure loop	201
6.12	Control valve Characteristics	201
6.13	Pressure vs. current calibration curve	202
6.14	Process response for a set-point of 25 psi (10mA) with STFPIC	203
6.15	Process response for a set point of 25 psi (10mA) with SOM-STFPIC ₁	204
6.16	Pressure evolution using SOM-STFPIC ₁ after load change at 100s	205
6.17	Pressure evolution after set-point changes for SOM-STFPIC ₁	205

List of Tables

Table No.	Captions	Page
2.1	Linguistic expressions	41
2.2	Fuzzy rules for computation of Δu	43
2.3	Fuzzy rules for computation of β	45
2.4	Performance analysis for different models of HVAC air pressure loop	56
2.5	Performance comparison between PID, ANF, FPIC and STFPIC for model variation	60
2.6	The value of the pendulum constants	67
2.7	Fuzzy rules for position and angle control	73
2.8	Fuzzy rules for computation of β	75
2.9	Performance analysis of the controllers for inverted pendulum control	78
2.10	Performance indices of the controllers for inverted pendulum control	79
3.1	Fuzzy rules for computation of Δu	86
3.2	Fuzzy rules for computation of β	86
3.3	Performance analysis for the linear 2 nd order process (3.9)	91
3.4	Performance analysis for the marginally stable 2 nd order process (3.10)	93
3.5	Performance analysis for the nonlinear 2 nd order process (3.11)	95
3.6	Performance analysis for the 2 nd order processes with load disturbance	95
3.7	Performance analysis with STFPIC α for the practical DC motor	101
3.8	Performance comparison for different controllers	101
4.1	Fuzzy control rules for computation of u	109
4.2	Performance analysis of (4.4) with PID	114
4.3	Performance analysis of (4.4) with FLCs	114
4.4	Performance analysis of (4.5) with PID without load disturbance	117
4.5	Performance analysis of the nonlinear process (4.5)	117
4.6	Performance analysis of the integrating process (4.6)	119
4.7	Performance analysis of the proposed controllers in crane control	129
4.8	Performance analysis of AFPDC for sine and saw tooth inputs	132
5.1	Performance analysis of linear process (5.10)	171
5.2	Performance analysis of (5.11) at different dead-time	174
5.3	Performance analysis of (5.11) with controllers developed by [105]	174
5.4	Performance analysis of marginally stable process (5.12)	176
5.5	Performance analysis of marginally stable process (5.12) with load	177
5.6	Performance analysis of linear (5.13) and nonlinear (5.14) system	178

5.7	Performance analysis of linear (5.17) and nonlinear (5.18) system	184
6.1	Parameters and their values of overhead crane	193
6.2	Performance analysis of the controllers in overhead crane control	195
6.3	Valve opening and corresponding pressure	202
6.4	Pressure transmitter and corresponding pressure gauge output	202
6.5	Performance analysis of the process	204

List of Abbreviations

AFPDC	Adaptive Fuzzy PD Controller
AFS	Adaptive Fuzzy Systems
ANF	Adaptive Neuro-Fuzzy
ANN	Artificial Neural Networks
BMU	Best Matching Unit
BPNN	Back Propagation Neural Network
CCM	Compatible Cluster Merging
DSFR	Degree of Similarity between Fuzzy Rules
DSFS	Degree of Similarity between Fuzzy Sets
FCM	Fuzzy C-Means
FCRM	Fuzzy C-Regression Model
FKBC	Fuzzy Knowledge Base Controller
FKBS	Fuzzy Knowledge Base System
FLC	Fuzzy Logic Controller
FOPDT	First Order Plus Derivative Time
FPIC	Fuzzy PI Controller
FPDC	Fuzzy PD Controller
FRC	Fuzzy Relational Clustering
GA	Genetic Algorithms
GKFCM	Gustafson–Kesel’s Fuzzy C-Means
HCM	Hard C-means
HiSSOL	Hierarchally Structural Self-Organizing Learning
HVAC	Heating Ventilation and Air-Conditioning
IAE	Integral Absolute Error
IFC	Improved Fuzzy Clustering
IMC	Internal Model Control
IR	Infrared
ISE	Integral Square Error
ITAE	Integral of the Time Multiplied Absolute Error
LFC	Load Frequency Control
LMS	Least Mean Square
MA	Mamdani-Assilian
MCM	Mountain Clustering Method
MF	Membership Function

MIMO	Multiple Input Multiple Output
MLP	Multilayer Perceptron
MSE	Mean Square Error
NN	Neural Network
NRMSE	Normalized Root Mean Square Error
PCM	Possibilistic C-means
PD	Proportional Derivative
PI	Proportional Integral
PID	Proportional Integral Derivative
RBFN	Radial Basis Function Network
RFCM	Relational Fuzzy C-Means
RMSE	Root Mean Square Error
SCM	Subtractive Clustering Method
SF	Scaling Factor
SFIN	Simplified Fuzzy Inference Network
SI	System Identification
SIMO	Single Input Multiple Output
SISO	Single Input Single Output
SOM	Self-Organizing Map
SOM-STFPIC ₁	STFPIC for MF_1
SOM-STFPIC ₂	STFPIC for MF_2
SOM-STFPIC ₃	STFPIC for MF_3
SOM-STFPDC ₁	STFPDC for MF_1
STAR	Self-Tuning Adaptive Resolution
STFLC	Self-Tuning Fuzzy Logic Controller
STFPDC	Self-Tuning Fuzzy PD Controller
STFPIC	Self-Tuning Fuzzy PI Controller
STFPIC α	Self-Tuning Fuzzy PI Controller with Dynamic Gain
SVM	Support Vector Machine
TS	Takagi–Sugeno
VQTAM	Vector-Quantized Temporal Associative Memory
ZN	Ziegler-Nicholas
ZNPIC	Ziegler-Nicholas Tuned PI Controller

CHAPTER 1

Introduction and scope of the thesis

1.1 Introduction

The conventional PID controllers are widely used in industry due to their simplicity in arithmetic, ease of use, good robustness, high reliability, stabilization and zero steady state error [1-2]. Ziegler-Nichols tuned PI or PID controller performs well around normal working conditions, but its tolerance to process parameter variations are severely affected. Conventional controllers require mathematical models, which may not be always available. The industrial processes are usually nonlinear and higher order systems with considerable dead-time, and their parameters may be changed with changes in ambient conditions or with time [3-4]. Thus, to have a satisfactory control performance, the control action should be a nonlinear function. It is not possible to incorporate this nonlinearity in a conventional controller. However, a conventional fuzzy logic controller (FLC) attempts to incorporate it by a limited number of *if-then* rules; although this may not always be sufficient to generate the required control action [5-8]. For the successful design of a FLC, the proper selection of input and output scaling factors and/or the tuning of other controller parameters, such as meaningful partition of the input-output linguistic variables, appropriate formulation of the rule-base, and definition of the membership functions, are crucial jobs [9-20]. Attempts are persistent in the research world to obtain a superior controller by choosing the correct set of rules and scaling factors in order to fine tune the fuzzy logic controller. Moreover, a number of approaches have been proposed to implement hybrid control structures that combine the robustness of conventional controllers with the intelligence of fuzzy logic techniques to control the nonlinear systems.

In the process industry, generally a skilled human operator always tries to manipulate the process input, usually by adjusting the controller gain based on the current process states to get the process ‘optimally’ controlled. The proper tuning of the output scaling factor (SF) is very important, as it is equivalent to the controller gain. This output gain has been given the highest priority because of its strong influence on the performance and stability of the system [21 -25]. *Mudi and Pal* [22] proposed a robust self-tuning scheme for FLCs, where an on-line fuzzy gain modifier is determined by 49 fuzzy *if-then* rules based on operator’s knowledge. For fine tuning, irrespective of the process under control, they further augmented the gain modifier by an empirically obtained constant multiplying factor. Motivated by the encouraging simulation results of [22], we decide to investigate the proposed self-tuning fuzzy controllers’ performance in practical systems. As a typical nonlinear system, we consider the heating, ventilation and air-conditioning (HVAC) system, where the supply air pressure control loop for HVAC is highly nonlinear in nature. We also consider the inverted pendulum, a benchmark system to researchers, for its strong degrees of non-linearity and inherent instability [6]. A control algorithm is properly judged when it efficiently and effectively controls such a complex system. By applying the proposed fuzzy controller and dual control scheme the inverted pendulum stabilization and swing-up problems are addressed.

Finding an appropriate multiplying factor for the fine tuning of FLCs [22] becomes a great challenge. Instead of using a fixed gain multiplier, we determine to find out a process specific multiplier that is directly related to the process dynamics. The relay-feedback approach is proposed here for this purpose. In this approach, output scaling factor of the FLC updated by fuzzy gain modifier is further parameterized by the process ultimate-gain and ultimate-period, *i.e.*, *critical point* of the system obtained through relay-feedback experiment. The effectiveness of the technique is demonstrated on various linear and nonlinear processes.

Mudi and Pal [22] used 49 rules for FLC and another 49 rules for its tuning. Sometimes, the tuning by such a large number of fuzzy rules may make the system more complex and slow. In view of this, a question comes in our mind, whether we really need so much of rules or is it possible to realize the same level of performance even with a much smaller set of rules. We investigate this issue by using lesser number of rules and also by applying different non-fuzzy adaptive schemes, and later on, proposed a rule extraction scheme that can extract a smaller but

adequate rule-base from a set of input-output data. Effectiveness of the proposed adaptive controller is studied with respect to its conventional fuzzy and non-fuzzy controllers in terms of several performance indices. Finally, the proposed control approach is demonstrated on a laboratory scale overhead crane. Moving a suspended load along a pre-specified path is not an easy task when strict specifications on the swing angle and on the transfer time need to be satisfied. In this thesis, twin adaptive fuzzy controllers are proposed to control the position of the trolley crane and swing angle of load. The proposed adaptive control scheme guarantees a fast and precise load transfer and the swing suppression during the load movement, despite of model uncertainties.

Usually, most of the fuzzy systems are developed by user defined fuzzy *if-then* rules for a given system. But there are some cases where there is no expert, or expert cannot express his or her knowledge explicitly. Hence in such cases, it is required to extract input-output relationship based on the information obtained from the system. System identification (SI) is an important research area primarily devoted to developing models of physical systems based on observed input-output data. During the past three decades a lot of research has been directed towards developing efficient system identification algorithms with a view to obtain models that closely match to the real physical systems. The fuzzy modeling is an integral part of system identification. The main idea of fuzzy modeling [14, 17, 26-29] is to describe the input-output behavior of a given system by a set of fuzzy *if-then* rules. An unknown system transforms input x_i to output y_i and let the system be denoted by S , thus $y=S(x)$. The problem here is to find an explicit (mathematical) or implicit (computational schemes/algorithms) model for S . Identification problem can be conceived as an optimization problem in which the error between the actual measured response of a system and the identified response of a model is minimized. Therefore, interest in system identification lies in minimizing the error norm of the outputs. The identification of such a linguistic model consists of two parts; structure identification and parameter estimation. For structure identification clustering techniques can be used. Among the different clustering models, we use Self-Organizing Map (SOM) since it operates in an unsupervised manner, thus minimizing the requirement for human guidance. SOM projects high-dimensional data onto a low-dimensional grid. The projected data preserves the topological relationship of the original data. We define fuzzy rules for each node/cluster. Since the feature selection and rule identification are integrated, such a system can establish subtle nonlinear

interaction between features and fuzzy rule-base. Initial parameters of the input-output membership functions (MFs) are estimated from the clustering results and are refined by some gradient descent based optimization schemes. The proposed method has a provision of selecting desired number of rules for a given process.

After rule extraction, it has been observed that many of the fuzzy sets are almost similar in nature. By investigating their similarity, for simplicity, rule merging scheme is suggested in this thesis. The number of rules is reduced through the merging of similar fuzzy sets. This thesis considers the linguistic modeling of nonlinear systems using a neuro-fuzzy approach and its effectiveness is successfully demonstrated through simulation experiments as well as real time implementation on an actual/practical system.

1.2 Background

1.2.1 Conventional controller

A proportional-integral-derivative (PID) controller is a generic control loop, with feedback mechanism, widely used in industrial control systems [30] as shown in Fig. 1.1. A PID controller calculates an ‘error’ value as the difference between a measured process variable and a desired set-point. The controller attempts to minimize the error by adjusting the process control inputs. The PID controller calculation involves three separate constant parameters: the proportional, the integral and derivative values, denoted by P , I , and D . Heuristically, these values can be interpreted in terms of time: P depends on the *present* error, I on the accumulation of *past* errors, and D is a prediction of *future* errors based on current rate of change. The weighted sum of these three actions is used to adjust the process via a final control element. By tuning the three parameters in the PID controller algorithm, the controller can provide control action designed for specific process requirements. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the set-point, and the degree of system oscillation. A PID controller will be called a P, PI or PD controller in the absence of the respective control actions. In process industries, PI and PID controllers are generally used due to their simple design and tuning methods. Due to the presence of measurement noise, PI controllers are more preferable than PID controllers. The absence of derivative action makes a PI controller simple and less sensitive to

noise [31]. In practice, nearly 90% of all industrial PID controllers have their derivative action turned off [32]. The most important step for a successful controller design is its tuning. Ziegler–Nichols (ZN) ultimate cycle method [33] is widely used to determine reasonably good settings of P, PI and PID controllers, to start with. ZN tuned PI controllers (ZNPICs) exhibit good performance for first-order processes, but they usually fail to provide satisfactory performance for high-order and/or nonlinear systems, which represent most of the practical processes. Specially, performances of ZNPICs under set-point change are not acceptable in many cases due to excessive oscillation associated with a large overshoot [34, 35].

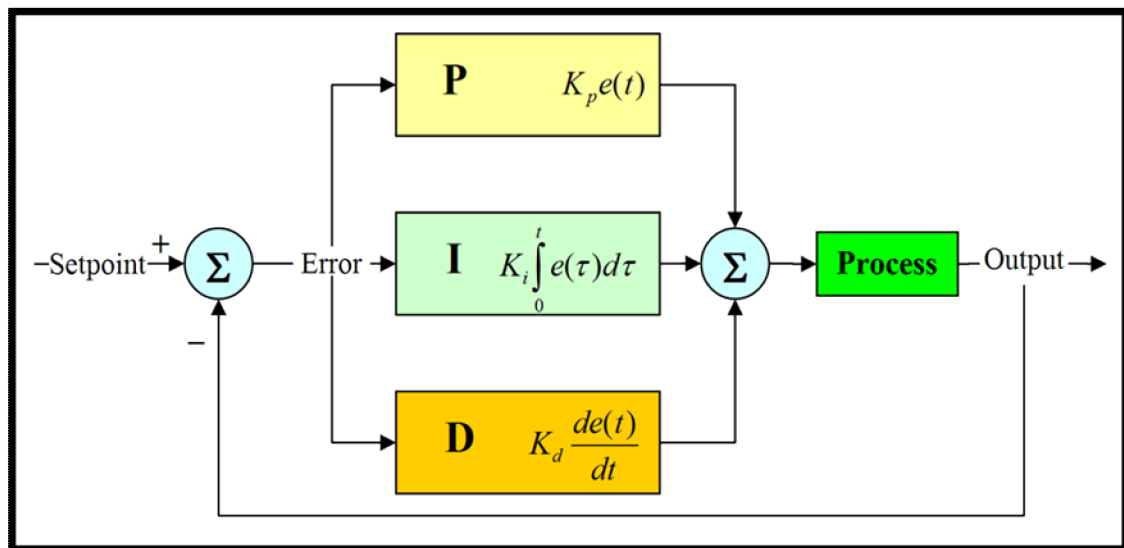


Figure 1.1: Block diagram of a PID controller.

1.2.2 Fuzzy logic controller

Complex real world problems require intelligent systems that combine knowledge, techniques, and methodologies from various sources. These intelligent systems are supposed to possess humanlike expertise within a specific domain, adapt themselves and learn to do better in changing environments and explain how they make decisions or take actions. This innovative approach to constructing computationally intelligent systems is called Soft Computing.

Zadeh introduced the subject of ‘Fuzzy Logic’ or fuzzy set theory, in 1965 [36]. The fuzzy logic tool is a soft computing tool that can be used to powerfully get things done. It deals with uncertainty and provides an inference structure that enables appropriate human reasoning

capabilities. The human brain interprets imprecise and incomplete sensory information provided by perceptive organs. The fuzzy set theory provides a systematic calculus to deal with such information linguistically, and it performs numerical computation by using linguistic labels stipulated by membership functions. Moreover, a selection of fuzzy *if- then* rules forms the key component of a fuzzy inference system that can effectively model human expertise in a specific application [37]. In simpler words, fuzzy logic reflects how people think. It attempts to model our sense of words, our decision making and our common sense. As a result, it is leading to new and more human intelligent systems.

Fuzzy logic unlike Boolean or crisp logic, deals with problems that have vagueness or imprecision. Normally in boolean logic we deal with statements, answers to which are either *true or false, yes or no*, or a *0 or 1*. However for many situations the answer is more uncertain. Fuzzy logic deals with uncertainty in engineering by attaching degree of uncertainty to the answer to a logical question. Commercially, fuzzy logic has been used with success to control machines and consumer products [38]. Its major advantages include:

- i. It is an alternative design methodology which is simpler and faster.
- ii. Fuzzy logic reduces the design development cycle.
- iii. It simplifies design complexity.
- iv. It is always a better alternative solution to nonlinear control.
- v. It improves control performances.
- vi. It simplifies implementation, and reduces hardware costs.

Principle Design parameters:

Membership Function (MF) is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs that are processes, defines functional overlap between inputs, and ultimately determines an output response. The rules use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output conclusion. Once the functions are inferred, scaled, and combined, they are defuzzified into a crisp output which drives the system. There are different MFs associated with each input and output response.

Some of the extensively used MFs, especially in real time implementation are:

- i. *Increasing MFs*
- ii. *Decreasing MFs*
- iii. *Triangular MFs*
- iv. *Pie / Trapezoidal MFs*
- v. *S-/ Gaussian MFs*

Fuzzification Module performs a scale transformation (*i.e.*, an input normalization) which maps the physical values of the current process state variables into a normalized domain, we intend to work in. The fuzzification module converts a point-wise (crisp), current value of a process state variable into a fuzzy set of linguistic variables, in order to make it compatible with the fuzzy set representation of the process state variable in the rule-antecedent. In real life world, the quantities that we consider may be thought of as crisp, accurate and deterministic, but actually they are not so. They possess uncertainty within themselves. The uncertainty may arise due to vagueness or imprecision; in this case the variable is probably fuzzy and can be represented by a membership function [39].

Knowledge Base of a fuzzy knowledge base controller (FKBC) consists of data-base and a rule-base. The basic function of the data-base is to provide the necessary information for the proper functioning of the fuzzification module, the rule-base, and the defuzzification module. This information includes: membership functions representing the meaning of the linguistic values of the process state and control output variables; and the physical domains and their normalized counterparts together with the normalization/denormalization factors. If the continuous domains of the process state and control output variables have been discretized then the data-base also contains information concerning the quantization look-up tables defining the discretization policy. The basic function of the rule-base is to represent in a structured way the control policy of an experienced process operator and/or control engineer in the form of a set of production rules such as:

If [process state] *then* [control output]

The *if* – part of such a rule is called the rule-antecedent and is a description of a process state in terms of a logical combination of atomic fuzzy propositions. The *then* – part of the rule is called the rule consequent and is again a description of the control output in terms of a logical

combination of fuzzy propositions. These propositions state the linguistic values which the control variables take whenever the current process state matches the process state description in the rule-antecedent.

Fuzzy Inference Engine is a popular computing frame work based on the concepts of fuzzy set theory, fuzzy *if-then* rules, and fuzzy reasoning [37]. There are two basic types of approaches employed in the design of the inference engine of a FKBC: *composition based inference* and *individual rule based inference*. In composition based inference, union of the relational representation of each rule can be used for the entire rule-base and the control output is then obtained as the inference from this composite relation. The second type of inference rule is predominantly used in applications of FKBC. Its basic function is to compute the overall value of the control output variable based on the individual contributions of each rule in the rule-base. Each such individual contribution represents the value of the control output variables as computed by a single rule. The output of the fuzzification module representing the current crisp values of the process state variables is matched to each rule antecedent, and a degree of match for each rule is established. Based on this degree of match, the value of the control output variable in the rule antecedent is modified.

Fuzzy Inference Methods:

Mamdani's fuzzy inference method is the most commonly used in fuzzy decision making systems. Mamdani's method is among the first control systems built using fuzzy set theory. It is proposed in 1975 by *Mamdani* and *Assilian* [40] as an attempt to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from experienced human operators. *Mamdani-type inference*, expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification. In Mamdani's model the fuzzy implication is modeled by Mamdani's minimum operator, the conjunction operator is *min.*, the *t-norm* from compositional rule is *min.* and for the aggregation of the rules the *max.* operator is used [41].

This can be mathematically explained as follows: For Mamdani implication the relation R is obtained as $\mu_R(x,y) = \min[\mu_A(x), \mu_B(y)]$; $x \in X$, $y \in Y$. Once the relationship between A and B is known for a given fact 'x is A' (A is a fuzzy set on X) the corresponding conclusion B (B is a

fuzzy set on Y) is obtained through composition operation as $B = A \circ R$; where ‘o’ denotes the composition operator.

Like fuzzy implications, various forms of composition operators also exist in the literature [42, 43]. The two most common forms of composition operator are the ‘*max-min*’ and ‘*max-product*’ or ‘*max-dot*’ compositions.

$$\text{max-min: } \mu_B(y) = \max\{\min(\mu_A(x), \mu_R(x,y))\}; \quad x \in X$$

$$\text{max-dot: } \mu_B(y) = \min\{\max(\mu_A(x), \mu_R(x,y))\}; \quad x \in X$$

Such composition operation can be done by the combination of fuzzy sets and fuzzy relation with the aid of ‘cylindrical extension (*ce*)’ and ‘projection (*proj*)’ [42, 44] as

$$B = A \circ R = \text{proj} [\text{ce} (A) \cap R] \text{ on } Y.$$

This process of inferring the conclusion (B) from a given fact (A) and the fuzzy relation (R) is termed as ‘compositional rule of inference’. Fuzzy inference scheme that uses Mamdani implication in conjunction with *max-min* composition is known as *Mamdani type inferencing* which is normally used in fuzzy control. The Mamdani model is typically used in knowledge-based (expert) systems.

Takagi–Sugeno (TS) Inference Method is another important method used in FLC design. In data-driven identification, the model due to *Takagi* and *Sugeno* has become popular [17]. In this model, the antecedent is defined in the same way as above, while the consequent is an affine linear function of the input variables:

$$R_i: \text{ If } \mathbf{x} \text{ is } A_i \text{ then } y_i = \mathbf{a}_i^T \mathbf{x} + b_i; \quad i = 1, 2, \dots, K.$$

where \mathbf{a}_i is the consequent parameter vector and b_i is a scalar offset. This model combines a linguistic description with standard functional regression: the antecedents describe fuzzy regions in the input space in which the consequent functions are valid. The output y is computed by taking the weighted average of the individual rules’ contributions:

$$y = \frac{\sum_{i=1}^k \beta_i(x) y_i}{\sum_{i=1}^k \beta_i(x)} = \frac{\sum_{i=1}^k \beta_i(x) (\mathbf{a}_i^T \mathbf{x} + b_i)}{\sum_{i=1}^k \beta_i(x)}$$

where $\beta_i(x)$ is the *degree of fulfillment* of the i^{th} rule. The antecedent fuzzy sets are usually defined to describe distinct, partly overlapping regions in the input space. The parameters \mathbf{a}_i are then approximate local linear models of the considered nonlinear system. The TS model can thus be regarded as a smooth piece-wise linear approximation of a nonlinear function or a parameter-scheduling model.

Defuzzification Module performs the so-called defuzzification, which converts the set of modified control output values into a single point-wise value. A defuzzification process produces a non-fuzzy control action that best represents the possibility distribution of an inferred fuzzy control action. Some defuzzification processes are:

a) Centre of Area/ Centre of Gravity defuzzification is the well-known defuzzification method. In the discrete case ($\mathbf{u} = \{u_1, u_2, \dots, u_l\}$). Then the crisp value of the control output (u^*) resulting from this defuzzification method will be given by

$$\begin{aligned} u^* &= \frac{\sum_{i=1}^l u_i \cdot \mu_U(u_i)}{\sum_{i=1}^l \mu_U(u_i)} \\ &= \frac{\sum_{i=1}^l u_i \cdot \max_k \mu_{CLU^{(k)}}(u_i)}{\sum_{i=1}^l \max_k \mu_{CLU^{(k)}}(u_i)} \end{aligned} \quad (1.1)$$

Here \sum denotes an algebraic integration, the overall control output \tilde{U} or μ_U is obtained as the

union of the m clipped control outputs $\tilde{U} = \bigcup_{k=1}^m \widetilde{CLU}^{(k)}$.

b) Centre of Sums is a similar but faster defuzzification method. The motivation for using this method is to avoid the computation of \tilde{U} . The idea is to consider the contribution of the area of each $\widetilde{CLU}^{(k)}$ individually. Mathematically, the Center of Area/Gravity method builds \tilde{U} by taking the union of all $\widetilde{CLU}^{(k)}$. Center of Sums, however, takes the sum of the $\widetilde{CLU}^{(k)}$, where $k =$

1 to n . Thus overlapping areas, if such exist, are reflected more than once by this method. The faster algorithm for this defuzzification method is one reason why most FKBC use this method. In discrete case Center-of-Sums is formally given by

$$u^* = \frac{\sum_{i=1}^l u_i \cdot \sum_{k=1}^n \mu_{CLU^{(k)}}(u_i)}{\sum_{i=1}^l \sum_{k=1}^n \mu_{CLU^{(k)}}(u_i)} \quad (1.2)$$

c) Height defuzzification is a method which takes the peak value of each $\widetilde{CLU}^{(k)}$ and builds the weighted sum of these peak values. Thus neither the support nor the shape of $\widetilde{CLU}^{(k)}$ plays a role in the computation of u^* . The Height method is both a very simple and very quick method.

$$u^* = \frac{\sum_{k=1}^m \mu_k^c \cdot f_k}{\sum_{k=1}^n f_k} \quad (1.3)$$

where μ_k^c and f_k are the centroid and height or the firing strength of the k^{th} rule respectively.

1.2.3 Self-tuning and adaptive fuzzy control

Nowadays, control engineers frequently use the conventional PI, PD and PID type fuzzy logic controllers to address different process nonlinearities by a limited number of *if-then* rules, but it may not always produce fruitful results with fixed valued scaling factors and uniform membership functions [12]. In spite of a number of merits, there are many limitations while designing a fuzzy logic controller, since there is no standard methodology for its various design steps, and no well-established criterion for selecting suitable values for its large number of design parameters.

Induction of a suitable tuning scheme may eliminate the limitations of fuzzy logic controllers. Existing tuning schemes are applicable for conventional controllers, which are linear in nature. But, fuzzy logic is a nonlinear technique, thus the existing tuning schemes cannot solve this problem. Different attempts have been made to tune the *if-then* control rules to achieve the desired control objectives [21]. But, the tuning of a large number of FLC parameters is a time-consuming, expensive, and cumbersome task. Ill-performed and ill-defined control systems greatly reduce system effectiveness, therefore control engineers recognized the importance of automatic tuning of the controllers. The automatic tuning or self-tuning of a fuzzy logic

controller aims to adapt the controller at different operating conditions and subsequently eliminates the disturbances occurring in the process.

A fuzzy logic controller is called ‘*adaptive*’, if any of its tunable parameters such as scaling factors, membership functions and *if-then* rules changes when the controller is in operation, and otherwise it is a non-adaptive or conventional fuzzy logic controller. An adaptive fuzzy logic controller that fine-tunes an already working controller by adjusting either its scaling factors or membership functions or both of them is called a *self-tuning or adaptive* fuzzy logic controller. On the other hand, a fuzzy logic controller is called *self-organizing* when its rules are changed automatically.

1.2.4 Neural network

Artificial neural networks (ANN) or Neural networks (NN) have a large number of highly interconnected processing elements with the ability to learn from training patterns or data. They are like human brain and can perform pattern-matching tasks. Neural networks combined with fuzzy logic can provide excellent performance in developing human made systems that can perform the same type of information processing as that of our brain. Distributed representation and learning capabilities are the two features of neural networks. Most importantly, neural networks can perform filtering operations that are beyond the capabilities of the conventional linear filtering techniques because of their non-linear nature. Artificial neural network (ANN) is composed of a large number of highly interconnected processing elements (neurons) working in union to solve specific problems.

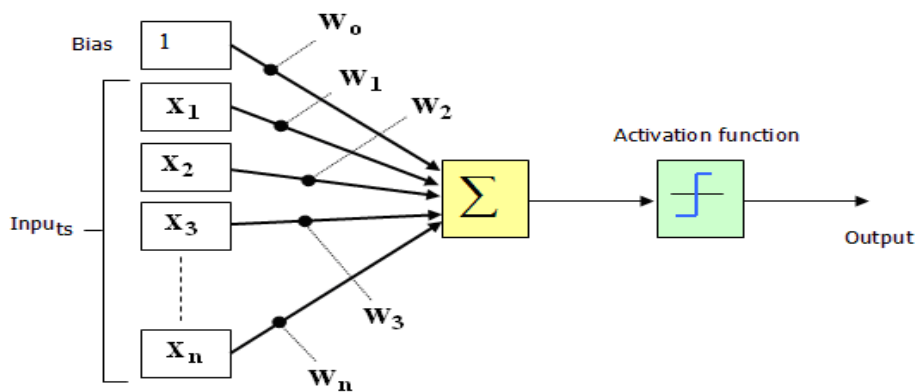


Figure 1.2: A simple artificial neural net.

An artificial neuron is characterized by:

1. Architecture (connection between neurons)
2. Training or learning
3. Activation function

Fig. 1.2 shows an artificial neuron with a set of ‘ n ’ inputs x_i , each representing the output of another neuron (the subscript i takes values between 1 and n and indicates the source of the vector input signal). Each input is weighed before it reaches the main body of the processing element by the connection strength or the weight factor analogous to the synaptic strength. The amount of information about the input that is required to solve a problem is stored in the form of weights. Each signal is multiplied by its associative weight $w_1, w_2, w_3, \dots, w_n$ before it is applied to the summing block. In addition, the artificial neuron has a bias term w_0 , a threshold value ‘ θ ’ that has to be reached or extended for the neuron to produce a signal. Thus the output of the summing block is:

$$S = w_0 + \sum_{i=1}^n x_i w_i \quad (1.4)$$

The resultant signal is then passed through a nonlinear activation function ‘ f ’. Then the output ‘ y ’ of the neuron may be represented as,

$$y = f(w_0 + \sum_{i=1}^n x_i w_i) = f(S) \quad (1.5)$$

The neuron firing condition is:

$$\sum_{i=1}^n x_i w_i \geq \theta \text{ [for linear activation function], or}$$
$$f(S) \geq \theta \text{ [for nonlinear activation function].}$$

Learning algorithms are used in neural network to update the weight parameters at the inter-connection level of the neurons during the training process of the network. The well-known and most often used learning mechanisms are the *supervised* and the *unsupervised*.

Supervised learning:

The main feature of the supervised learning mechanism is the training by examples. This means that an external expert provides the network with a set of input stimuli for which the output is a priori known. During the training process, the outputs are continuously compared with the

desired value. An appropriate learning rule (such as gradient decent) uses the error between the actual output and the desired output to adjust the connection weights so as to obtain, after a number of iterations, the closest match between the target output and the actual output. Supervised learning is particularly useful for feedforward networks. The *back-propagation neural network* (BPNN) algorithm, which is based on the gradient decent optimization technique, *radial basis function network* (RBFN) and the *least mean square* (LMS) algorithm are among the most commonly used supervised learning rules.

Unsupervised learning:

Unsupervised or self-organized learning does not involve any external teacher and relies instead upon local information and internal control. The training data and input patterns are presented to the system, and through predefined guidelines, the system discovers emergent collective properties and organizes the data into clusters or categories. Unsupervised learning algorithms have been known as open loop adaptation learning schemes. An unsupervised learning scheme operates as follows:

- A set of training data is presented to the system at the input layer.
- The network connection weights are then adjusted through some sort of competition among the nodes of the output layer, where the successful candidate will be the node with the highest value. In the process, the algorithm strengthens the connection between the incoming pattern at the input layer and the node output corresponding to the winning candidate.
- In addition to the strengthening of the connections between the input layer and the winning output node, the learning scheme may be used to adjust the weights of the connections leading to the neighboring nodes at the output layer. It has the major property of making groups of output nodes (*clusters*) behave as single entities with particular features.

Clusters, or conceptually meaningful groups of objects that share common characteristics, play an important role in how people analyze and describe the world. Indeed, human beings are skilled at dividing objects into groups (clustering) and assigning particular objects to these groups (classification). Clusters are potential classes and cluster analysis is the study of techniques for automatically finding classes. Cluster analysis provides an abstraction from

individual data objects to the clusters in which those data objects reside. Additionally, some clustering techniques characterize each cluster in terms of a cluster prototype; *i.e.*, a data object that is representative of the other objects in the cluster. Clustering techniques are validated on the basis of the following **assumptions**:

1. *Similar inputs to the target system to be modeled should produce similar outputs.*
2. *These similar input-output pairs are bundled into clusters in the training dataset.*

Assumption-1 states that the target system to be modeled is a smooth input-output mapping; this is generally true for most of the real world systems.

Assumption-2 requires the data to conform to some specific type of distribution; however, this is not always true. Therefore clustering techniques used for structure identification in neural or fuzzy modeling are highly heuristic, and finding data to which clustering techniques cannot be applied satisfactorily is not uncommon.

Clustering techniques can be *off-line* or *on-line*.

Off-line clustering techniques: Mostly four types of *off-line* clustering techniques [45 - 49] are used for fuzzy modeling: *K-Means Clustering*, *Fuzzy C-Means Clustering*, *Mountain Clustering* and *Subtractive Clustering*.

On-line (unsupervised) clustering techniques: When no external teacher or critic's instruction is available, only input vectors can be used for learning, such an approach is learning without supervision or unsupervised learning. This type of learning frequently employed for data clustering, feature extraction and similarity detection. Some of the unsupervised learning techniques are: *Competitive learning*, *Kohonen self-organizing feature map* and *Principal component analysis*.

- **Competitive Learning** is an unsupervised learning that updates weights only on the basis of the input patterns, with no available information regarding the desired outputs. Competitive Learning is usually implemented with *neural networks* that contain a hidden layer which is commonly known as "competitive layer". We can find the best matching unit by two ways:

- i) Comparing the inner product $\mathbf{X}^T \mathbf{W}_{ij}$ of the impinging input \mathbf{X} with each weight vector \mathbf{W}_{ij} . The winning neuron is the one that has the largest inner product.

ii) Euclidean distance criterion: the winning neuron is the one that that minimizes the distance $\|X - W_{ij}\|$.

For every input vector, the competitive neurons ‘compete’ with each other to see which one of them is the most similar to that particular input vector. The winner neuron sets its output ‘1’ and all the other competitive neurons set their output ‘0’. This strategy is also called *winner-take-all* since only the winning neuron is updated. A limitation of competitive learning is that some of the weight vectors never get updated. This learning method lacks the capability to add new clusters when necessary.

- ***Kohonen self-organizing feature map*** is another competition based network paradigm for data clustering. Self Organizing Map (SOM) by *Teuvo Kohonen* [50] provides a data visualization technique which helps to understand high dimensional data by reducing the dimensions of data to a map. SOM also represents clustering concept by grouping similar data together. Therefore it can be said that SOM reduces data dimensions and displays similarities among data. With SOM, clustering is performed by having several units compete for the current object. Once the data have been entered into the system, the network of artificial neurons is trained by providing information about inputs. The weight vector of the unit is closest to the current object becomes the winning or active unit. During the training stage, the values for the input variables are gradually adjusted in an attempt to preserve neighborhood relationships that exist within the input dataset. As it gets closer to the input object, the weights of the winning unit are adjusted as well as its neighbors. Getting the Best Matching Unit (BMU) is done by running through all weight vectors and calculating the distance from each weight to the sample vector. The weight with the shortest distance is the winner. There are numerous ways to determine the distance; however, the most commonly used method is the *Euclidean Distance*. We have discussed in details about SOM and SOM algorithm in **Section 5.2 (Chapter-5)**. *Teuvo Kohonen writes "The SOM is a new, effective software tool for the visualization of high-dimensional data. It converts complex, nonlinear statistical relationships between high-dimensional data items into simple geometric relationships on a low-dimensional display. As it thereby compresses information while preserving the most important topological*

and metric relationships of the primary data items on the display, it may also be thought to produce some kind of abstractions."

1.2.5 System identification and Neuro-fuzzy systems

System Identification is prerequisite before going to design a controller for a plant, say for a scenario, an on-line plant requires a controller for improving its performance. The controller cannot be designed on-line for a running plant as it may disturb the entire production which may be cost effective, so a model is required which represents the on-line plant. Here comes the concept of modeling a plant.

System identification concerns with the determination of a system structure on the basis of input-output data samples. The identification task is to determine a suitable estimate of finite dimensional parameters which completely characterize and describe the plant. The selection of the estimate is based on comparison between the actual output value and a predicted value on the basis of input data up to that instant.

The purposes of system identification/modeling are multiple:

- To predict a system's behavior, as in time series prediction and weather forecasting.
- To explain the interactions and relationships between inputs and outputs of a system. For example, a mathematical model can be used to examine whether the demand indeed varies proportionally to the supply in an economic system.
- To design a controller based on the model of a system, as in aircraft and ship control. Also to control computer simulation of the system, we need a model based on the system.

System identification generally involves two steps:

- A. Structure Identification:** In this step, we need to apply a priori knowledge about the target system to determine a class of models within which the search for the most suitable model is to be conducted. Usually this class of models is denoted by a parameterized function $y=f(\mathbf{u}, \boldsymbol{\theta})$; where y is the model's output, \mathbf{u} is the input vector, and $\boldsymbol{\theta}$ is the parameter vector. The determination of the function f is problem dependent, and the

function is based on designer's experience and intuition as well as the laws of nature governing the target system.

B. Parameter Estimation: In the second step, when the structure of the model is known, we need to apply optimization techniques to determine the parameter vector θ , such that the resulting model y can describe the system appropriately.

In general, system identification is not a one pass process; it needs to do both structure identification and parameter estimation repeatedly until a satisfactory model is found.

Neuro-Fuzzy system represents a hybrid intelligent system combining the main features of ANN with those of fuzzy knowledge base system (FKBS). It is well recognized that neither fuzzy reasoning systems nor neural networks are by themselves capable of solving problems involving both linguistic and numerical knowledge at the same time. The aim of this integration is to either adapt fuzzy systems to the changing environment, or to identify fuzzy rules and membership functions. The integration of fuzzy logic systems with neural networks reduces the limitations of fuzzy systems in terms of lack of learning while strengthening the neural network features in terms of explicit knowledge representation. Major strengths and weaknesses of both neural network and fuzzy logic system are summarizing below:

	Fuzzy Logic	Neural Network
Representation	<i>Linguistic description of knowledge</i>	<i>Knowledge distributed within computational units</i>
Adaptation	<i>Some adaption</i>	<i>Adaptive</i>
Knowledge representation	<i>Explicit and easy to interpret</i>	<i>Implicit and difficult to interpret</i>
Learning	<i>Non-existent</i>	<i>Excellent tools for imparting learning</i>
Verification	<i>Easy and efficient</i>	<i>Not straightforward</i>

It is difficult to construct model of an unidentified ill-defined system, but it is possible to obtain output data from the system for a given set of known input data. It is important to extract input-

output relational data from any unknown or ill-defined system. This relational information helps to characterize or model the system. Neuro-fuzzy system is an important tool to investigate any nonlinear and ill-defined system. Usually to construct a neuro-fuzzy system, we use a set of numerical data consisting of an input-output space. The construction of the system involves two essential phases: *structure identification*, which aims to determine the fuzzy rules structure, and the *parameter learning phase* used to tune and optimize the parameters of each fuzzy rule constructed in the structure identification phase. The input-output behavior of a given system can be described by a set of fuzzy rules [26, 45]. These two phases are carried out in sequence with the learning structure phase coming first followed by the learning parameter phase.

1.3 Literature survey

1.3.1 Self-tuning fuzzy control

Here we have reviewed the literature for different approaches of developing self-tuning FLCs: *Yoshida et al.* [51] proposed the gain tuning method assuming all processes as *first-order* systems with dead-time. The input and output SFs are calculated by some empirical relations involving process parameters. However, good control performances for higher order systems cannot be ensured by this technique. *Iwasaki and Morita* [52] considered linear first-order plant models with dead-time in their auto-tuning scheme. Here, the parameters of an assumed plant model are iteratively revised through fuzzy inference using differences between the actual plant features (rise time and overshoot) and the plant model features. The overall performances of the controller will be dependent on the appropriateness of the assumed process model. *Hayashi* [53] designed an auto-tuning fuzzy controller by considering two tuning functions. From the approximate parameters of the identified plant model (first-order lag with dead-time) the input and output SFs are calculated using the concept of *Chien–Hrones–Reswick* tuning rules for a conventional PI controller. Then the crisp consequent parts are modified by fuzzy rules using overshoot and rise time to improve the control performance. *Nomura et al.* [54] proposed the self-tuning method of FLCs, which is a well known gradient decent technique to optimize both the fuzzy antecedent and crisp consequent parts. The controller is tuned iteratively by minimizing the square error between the FLC output and the desired output given by the training data. This method simultaneously modifies the crisp consequent values and, centers and widths

of triangular input fuzzy sets. This off-line tuning method may be very good for time invariant control systems, but its applicability is limited due its dependency on the availability of a reliable set of training data. Performance evaluation of self-tuning fuzzy controller is described by *Daugherty et al.* [55] where the scaling factors of the inputs are changed in the tuning procedure. The FLC has two control inputs: the current error and the change of error. The control action is the change in the manipulated variable. The tuning of the two scaling factors for the two control inputs is done automatically by a fuzzy set of meta rules. The performance measures for tuning are the overshoot, rise time and the amplitude of oscillation of the transient response of the process. By changing the scaling factor of each controller input, the weight given to the input of the controller is changed. For example, if the system response is slower than desired, the effect of the error on the system must be increased. Hence, the error scaling factor is increased. Similarly, if the overshoot or amplitude of oscillation is higher, the effect of the change of error on the controller should be bigger. Hence, the appropriate scaling factor is determined.

Maeda et al. [56] introduced the application of neural nets to the design and tuning of fuzzy system. He described how neural networks (NN) and fuzzy logic have been applied to consumer products. *Zheng* [57] presented a self-tuning fuzzy logic controller (STFLC) which can emulate not only the control experience of human expert, but also the strategies or thinking of the expert to be utilized in developing a fuzzy controller. The remarkable feature of a STFLC is its dynamic knowledge, which is constructed by a multi-layer rule-base. He suggested to tune the parameters of PI-type FLCs in order to their significance; *i.e.*, first parameters with a global effect and then ones with only local effect and, hence, given the maximum importance to the tuning of SFs. *Zheng* did not provide any algorithm for tuning of FLCs, but discussed various factors, their interaction, and their impact on the controller performance that should be considered while designing tuning algorithms for FLCs. Input and output SFs are recommended to be selected from the knowledge of conventional PI-controller parameters, if available, otherwise through trial and error. Simulation result with tuned MFs shows a marginal improvement in transient response of a second-order linear process where tuning resulted in symmetric (triangle) MFs with unequal base. To be more specific, the width of MFs increased around steady state (*i.e.*, $e=0$, $\Delta e=0$) and such MFs contradict the usual practice. Thus, the proposed MFs tuning scheme cannot guarantee improved performance under load disturbance, which is a very important criterion for the performance evaluation of any control system. *Maeda* and *Murakami* [58]

proposed fuzzy rule-based schemes for adjustment of input–output SFs as well as for tuning of control rules for Takagi–Sugeno (TS) model. The fuzzy rule-base for tuning has three sets of rules based on three different performance measures: *overshoot*, *rise time*, and *amplitude*. After the tuning of SFs, the crisp consequent parts of the control rules are modified in each sampling time considering a fuzzy performance index and the deviation of the actual control response from a predefined target response. *He et al.* [59] proposed a scheme for self-tuning of a conventional PID controller using fuzzy rules. The proportional sensitivity, integral time and derivative time are initially calculated using Ziegler–Nichols (ZN) tuning formula. These three parameters are then modified on-line by a single parameter, which is updated by a rule-base defined on error and change of error. It is reported that there is a considerable improvement in the overall performance of the controller over its conventional counterpart. The algorithm is tested by simulation under step changes in the set-point and load disturbance. Results obtained by this method, shows a significant reduction in overshoots of second-order processes with dead time, but at the cost of increased rise times. Neither experimental trials nor any evidence of the controller robustness were examined.

Tönshoff and *Walter* [60] presented a method, which uses neural networks and other intelligent technologies for adjusting the design parameters of FLC for grinding control. Two distinct controllers are used. In case of high deviations between the normal and controlled value, the first controller is activated. Its input may lie in the entire range and it gives the controller output as an absolute value. At a low deviation, an incremental controller, whose input covers only a limited range is used. The fuzzy controller proposed is able to reproduce the behavior of dead-beat controller fully, but it seems to be difficult to implement on-line as it depends on hard computation requirements. *Lee* [61] proposed two augmented versions of the conventional fuzzy PI controller using resetting factors with a view to eliminating the overshoot caused by the accumulation of control input in a fuzzy PI-type controller. The first of the two fuzzy controllers determines the resetting rate based on error and change of error, while the second one uses error and control input. The computation of the resetting factor is driven by a fuzzy rule-base. The controller remarkably improves the transient response of a second-order linear system with integrating element. But the authors, *Mudi* and *Pal* [62] in their paper have clearly shown with extensive simulation conducted on different types of second-order linear as well as nonlinear

systems with and without integration that the controller used by *Lee* with resetting action is almost similar to a conventional fuzzy PD controller.

Lui et al. [63] introduced a novel self-tuning adaptive resolution (STAR) fuzzy control algorithm. One of the unique features in STAR is that the fuzzy linguistic concepts change constantly in response to the states of input signals. This is achieved by modifying the corresponding membership functions. This adaptive resolution capability is used to realize a control strategy that attempts to minimize both the rise time and the overshoot. In this approach, the rule-base of the controller will not change, but the definition of the linguistic concepts adapt constantly according to requirements. This approach has been applied for a simple two input-one output fuzzy controller. Experimental results show the cascaded controller is robust against disturbances and uneven load. Compared to the conventional fuzzy controller, the STAR approach reduces the positional overshoot and also the angular error in steady state. *Ramkumar and Chidambaram* [64] developed fuzzy self-tuning PI controller with the basic idea of parameterize the ZN tuning formula by two parameters α and β and then to use an on-line fuzzy inference mechanism to tune the PI controller parameters *i.e.* proportional gain and reset time. The fuzzy self-tuning method uses the process output error as input and the tuning parameters α and β as outputs. The ranges of membership functions are selected based on the simulation study. The presented fuzzy logic controller is robust to process parameter uncertainties. *Jung et al.* [65] presented a self-tuning fuzzy water level controller based on the real-time tuning of the scaling factors for the steam generator of a nuclear power plant. They proposed a new real-time tuning of the scaling factors. The new tuning method uses a variable reference tuning index and an instantaneous system fuzzy performance. The controller is simulated on the Compact Nuclear Simulator at the Korea Atomic Energy Research Institute. The simulation results show that the proposed tuning method improves the performance of the water level controller. *Chiricozzi et al.* [66] proposed a new gain self-tuning method for PI controllers based on the fuzzy inference mechanism. The purpose is to design a fuzzy rule-based parameter adaptation scheme for non-fuzzy PI controller on-line. The aim of the method is to improve the step response gradually and to assure a certain system overshoot target reached with a reasonable rise time. This algorithm is tested in the permanent magnet synchronous motor drive speed control scheme with different set-points and extreme initial conditions. *Shimajima et al.* [67] proposed a new supervised self-

tuning fuzzy modeling, which consists of some membership functions expressed by the radial basis function with insensitive region. Genetic algorithms (GA) take care of learning. The steepest descent method is also utilized for tuning the shapes of the membership functions and consequent parts of the rules. There are two tuning methods. One is the coarse tuning with GA and the other is the fine-tuning by the gradient descent method. No evidences of experimental trials neither of robustness of this algorithm are provided. Moreover, its on-line applicability is not indicated.

Palm [68] proposed to achieve the optimal adjustment in the input SF with the help of input–output cross-correlation function, though he assigned a higher priority to the tuning of output SF over that of input SFs. Here, the input data’s are assumed to follow a Gaussian distribution whose parameters are unknown. An optimal input SF is obtained by maximizing the cross-correlation function, which is a measure of the statistical dependence between input and output. *Li and Gatland* [69] have given more emphasis on the tuning of input and output SFs than that of MFs or rule-base. They basically suggested a trial and error method for tuning of input and output SFs for a fuzzy PID controller developed from two FLCs in parallel- one is a PI-type and the other is a PD-type.

Miyata and Ohkita [70] proposed a steepest descent based method for the generation of piecewise linear membership functions. In this algorithm, MFs of the premise for each fuzzy rule are tuned independently. Comparing with the conventional triangular form and the Gaussian distribution of MFs, an expansion of the expressiveness is indicated. This greatly reduces the iterative computation. However, the determination of some initial learning coefficients remains unsolved. *Wang and Chai* [71] developed a learning algorithm to train the simplified fuzzy inference network (SFIN), used for implementation of the fuzzy logic controller, to match the given input-output pairs. This learning algorithm firstly considers the FLC as a four-layer feedforward network and secondly uses the chain rule to determine gradients of the output errors of the SFIN with respect to its design parameters. *Routray et al.* [72] introduced a fuzzy logic based approach for the on-line tuning of the control parameters. A satisfactory accuracy of the parameter adaptation is obtained by referring the fuzzy subsets to the normalized values of the variables involved in the fuzzy logic. The tuning rule-base tries to emulate operator experience

on gain tuning. A comparative study has been done with and without tuning, using an electromagnetic transient simulation program. No experimental tests are provided.

Chen and Lin [73] presented a methodology to tune the initial membership functions of a fuzzy logic controller. The membership functions of the controller output are adjusted according to the performance index of sliding mode control, thereby trying to propose a real time simultaneous tuning method. The input variables *i.e.* error and change in error define this performance index. The general gradient method is adopted to alter the output fuzzy set in the direction of the gradient of the performance index. The proposed algorithm has not been tested on experimental systems, and no evidence of its robustness has been provided. *Leu et al.* [74] presented the adaptive fuzzy-neural controllers tuned on-line for a class of unknown nonlinear dynamical systems. To approximate the linearization of the unknown nonlinear dynamical system, the fuzzy approximation is established. Furthermore, the control law and update law are derived to tune on-line both the B-spline membership functions and the weighting factors of the adaptive fuzzy-neural controllers. The superiority of the on-line tuning of both membership functions and weighting factors over the tuning of only the weighting factors is demonstrated. The proposed algorithm has not been tested on practical systems. The robustness property of the controller is also not mentioned. *Takagi et al.* [75] presented a skill-based PID control scheme, which extracts skills of human experts as PID gains. This controller was designed by using a three-layered artificial neural network together with a conventional PID controller. The digital PID controller produces a control signal by using the proportional, integral and derivative actions and the main task of the neural network is to tune the parameters- K_C (proportional gain), T_i (reset time) and T_d (rate time) of the PID controller.

Ying [76] investigated the analytical structure of the Takagi-Sugeno type fuzzy controllers. The TS fuzzy controllers employ a new and simplified TS control rule scheme in which all the rule consequents use a common function and are proportional to one another, greatly reducing the number of parameters needed in the rules. The proposed controller scheme does not match the problem of fuzzy self-tuning exactly, but it is an alternative way to build up a new improved fuzzy controller in the hybrid system using the variable gain as a tunable parameter. *Chung et al.* [77] proposed a self-tuning fuzzy controller with a smart and easy structure. The tuning scheme allows tuning the scaling factors by only seven rules. The aim of the controller is to obtain a

satisfactory performance in term of *rise time*, *overshoot* and *steady-state error* for the step response. The structure of this controller consists of two fuzzy logic controllers: one is a PI-type fuzzy controller at low level directly applied to the process; the other one is the fuzzy supervisory tuner controller which adjusts the scaling factors of each MF of the low level controller. This means that the self-tuning controller adjusts three scaling factors for the three linguistic variables of the PI-type fuzzy controller, *i.e.* scaling factor of error, change of error and change of manipulated variable. This controller shows some robustness but no experimental trials are made.

Mudi and Pal [22] presented a simple but robust model for self-tuning FLCs. According to them, the adaptive tuning of a FLC is based on an on-line adjustment of the output SF of a FLC by fuzzy rules according to the current trend of the controlled process. The rule-base for tuning the output SF is defined based on the error (e) and the change of error (Δe) of the controlled variable using the most common and unbiased membership functions (MFs). The proposed self-tuning technique is applied to both PI and PD-type FLC and tested through simulation experiments on a wide range of different linear and nonlinear second order processes including a marginally stable system. The performance of the proposed self-tuning FLC (STFLC) is compared with the corresponding conventional FLC in terms of several performance measures such as *peak overshoot*, *settling time*, *rise time*, *integral absolute error (IAE)* and *integral-of-time absolute error (ITAE)* in addition to the responses due to step set-point changes and load disturbances. The proposed algorithm has not been tested on any real or actual systems.

Woo et al. [78] proposed a new fuzzy controller structure, namely PID type fuzzy controller by relating to the conventional PID control theory. In order to improve the performance of the transient state and the steady state of the PID type controller further, they developed a method to tune the scaling factors of the PID type fuzzy controller on-line. *Chang et al.* [79] proposed a self-tuning method for a class of nonlinear PID control systems based on Lyapunov approach. Here, the three PID control gains are adjustable parameters and will be updated on-line with a stable adaptation mechanism such that the PID control law tracks certain feedback linearization control, which is previously designed. The stability of closed loop nonlinear PID control system is analyzed and guaranteed by introducing a supervisory control and a modified adaptation law with projection. *Yesil et al.* [80] proposed a self-tuning fuzzy PID type controller for solving the

load frequency control (LFC) problem. The fuzzy PID type controller is constructed as a set of control rules, and the control signal is directly deduced from the knowledge-base and the fuzzy inference. Moreover, there exists a self-tuning mechanism that adjusts the input scaling factor corresponding to the derivative coefficient and the output scaling factor corresponding to the integral coefficient of the PID type fuzzy logic controller in an on-line manner. The self-tuning mechanism depends on the peak observer idea, and this idea is modified and adapted to the LFC problem.

Huang and Chen [81] proposed a novel model-free adaptive sliding controller to suppress the position oscillation of the spring-mass in response to road surface variation. Since the hydraulic actuating suspension system has nonlinear and time-varying behavior, it is difficult to establish an accurate dynamic model for a model-based sliding mode control design. This control strategy employs the functional approximation technique to establish the unknown function for releasing the model-based requirement. In addition, a fuzzy scheme with on-line learning ability is introduced to compensate the functional approximation error for improving the control performance and reducing the implementation difficulty. The important advantages of this approach are to achieve the sliding mode controller design without the system dynamic model requirement and release the *trial and error* work of selecting approximation function. *Pal and Mudi* [23] presented a self-tuning Fuzzy PI controller for the supply air pressure control loop for Heating, Ventilation and Air-Conditioning (HVAC) system. The self-tuning Fuzzy PI controller (STFPIC) adjusts the output scaling factor on-line by fuzzy rules according to the current trend of the controlled process [22]. Comparing with PID and Adaptive Neuro-Fuzzy (ANF) Controllers, results show that STFPIC performances are better under normal conditions as well as when the HVAC system encounters large parameter variations.

1.3.2 Rule extraction and fuzzy modeling

There have been several attempts to identify a rule-based system to characterize the relation between the input and output by various clustering methods [14, 82-91]. Among the different clustering methods, the K-means algorithm has a long history, but is still the subject of current research. The original K-means algorithm was proposed by *MacQueen* [82]. The recent variations of K-means are new incremental versions of K-means proposed by *Dhillon et al.* [83].

K-means is simple and can be used for a wide variety of data types, but is not suitable for all types of data. It cannot handle clusters of different sizes and densities.

The clustering method, called Fuzzy *C*-means (FCM), was introduced by *Dunn* [84] in 1973. An extensive discussion on of fuzzy clustering, including a description of fuzzy *C*-means and formal derivations of the formulas can be found in the book on fuzzy cluster analysis by *Hoppner et al.* [85]. FCM has much the same strengths and weaknesses as K-means, although it is somewhat more computationally intensive.

Sugeno and *Yasukawa* [14] used FCM algorithm to determine the structure of the system. The number of clusters, c , (*i.e.*, the number of rules) is determined by minimizing the validity function $S(c)$:

$$S(c) = \sum_{k=1}^n \sum_{i=1}^c (\mu_{ik})^m \left(\|y_k - v_i\|^2 - \|v_i - \bar{\mathbf{y}}\|^2 \right)$$

In the above equation, n = total number of data points to be clustered, c = number of clusters, μ_{ik} = membership value of the k^{th} data point, y_k , to the i^{th} cluster, m = the fuzzy exponent used in the FCM algorithm, $v_i = i^{\text{th}}$ cluster center and $\bar{\mathbf{y}}$ = the grand mean vector of all data points. The number of clusters is determined by minimizing $S(c)$ with respect to c . For each c , the FCM centroids V and the partition matrix U are first obtained and then used them to compute $S(c)$. The value $c = c'$ at which $S(c)$ attains the minimum value is taken as the right number of rules. The membership values of the input clusters were obtained by projecting the membership values of the extracted clusters on the input axes. They approximated these clusters by trapezoidal fuzzy sets and used a heuristic method to adjust the parameters of the trapezoidal MFs. *Yoshinari et al.* [86] clustered using fuzzy *C*-linear varieties, where each cluster is interpreted as a multi-dimensional local linear relationship. *Sin* and *de Figueiredo* [92] used FCM for clustering and suggested to use the *Xie–Beni index* [93] for selecting the number of clusters. Each cluster obtained from input-output dataset is represented by a *Takagi–Sugeno* type rule [17]. They estimate the consequent functions by minimizing an objective function E_k . For a given data set

$x_k \in \mathfrak{R}^p$, the firing strength of the i^{th} rule is computed as $\alpha_i = \mu_k \left(\mathbf{x}^* = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} = u_i(\mathbf{x}) \end{pmatrix} \right)$. First they

compute the output of the i^{th} rule as $y=u_i(x)$. This y is then augmented to x and the augmented vector is used to find the firing strength of the i^{th} rule.

Krishnapuram and *Keller* [94] introduced a possibilistic approach to clustering *i.e.*, the possibilistic C-means (PCM). *Yager* and *Filev* [87] used the mountain clustering method (MCM). The optimal number of clusters is chosen based on a user-defined threshold on the mountain potential. They investigate the cluster centroid $\mathbf{v}_i^* = \begin{pmatrix} \mathbf{v}_i^x \in \mathfrak{R}^p \\ \mathbf{v}_i^y \in \mathfrak{R} \end{pmatrix}$, $i = 1, 2 \dots c$ from a

MISO system and converted it into a fuzzy rule of the form: If x is CLOSE to \mathbf{v}_i^x then y is CLOSE to \mathbf{v}_i^y . *Chiu* [89] proposed Subtractive Clustering Method (SCM), where each data point is considered as a potential cluster. After calculating the potential of all data points, the point with the highest potential is selected as first cluster center. Then after revision, the data point with the highest remaining potential is selected as second cluster center, and so on. If the i^{th} cluster center x_i^* is found in the group of data for class $c1$, then the corresponding rule can be written as R_i : *If x is CLOSE to x_i^* then class is $c1$* . The fuzzy set CLOSE to x_i^* is modeled by a Gaussian type MF. No explicit cluster validity index is used here. The number of clusters SCM settles to is dependent on the parameters of the mountain function. *Nakamori* and *Royke* [95] used the objective variables and a subset of explanatory and conditional variables. They used a hyper-ellipsoidal crisp clustering algorithm, which dynamically determines the number of clusters based on several user-defined parameters. Each cluster is then translated into a TS rule with linear function for the consequents, and if the TS model is not satisfactory, they use the Mamdani-Assilian (MA) model.

Babuska and *Kaynak* [90] clustered using *Gustafson–Kesel’s* [96] fuzzy C-means (GKFCM) algorithm for TS modeling. Then the compatible cluster merging criterion of *Krishnapuram* and *Freg* [97] is used to merge compatible clusters. After this, GKFCM algorithm is again run with the reduced number of clusters. The process is repeated until no more clusters can be merged. The fuzzy partition thus obtained is used to generate MFs for the antecedent variables. Finally, the consequent parameters are estimated using the least square technique. *Kaynak et al.* [91] and *Babuska et al.* [98] discussed methods for system optimization through removal and merging of fuzzy sets extracted by clustering of input-output datasets using a measure of similarity between

fuzzy sets. *Runkler* and *Palm* [99] designed a regular fuzzy system with complete rule-base, defined with equispaced unimodal MFs and the first order TS consequents. *Delgado et al.* [100] presented several methods for fuzzy modeling that use clusters with FCM. They also proposed to cluster input and output separately to generate fuzzy sets for antecedents and for the consequents. Several cluster validity indices [100-102, and 91] are used to get a good choice for the number of clusters (rules). Finally, they optimized the system with respect to root mean square error (RMSE) using genetic algorithm (GA). *Wong* and *Chen* proposed a hybrid clustering and gradient descent approach for constructing a multi-input fuzzy model automatically [103]. *Pal et al.* applied FCM for their rule extraction scheme in nonlinear system [104-106]. FCM is widely used in data mining process [107], but, it is very difficult to partition large number of datasets effectively and, also, often datasets are affected by noise and outliers [108, 109]. Choice of the right value for the number of clusters is very important for successful identification of the system and in the case of FCM there are many validity indices that have been used for this purpose. Proper selection of validity indices is very important in the case of FCM and there are no such guidelines for that.

Krishnapuram et al. [110] proposed a relational fuzzy C-means (RFCM) algorithm for partitioning large datasets. *Dave* and *Sen* introduced a new algorithm for relational data, named fuzzy relational data clustering (FRC) algorithm, having an identical objective function as RFCM, but without the restriction of RFCM [111]. *Frigui et al.* introduced a semi-supervised approach for clustering and aggregating of relational data and they assumed that data is represented by multiple dissimilarity matrices form [112, 113]. Some improvement has also been done using Kernel-based fuzzy C-means clustering for large noisy datasets [114, 115]. When the kernel function integrates with Fuzzy C-means, it behaves almost like support vector machine (SVM). SVM is a useful method for generation of fuzzy rules automatically with fuzzy singletons; though the numbers of fuzzy rules are generally high [116]. SVM faces problem of speed and size in case of training huge datasets.

Usually, conventional model-based clustering methods described above, may not be able to maintain the topological relationship among different clusters, after the clustering procedure [117]. Some work has been reported in this field to preserve topological relationship using conventional model-based clustering methods. *Celikyilmaz* and *Turksen* [118] applied FCM

initially and then re-clustered the datasets using improved fuzzy clustering (IFC) to capture interactive membership values. Apart from maintaining topological relationship, selection of proper clustering index is also a major concern in the supervised clustering methods. *Kohonen* suggested an unsupervised clustering method self-organizing map (SOM) that can maintain the topological relationship between the clusters [119]. The use of hierarchical agglomerative clustering using K-means is investigated by *Vesanto* and *Alhoniemi* [120]. As with incremental K-means, data objects are processed one at a time and the closest centroid is updated. They used SOM to produce the prototypes. Unlike K-means, SOM imposes a topographic ordering on the centroids and nearby centroids are also updated.

Yang and *Bose* [121] used SOM to generate fuzzy membership function directly during the learning phase. A key step in their proposed scheme is to combine the input feature vector (\mathbf{x}_n) with the vector (\mathbf{y}_n) coding the class labeling information. The dimensions of \mathbf{x}_n and \mathbf{y}_n are respectively, the number of input features d and the number of class labels c . That is, a new vector \mathbf{z}_n of dimension $c+d$ is constructed. In the learning phase, the newly constructed \mathbf{z}_n will be the input feature vector to SOM and after the learning phase, the SOM can be considered as a membership generation network just like its supervised counterpart, *i.e.*, the feed-forward multilayer neural network trained with a supervised learning algorithm. In the retrieving phase, the input feature vector is only \mathbf{x}_n . Therefore, the input feature vector will find the best matching neuron by calculating the minimum Euclidean distance, considering only the weight sub-vector related to input features.

Hashimoto et al. [122] applied SOM-based clustering method for nonlinear system identification. SOM belongs to the class of unsupervised learning networks and the network produces a low-dimension representation of the input space that preserves the ordering of the original structure of the network [123, 124]. It projects high-dimensional data onto a low-dimensional grid. The projected data preserves the topological relationship of the original data; therefore, this ordered grid could be used as a convenient visualization surface for showing various features of the training data. The SOM possesses a regular structure, which is readily extendable and has been successfully scaled-up to implement different real world applications. SOM can be used as an efficient rule-extraction scheme as it not only clusters large data but also

provides useful information about clustered data [125]. We have elaborated this topic in *chapter-5*.

1.4 Objectives of the thesis

Fuzzy controllers do not have well defined tuning mechanisms. To overcome this tuning problem an on-line gain adjustment scheme has been developed using fuzzy *if-then* rules [22]. In this self-tuning scheme gain will be automatically adjusted according to the process trend.

- ✚ Our initial objective is to test the performance of this self-tuning fuzzy controller in simulated as well as in real time processes.

To further modify this tuning scheme:

- ✚ We plan to develop a relay feedback tuning approach to parameterize the output gain of a self-tuning fuzzy controller according to the process under control.
- ✚ In place of fuzzy rule-based tuning, we attempt to develop a non-fuzzy auto-tuning method based on process dynamics and thereby reduced the number of rules significantly.

The main idea of fuzzy modeling is to describe the input-output behavior of a given system by a set of fuzzy rules. But, there are some cases where there is no expert, or due to highly complex and ill-defined nature of the system expert cannot construct any linguistic model of the system. In such cases, our objective is to make a proto-type of the system from the available input-output data, which is expected to behave like the original system. To achieve this goal:

- ✚ We plan to develop a SOM-based rule extraction scheme that can pick up essential rules to make an effective linguistic model of the system from the available input-output data.
- ✚ We apply this developed scheme to identify a highly nonlinear gain surface and control surface of self-tuning fuzzy controller.
- ✚ To evaluate the effectiveness of the identified linguistic model we observe its performance on different simulated and real time processes.
- ✚ We attempt to reduce the MFs / rules using similarity measure among fuzzy sets to make the identified model simpler.

1.5 Motivation of the present work

Most of the industrial processes under automatic control are nonlinear, complex and higher order systems and most of them have considerable dead-time. In industrial environment these parameters are often varied with time. Among the different parameters in a physical system, dead-time is the most difficult one, as it delays the required control action to a process. Though, FLC tries to incorporate these parameter variations of process, but it may fail to give desired result due to its limited number of *if-then* rules. A FLC consists of various parameters like MFs, linguistic data-bases, rule-bases, fuzzification and defuzzification strategy, which are not well defined till date. Considering all these parameters, designing an optimal FLC analytically becomes very difficult. These limitations of the conventional FLCs motivated us to design on-line tuning schemes for fuzzy controller that can control real time systems satisfactorily.

The first step in designing the controller is to model the plant. System identification is the process of building models of dynamical systems from input-output data. However, there are many ill-defined systems; the modeling of such systems is very difficult in absence of experts' knowledge about the data pattern. This fact motivated us to develop an unsupervised mechanism that can pick up required number of *if-then* rules along with their antecedent and consequent MFs from the input-output data for developing a prototype of the original system.

1.6 Thesis organization

Next we provide brief chapter-wise information about the content of this thesis.

Chapter-1 gives an overview about the scope and objectives of the thesis. The different background components of our thesis are discussed briefly in this chapter. We review the literature for different approaches of developing self-tuning FLCs. This chapter provides a survey on different techniques used to identify the unknown systems from the available input-output data. The importance of clustering techniques for rule extraction and system modeling is also reviewed.

Chapter-2 provides tuning procedure of fuzzy controller and its applications to practical systems. In this chapter, an on-line self-tuning scheme for fuzzy controller is discussed. The proposed self-tuning Fuzzy PI controller (STFPIC) and self-tuning Fuzzy PD controller

(STFPDC) adjust their output scaling factors on-line by a fuzzy gain modifier (β) according to the current trend of the controlled process. Later, the performance of the proposed control scheme is investigated in simulated process and also in real time process. A comparative study of the investigated processes is made with respect to different performance measures such as rise time (t_r), settling time (t_s), peak overshoot (% OS), integral absolute error (IAE) and integral-of-time multiplied absolute error ($ITAE$) and integral square error (ISE).

Chapter-3 is an extension to the work proposed in *chapter-2*. Here the fuzzy output gain modifier (β) is further augmented by a multiplicative factor (α), which is directly related to the system dynamics and derived by relay feedback experiment. STFPIC uses process specific appropriate gain multiplicative factor (α) instead of a fixed numerical value. The modified STFPIC uses only 50 rules in place of 98 rules used earlier [22, 23]. Robustness of the proposed controller is demonstrated by testing on a wide range of processes including nonlinear and marginally stable systems with a considerable variation in dead-time. This chapter also shows a real time implementation of the proposed controller for the speed control of DC motor.

Chapter-4 introduces a new auto-tuning scheme for PD-type fuzzy controller. Instead of using large number of fuzzy *if-then* rules for gain adjustment as demonstrate in previous chapters, here, we propose a simple non-fuzzy scheme for the design of an adaptive fuzzy PD controller (AFPDC). In the proposed AFPDC, output SF of the controller continuously updates by a non-fuzzy multiplicative factor β , which is directly related to the normalized error and normalized change of error of the system under control. An important point of our proposed scheme is that it significantly reduces the number of rules from the previous cases. The proposed scheme is applied on different second order integrating, nonlinear and non-minimum phase systems with variable dead-time. In this chapter, twin adaptive fuzzy controllers are also implemented for a laboratory scale crane (FEEDBACK, UK) to control the position of the trolley crane and swing angle of load more precisely. Note that, it is very difficult to transfer a suspended load along a pre-specified path by maintaining minimum swing angle and transfer time of load.

Chapter-5 presents a new scheme of fuzzy modeling, based on the Kohonen self-organizing map [119]. The main idea of the fuzzy modeling is to describe the input-output behavior of a given system by a set of fuzzy rules. In this proposed work, we can find suitable computational

(linguistic) model for any unidentified system through structure identification and parameter estimation stages. For structure identification, Self-Organizing Map (SOM) based clustering technique is used. Here, the system identification method is integrated with rule extraction method in such a way that it can pick up essential rules for an unknown system. The scheme has been successfully tested by identifying the rules required to realize the gain factor of a self-tuning fuzzy PI controller, which is highly nonlinear in nature. Identified gain rules along with initial control rules are used to investigate different linear, nonlinear and marginally stable systems. Comparative studies of simulation results ensure that the proposed rule extraction technique is capable to model any complex process successfully. In this chapter, an attempt has been made for further MF reduction using similarity measure. The proposed modeling scheme is also applied on a function approximation problem to justify its generalization capability.

Chapter-6 demonstrates the effectiveness of the rule extraction technique in two crucial industrial processes. The developed SOM based control scheme in *chapter-5* is used here to control the position as well as swing angle of a laboratory based overhead crane. The effectiveness of the proposed approach is also tested in an industrial pressure control loop. The results show that even with significant reduction of rule-base, the controllers' exhibit effective and improved performance in real time systems compared to its conventional fuzzy counterpart.

Chapter-7 concludes by highlighting the contributions of this thesis together with some scope for future work.

CHAPTER 2

Self-Tuning fuzzy logic controller and its application to HVAC and inverted pendulum

2.1 Introduction

Fuzzy logic controllers have been implemented successfully in a number of complex and nonlinear processes [12]. Application of fuzzy logic for inverted pendulum control still by far remains most popular, as they do not require precise knowledge of the system parameters [6]. Among the proportional-integral (PI), proportional-derivative (PD), and proportional-integral-derivative (PID) type of FLCs, just like the widely used conventional PI controllers [126] in process control systems, PI-type FLCs are most common and practical followed by the PD-type FLCs because proportional (P) and integral (I) actions are combined in the proportional-integral (PI) controller to take advantages of the inherent stability of proportional controllers and the offset elimination ability of integral controllers. The performance of PI-type FLCs for higher order systems, systems with integrating elements or large dead-time, and also for nonlinear systems may be very poor due to large overshoot and excessive oscillation [127]. PD-type FLCs are suitable for a limited class of systems [128] and they are not recommendable in presence of measurement noise and sudden load disturbances. PID-type FLCs are rarely used due to the difficulties associated with the generation of an efficient rule-base and the tuning of its large number of parameters.

For the successful design of FLCs proper selection of input and output scaling factors are very important. Among the various tunable parameters, SFs have the highest priority due to their global effect on the control performance. Unlike conventional control, which is based on mathematical model of a plant, a FLC usually embeds the intuition and experience of a human

operator and sometimes those of designers and researchers. While controlling a plant, a skilled operator always tries to manipulate the controller output (manipulated variable) with a view to minimize the error within the shortest possible time. By analogy with the human operator, the output SF should be considered a very important parameter of the FLC since its function is similar to that of the controller gain. Moreover, it is directly related to the stability of the control system. So the output SF should be determined very carefully for the successful implementation of a FLC.

Designing a general tuning method for FLCs is a very difficult task because, it have no fixed structures like conventional PI, PD, and PID controllers. FLCs are knowledge based system and its different parameters like membership functions, number of linguistic values, rule-bases, fuzzification strategy, and defuzzification method are not well defined till date. Probably due to these reasons designing an optimal FLC analytically becomes very difficult. These limitations of the conventional FLCs motivated researchers [22] to investigate methods of tuning based on experts' knowledge rather than mathematical models. In their scheme, the FLC is tuned on-line by dynamically adjusting its output SF by a gain updating factor. The self-tuning mechanism is applied to both PI and PD-type FLCs for simulation experiments with nonlinear processes [22, 129]. The proposed FLCs are also tested for marginally stable and unstable systems where well-known Ziegler–Nichols tuned PI or PID controllers exhibit very poor performance [130, 131]. The simulation results show that in each case the proposed control scheme outperforms its conventional counterpart [22]. However, its performance is not yet investigated in any practical system. In this chapter, we demonstrate this self-tuning scheme in a real time *Inverted Pendulum* and also in a *Heating, Ventilation and Air-Conditioning* system.

Heating, Ventilation and Air-conditioning (HVAC) system is a nonlinear and time variant system. It is difficult to achieve desired tracking control performance since appropriate tuning or adjusting PID parameters on-line is a difficult problem. Industrial pressure control systems constitute the heart of many process plants [126]. The performance of the Self-tuning Fuzzy PI controller is investigated in the supply air pressure control loop [23] for HVAC system.

The effectiveness of the self-tuning scheme is also tested in a laboratory based inverted pendulum, which is a nonlinear and highly unstable system [6]. Its basic structure consists of two

flexible pendulum arms mounted to a cart on a moving rail. In the absence of a stabilizing controller, the pendulum arms, which have their centre of mass above their pivot point, are unable to maintain their upright position. Control of an inverted pendulum is a very common but difficult control engineering problem, based on flight simulation of rockets and missiles during the initial stages of flight, wherein the aim is to stabilize the inverted pendulum such that the position of the carriage on the track is controlled quickly and accurately. The pendulum should remain erected in its inverted position during such movements.

2.2 Types of fuzzy logic controllers

2.2.1 PD type fuzzy controller (FPDC)

The equation of a conventional PD-controller is

$$u(t) = K_p \left(e(t) + T_d \frac{de(t)}{dt} \right) \quad (2.1)$$

Following backward difference rule, the discrete form of *equation* (2.1) at k^{th} sampling instant can be written as

$$u(k) = K_p \left(e(k) + T_d \frac{e(k) - e(k-1)}{\Delta t} \right)$$

$$\text{or, } u(k) = K_p e(k) + K_D \Delta e(k).$$

Where, $K_D = \frac{K_p T_d}{\Delta t}$ is the derivative gain, $\Delta e(k) = e(k) - e(k-1)$ and Δt is the sampling time.

Thus the linguistic variables of the rule antecedent (*if*-part of a rule) of FPDC are:

- error, denoted by $e(k)$, where $e(k) = r(k) - y(k)$
- change-of-error, denoted by $\Delta e(k)$

Where $y(k)$ and $r(k)$ are the system output and the set-point respectively.

The control output, denoted by $u(k)$ represents the linguistic variable of the rule-consequent (*then*-part of the rule).

Then a PD-like FLC consists of rules with following linguistic description:

If the value of *error* (e) has the property of being <linguistic value> and the value of *change-of-error* (Δe) has the property of being <linguistic value>; then the value of *control output* (u) has the property of being <linguistic value>.

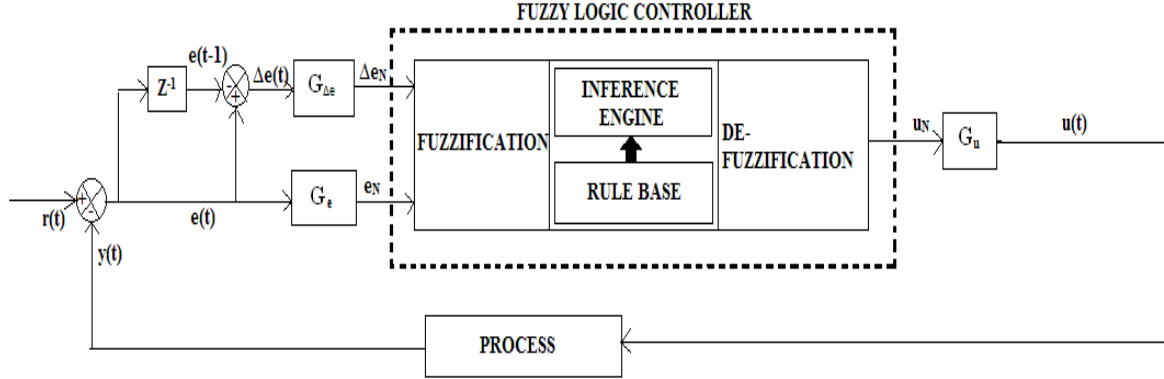


Figure 2.1: Block diagram of PD type fuzzy logic controller.

2.2.2 PI type fuzzy controller (FPIC)

A conventional PI-controller uses an analytical expression of the following form to compute the control action

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(t) dt \right) \quad (2.2)$$

The discrete-time version of the *equation* (2.2) at k^{th} and $(k-1)^{\text{th}}$ instant respectively can be written as

$$u(k) = K_p e(k) + K_I \sum_{i=0}^k e(i) \quad (2.3)$$

$$u(k-1) = K_p e(k-1) + K_I \sum_{i=0}^{k-1} e(i) \quad (2.4)$$

Where, $K_I = \frac{K_p \Delta t}{T_i}$ is the integral gain and Δt is the sampling time.

Now the incremental form of the discrete PI controller can be written as

$$\Delta u(k) = u(k) - u(k-1);$$

$$\Delta u(k) = K_p \{e(k) - e(k-1)\} + K_I \left\{ \sum_{i=0}^k e(i) - \sum_{i=0}^{k-1} e(i) \right\};$$

$$\text{or, } \Delta u(k) = K_p \Delta e(k) + K_I e(k) \quad (2.5)$$

Thus the linguistic variables of the rule antecedent (*if*-part of a rule) of a FPIC are

- error, denoted by $e(k) = r(k) - y(k)$
- change-of-error, denoted by $\Delta e(k) = e(k) - e(k-1)$

Finally, the control output, denoted by $u(k)$ is obtained by

$$u(k) = u(k-1) + \Delta u(k) \quad (2.6)$$

In equation (2.6), $\Delta u(k)$ is the incremental change in controller output. We emphasize here that this accumulation (2.6) of controller output takes place outside the FLC and is not reflected in the rules themselves. On the other hand, if the output of the FLC is $u(k)$ (not $\Delta u(k)$) and there is no accumulation of controller output, then it becomes a PD-type FLC.

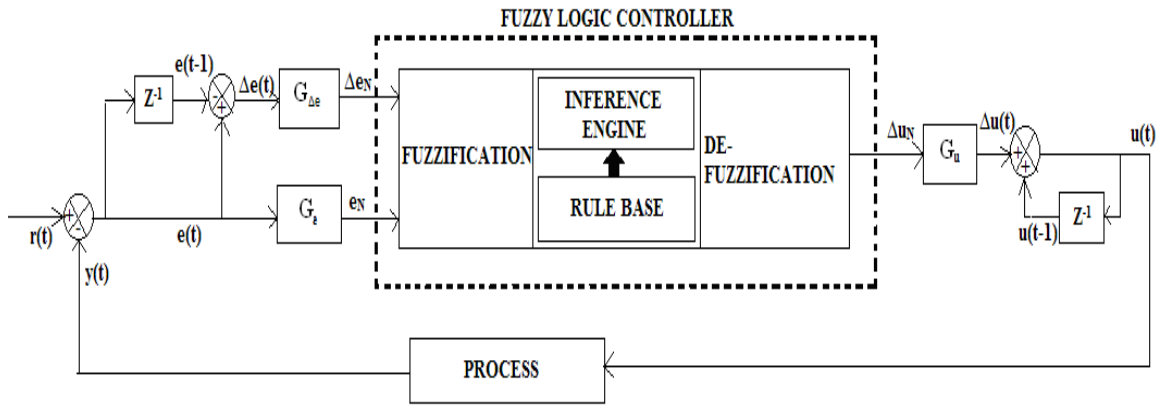


Figure 2.2: Block diagram of PI type fuzzy logic controller.

A PI-like FLC (FPIC) consists of rules with following linguistic description:

If the value of error (e) has the property of being<linguistic value> and the value of change-of-error (Δe) has the property of being<linguistic value>; then the value of change in control output (Δu) has the property of being<linguistic value>.

2.3 The proposed self-tuning scheme of FLCs

In spite of a number of merits, there are many limitations while designing a fuzzy controller, since there is no standard methodology for its various design steps, and no well-defined criterion for selecting suitable values for its large number of tunable parameters. Attempts are persistent in

the research world to obtain a superior controller; as choosing the correct set of rules and scaling factor is not an easy task in order to fine tune the fuzzy logic controller [22, 25]. Therefore, tuning of a large number of FLC parameters can be a tough task [21]. Our objective here is to tune only the output SF for given input SFs to achieve better control performance, though we know, the characteristics of a PI or PD type FLC depends on both input and output SFs [57]. The gain (output SF) of the FLC is adjusted on-line according to the current states of the process, thereby making them self-tuning FLC. The block diagram of the proposed self-tuning FLC is shown in Fig. 2.3, which basically have two major blocks with same inputs - **Fuzzy logic controller block** and **Gain tuning block**. The design components of STFPIIC are described next.

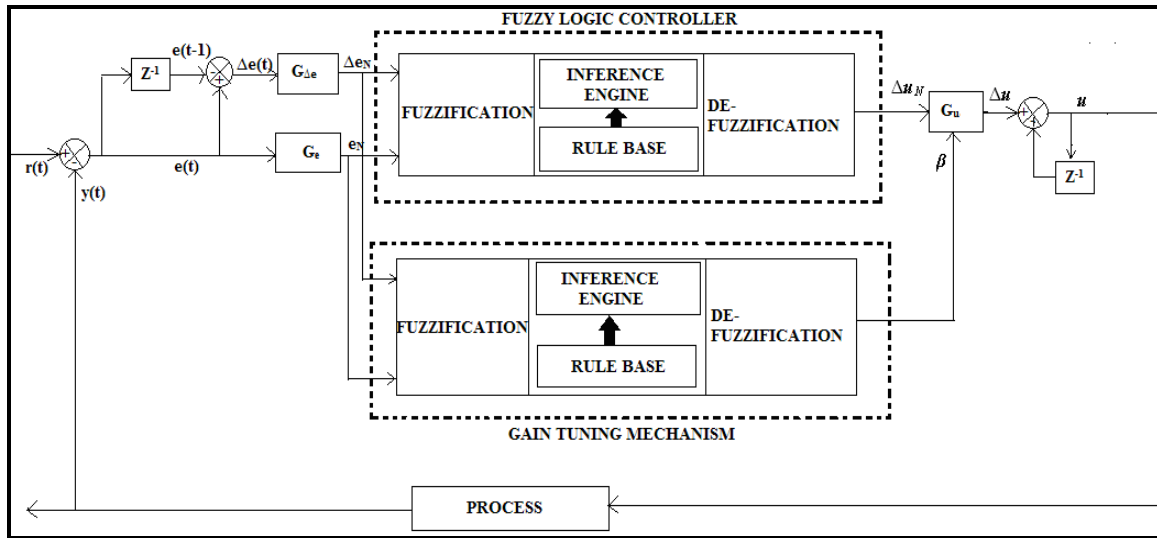


Figure 2.3: Block diagram of the proposed self-tuning PI-type FLC (STFPIIC).

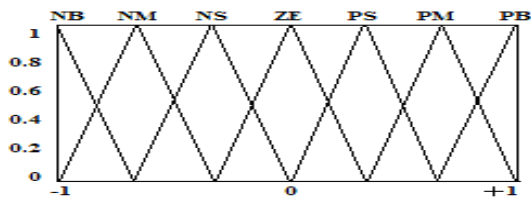


Figure 2.4: Membership functions of inputs (e , Δe) and output (Δu).

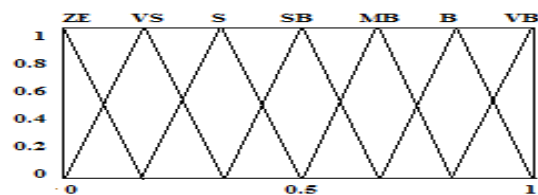


Figure 2.5: Membership functions of gain updating factor, β .

2.3.1 Membership functions

There can be different types of MFs, but for simplicity, here we have used triangular type of MFs with equal base and 50% overlap with neighboring MFs. As shown in Fig. 2.4 and Fig. 2.5, the

MFs for e , Δe and $u/\Delta u$ are defined on the common interval $[-1, 1]$; whereas the MFs for the gain updating factor (β) is defined on $[0, 1]$ respectively. In both the cases, we have divided the input and output spaces into 7 fuzzy regions and assigned membership functions accordingly. The term sets (shown in Table 2.1) of e , Δe and $u/\Delta u$ contain the same linguistic expressions for the magnitude part of the linguistic values, *i.e.*, $LE = L\Delta E = L\Delta U = \{NB, NM, NS, ZE, PS, PM, PB\}$ as shown in Fig. 2.4. Similarly the MFs for gain updating factor is mapped into 7 fuzzy regions $\{ZE, VS, S, SB, MB, B, VB\}$, shown in Fig. 2.5.

Table 2.1: Linguistic expressions

Linguistic Expression	Stands for	Linguistic Expression	Stands for
NB	Negative big	ZE	Zero
NM	Negative medium	VS	Very small
NS	Negative small	S	Small
ZE	Zero	SB	Small big
PS	Positive small	MB	Medium big
PM	Positive medium	B	Big
PB	Positive big	VB	Very big

2.3.2 Scaling factors

The scaling factors also known as gains, which describe the input normalization and output denormalization, play an important role similar to that of the gain coefficients in a conventional controller. Fig. 2.3 shows that the output SF of the fuzzy controller is modified by a self-tuning mechanism. The MFs for both scaled inputs (e_N and Δe_N) and output (Δu_N or u_N) of the controller have been defined on the common interval $[-1, 1]$. The values of the actual inputs (e and Δe) are mapped onto $[-1, 1]$ by the input SFs G_e and $G_{\Delta e}$, respectively. Actually, suitable values of SFs are chosen based on the knowledge of process control or by trial and error method. The relationships between the SFs and the input and output variables of the self-tuning fuzzy controller are as follows:

$$\begin{aligned}
 e_N &= G_e e \\
 \Delta e_N &= G_{\Delta e} \Delta e \\
 \Delta u &= (\beta G_u) \Delta u_N \text{ (for STFPIC)}
 \end{aligned} \tag{2.7}$$

$$u = (\beta G_u)u_N \text{ (for STFPDC)} \quad (2.8)$$

Where G_e and $G_{\Delta e}$ are input scaling factors and G_u is output scaling factor.

2.3.3 The rule-bases

Before designing a rule-base for any system, expert should have proper understanding and knowledge about its characteristics. For example, in this section, we have designed rule-bases for a typical second order under-damped system. The linguistic values of the fuzzy knowledge-based system, are expressed as tuples of the form {value sign, value magnitude}, e.g., {positive big}, {negative small}, etc. The value sign can either be positive or negative and the value magnitude can be expressed by linguistic terms such as Big, Medium, Small, etc. In this case, the rule-base is denoted by the two rule matrix in the form of {NB, NM, NS, ZE, PS, PM, PB} and {ZE, VS, S, SB, MB, B, VB}.

From the response of a typical second order under-damped system, it is revealed that linguistic value of error with a “*positive sign*” means that the current process output has a value below the set-point (r). On the other hand, linguistic value of $e(k)$ with a “*negative sign*” means that the current value of $y(k)$ is above the set-point. Linguistic value of change of error with a “*negative sign*” means that the current process output $y(k)$ has increased when compared with its previous value $y(k-1)$. Linguistic value of $\Delta e(k)$ with a “*positive sign*” means that $y(k)$ has decreased its value when compared to $y(k-1)$. A linguistic value of “*zero*” for $e(k)$ means that the current process output is at the set-point. A “*zero*” for $\Delta e(k)$ means that the current process output has not changed from its previous value.

Few possible combinations of positive/negative values of $e(k)$ and $\Delta e(k)$ in the antecedent part of the rule are discussed below:

- The combination (positive $e(k)$, negative $\Delta e(k)$) means that the current process output $y(k)$ is below the set-point, since $e(k) = r(k) - y(k) > 0$, and increasing, since $\Delta e(k) = -(y(k) - y(k-1)) < 0$. Thus the current process output is approaching the set-point from below.
- The combination (negative $e(k)$, positive $\Delta e(k)$) means that the current process output $y(k)$ is above the set-point and decreasing. Thus the process output is approaching the set-point from above.

- The combination (negative $e(k)$, negative $\Delta e(k)$) means that the current process output $y(k)$ is above the set-point and increasing. Thus the process output is moving further away from the set-point and approaching overshoot.
- The combination (positive $e(k)$, positive $\Delta e(k)$) means that the current process output $y(k)$ is below the set-point and decreasing. Thus the process output is moving further away from the set-point and approaching undershoot.

The term sets of $e(k)$, $\Delta e(k)$ and $\Delta u(k)$ all have been chosen equal in size and contain the same linguistic expressions. We can present the rule-base for the controller in a tabular form as shown in Table 2.2. For example, the cell defined by the intersection of the row and the column represents a rule such as,

If $e(k)$ is PM and $\Delta e(k)$ is NS then $\Delta u(k)$ is PS

In rule antecedent part, $e(k)$ is positive and $\Delta e(k)$ is negative means that the current process output is approaching the set-point from below. Thus, since the process output is moving in the direction of the set-point, the rule consequent prescribes a small $\Delta u(k)$ to be added to the previous control output so that $y(k)$ is moved further up in the direction of the set-point without going above it.

Table 2.2: Fuzzy rules for computation of Δu

$\Delta e/e$	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NM	NS	NS	ZE
NM	NB	NM	NM	NM	NS	ZE	PS
NS	NB	NM	NS	NS	ZE	PS	PM
ZE	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PS	PM	PB
PM	NS	ZE	PS	PM	PM	PM	PB
PB	ZE	PS	PS	PM	PB	PB	PB

Gain rule- Similarly, as shown in Table 2.3, the cell defined by the intersection of the row and the column represents a gain rule such as,

If $e(k)$ is PB and $\Delta e(k)$ is NM then β is VS

With a view to improving the overall control performance, we use the rule-base in Table 2.3 for computation of β . It is designed in conjunction with the rule-base in Table 2.2. Some important considerations have been taken into account for determining the rules. For example, to make the controller produce a lower overshoot and reduce the settling time, the controller gain is set at a small value when the error is big (either +ve or -ve), but $e(k)$ and $\Delta e(k)$ are of opposite signs. For example, If $e(k)$ is PB and $\Delta e(k)$ is NS then β is VS or if $e(k)$ is NM and $\Delta e(k)$ is PM then β is S. To minimize the effects of delayed control action due to inherent process dead-time or measuring lag such small gain is essential to maintain the controller performance within the acceptable limit, especially when the process dead-time becomes considerably large. Observe that when the error is big but $e(k)$ and $\Delta e(k)$ are of the same sign (*i.e.*, the process is now not only far away from the set-point but also it is moving farther away from it), the gain should be made very large to prevent from further worsening the situation. This has been realized by rules of the form: *If $e(k)$ is PB and $\Delta e(k)$ is PS then β is VB or if $e(k)$ is NM and $\Delta e(k)$ is NM then β is VB.* Depending on the process trend, there should be a wide variation of the gain around the set-point (*i.e.*, when $e(k)$ is small) to avoid large overshoot and undershoot. For example, overshoot will be reduced by the rule *If $e(k)$ is ZE and $\Delta e(k)$ is NM then β is B.* This rule indicates that the process has just reached the set-point but it is moving away upward from the set-point rapidly. In this situation, large gain will prevent its upward motion more severely resulting in a smaller overshoot. Similarly, a large undershoot can be avoided using the rules of the form: *If $e(k)$ is NS and $\Delta e(k)$ is PS then β is VS.* This type of gain variation around the set-point will also prevent excessive oscillation and as a result the convergence rate of the process to the set-point will be increased. Note that unlike conventional FLCs, here the gain of the proposed controller around the set-point may vary considerably depending on the trend of the controlled process. Such a variation justifies the need for variable SF.

Further modification of the rule-base for β in Table 2.3 may be required, depending on the type of response the control system designer wishes to achieve. It is very important to note that the rule-base for computation of β will always be dependent on the choice of the rule-base for the controller. For example, the rule-base in Table 2.3 is justified and defined for the controller rule-

base in Table 2.2. Any significant change in the controller rule-base may call for changes in the rule-base for β accordingly.

Table 2.3: Fuzzy rules for computation of β

$\Delta e/e$	NB	NM	NS	ZE	PS	PM	PB
NB	VB	VB	VB	B	SB	S	ZE
NM	VB	VB	B	B	MB	S	VS
NS	VB	MB	B	VB	VS	S	VS
ZE	S	SB	MB	ZE	MB	SB	S
PS	VS	S	VS	VB	B	MB	VB
PM	VS	S	MB	B	B	VB	VB
PB	ZE	S	SB	B	VB	VB	VB

2.3.4 The self-tuning mechanism

From section 2.3.2, it is found that the effective gain of the self-tuning controller is βG_u . Though the value of G_u is constant for PI or PD type of FLCs, but the gain of the self-tuning FLC does not remain constant, when the controller is in operation, rather it is modified in each sampling time by the gain updating factor β , depending on the trend of the process output. This on-line gain variation updates the controller output according to the desired performance specifications. The functional relationship of β can be viewed as, $\beta(k) = f(e(k), \Delta e(k))$, where, ' f ' is a nonlinear function of e and Δe , which is described by the rule-base shown in Table 2.3.

The variation of β with e and Δe is shown in the surface plot Fig. 2.6, which is seen to be highly nonlinear in nature. For example, if error is *positive big* and change of error is *negative big* then the system is moving fast toward the set-point and, hence, gain should be kept very small to avoid possible large overshoot by using rule-base such as.

If $e(k)$ is PB and $\Delta e(k)$ is NB then β is ZE

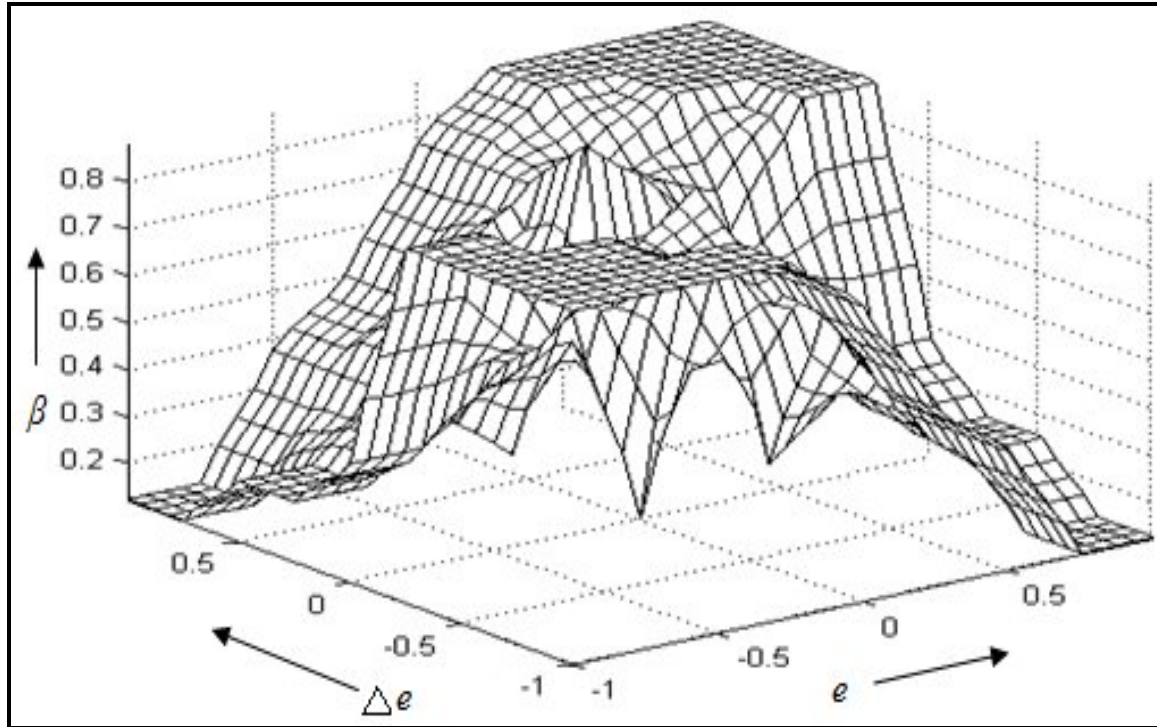


Figure 2.6: Gain surface of $\{e, \Delta e, \beta\}$ with 49 gain rules.

The following steps can be used for tuning of the proposed FLC:

Step 1: Tune the SFs of the self-tuning FLC without the gain tuning mechanism for a given process to achieve a reasonably good control performance. In doing so first, G_e should be selected in such a way that the error almost covers the entire domain $[-1, 1]$ to make efficient use of the rule-bases. Then $G_{\Delta e}$ and G_u are to be tuned to make the transient response of the system as good as possible, since there is no existing well-defined method for the determination of SFs. Suitable values of $G_{\Delta e}$ and G_u are to be selected from the knowledge of the process to be controlled and sometime through trial and error. Usually, we get a good controller without self-tuning mechanism and for further improvement a self-tuning mechanism is introduced in *Step 2*.

Step 2: Set the output SF (G_u) of the self-tuning FLC nearly ***three times*** greater than that obtained in *Step 1* keeping the values of G_e and $G_{\Delta e}$ same as those of the conventional FLC. The value of β is calculated according to the rule-base depicted in Table 2.3. It has been found that this large increase in output scaling factor (***three times***) is required due to very small multiplication factor arises from gain updating factor, which is defined in the range of $[0, 1]$. The design rule-base is flexible in nature and one can modify it for getting desired performance.

In the next two sections two practical problems (*i.e.*, HVAC and Inverted Pendulum) are addressed in detail. In each case, first we provide a brief statement about the system description. Then we illustrate the effectiveness of the proposed STFPIC/STFPDC for such systems.

2.4 HVAC system

The objectives of HVAC (heating, ventilation, and air conditioning) system are to provide an acceptable level of occupancy comfort and process function to maintain good indoor air quality, and to keep system costs and energy requirements to a minimum. HVAC is the technology of indoor and automotive environmental comfort. HVAC system design is a sub-discipline of mechanical engineering, based on the principles of thermodynamics, fluid mechanics, and heat transfer. HVAC is important in the design of medium to large industrial and office buildings such as skyscrapers and in marine environments such as aquariums, where safe and healthy building conditions are regulated with respect to temperature and humidity, using fresh air from outdoors. The three central functions of heating, ventilating, and air-conditioning are interrelated, especially with the need to provide thermal comfort and acceptable indoor air quality within reasonable installation, operation, and maintenance costs [132, 133]. Here we are discussing these three parts in brief.

Heating:

There are many different types of heating systems. Central heating is often used in cool climates to heat houses and public buildings. Such a system contains a boiler, furnace, or heat pump to warm water, steam, or air in a central location such as a furnace room in a home or a mechanical room in a large building. These systems also contain either duct for forced air systems or piping to distribute a heated fluid to radiators to transfer this heat to the air. Most modern hot water boiler heating systems have a circulator, which is a pump, to move hot water through the distribution system. This distribution system can be via radiators, convectors, hot water coils or other heat exchangers. The air supply is typically filtered through air cleaners to remove dust and pollen particles [134]. Electrical heaters are often used as backup or supplemental heat for heat pump systems. Heat pumps can extract heat from the exterior air or from the ground. Initially, heat pump HVAC systems were used in moderate climates, but with improvements in low

temperature operation and reduced loads due to more efficient homes, they are increasing in popularity in other climates.

Ventilation:

Ventilation is the process of changing or replacing air in any space to control temperature or remove any combination of moisture, odors, smoke, heat, dust, airborne bacteria, or carbon dioxide, and to replenish oxygen. Ventilation includes both the exchange of air with the outside as well as circulation of air within the building. It is one of the most important factors for maintaining acceptable indoor air quality in buildings. Methods for ventilating a building may be divided into *mechanical / forced* and *natural* types [135].

Mechanical or forced ventilation is provided by an air handler and used to control indoor air quality. Excess humidity, odors, and contaminants can often be controlled via dilution or replacement with outside air. However, in humid climates much energy is required to remove excess moisture from ventilation air. Kitchens and bathrooms typically have mechanical exhausts to control odors and sometimes humidity. Factors in the design of such systems include the flow rate (which is a function of the fan speed and exhaust vent size) and noise level. Direct drive fans are available for many applications, and can reduce maintenance needs. Ceiling fans and table/floor fans circulate air within a room for the purpose of reducing the perceived temperature by increasing evaporation of perspiration on the skin of the occupants.

Natural ventilation of a building with outside air is without the use of fans or other mechanical systems. It can be achieved with operable windows or trickle vents when the spaces to ventilate are small and the architecture permits. In more complex systems, warm air in the building can be allowed to rise and flow out upper openings to the outside thus causing cool outside air to be drawn into the building naturally through openings in the lower areas. These systems use very little energy but care must be taken to ensure comfort.

Air Conditioning:

Air conditioning and refrigeration are provided through the removal of heat. Refrigeration conduction media such as water, air, ice, and chemicals are referred to as refrigerants. A refrigerant is employed either in a heat pump system in which a compressor is used to drive thermodynamic refrigeration cycle, or in a free cooling system which uses pumps to circulate a

cool refrigerant (typically water or a glycol mix). An air conditioning system, or a standalone air conditioner, provides cooling, ventilation, and humidity control for all or part of a building. The refrigeration cycle uses four essential elements to cool. The system refrigerant starts its cycle in a gaseous state. The compressor pumps the refrigerant gas up to a high pressure and temperature. From there it enters a heat exchanger (sometimes called a "condensing coil" or condenser) where it loses energy (heat) to the outside, cools, and condenses into its liquid phase. The liquid refrigerant is returned to another heat exchanger where it is allowed to evaporate; hence the heat exchanger is often called an "evaporating coil" or evaporator. A metering device regulates the refrigerant liquid to flow at the proper rate. As the liquid refrigerant evaporates it absorbs energy (heat) from the inside air, returns to the compressor, and repeats the cycle. In the process, heat is absorbed from indoors and transferred outdoors, resulting in cooling of the building. In variable climates, the system may include a reversing valve that switches from heating in winter to cooling in summer. By reversing the flow of refrigerant, the heat pump refrigeration cycle is changed from cooling to heating or vice versa.

Dehumidification (air drying) in an air conditioning system is provided by the evaporator. A dehumidifier is an air-conditioner-like device that controls the humidity of a room or building. It is often employed in basements which have a higher relative humidity because of their lower temperature (and propensity for damp floors and walls). In food retailing establishments, large open chiller cabinets are highly effective at dehumidifying the internal air. Conversely, a humidifier increases the humidity of a building.

Air conditioned buildings often have sealed windows, because open windows would work against an HVAC system intended to maintain constant indoor air conditions. For example, a building in a high dust environment, or a home with furry pets, will need to have the filters changed more often than buildings without these dirt loads. Failure to replace these filters as needed will contribute to a lower heat exchange rate, resulting in wasted energy, shortened equipment life, and higher energy bills; low air flow can result in 'iced-up' or 'iced-over' evaporator coils, which can completely stop air flow. Additionally, very dirty or plugged filters can cause overheating during a heating cycle, and can result in damage to the system or even fire. Because an air conditioner moves heat between the indoor coil and the outdoor coil, both must be kept clean. This means that, in addition to replacing the air filter at the evaporator coil, it

is also necessary to regularly clean the condenser coil. Failure to keep the condenser clean will eventually result in harm to the compressor, because the condenser coil is responsible for discharging both the indoor heat (as picked up by the evaporator) and the heat generated by the electric motor driving the compressor.

2.4.1 HVAC system control

A second order plus dead-time model of HVAC is well established by *Bi* and *Cai* [136]. In real application however, both fans and dampers exhibit nonlinear properties for different working points, even a well-tuned PID controller may not be able to achieve a desired performance for set-point and process variations. It has been proved that fuzzy logic controller is very suitable for nonlinear system and even with unknown structure [137-139]. A typical cooling only HVAC system is shown in Fig. 2.7. In the system, the outside air is mixed with the building return air. Then the mixed air (supply air) is sucked through the cooling coil via a filter by a supply air fan. The cooled air is then supplied to different zones as shown in the figure.

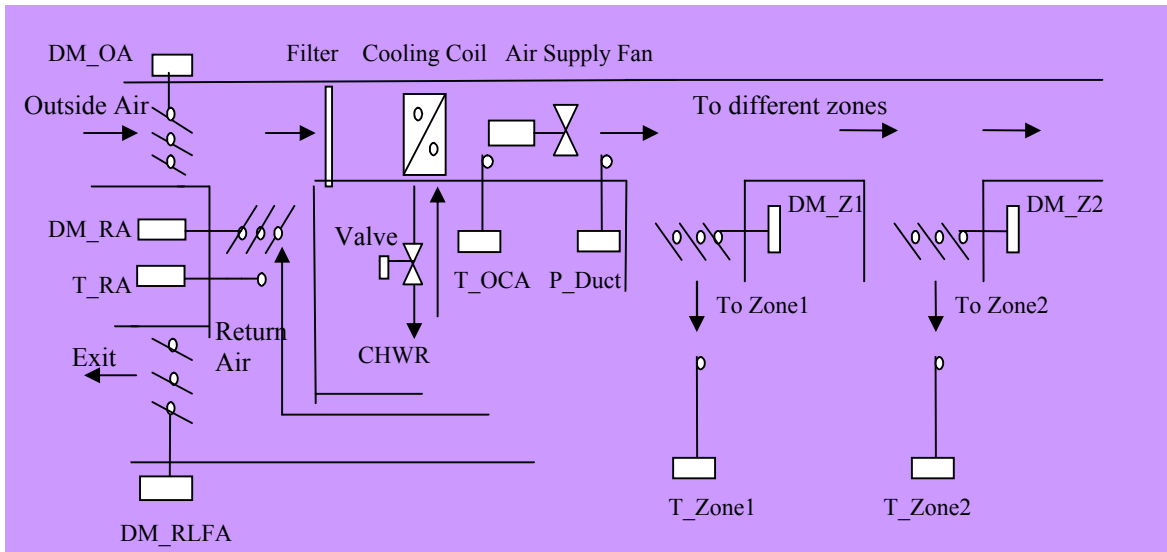


Figure 2.7: A typical HVAC system.

2.4.2 Results and discussion

A supply air pressure loop of HVAC system is shown in Fig. 2.8. In this HVAC system, the supply air pressure is regulated by the speed of a supply air fan. Increasing the fan speed will increase the supply air pressure, and vice versa. The dynamics of the control signal feeding to the

fan to the supply air pressure can be modeled as a second order plus dead-time plant. The self-tuning mechanism that already discussed in the previous section is applied to PI type fuzzy logic controller for investigation of this system [23]. In general, the transfer function of the supply air pressure loop is expressed as:

$$G(s) = \frac{Ke^{-\delta s}}{(1 + \tau_1 s)(1 + \tau_2 s)} \quad (2.9)$$

where, the term K is gain constant, τ_1 and τ_2 are time constants, and δ is dead-time.

Different performance parameters such as rise time (t_r), settling time (t_s), % peak overshoot (%OS), integral absolute error (IAE), integral of the time multiplied absolute error ($ITAE$) and integral square error (ISE) are observed to analyze the performances of different controllers. The integral criterions IAE , ISE and $ITAE$ are considered because mere visual observations of response curves are not always enough to make a good comparison. Large errors contribute heavily to IAE ; on the other hand $ITAE$ penalizes errors that occur late in time. Thus IAE and $ITAE$ reveal overall performance characteristics of the control system.

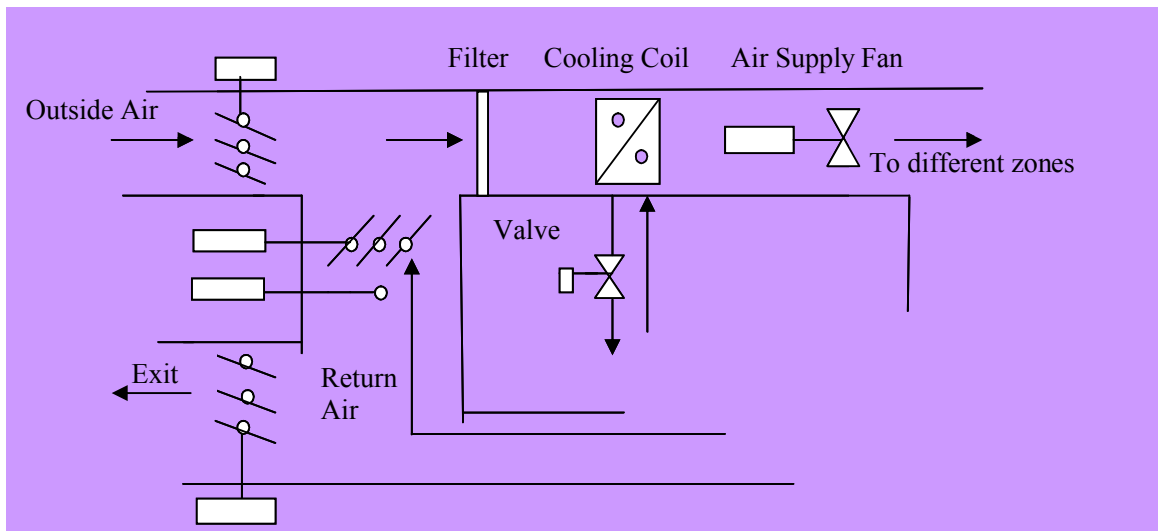


Figure 2.8: A supply air pressure loop of typical HVAC system.

In this section, performance analysis of STFPIC is made under normal condition and changing of HVAC process model. We considered four variations for *equation* (2.9) that represent different HVAC models.

- a. Case 1: $K=0.81$, $\delta=0.2$, $\tau_1=0.97$ and $\tau_2=0.1$
- b. Case 2: $K=0.81$, $\delta=0.2$, $\tau_1=0.2$ and $\tau_2=2.0$
- c. Case 3: $K=1.2$, $\delta=0.3$, $\tau_1=0.97$ and $\tau_2=0.1$
- d. Case 4: $K=1.2$, $\delta=0.4$, $\tau_1=0.97$ and $\tau_2=0.1$

Fig. 2.9(b), Fig. 2.10(b), Fig. 2.11(b) and Fig. 2.12(b) show that the STFPIIC controls the supply air pressure loop of HVAC satisfactorily compare to FPIC, both under normal and as well as under model variations. Table 2.4 refers that both the rise time and settling time are very much satisfactory. Peak overshoots are also negligible when STFPIIC is used. In all the four cases, we observe very less *IAE*, *ITAE* and *ISE* values, when STFPIIC is applied.

$$\text{Case 1 } G(s) = \frac{0.81e^{-0.2s}}{(0.97s+1)(0.1s+1)}$$

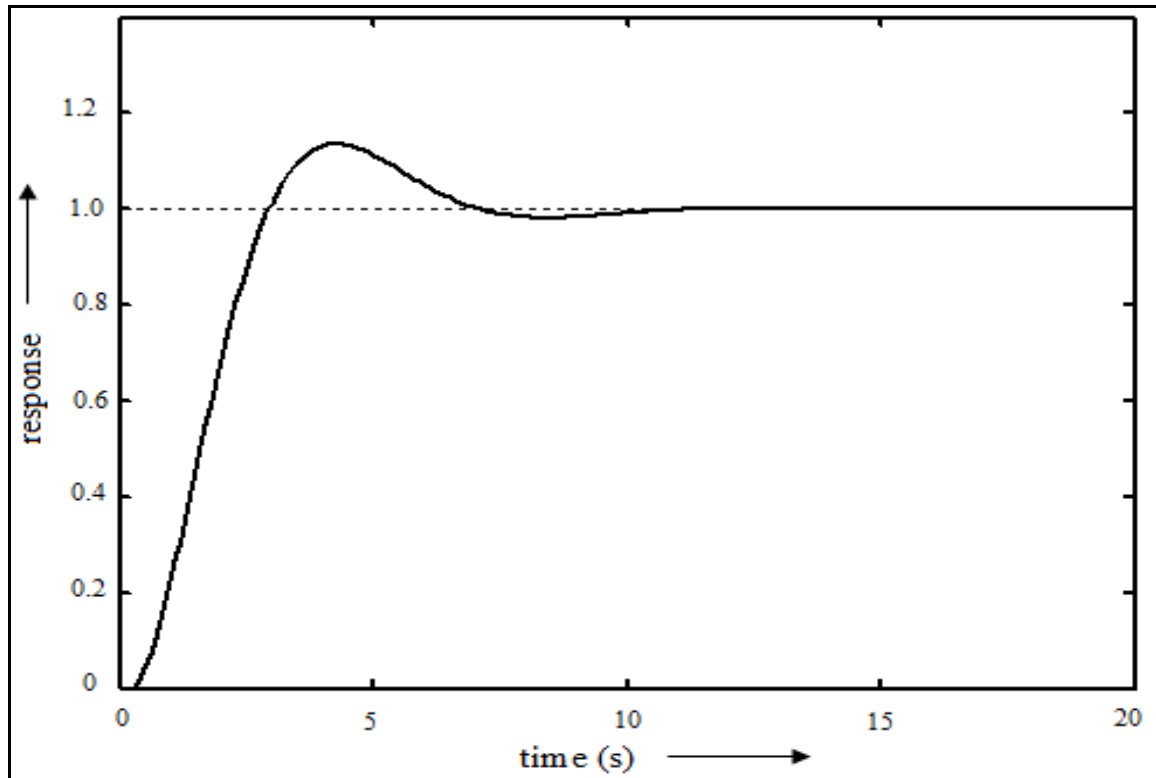


Figure 2.9 (a): Performance of HVAC with FPIC.

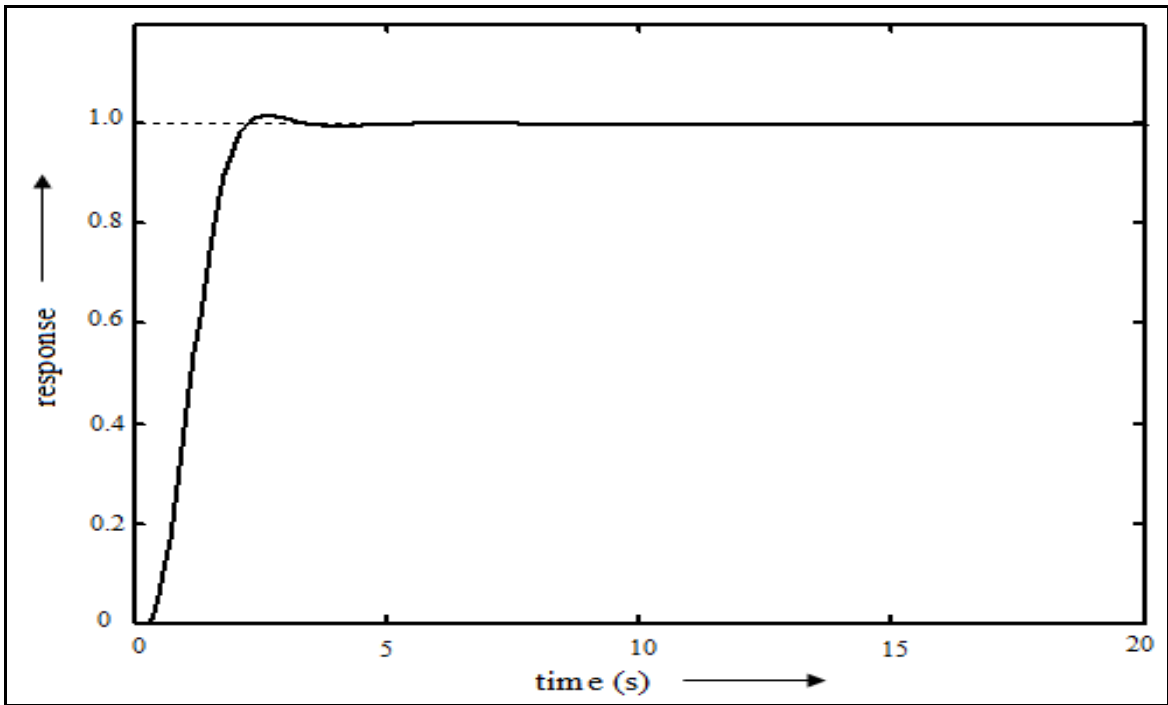


Figure 2.9 (b): Performance of HVAC with STFPIC.

Case 2 $G(s) = \frac{0.81e^{-0.2s}}{(0.2s+1)(2s+1)}$

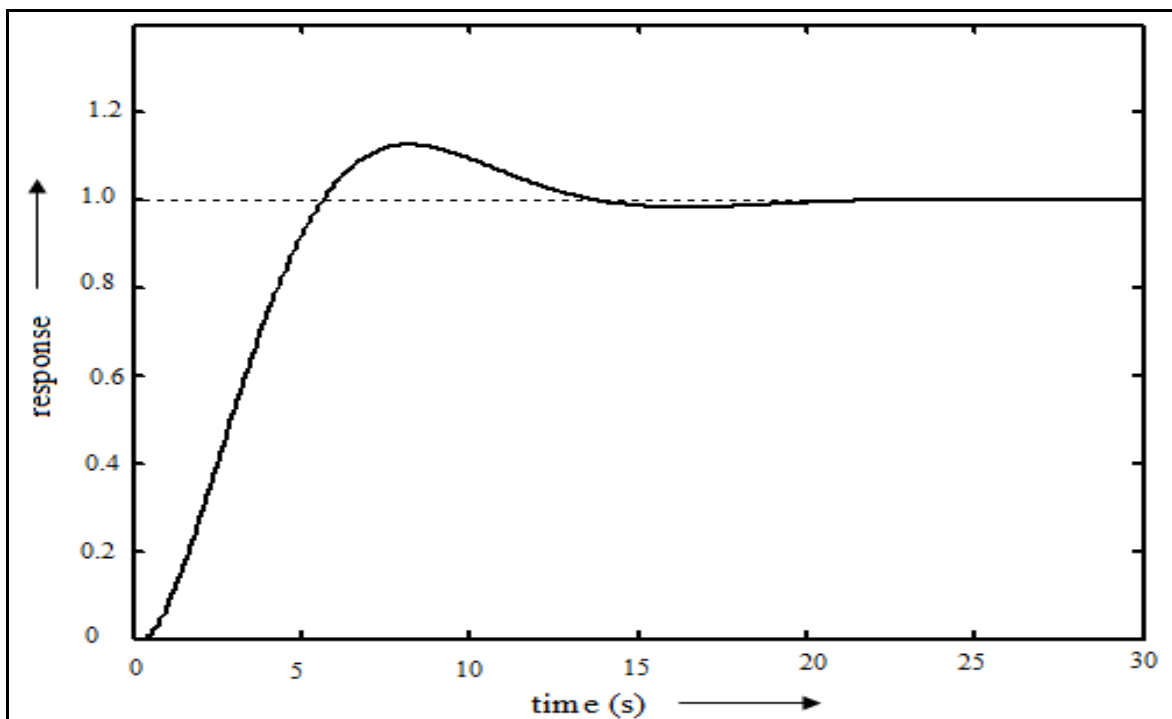


Figure 2.10 (a): Performance of HVAC with FPIC.

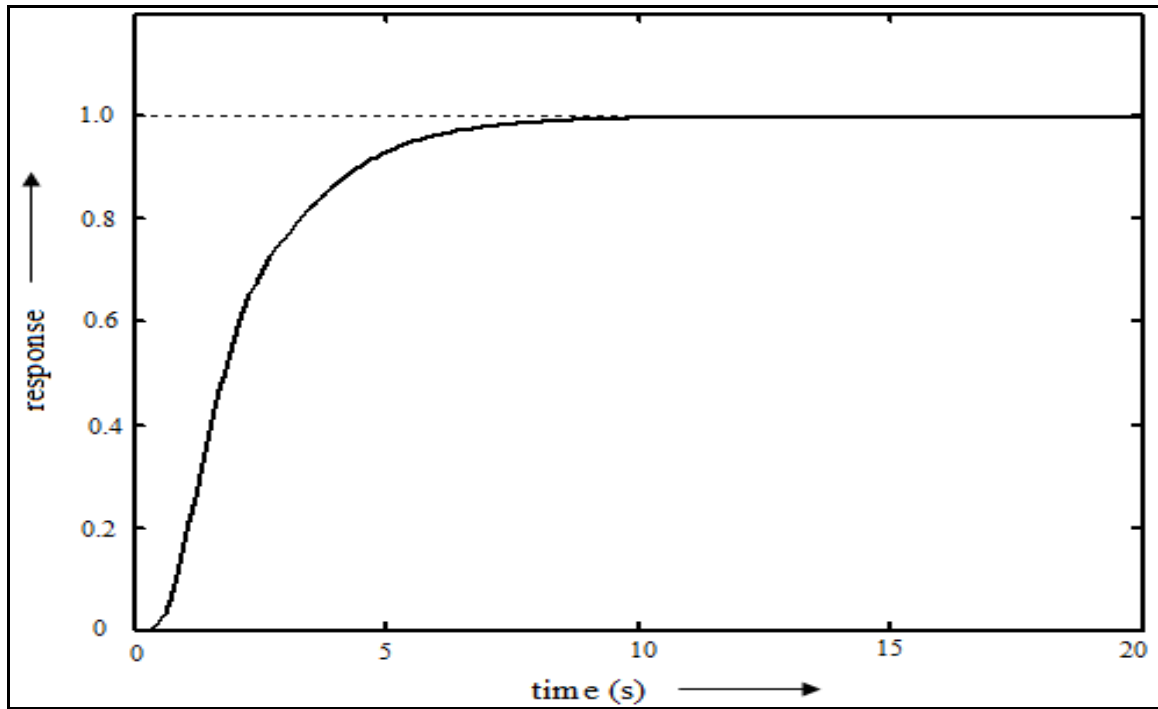


Figure 2.10 (b): Performance of HVAC with STFPI.

Case 3 $G(s) = \frac{1.21e^{-0.3s}}{(0.97s+1)(0.1s+1)}$

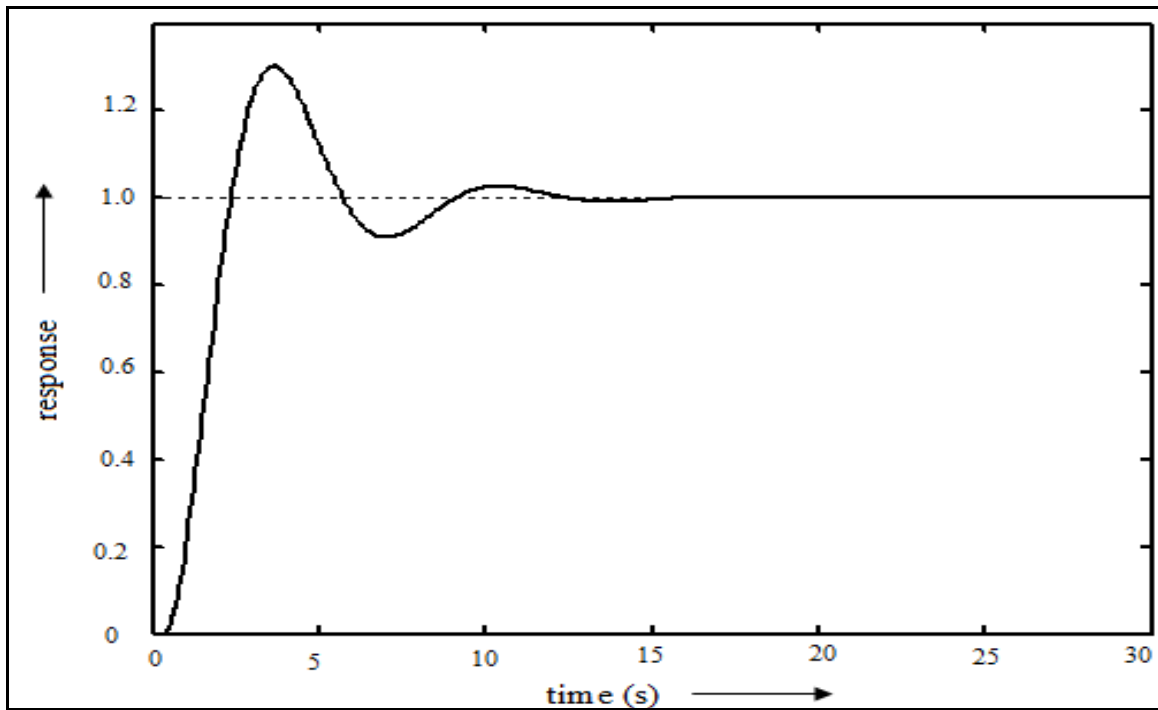


Figure 2.11 (a): Performance of HVAC with FPI.

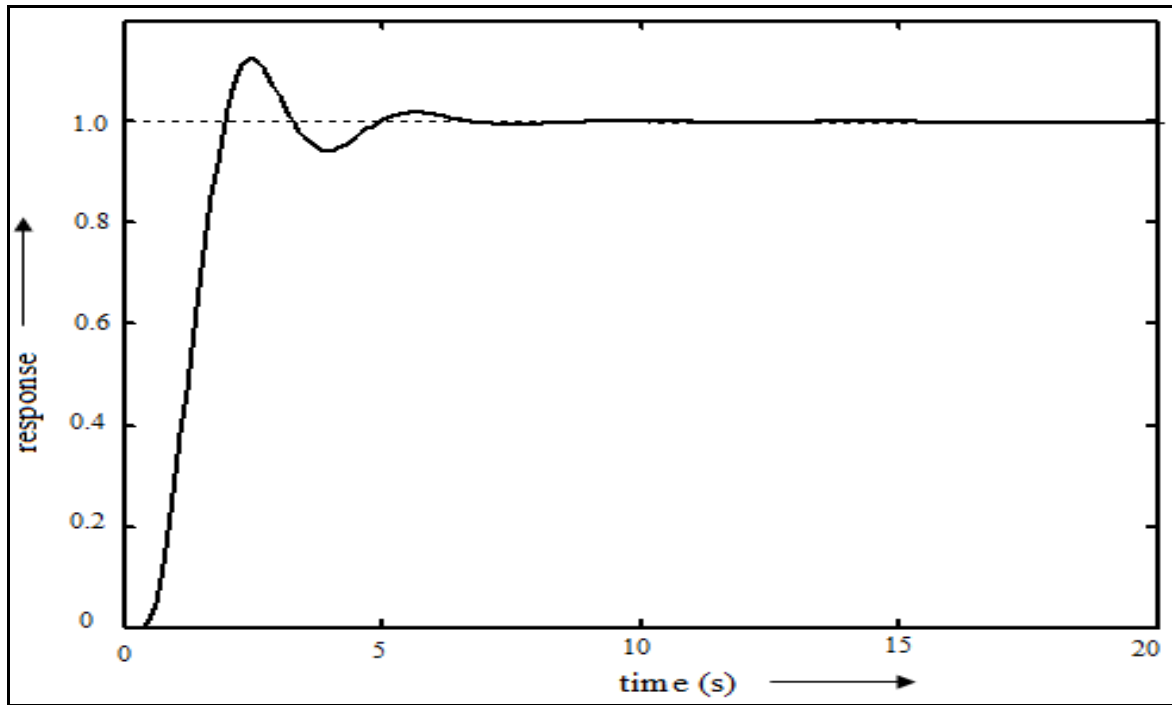


Figure 2.11 (b): Performance of HVAC with STFPIC.

Case 4 $G(s) = \frac{1.21e^{-0.4s}}{(0.97s + 1)(0.1s + 1)}$

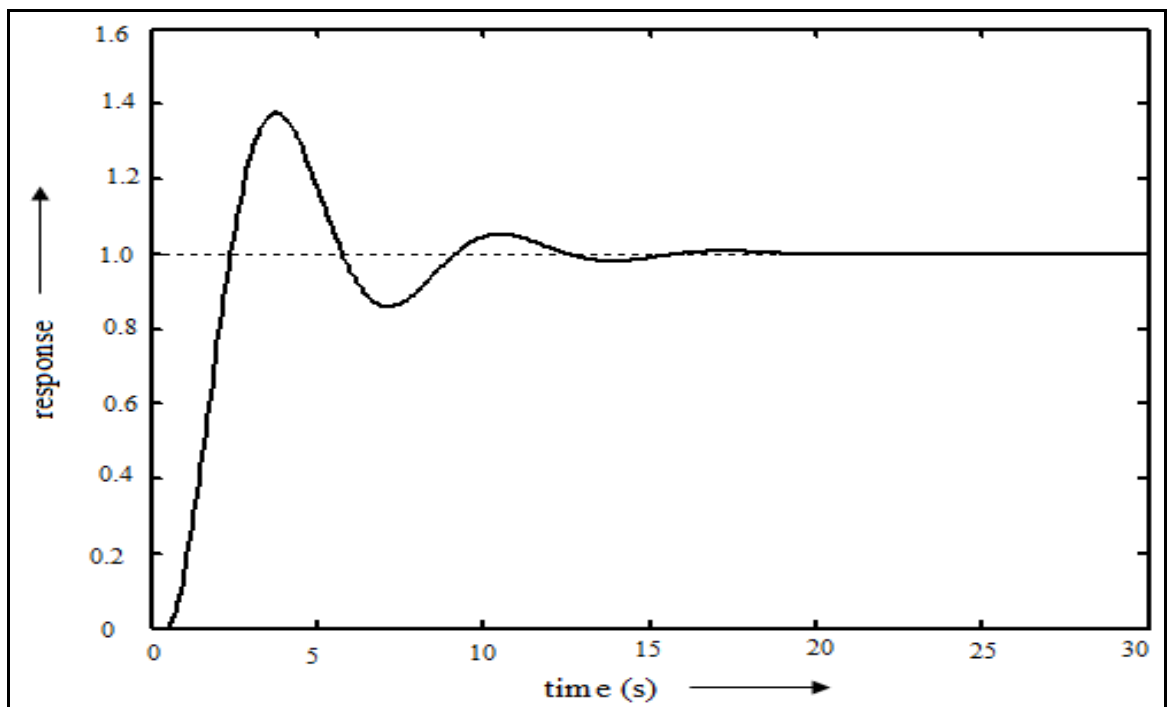


Figure 2.12 (a): Performance of HVAC with FPIC.

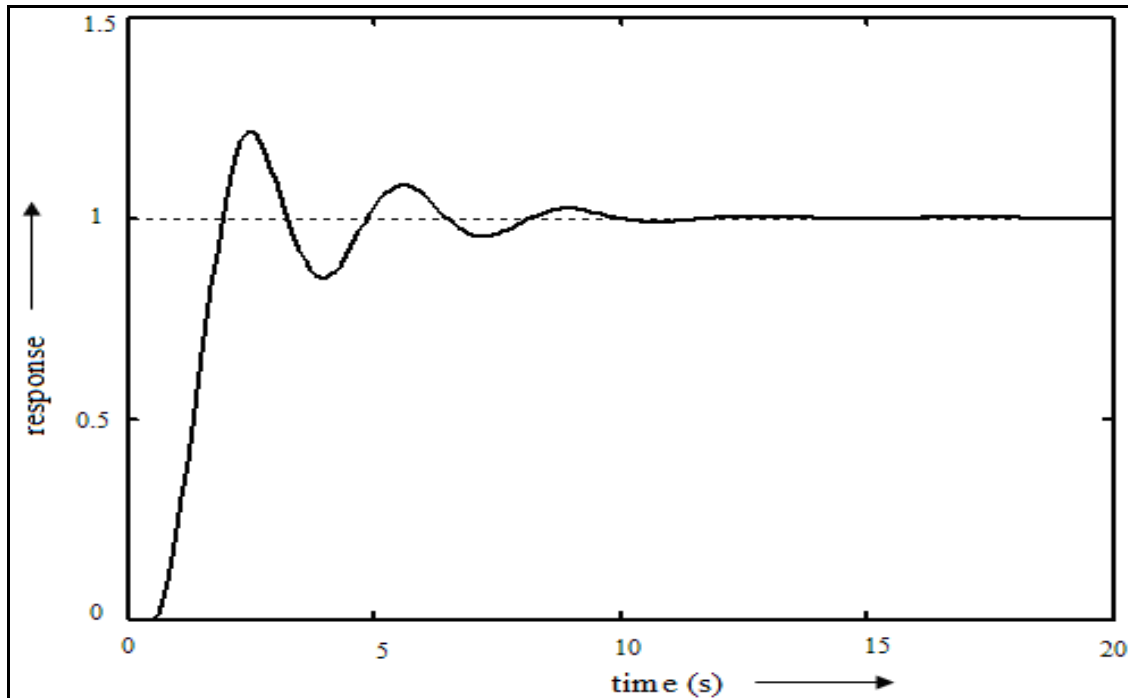


Figure 2.12 (b): Performance of HVAC with STFPIC.

Table 2.4: Performance analysis for different models of HVAC supply air pressure loop

Process Model	Controller Type	t_r (s)	t_s (s)	%OS	IAE	ITAE	ISE
Case 1	FPIC	2.8	6.6	13.6	2.03	3.50	1.30
	STFPIC	2.0	2.2	1.74	1.11	0.76	0.78
Case 2	FPIC	5.2	12.6	12.7	3.68	12.10	2.28
	STFPIC	5.5	6.6	0.0	2.09	3.50	1.23
Case 3	FPIC	2.3	10.9	30.24	2.41	5.91	1.39
	STFPIC	1.9	4.7	12.41	1.32	1.36	0.87
Case 4	FPIC	2.3	11.9	37.6	2.87	8.78	1.60
	STFPIC	1.9	7.9	21.69	1.67	2.93	0.97

The above four cases are also studied *with load variations* as presented in Fig. 2.13 to Fig. 2.16. These figures indicate that even with load variations HVAC supply air pressure loop is working satisfactorily under STFPIC.

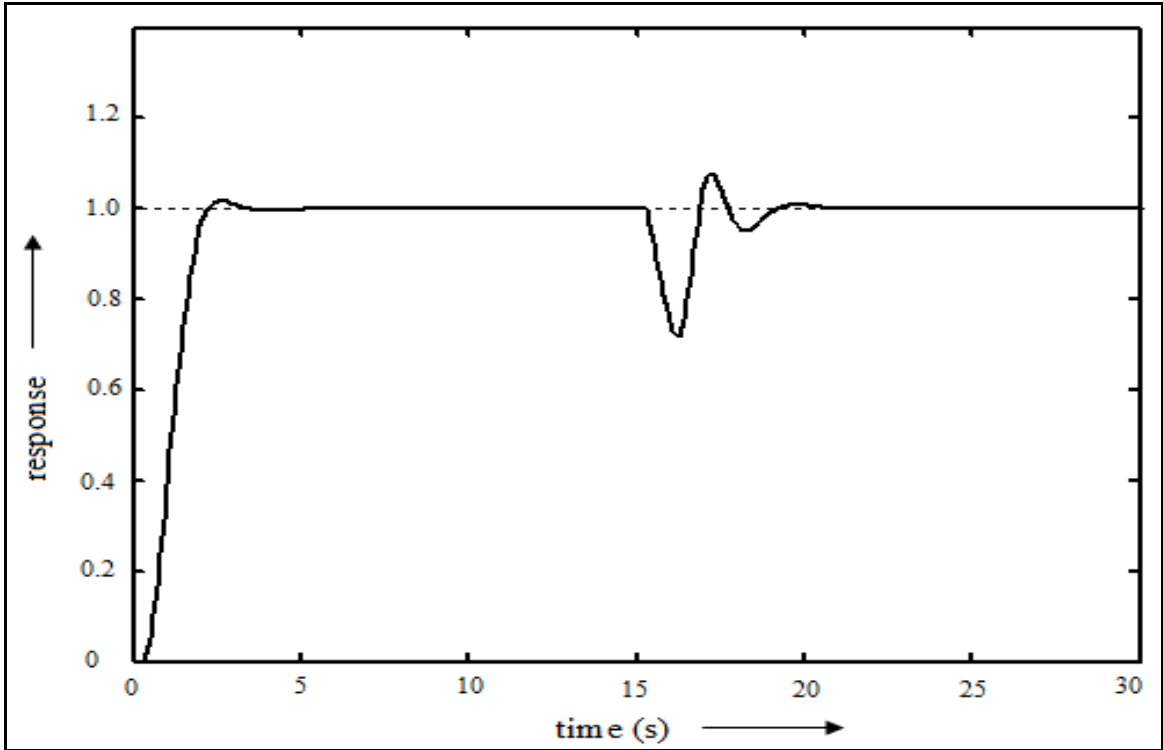


Figure 2.13: Performance of HVAC (*Case 1*) with STFPIC for load variation at 15s.

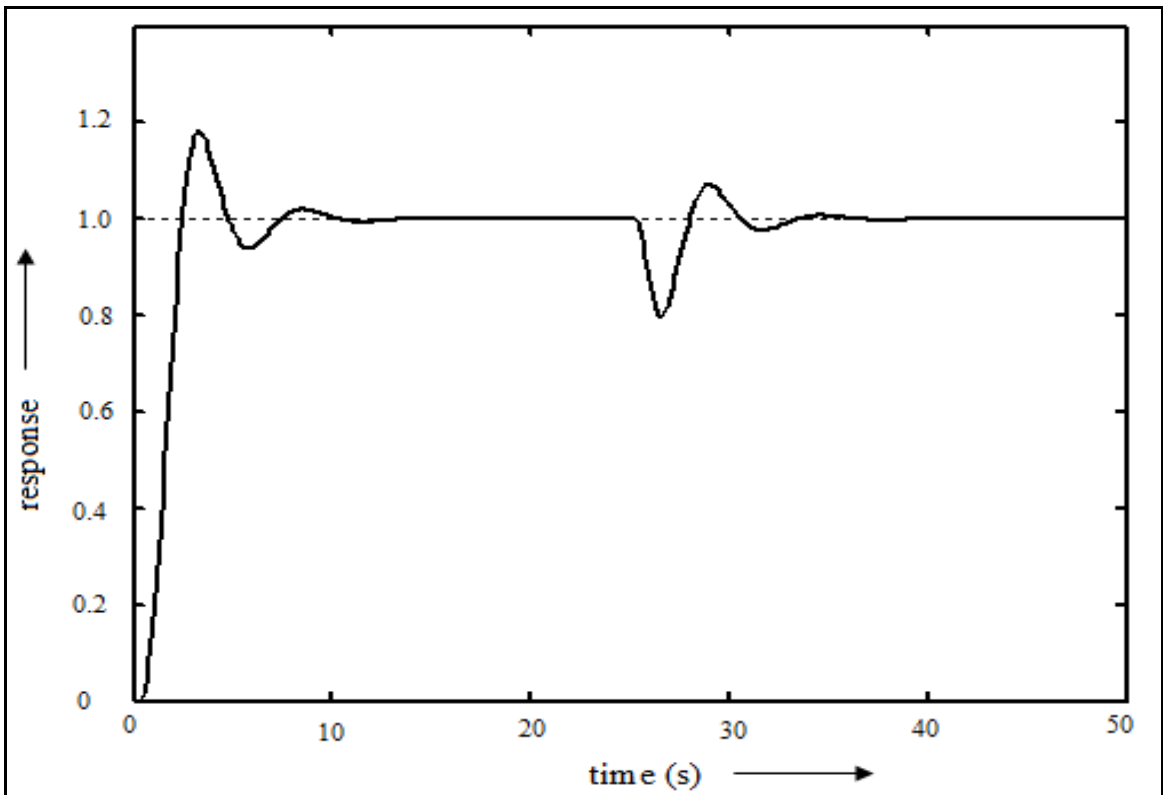


Figure 2.14: Performance of HVAC (*Case 2*) with STFPIC for load variation at 25s.

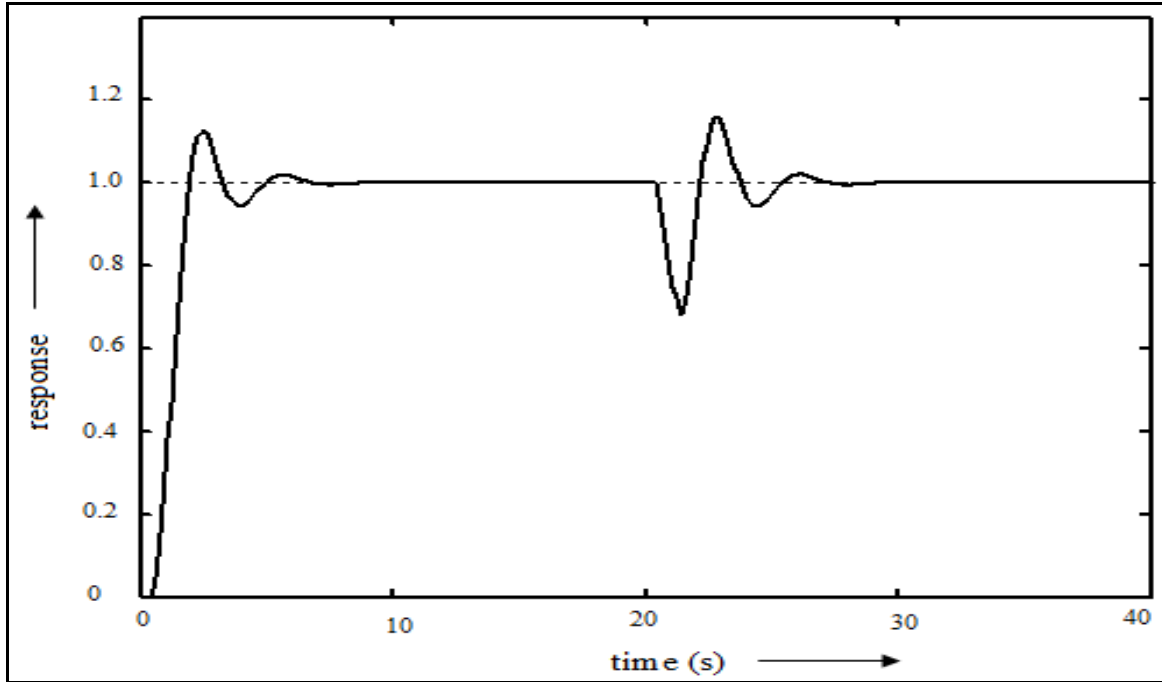


Figure 2.15: Performance of HVAC (*Case 3*) with STFPIC for load variation at 20s.

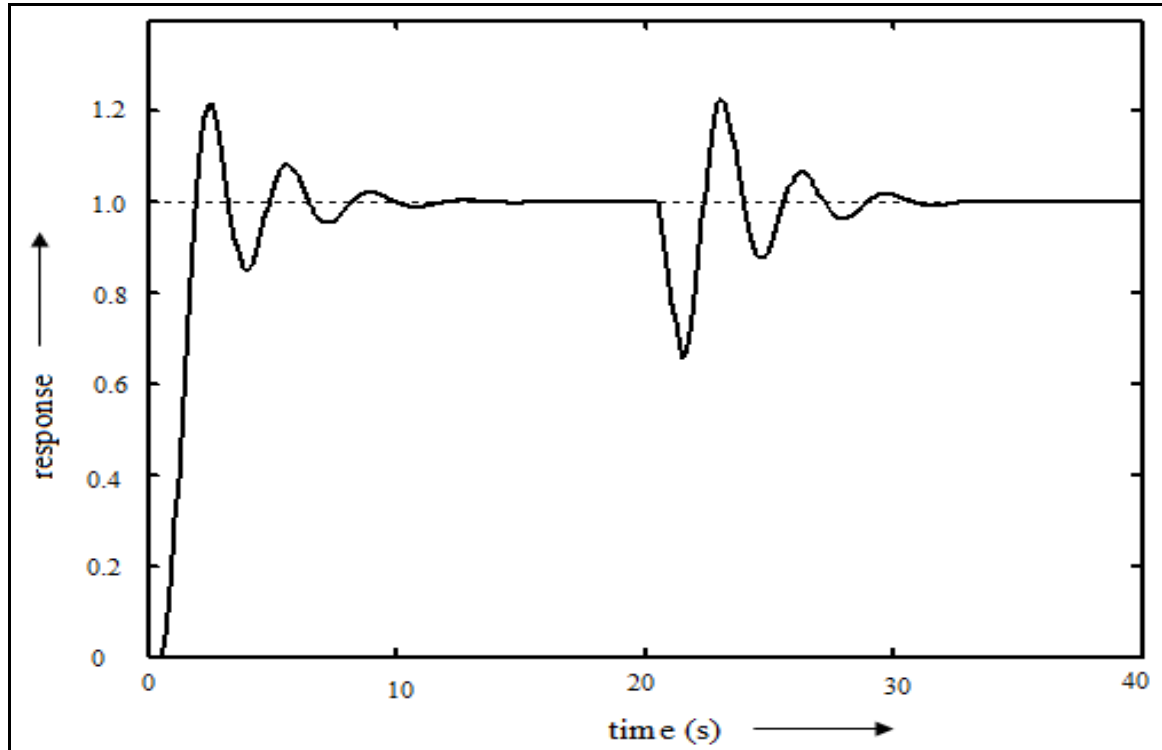


Figure 2.16: Performance of HVAC (*Case 4*) with STFPIC for load variation at 20s.

2.4.3 Comparative study

In order to demonstrate the effectiveness and robustness, the performance of the propose STFPIC is compared with those of existing methods like- PID controller, FPIC and *Jian and Cai's* Adaptive Neuro-Fuzzy (ANF) controller [140] for supply air pressure loop control. The study is made under changing process model and comparative results are provided in Table 2.5. Fig.

2.17 shows the characteristics of model $\frac{1.21e^{-4s}}{(0.97s+1)(0.1s+1)}$ for application of PID, FPIC and STFPIC.

Due to the application of STFPIC, substantial improvements are observed in settling time and also in peak overshoot. Moreover, it is important that when the process encounters large parameter variations, the proposed method provides much robustness compared to PID controller and even it outperforms Adaptive Neuro-Fuzzy (ANF) controller designed by *Jian and Cai* [140], as shown in Table 2.5 and Fig. 2.17.

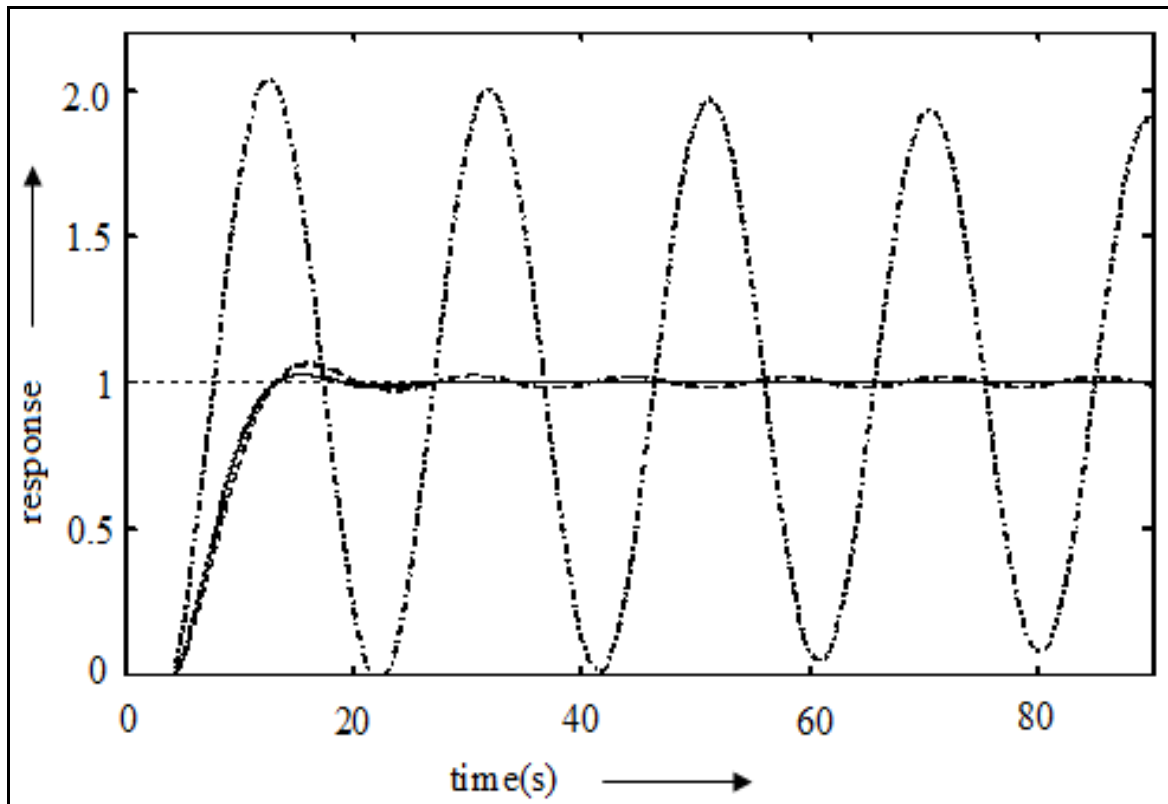


Figure 2.17: Response of HVAC system under PID (dashed-dot), FPIC (dashed) and STFPIC (solid).

Table 2.5: Performance comparison between PID, ANF, FPIC and STFPIK for model variation

Transfer Function	Controller Type	t_s (s)	%OS
$\frac{0.81e^{-2s}}{(0.97s+1)(0.1s+1)}$	PID	57.5	70.6
	ANF	7.5	3.5
	FPIC	13.2	2.44
	STFPIK	7.3	1.16
$\frac{0.81e^{-2s}}{(0.2s+1)(2s+1)}$	PID	60.9	63.8
	ANF	10.6	0.9
	FPIC	10.3	1.04
	STFPIK	10.5	0.03
$\frac{1.21e^{-3s}}{(0.97s+1)(0.1s+1)}$	PID	>100	78.0
	ANF	19	56.0
	FPIC	15.5	4.86
	STFPIK	10.9	1.82
$\frac{1.21e^{-4s}}{(0.97s+1)(0.1s+1)}$	PID	>100	103.4
	ANF	32	59
	FPIC	24.6	6.25
	STFPIK	12.5	2.59

2.5 Inverted pendulum

The inverted pendulum is an interesting topic to researchers for its strong degree of nonlinearity and inherent instability and yet simplicity of structure. So it can justifiably be used for determining the effectiveness of a control algorithm. Control of an inverted pendulum is a very common control engineering problem based on flight simulation of rockets and missiles during the initial stages of flight, wherein the aim is to stabilize the inverted pendulum such that the position of the carriage on the track is controlled quickly and accurately. The inverted pendulum is also incorporated in heavy cranes lifting containers in shipyards, self-balancing robots and in future transport systems like seaways and jetpacks.

Application of fuzzy logic for inverted pendulum control still by far remains most popular, as they do not require precise knowledge of the system parameters. However, performance of PI-type FLC for higher order, nonlinear, and unstable systems, like inverted pendulum is not satisfactory due to large overshoot, excessive oscillation, and slower response. As already stated, the inverted pendulum model is nonlinear in nature and its parameters may vary with time. An expert defined fuzzy tuning scheme for PD type FLC is proposed here to control such a complex and nonlinear system [22, 141, 142].

2.5.1 Inverted pendulum and its mathematical model

The pendulum is a modern exemplary area of research for determining the effectiveness of various control algorithms. The pendulum experiment can be divided into two separate control problems. First is the crane control problem, in which the goal is to move the cart into a desired position with as little oscillation of the load (pendulum arms) as possible. The other is to stabilize the inverted pendulums in an upright position [143, 144]. The crane control problem is very often encountered in industrial applications where load movement is incorporated. It is especially difficult to realize when cranes are placed on ships and the effect of waves is considered. The inverted pendulum task can be seen as a self-erecting control problem, which is present in missile launching and control applications. Furthermore, the pendulum application involves a swing-up control aspect if initially the pendulum hangs freely in the vertical position. These two control problems (inverted pendulum and crane control) have one very important difference, which is the stability. The pendulum serving as a crane is stable without a working controller. Due to energy loss through friction and air resistance it will always end up at an equilibrium point. The inverted pendulum is inherently unstable. Left without a stabilizing controller it will not be able to remain in an upright position when disturbed. Thus, the control objective is to apply a force to move the cart so that the pendulum arms remain in the vertical unstable position while simultaneously being driven to the desired cart location. Furthermore, as already mentioned it should also include a swing-up control aspect if initially the pendulum hangs freely in the vertical position.

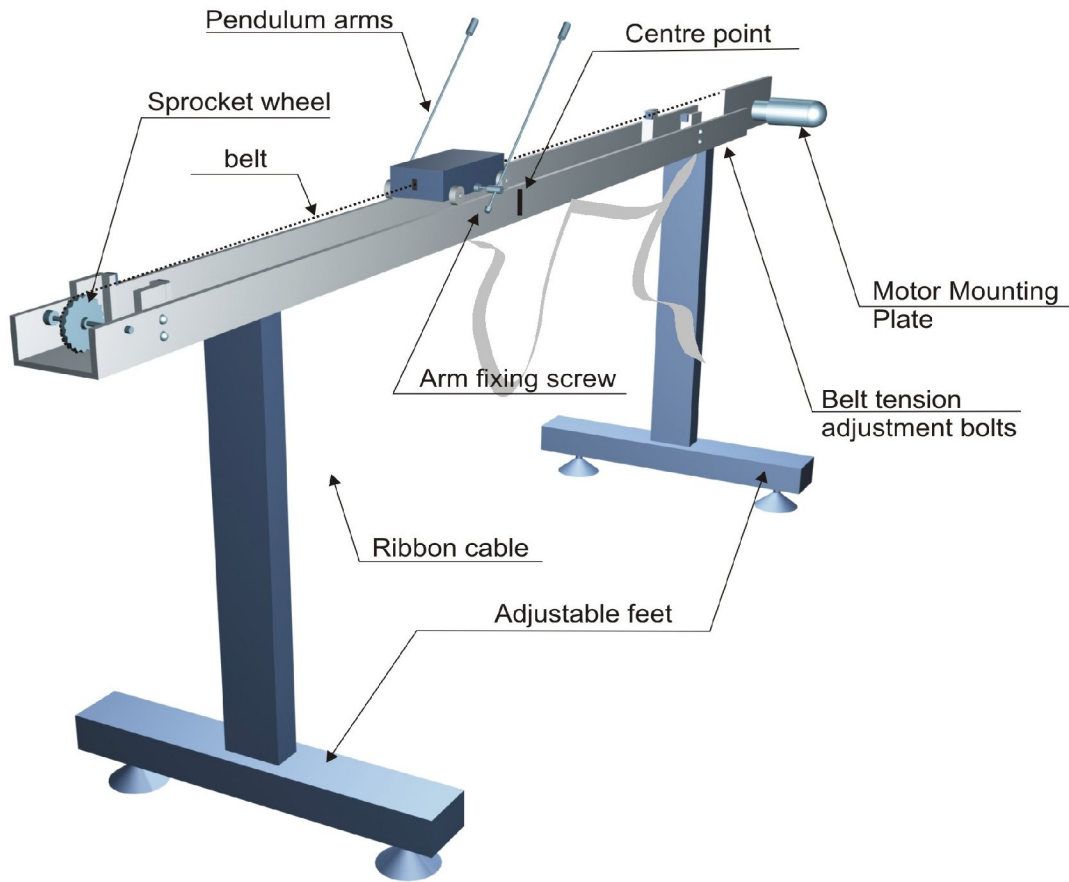


Figure 2.18: Digital pendulum mechanical unit.

A laboratory scale inverted pendulum/crane set-up (FEEDBACK, UK) shown in Fig. 2.18 consists of a cart moving along the 1m length track and a load is attached with cart through shaft [143]. The cart can move back and forth causing the load to swing. The movement of the cart is caused by pulling the belt in two directions by the DC motor attached at the end of the rail. By applying a voltage to the motor we control the force with which the cart is pulled. The value of the force depends on the value of the control voltage. The two variables that are read using optical encoders are the load angle and the cart position on the rail is installed on the cart. The controller's task will be to change the DC motor voltage depending on these two variables, in such a way that the desired control task is fulfilled (*stabilizing in an upright position, swinging or crane control*) [21]. An optical encoder consists of a light source, light detector and a slit disk placed between them. This way the relative position with respect to the initial point can be measured by counting the pulses on the light detector. Additionally, safety limit switches are

mounted on either side of the track which cut power to the motor when the cart crosses them. Initially the control signal is set to $-2.5v$ to $2.5v$ and the generated force magnitude of around $-20N$ to $+20N$. The cart position is physically bounded by the rail length and is equal to $-0.5m$ to $+0.5m$. The system is interfaced to a personal computer by means of a data acquisition card and is driven by Matlab/Simulink based real time software [144].

Mathematical model

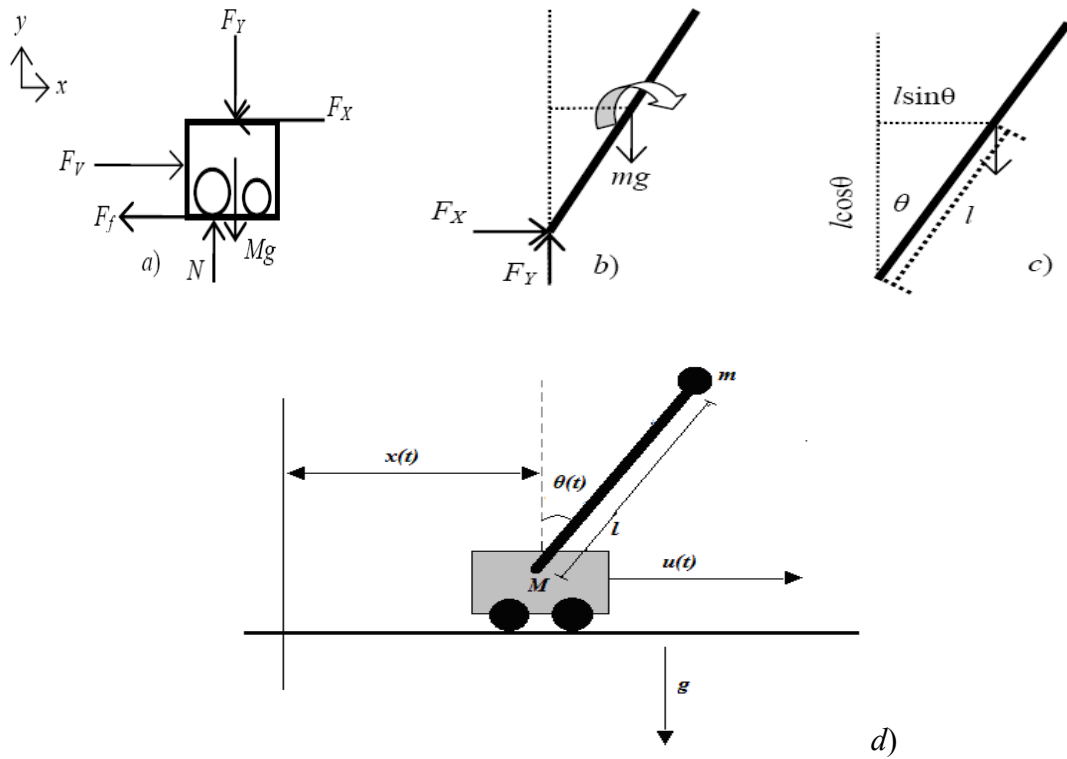


Figure 2.19: Free body diagrams of (a) the cart (b) and (c) the pendulum, (d) schematic of the inverted pendulum system.

m is the mass of the pendulum rod in kg ,

M is the mass of the moving cart in kg ,

F_V is the force applied to the cart in N ,

F_f is the force due to friction in N ,

b is the pendulum damping constant,

g is the acceleration due to gravity in m/s^2 ,

l is the distance along the pendulum to the centre of gravity,

x is the cart position on the rail in m ,

θ is the angle of the inverted pendulum measured from the vertical y -axis in *radians*,

u is the control voltage in v .

Let us assume that the co-ordinates of the centroid (centre of gravity) of the pendulum, (x_G, y_G) , are given by

$$x_G = x + l \sin \theta \quad (2.10)$$

$$y_G = l \cos \theta \quad (2.11)$$

For the horizontal and vertical motion of the cart, applying Newton's second law of motion;

$$\sum F = M \frac{d^2 x}{dt^2} \quad (2.12)$$

For the horizontal motion of the cart

$$M \frac{d^2 x}{dt^2} = F_v - F_x - F_f \quad (2.13)$$

Where:

$$F_f = K \cdot \frac{dx}{dt} \quad (2.14)$$

$$F_x = m \frac{d^2 x_G}{dt^2} \quad (2.15)$$

Evaluating the derivative in *equation (2.15)*

$$\begin{aligned} \frac{dx_G}{dt} &= \frac{d(x + l \sin \theta)}{dt} \\ &= \frac{dx}{dt} + l \frac{d(\sin \theta)}{dt} \\ &= \frac{dx}{dt} + l \cos \theta \frac{d\theta}{dt} \\ \frac{d^2 x_G}{dt^2} &= \frac{d}{dt} \left(\frac{dx}{dt} + l \cos \theta \frac{d\theta}{dt} \right) \\ &= \frac{d^2 x}{dt^2} + l \frac{d}{dt} \left(\cos \theta \frac{d\theta}{dt} \right) \\ &= \frac{d^2 x}{dt^2} + l \left(\frac{d \cos \theta}{dt} \frac{d\theta}{dt} + \cos \theta \frac{d^2 \theta}{dt^2} \right) \end{aligned}$$

$$\begin{aligned}
&= \frac{d^2 x}{dt^2} + l \left\{ -\sin \theta \left(\frac{d\theta}{dt} \right)^2 + \cos \theta \frac{d^2 \theta}{dt^2} \right\} \\
&= \frac{d^2 x}{dt^2} - l \sin \theta \left(\frac{d\theta}{dt} \right)^2 + l \cos \theta \frac{d^2 \theta}{dt^2}
\end{aligned} \tag{2.16}$$

Thus *equation 2.15* becomes

$$F_x = m \left(\frac{d^2 x}{dt^2} - l \sin \theta \left(\frac{d\theta}{dt} \right)^2 + l \cos \theta \frac{d^2 \theta}{dt^2} \right) \tag{2.17}$$

Using *equation (2.17)*, *equation (2.18)* can be simplified to give:

$$\begin{aligned}
M \frac{d^2 x}{dt^2} &= F_v - m \left(\frac{d^2 x}{dt^2} - l \sin \theta \left(\frac{d\theta}{dt} \right)^2 + l \cos \theta \frac{d^2 \theta}{dt^2} \right) - K \frac{dx}{dt} \\
M \frac{d^2 x}{dt^2} &= F_v - m \frac{d^2 x}{dt^2} + ml \sin \theta \left(\frac{d\theta}{dt} \right)^2 - ml \cos \theta \frac{d^2 \theta}{dt^2} - K \frac{dx}{dt}
\end{aligned} \tag{2.18}$$

The final form for the horizontal motion of the cart can be given as:

$$(M + m) \frac{d^2 x}{dt^2} + K \frac{dx}{dt} = F_v + ml \sin \theta \left(\frac{d\theta}{dt} \right)^2 - ml \cos \theta \frac{d^2 \theta}{dt^2} \tag{2.19}$$

For the vertical motion of the pendulum, *equation (2.12)* can be written as

$$F_y - mg = m \frac{d^2 y_G}{dt^2} \tag{2.20}$$

Evaluating the derivative in *equation (2.20)*

$$\begin{aligned}
\frac{dy_G}{dt} &= \frac{d(l \cos \theta)}{dt} \\
&= -l \sin \theta \frac{d\theta}{dt} \\
\frac{d^2 y_G}{dt^2} &= \frac{d}{dt} \left(-l \sin \theta \frac{d\theta}{dt} \right) \\
&= -l \left(\frac{d \sin \theta}{dt} \frac{d\theta}{dt} + \sin \theta \frac{d^2 \theta}{dt^2} \right) \\
&= -l \left(\cos \theta \left(\frac{d\theta}{dt} \right)^2 + \sin \theta \frac{d^2 \theta}{dt^2} \right)
\end{aligned}$$

$$= -l \cos \theta \left(\frac{d\theta}{dt} \right)^2 - l \sin \theta \frac{d^2\theta}{dt^2} \quad (2.21)$$

Using *equation (2.21)*, *equation (2.20)* can be rewritten to give

$$F_y - mg = m \left(-l \cos \theta \left(\frac{d\theta}{dt} \right)^2 - l \sin \theta \frac{d^2\theta}{dt^2} \right) \quad (2.22)$$

Thus the vertical reaction force F_y can be written as

$$F_y = mg + m \left(-l \cos \theta \left(\frac{d\theta}{dt} \right)^2 - l \sin \theta \frac{d^2\theta}{dt^2} \right) \quad (2.23)$$

Now we know,

$$\text{Sum of moments} = I \frac{d^2\theta}{dt^2} \quad (2.24)$$

Where the moment due to a given force is defined as:

$$M = F \times r \quad (2.25)$$

Where: F is the force vector,

r is the position vector of the object with respect to the point, about which the moments are being summed,

I is the angular momentum of the object

For the pendulum, summing the moment around its centre of gravity,

$$F_y l \sin \theta - F_x l \cos \theta = I \frac{d^2\theta}{dt^2} \quad (2.26)$$

Substituting *equation (2.23)* for F_y and *equation (2.17)* for F_x into *equation (2.26)* gives

$$\left(mg + m \left(-l \cos \theta \left(\frac{d\theta}{dt} \right)^2 - l \sin \theta \frac{d^2\theta}{dt^2} \right) \right) l \sin \theta - \left(m \left(\frac{d^2x}{dt^2} - l \sin \theta \left(\frac{d\theta}{dt} \right)^2 + l \cos \theta \frac{d^2\theta}{dt^2} \right) \right) l \cos \theta = I \frac{d^2\theta}{dt^2}$$

The above equation can be simplified to get the equation for angular position

$$(I + ml^2) \frac{d^2\theta}{dt^2} = mgl \sin \theta - ml \cos \theta \frac{d^2x}{dt^2} \quad (2.27)$$

Therefore the equations of motion for the inverted pendulum on a moving cart can be written as:

$$(M + m) \frac{d^2x}{dt^2} + K \frac{dx}{dt} = F_v + ml \sin \theta \left(\frac{d\theta}{dt} \right)^2 - ml \cos \theta \frac{d^2\theta}{dt^2} \quad (2.28)$$

$$(I + ml^2) \frac{d^2\theta}{dt^2} = mgl \sin \theta - ml \cos \theta \frac{d^2x}{dt^2} \quad (2.29)$$

The values of the constants in the above equations, considered henceforth are given by Table 2.6

Table 2.6: The value of the pendulum constants

m is the mass of the pendulum rod in kg	0.2
M is the mass of the moving cart in kg	2.3
g is the acceleration due to gravity in m/s^2	9.81
l is the distance along the pendulum to the centre of gravity in m	0.3
I is the moment of inertia of the pole in kgm^2	0.0099
K is the cart friction coefficient	0.00005

The inverted pendulum is an unstable system which, in terms of behavior, means that the plant left without any controller reaches an unwanted, very often destructive state. Thus for such plants it is useful to carry out simulation tests on the models before approaching the real plant. For the purpose of controller design the model has to be linearized and presented in the form of transfer functions. But such a linear equivalent of the nonlinear model is valid only for small deviations of the state values from their nominal values also called the equilibrium point (for inverted pendulum $\theta = 0 = \Delta\theta$). Thus for the appropriate controller design we develop the nonlinear simulation model from the above equations of motion as given by Fig 2.20.

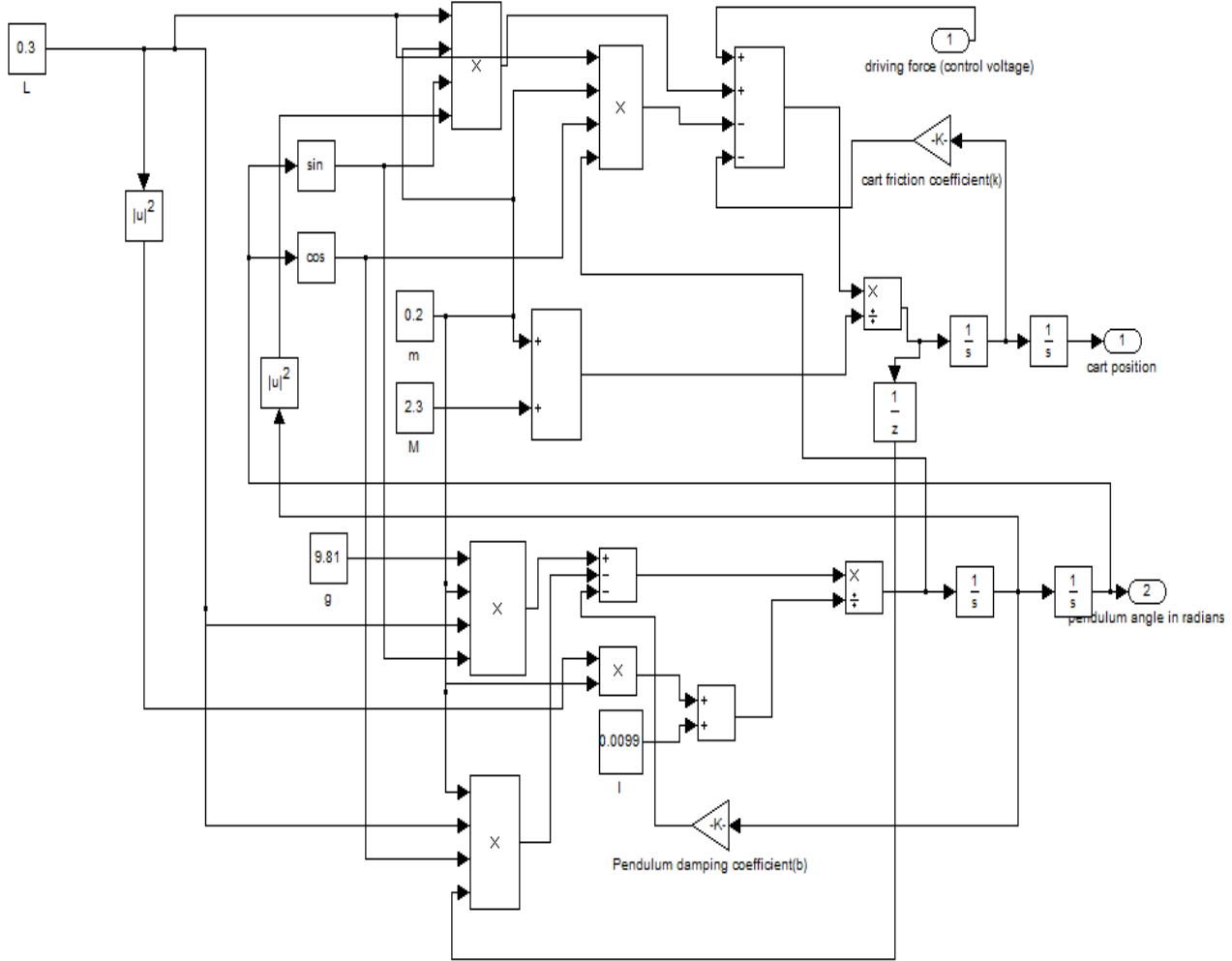


Figure 2.20: The inverted pendulum simulink model using *equations 2.28 and 2.29.*

Inverted pendulum control problem is divided into two separate control schemes: First *the swing up control* that allows the pendulum to reach the upright position; and second is *the inverted pendulum stabilization* around the equilibrium point [6]. The control action begins in the swing up control mode if initially the pendulum arms are hanging free. The stabilization mode comes into effect once the pendulum reaches $[-10^\circ, +10^\circ]$ of the final vertical upright position.

2.5.2 Inverted pendulum control by PID, FPDC and STFPDC

The inverted pendulum is one of the most difficult systems to control in the field of control engineering. In this section the effectiveness of PID, FPDC and STFPDC is tested on inverted pendulum using dual control scheme.

2.5.2.1 PID controller

The inverted pendulum system is nonlinear and unstable with one input signal and several output signals. Due to its importance in the field of control engineering, it has been, one of the most primitive areas of study, wherein the aim is to analyze its model and propose a linear compensator according to the PID control law. From the dynamic equations of this system, it is found that there are two control parameters in the inverted pendulum-cart system. One is the inverted pendulum angle (θ) and the other is the cart position on the rail (x), however, there is only one control action allowed. Fig. 2.21 and Fig. 2.23 show the basic block diagram of the inverted pendulum controller system for stabilization.

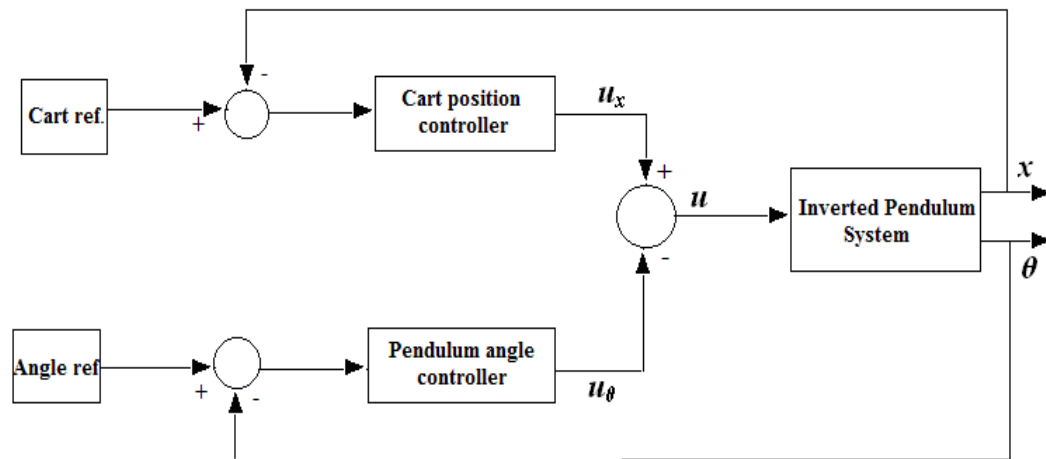


Figure 2.21: Basic block diagram of the inverted pendulum control Scheme.

The system is controlled by two separate controller, pendulum angle controller and cart position controller. The control action u_x for the cart subsystem and the control action u_θ for the pendulum angle subsystem need to be combined into one control action u for the inverted pendulum system. It can be seen that to provide a control action to push the cart toward left-hand side will move the pendulum to the right-hand side. This instinctive knowledge indicates that the control actions to move the cart and pendulum to the same direction have opposite sign [143]. Since the main purpose for the position control of the inverted pendulum-cart system is to balance the pendulum at the straight upward direction, the combination of u_x and u_θ is defined as $u = u_x - u_\theta$. The PID controller parameters (K_P, K_I, K_D) for both the controllers have been adjusted to the optimum values for the desired control response. Two controller blocks (one is for inverted pendulum stabilization and another is for pendulum swing up control) as presented in Fig.2.21,

are proposed for efficient inverted pendulum control. Different simulation blocks for inverted pendulum control are shown in Fig. 2.22. Block diagram of the inverted pendulum controller system for stabilization and block diagram for the inverted pendulum swing up control scheme are shown in Fig.2.23 and 2.24 respectively.

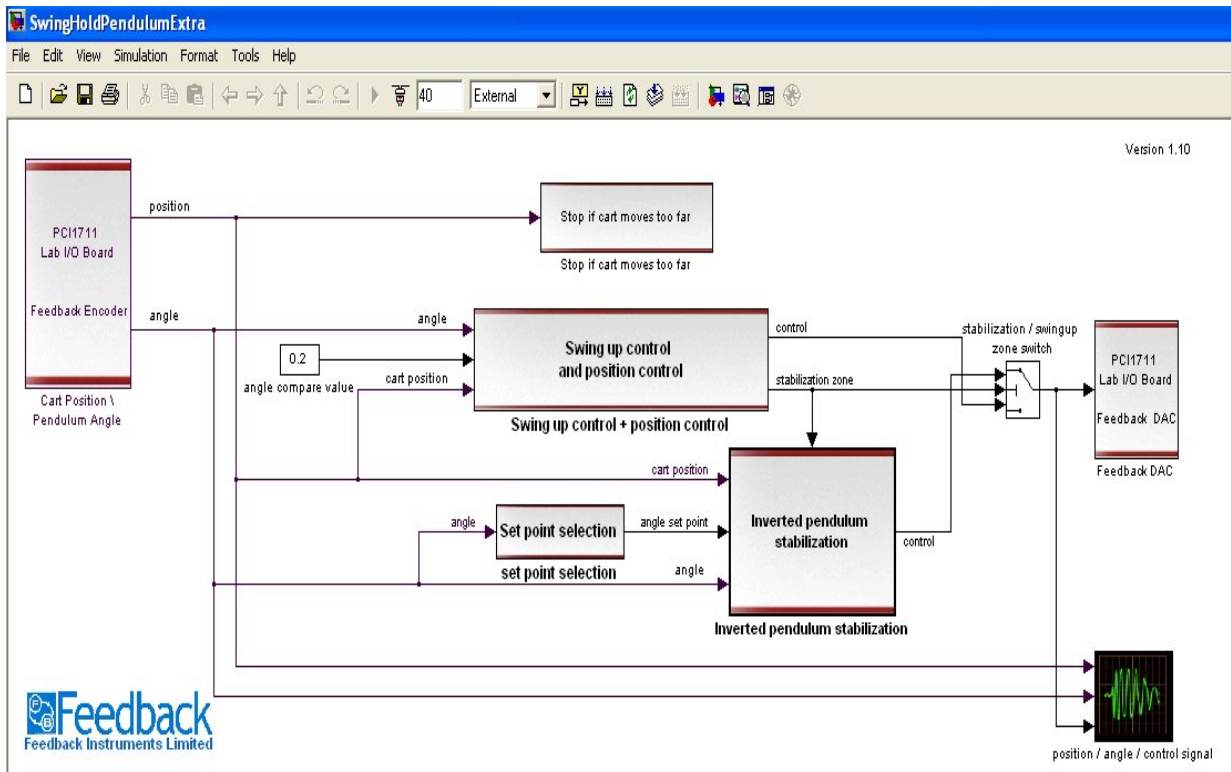


Figure 2.22: Subsystem blocks for inverted pendulum (swing up and stabilization) control.

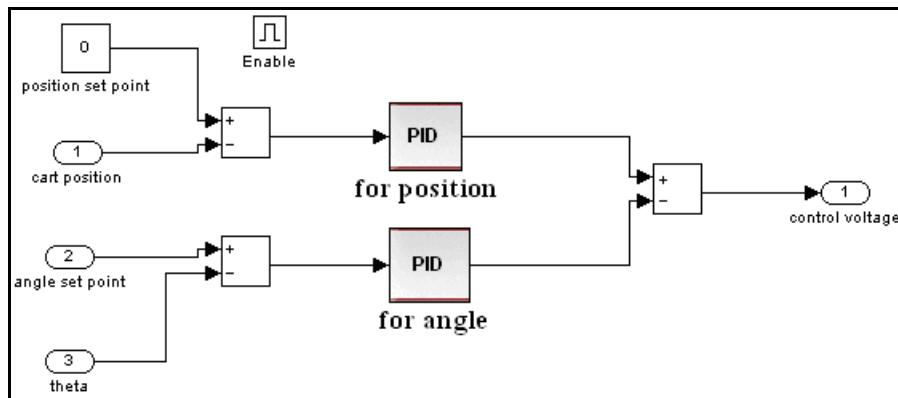


Figure 2.23: Block diagram of the inverted pendulum control scheme for stabilization.

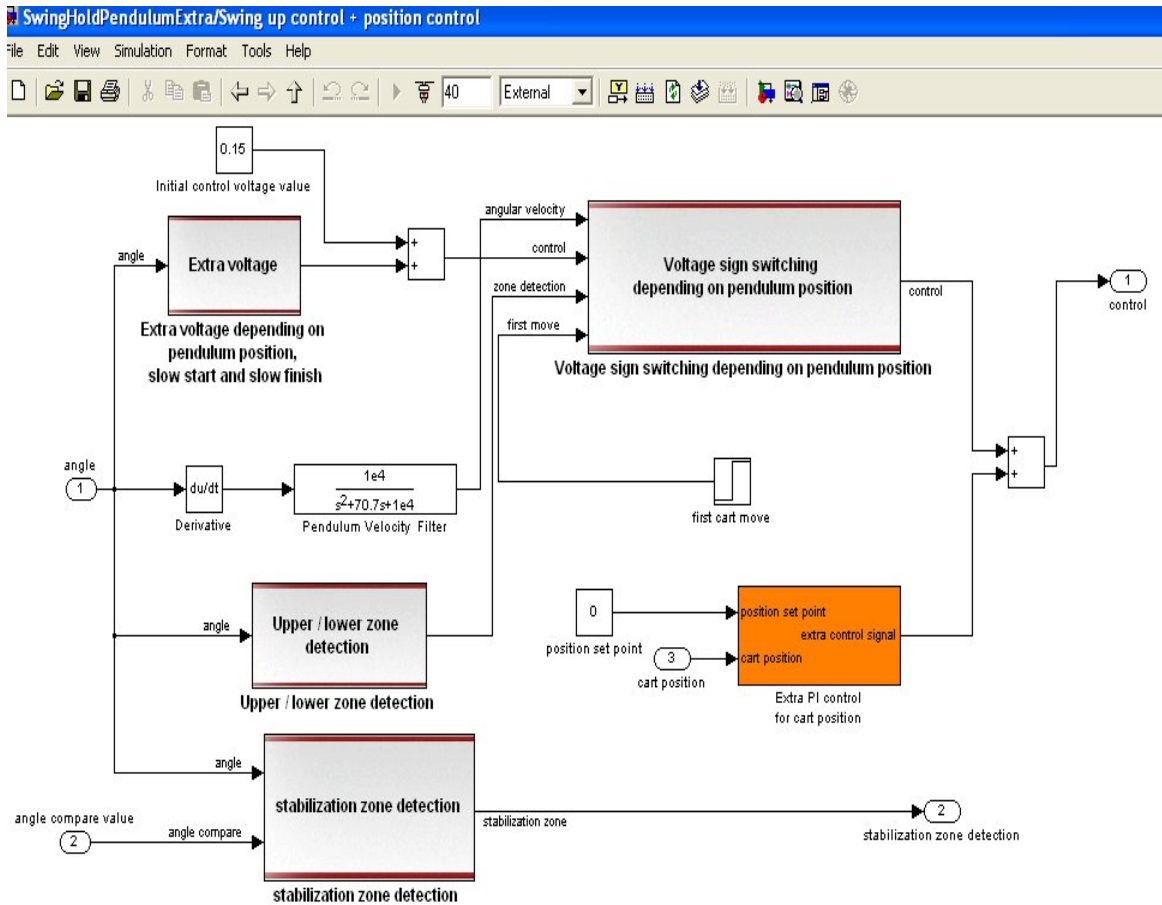


Figure 2.24: Block diagram of the inverted pendulum control scheme for swing up.

2.5.2.2 FPDC and STFPDC

The proposed hybrid controller, for efficient control of cart position and inverted pendulum angle is a PD type fuzzy logic controller (FPDC). As already mentioned the inverted pendulum control problem is divided into two separate control schemes [142], the swing up control (Fig. 2.24), which allows the pendulum to reach the upright position with as minimal angular velocity as possible; and the inverted pendulum stabilization (Fig.2.23) around the equilibrium point within a specified accuracy. Both these control objectives have to be attained simultaneously. A logical switch is used to change over between the swing up control mode and the inverted pendulum stabilization mode. The stabilization mode comes into effect once the pendulum reaches $[-10^\circ, +10^\circ]$ of the final vertical upright position. The control action begins in the swing up control mode if initially the pendulum arms are hanging free. Here, zone detection signals in combination with a position controller acts on the inverted pendulum system to achieve desired

sway of the arms. Once in the stabilization mode, the system is acted upon by twin controllers; viz position controller and angle controller (Fig. 2.25).

Inverted pendulum control for stabilization:

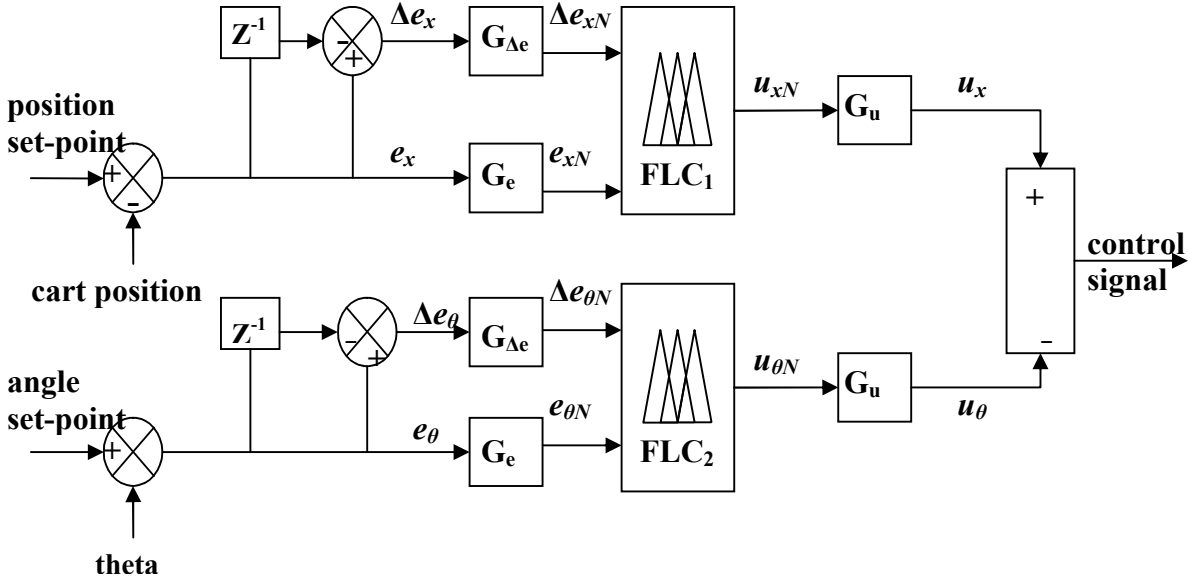


Figure 2.25: Block diagram of FPDC twin controller for inverted pendulum stabilization.

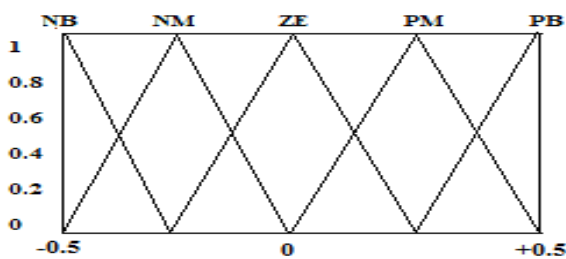


Figure 2.26: MFs of $e_x, \Delta e_x, u_x$.

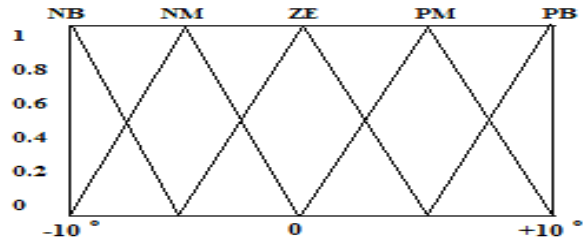


Figure 2.27: MFs of $e_\theta, \Delta e_\theta, u_\theta$.

Each of the fuzzy logic controller used for performing the control action consists of 5 symmetrical triangular membership functions (MFs) of equal base and 50% overlap, hence generating a set of $5^2=25$ fuzzy rules (Table 2.7). The inputs to the controller are, the amount of deviation in the cart position or pendulum angle from the desired set-point at a sampling instant k (denoted by e) and their corresponding change of error (denoted by Δe). Depending upon these two input parameters each of the fuzzy logic controller is capable of generating a control voltage

(denoted by u) that will effectively drive the system towards stability. The MFs of $e_x, \Delta e_x, u_x$ for position controller are shown in Fig. 2.26 and MFs of $e_\theta, \Delta e_\theta, u_\theta$ for angle controller are shown in Fig. 2.27.

Table 2.7: Fuzzy rules for position and angle control

$\Delta e/e$	NB	NM	ZE	PM	PB
NB	NB	NB	NB	NM	ZE
NM	NB	NB	NM	ZE	PM
ZE	NB	NM	ZE	PM	PB
PM	NM	ZE	PM	PB	PB
PB	ZE	PM	PB	PB	PB

The various input and output scaling factors play a role very similar to the gain coefficients in a conventional controller. The appropriate selection of these scaling factors is done based on the operator's knowledge of the system under study to obtain the best performance of the control system. The relationships between scaling factors ($G_e, G_{\Delta e}, G_u$) and the input and output variables ($e, \Delta e, u$) are shown below.

$$\Delta e(t) = e(t) - e(t-1)$$

$$e_N = G_e \cdot e$$

$$\Delta e_N = G_{\Delta e} \cdot \Delta e$$

$$u = G_u \cdot u_N$$

Inverted pendulum swing up control

Fig. 2.28 shows the block diagram of FPDC for swing up control. The output scaling factor (G_u) should be determined very carefully for the proper implementation of control logic; since it is directly related to the stability of the system. *Authors* in [22, 23] have proposed a self-tunable fuzzy based inference system to adjust the gain G_u on-line according to the current states of controlled processes. A very similar scheme is applied here for tuning of the FPDC position controller which is expected to give a better swing up control by modifying the controller gain by

a factor β , which is now given by *equation* (2.30). This self-tuning mechanism for FPDC will be denoted as STFPDC in our discussion henceforth (Fig. 2.29).

$$u = \beta \cdot G_u * u_N \tag{2.30}$$

where, β is a function of error(e) and change of error(Δe) of the system response. The fuzzy rule-base for computation of the gain updating factor β consists of 7 symmetrical triangular MFs of equal base and 50% overlap, hence generating a set of $7^2=49$ fuzzy rules as recorded in Table 2.8. The MFs of e , Δe and β are shown in Fig. 2.30 and Fig. 2.31 respectively.

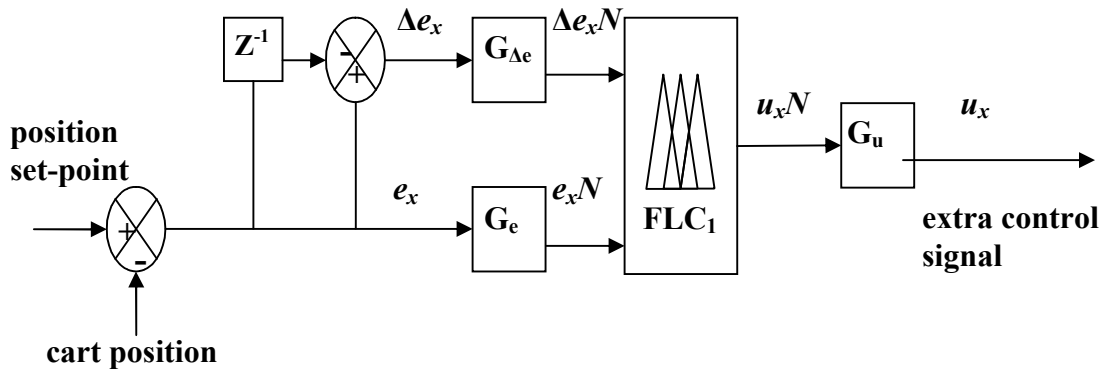


Figure 2.28: Block diagram of FPDC for swing up control.

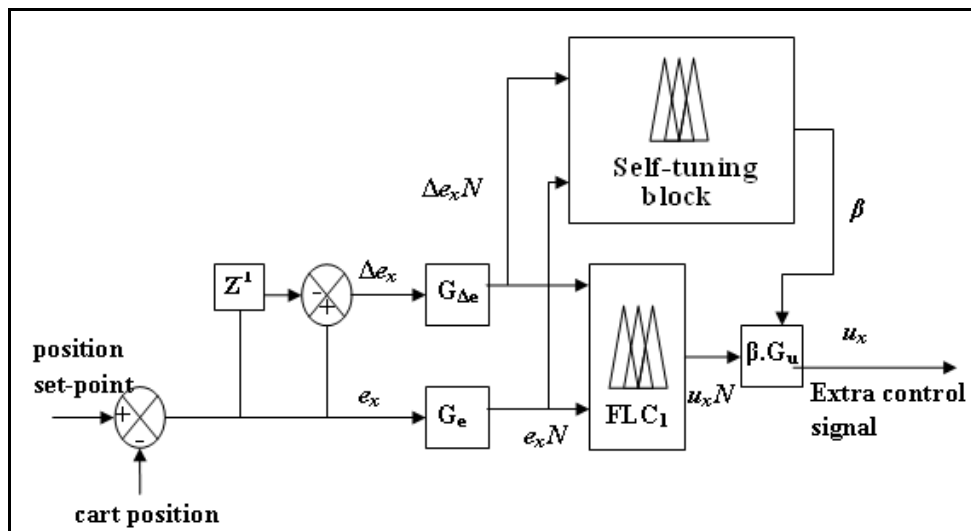


Figure 2.29: Block diagram of STFPDC for swing up control.

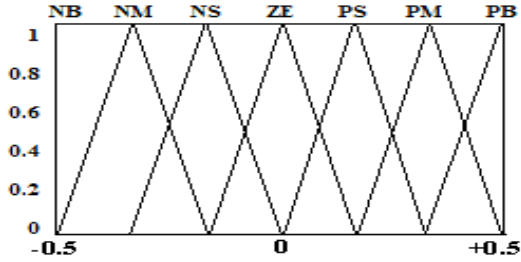


Figure 2.30: MFs of e and Δe .

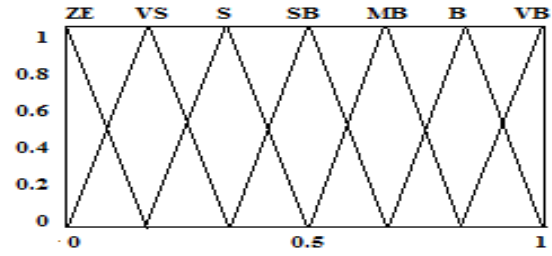


Figure 2.31: MFs for β .

Table 2.8: Fuzzy rules for computation of β

$\Delta e/e$	NB	NM	NS	ZE	PS	PM	PB
NB	VB	VB	VB	B	SB	S	ZE
NM	VB	VB	B	B	MB	S	VS
NS	VB	MB	B	VB	VS	S	VS
ZE	S	SB	MB	ZE	MB	SB	S
PS	VS	S	VS	VB	B	MB	VB
PM	VS	S	MB	B	B	VB	VB
PB	ZE	S	SB	B	VB	VB	VB

2.5.3 Comparative study

In this section we present the results of each of the control scheme discussed above and perform a comparative study to illustrate the effectiveness of our proposed scheme over the others.

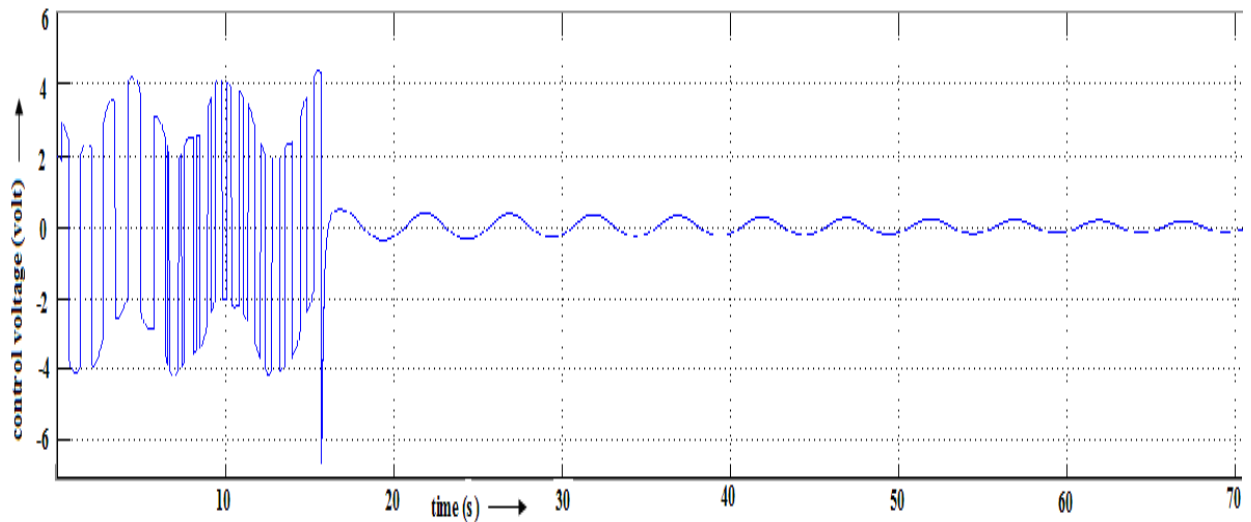


Figure 2.32: Plot of control voltage against time for PID.

Fig. 2.32 depicts that PID control action (voltage) applied to the inverted pendulum assembly is oscillatory in nature. Fig. 2.33 and Fig.2.35 represent the variation in the pendulum cart position and pendulum sway angle respectively under the influence of the PID control action with sampling time of $0.05s$ for $70s$. The position response curve of Fig. 2.33 shows that under PID control action, the pendulum cart sub-system exhibits large initial overshoot as well as undershoot and the cart never comes in completely steady position. The pendulum sway angle shown in Fig. 2.35 remains in a state of continued oscillation for a long period before eventually settling down. The inverted pendulum takes around $16s$ to swing up.

Fig. 2.34 shows the plot of pendulum cart position against time. The results show that the pendulum cart is stabilized within approximately $8s$ in case of STFPDC while it takes much longer time in case of FPDC. The pendulum cart oscillation is also limited to $0.2m$, which was much more in case of FPDC and PID controller. Fig. 2.36 and 2.37 show the plots of the inverted pendulum angle (in radian) against time for each of the control scheme separately for the sake of better observation. Figs. 2.35, 2.36 and 2.37 clearly reveal that in case of STFPDC the system takes the least time to reach and remain in its vertical upright position. The detailed performance indices are tabulated in Table 2.9 and Table 2.10. Note that various integral performance indices of the system response like IAE , ISE and $ITAE$ are significantly reduced for STFPDC when compared to FPDC.

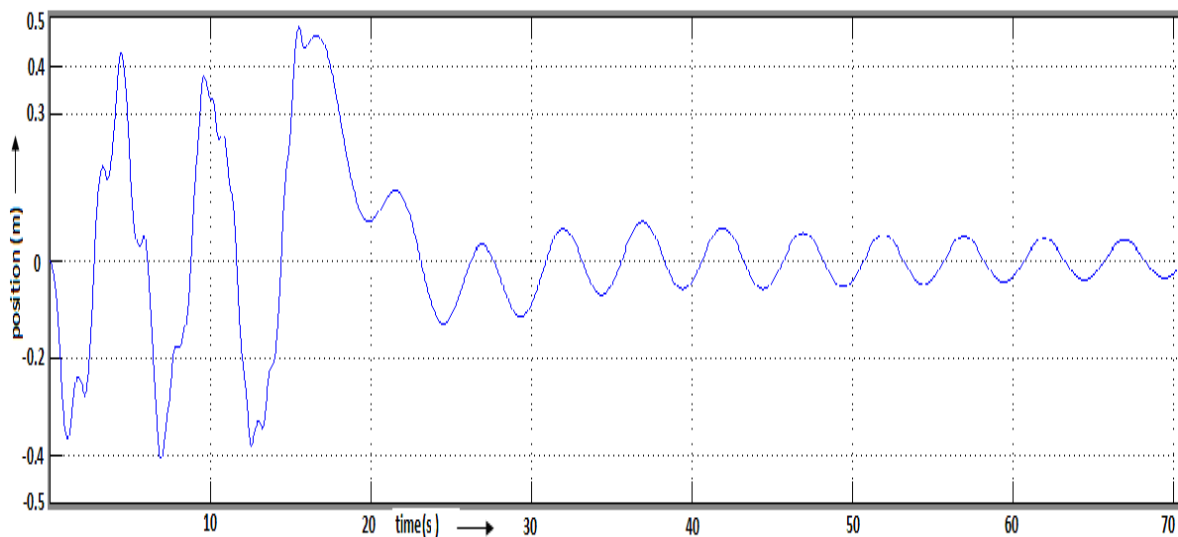


Figure 2.33: Plot of pendulum cart position against time for PID control.

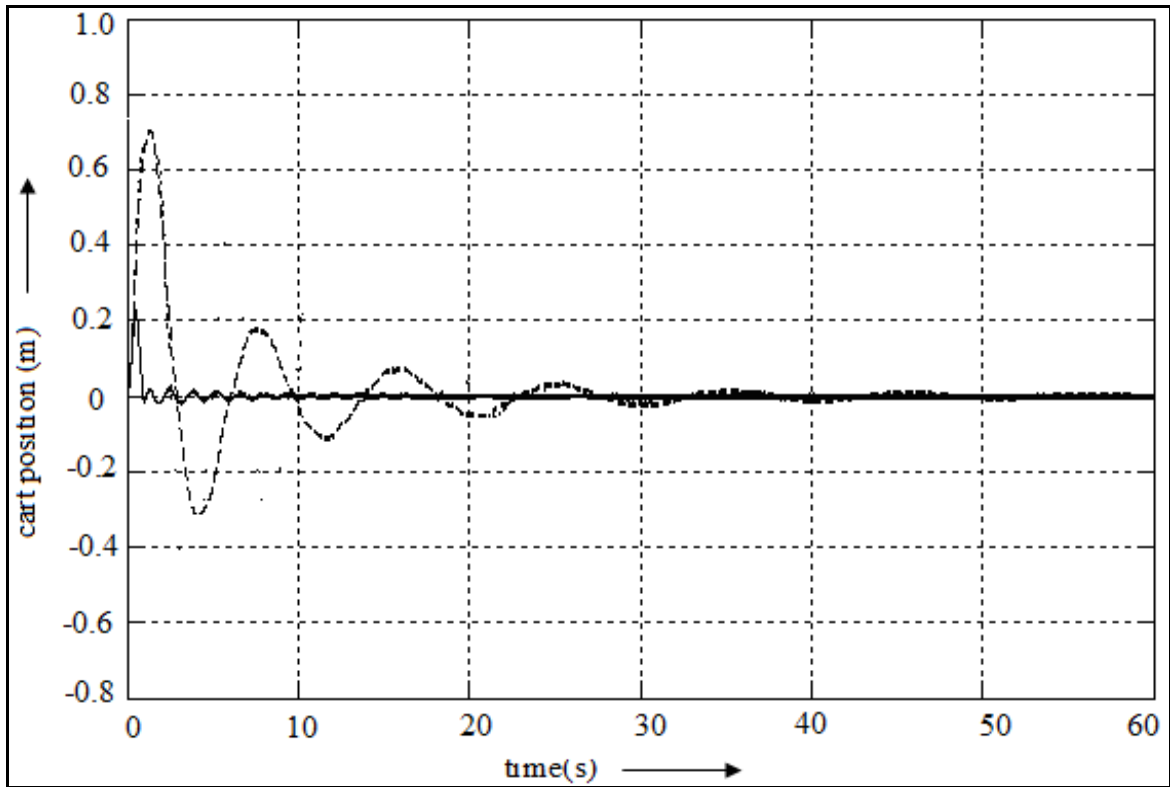


Figure 2.34: Plot of pendulum cart position against time (dashed-FPDC; solid-STFPDC).

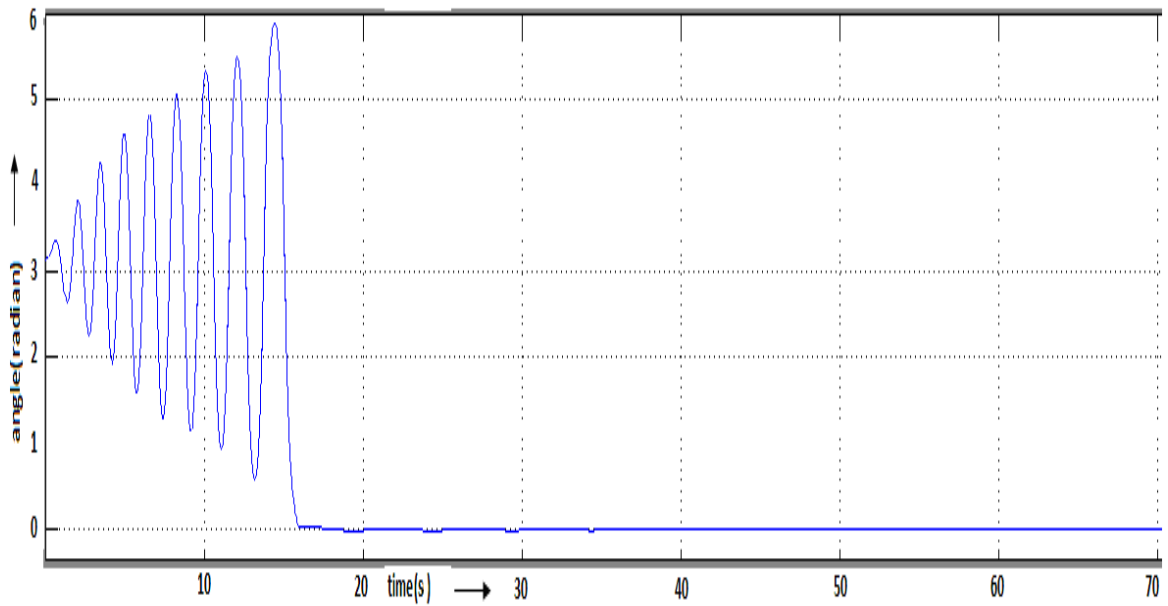


Figure 2.35: Plot of inverted pendulum sway angle against time during swing up for PID.

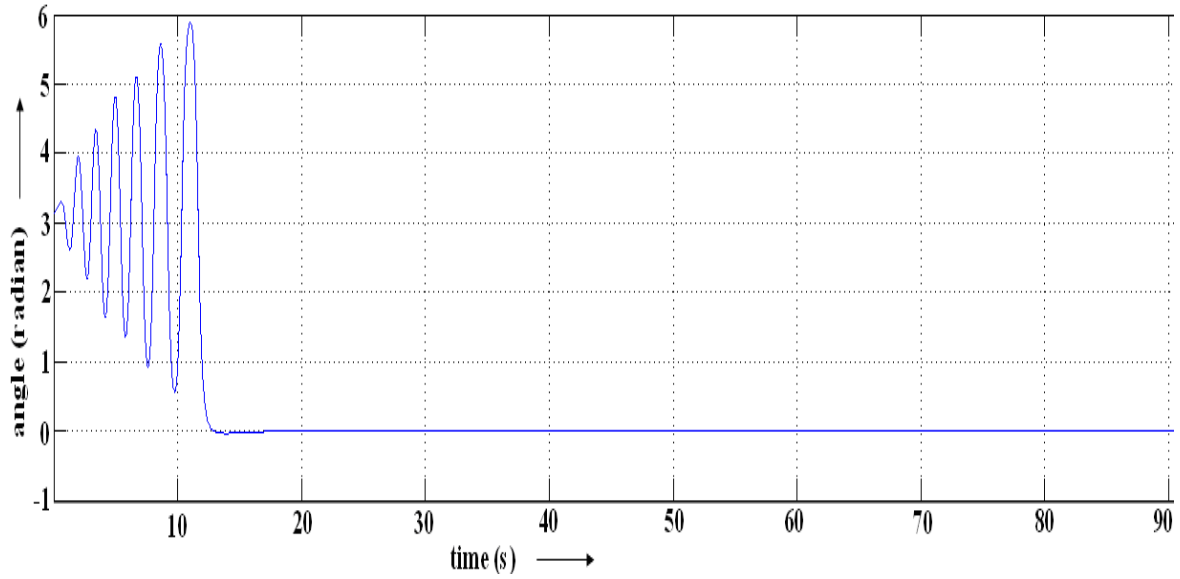


Figure 2.36: Plot of inverted pendulum sway angle against time during swing up for FPDC.

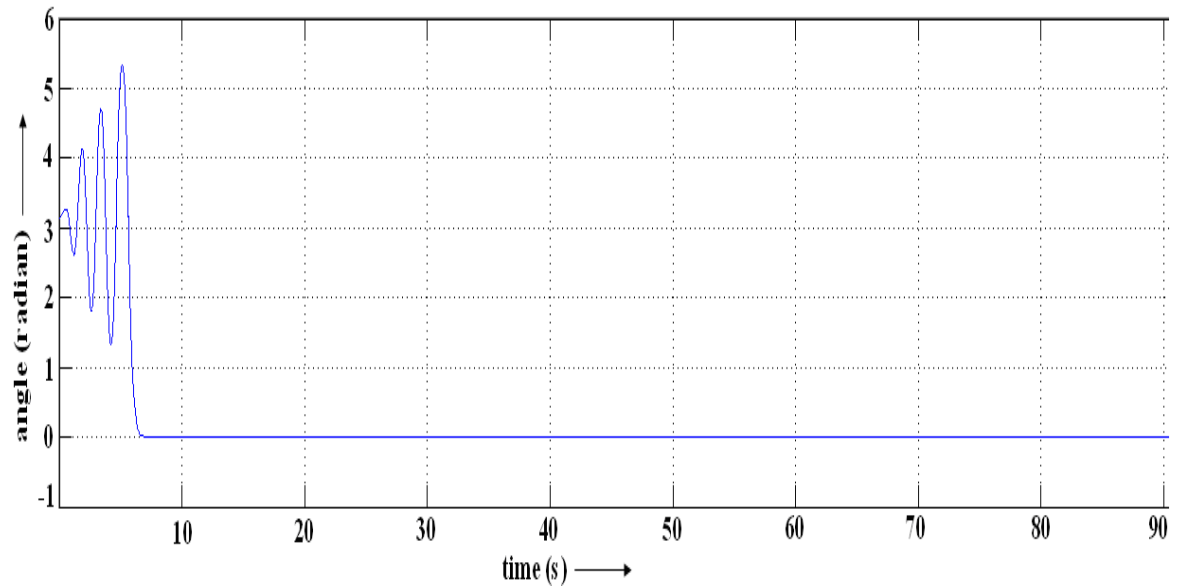


Figure 2.37: Plot of inverted pendulum sway angle against time during swing up for STFPDC.

Table 2.9: Performance analysis of the controllers for inverted pendulum control

Type of controller	t_p (s)	t_s (s)	Swing up time (s)
PID	4.2	>70	16
FPDC	1.3	35.71	12
STFPDC	0.5	8.12	7

Table 2.10: Performance indices of the controllers for inverted pendulum control

Type of controller	IAE	ISE	ITAE
FPDC	3.1523	0.0903	28.8326
STFPDC	0.1751	0.0019	0.4249

2.6 Conclusion

In our self-tuning scheme for FLC's, the output SF, which may be considered equivalent to the controller gain is tuned on-line by fuzzy rules defined on e and Δe . The most important feature of the proposed scheme is that it depends neither on the process being controlled nor on the controller used. Conceptually, our scheme differs from others as it attempts to implement the operator's strategy while running a plant. The proposed self-tuning scheme is applied to both PI- and PD-type FLC's for controlling two practical processes. At first, we demonstrate the PI-type self-tuning fuzzy controller in a Heating, Ventilation and Air-conditioning system, which is a nonlinear and time variant system. The performance of STFPIC is successfully investigated in the supply air pressure loop for HVAC system. The usefulness of the self-tuning scheme is also tested in a laboratory based inverted pendulum, which is highly nonlinear and highly unstable system. The self-tuning fuzzy PD controller (STFPDC) is applied to stabilize the inverted pendulum, exhibits effective and improved performance compared to its conventional fuzzy and non-fuzzy controllers. Also, the proposed twin control scheme for inverted pendulum control reduces the computational complexity and processing time by decreasing the number of fuzzy *if-then* rules.

CHAPTER 3

Development of self-tuning fuzzy controller through relay feedback approach - STFPIC α

3.1 Introduction

PID controllers are mostly used for different industrial applications [145]. In the absence of expert or the complete knowledge of the process these types of controllers are the best choices [146]. In a PID controller, the proportional part is responsible for following the desired set-point, while the integral and derivative part account for the accumulation of past errors and the rate of change of error in the process respectively. But, as discussed in previous chapter, these conventional controllers are not well suited for ill-defined and nonlinear systems. In such cases, we may think about some knowledge-based system [12, 147]. In *chapter-2*, we have noticed that depending on the input error (e) and change of error (Δe) of a process, an operator always tries to modify the output SF *i.e.* controller gain to enhance the system performance and to achieve stable control output using self-tuning fuzzy logic controller. In some earlier research [22, 23], it observed that the on-line *fuzzy gain modifier* (β) *is further augmented by some multiplicative factor which is chosen by trial in order to maintain almost the same rise time.*

In our scheme to replace this arbitrary gain multiplier, a relay feedback approach is implemented [148]. The fuzzy gain modifier (β) [22] is further augmented by a multiplicative factor (α), which is directly related to the system dynamics and derived by relay feedback experiment. In 1984, Åström and Hägglund [149] presented a relay feedback system to generate sustained oscillation as an alternative to the conventional continuous cycling technique for controller tuning. The system dynamics like: ultimate period (P_u), ultimate gain (K_u) and ultimate frequency (ω_u) can be found easily from the principal harmonic approximation. But, besides the

estimation of tuning parameters, identification of mathematical model in an auto-tune system is also desirable to achieve an improved control performance [150]. In our proposed scheme of STFPIC design, we have used the system dynamics to find out process specific appropriate gain multiplicative factor (α), instead of a fixed value (*i.e.*, 3) used in [22, 23]. Here, the proposed STFPIC that has used process specific appropriate gain multiplicative factor (α), is termed as STFPIC α .

Apart from α determination, another highlighting point of our scheme is significant number of rule reduction. The proposed scheme of fuzzy controller is implemented with almost 50% reduced rules as compared to earlier design [22, 23]. Robustness of the STFPIC α is demonstrated [148] for a wide range of processes including nonlinear and marginally stable system with a considerable variation in dead-time and also in a real time application.

3.2 Relay feedback tuning

The performance of the control systems used for process industries depend on its proper design and tuning. The most crucial phase of a successful controller design is its tuning. More than 70 years ago, in 1942 *Ziegler* and *Nichols* [151] proposed the ultimate cycling method for controller tuning. The PID controller tuning parameters are dependent on two factors, ultimate gain K_u and ultimate period P_u . However, this iterative tuning method is often difficult to apply in practice because it is time consuming, particularly for a process with large time constant. On the other hand, for designing model based controller, like internal model control (IMC), it is need to identify the process model, which is often based on the critical point, *i.e.*, K_u and P_u . Therefore, estimation of critical point plays an important role on the performance of process control systems. *Åström and Hägglund* [149] developed an attractive and simple experimental relay feedback method to determine K_u and P_u . The relay feedback test has many positive features that led to its widespread use:

- i) Only one parameter has to be specified (relay height).
- ii) The time it takes to run the test is short.
- iii) The test is a closed loop, so the process is not driven away from the set-point.

However, the precision of this technique for normal processes is not very accurate. Many research works on modifying the relay feedback auto-tuning method have been reported in the literature [152-154]. An automatic tuning procedure can be divided into two stages: *the identification phase* and *the controller design phase*.

In the identification phase, the Åström-Hägglund auto-tuner is based on the observation that a feedback system in which the output (y) lags behind the input (u) by $-\pi$ radian may oscillate with a period P_u . This is a well-known observation; however, the oscillation is carried out in a new manner. To generate the sustained oscillation, a relay feedback test is performed (Fig. 3.1). Initially, the input (u) is increased by h ($u=u^{ss} + h$), where u^{ss} is the steady-state value of u . As soon as the output is moving upward, the input is switched to the lower position ($u=u^{ss} - h$), as shown in Fig. 3.2. This procedure is repeated until the cycling reaches a steady form. From the relay feedback test, the familiar ultimate gain (K_u) and ultimate frequency (ω_u) are readily available. They can be approximated as:

$$K_u = 4h/\pi A \quad (3.1)$$

$$\omega_u = P_u/2\pi \quad (3.2)$$

In *equation* (3.1) and (3.2), A is the amplitude of the oscillation and P_u is the period respectively as shown in Fig. 3.2. Following the identification phase, the controller can be designed, from K_u , and ω_u .

In the design phase, the classical ZN method probably is the simplest and the most popular choice. The controller gain (K_c) and reset time (T_i) of a PI controller can be found from *equation* (3.3) and (3.4) with the simple calculation as follows:

$$K_c = K_u/2.2 \quad (3.3)$$

$$T_i = P_u/1.2 \quad (3.4)$$

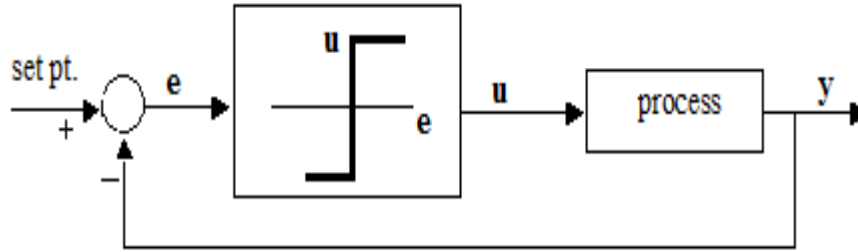


Figure 3.1: Block Diagram of relay feedback test.

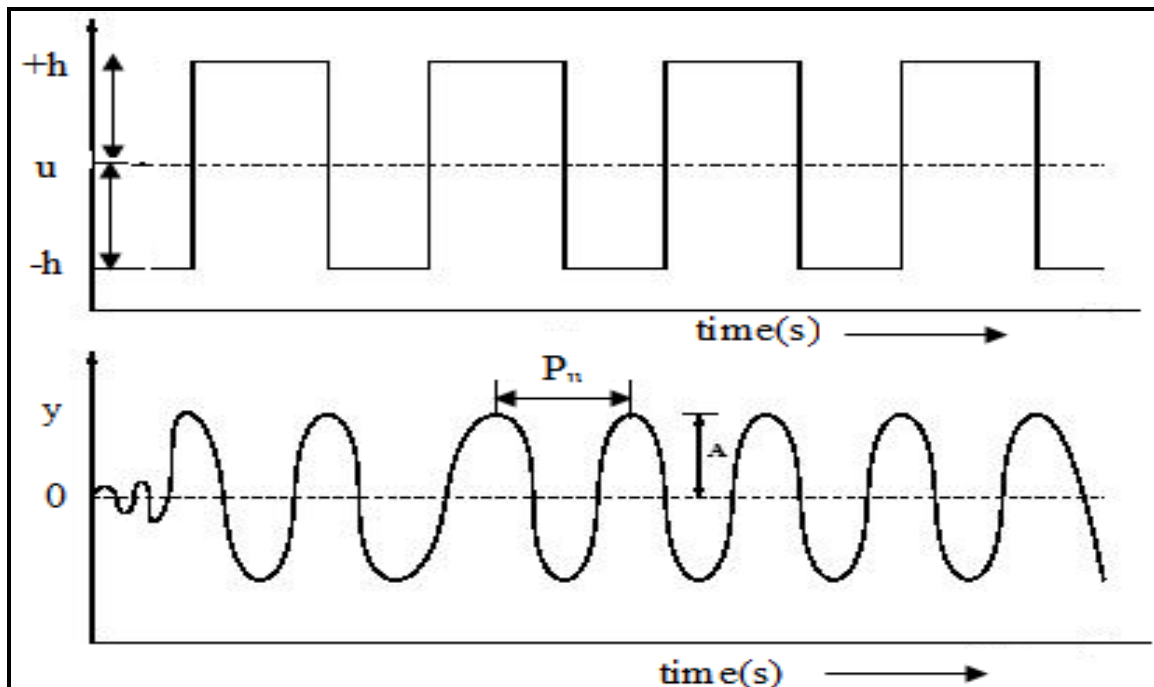


Figure 3.2: Relay feedback test response.

3.3 Self-tuning scheme with dynamic gain (α)

In this section, we incorporate a multiplicative factor (α) in the output of the controller that is obtained using relay feedback tuning. The block diagram of the self-tuning fuzzy PI controller (STFPIC) has already been shown in Fig. 2.3 in *chapter-2*. Here, the block diagram of the self-tuning fuzzy PI controller with the process dynamics based multiplicative factor (STFPIC α) is presented in Fig. 3.3. In this scheme, output SF of STFPIC is further modified by the process specific appropriate gain multiplicative factor (α) derived from relay feedback mechanism. Different design aspects are elaborated in the following subsections.

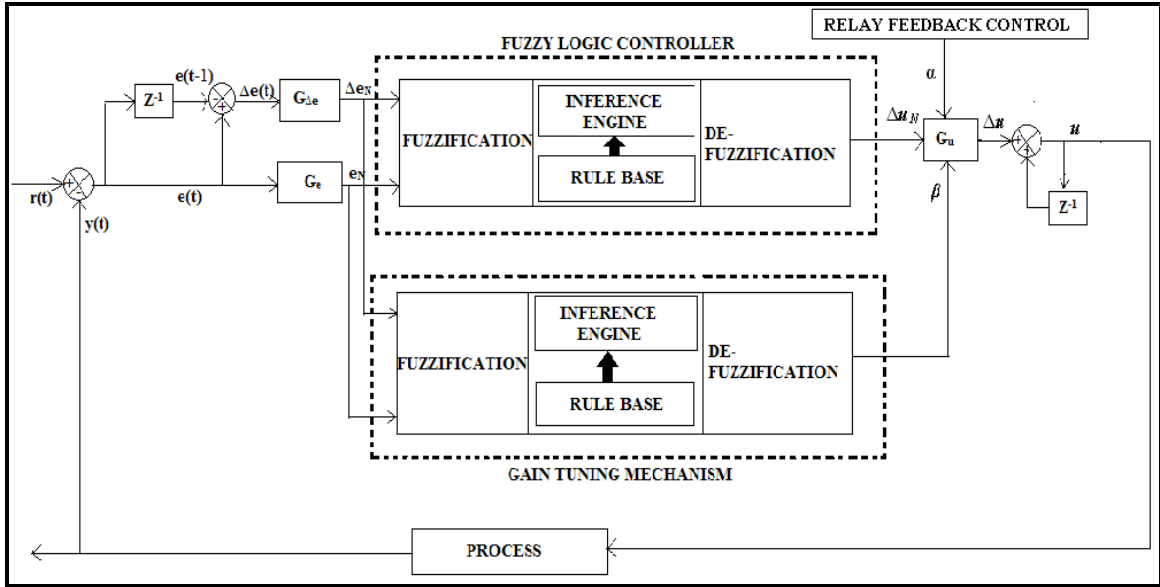


Figure 3.3: Block Diagram of STFPICa.

Fuzzy Membership Functions

The MFs for controller inputs error (e), change of error (Δe) and controller output (Δu) are defined on the normalized domain $[-1, 1]$, whereas the MFs of β is defined on $[0, 1]$ as shown in Fig. 3.4 and Fig. 3.5 respectively. The only difference with the earlier design (*chapter-2*) is that here we have used only five linguistic variables in place of seven. The term sets of e , Δe , Δu for PI type FLC contain the same linguistic expressions for the magnitude part of the linguistic values, i.e., $LE = L\Delta E = L\Delta U \{NB, NM, ZE, PM, PB\}$. Similarly, MFs of β are mapped to the MFs $\{Z, S, M, B, VB\}$.

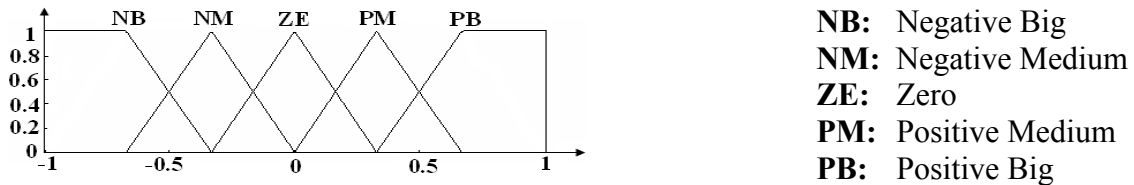


Figure 3.4: MFs of e , Δe and Δu .

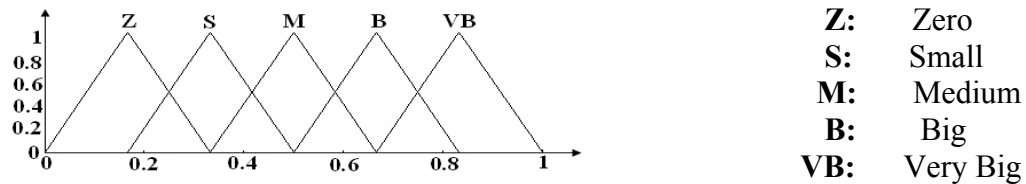


Figure 3.5: MFs of β .

Scaling Factor

The normalized inputs (e_N and Δe_N) and normalized output (Δu_N) of the STFPICA are defined on the normalized domain $[-1, 1]$. The relationship between the SFs (G_e , $G_{\Delta e}$, G_u) and the input and output variables of the STFPICA are as follows:

$$e_N = G_e \cdot e$$

$$\Delta e_N = G_{\Delta e} \cdot \Delta e$$

$$\Delta u = (\alpha \cdot \beta \cdot G_u) \cdot \Delta u_N$$

PI-type FLCs use output scaling factor G_u only, whereas the output SF of STFPICA is obtained by multiplying G_u with α and β .

Rule-Bases

Fuzzy *if-then* rule-bases for computing Δu and gain updating factor (β) are tabulated in Table 3.1 and Table 3.2 respectively. The design rule-bases are flexible in nature and these can be modified according to process requirement. In STFPICA, 25 control rules and 25 gain rules are applied, instead of 49 control rules and 49 gain rules [22, 23] used in STFPIIC.

Table 3.1: Fuzzy rules for computation of Δu

$\Delta e/e$	NB	NM	ZE	PM	PB
NB	NB	NB	NB	NM	ZE
NM	NB	NB	NM	ZE	PM
ZE	NB	NM	ZE	PM	PB
PM	NM	ZE	PM	PB	PB
PB	ZE	PM	PB	PB	PB

Table 3.2: Fuzzy rules for computation of β

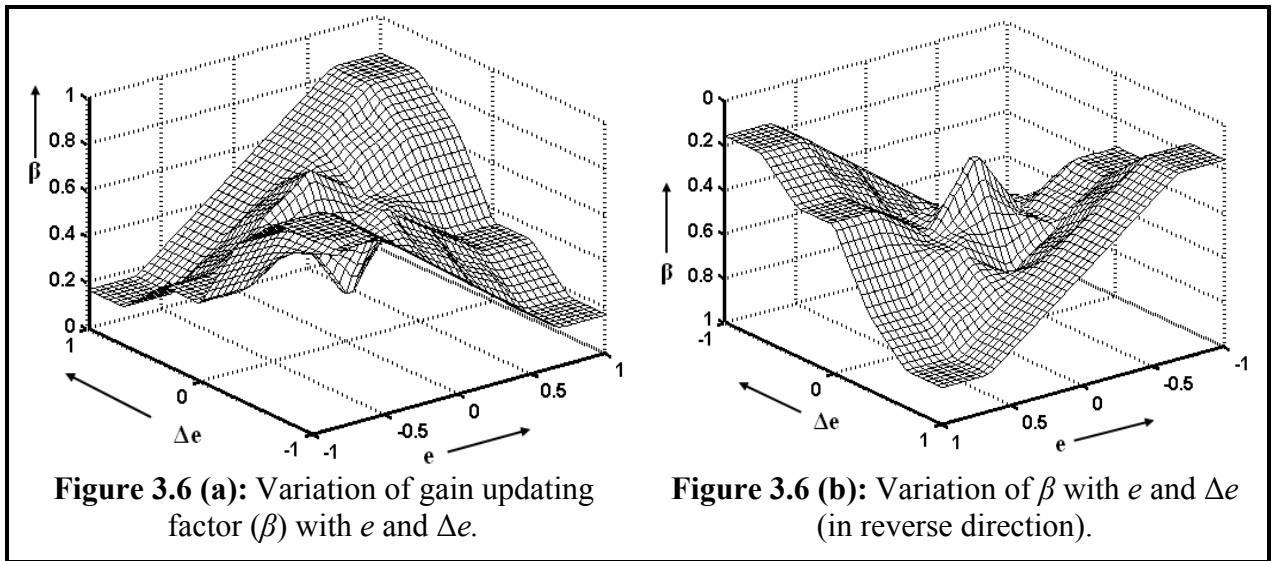
$\Delta e/e$	NB	NM	ZE	PM	PB
NB	VB	B	M	S	Z
NM	B	M	B	M	S
ZE	S	M	Z	M	S
PM	S	M	B	M	B
PB	Z	S	M	B	VB

Gain updating factor (β)

The computation of β is done using the 25 fuzzy rule-bases mentioned in Table 3.2 in the form:

If e is E and Δe is ΔE Then β is β .

Fig. 3.6 (a & b) illustrates the characteristics of β as a function of e and Δe . The highly nonlinear rule-base for β is designed to achieve a lower overshoot, reduced settling time but not at the cost of increased rise time. Basically the rule-base for β should be developed by the designer according to the type of response one wishes to achieve.



Computation of multiplicative factor (α)

Earlier study [105, 155] reveals that a fixed value (*i.e.* 3) of α is considered irrespective of the type of controlled system though its influence to the process response is significant. In this design, we try to eliminate this problem by choosing a process specific value of α that depends on the process dynamics. For computation of α , relay feedback test is performed on the individual processes with various dead-times, to find out ultimate gain (K_u) and ultimate period (P_u) and then the following relation is proposed:

$$\alpha = (K_u P_u) / 2.5 \quad (3.5)$$

Control surface observation of FPIC and STFPICa

The control surfaces, *i.e.*, controller output (Δu) versus inputs (e and Δe) for the FPIC and STFPICa are shown in Fig. 3.7 and Fig. 3.8 respectively. Careful observation reveals that the control surface of STFPICa (Fig. 3.8) is more nonlinear as well as smooth in nature due to

additional 25 gain rules as presented in Table 3.2 and Fig. 3.6. Fig. 3.7 indicates that the only limited number of rules in FPIC may not be sufficient to provide such nonlinear control action which may be needed for achieving desired performance.

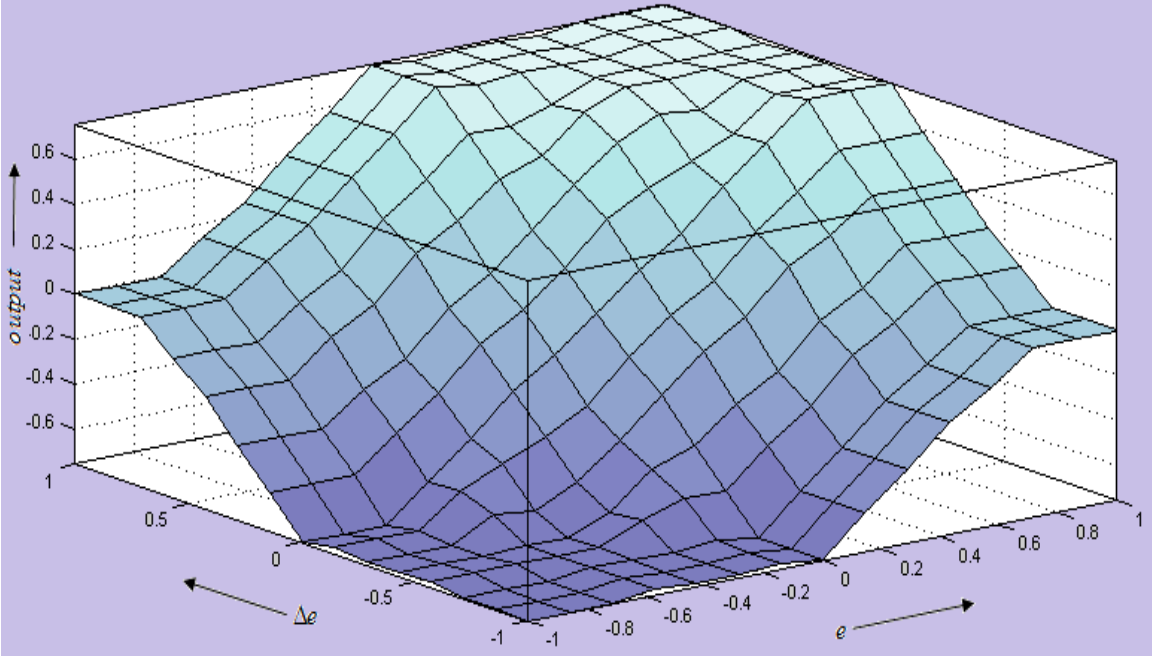


Figure 3.7: Control surface of FPIC.

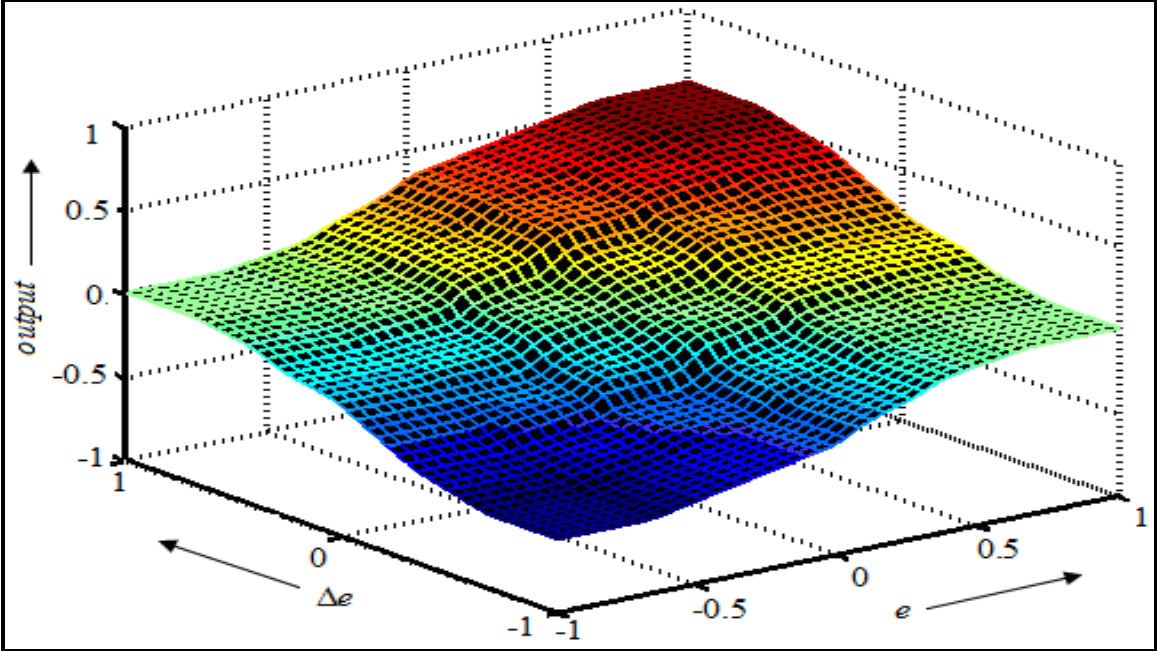


Figure 3.8: Control surface of STFPICα.

3.4 Simulation experiments with FPIC, STFPIC and STFPIC α

The performance of STFPIC α is compared with other controllers for various processes with different values of dead-time. The controllers are evaluated in terms of different performance indices. To establish the robustness of the proposed scheme, the same MFs (Fig. 3.4 and Fig. 3.5) and rule-bases (Table 3.1 and Table 3.2) are used for all the processes. As we know that rise time and peak overshoot normally cannot be reduce simultaneously because of their conflicting nature. However, in self-tuning FLCs, we attempt to reduce peak overshoot without sacrificing the rise time.

In this proposed scheme the value of α is not fixed, it is process dependent. Therefore, the value of α [$\alpha=(K_u.P_u)/2.5$] is calculated for STFPIC α , by applying relay feedback tuning mechanism to each of the investigating processes. For simplicity, irrespective of the controller its output SF is represented as G_u in Table 3.3 to Table 3.5. However, actually it will be as follows:

$$\text{Output SF of FPIC} = G_u \quad (3.6)$$

$$\text{Output SF of STFPIC} = 3\beta G_u \quad (3.7)$$

$$\text{Output SF of STFPIC}\alpha = \alpha\beta G_u \quad (3.8)$$

The above scheme is demonstrated in a linear, nonlinear and marginally stable system with varying dead-time. The performances are also checked with load disturbances.

Second Order Linear Process

Consider a 2nd order linear process with dead-time L :

$$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.2y = u(t - L) \quad (3.9)$$

The process is studied with unit step input for varying dead-time; $L=0, 0.1$ and 0.3 . The corresponding results for FPIC (with 49 rules), STFPIC (with 98 rules and constant gain *i.e.*3) and STFPIC α (with 50 rules) are tabulated in Table 3.3. Step responses of linear system with various dead-time are shown in Fig. 3.9 and the response due to load change (for $L=0.2$ at $t=40s$) is plotted in Fig. 3.10. The results due to load variation are presented in Table 3.6. From Fig. 3.9 and Fig. 3.10, we have observed that STFPIC α can control a linear system more efficiently as

compared to other controllers at normal condition and even with load disturbances. The STFPIC α with 50 rules is not only matched the STFPIC performance, but also improves the rise time.

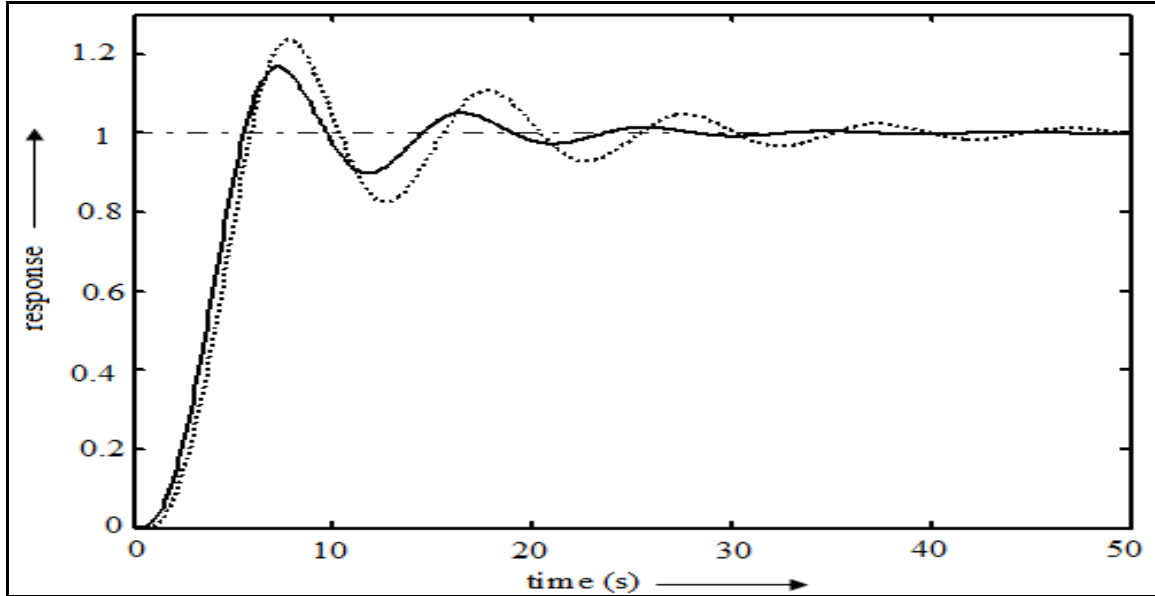


Figure 3.9: Responses of process (3.9) with STFPIC α for $L=0$ (solid) and $L=0.3$ (dotted).

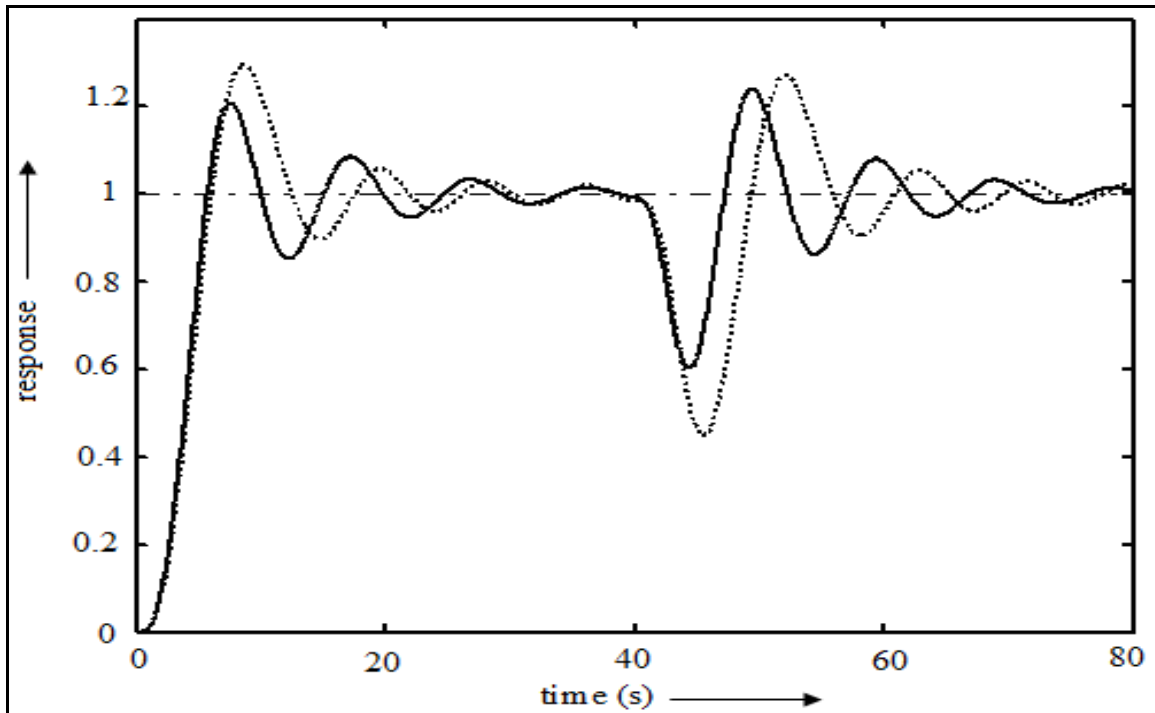


Figure 3.10: Responses of process (3.9) with $L=0.2$ with load disturbance at $t=40s$ (FPIC-dotted and STFPIC α -solid).

Table 3.3: Performance analysis for the linear 2nd order process (3.9)

L	FLC	G _u	%OS	t _r (s)	t _s (s)	IAE	ITAE	ISE
0.0	FPIC	0.02	25.97	5.8	20.1	4.81	24.13	2.70
	STFPIC	0.06	15.01	5.6	19.6	4.15	17.81	2.42
	STFPIC _α	0.073	16.76	5.3	18.1	4.15	18.69	2.39
0.1	FPIC	0.02	27.56	5.8	24.1	5.14	31.71	2.80
	STFPIC	0.06	17.30	5.6	19.9	4.45	23.94	2.45
	STFPIC _α	0.069	18.89	5.5	20.9	4.50	24.26	2.50
0.3	FPIC	0.02	32.10	5.9	42.0	6.44	79.93	3.05
	STFPIC	0.06	23.24	5.6	30.1	5.38	44.19	2.74
	STFPIC _α	0.064	23.65	5.6	31.5	5.42	44.88	2.75

Second Order Marginally Stable System

Let us consider a marginally stable system:

$$\frac{d^2y}{dt^2} + \frac{dy}{dt} = u(t-L) \quad (3.10)$$

Marginally stable system with dead-time is a difficult system to control for the existence of poles on the imaginary ($j\omega$) axis. The performance indices with various dead-times are listed in Table 3.4. Performance under load disturbance (applied at $t = 55s$) is shown in Fig. 3.12 and in Table 3.6. From Figs. 3.11 and 3.12, and from the Tables 3.4 and 3.6 we observe that in each case the proposed controller performs more satisfactorily compared to other controllers. Also STFPIC_α with almost 50% rules compared to STFPIC provides comparable results as shown in Table 3.4. From this study, we find that STFPIC_α considerably decreases the overshoot though we knew that overshoot and rise time conflict each other.

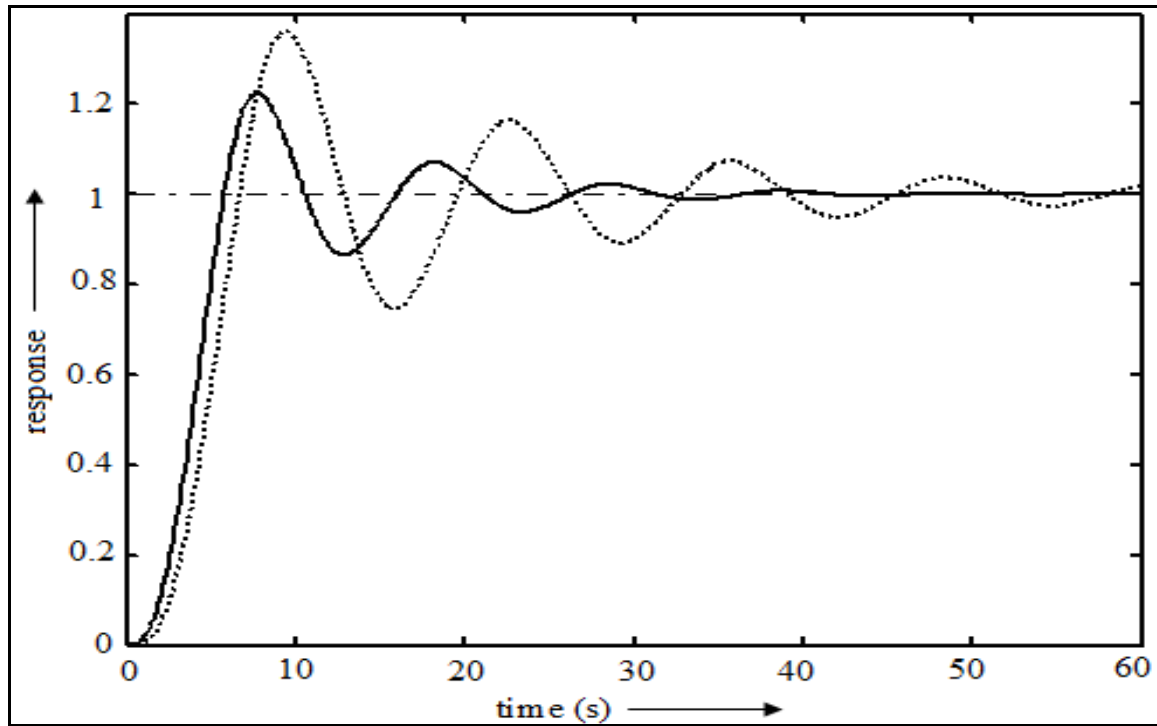


Figure 3.11: Responses of process (3.10) with STFPIC α for $L=0$ (solid) and $L=0.3$ (dotted).

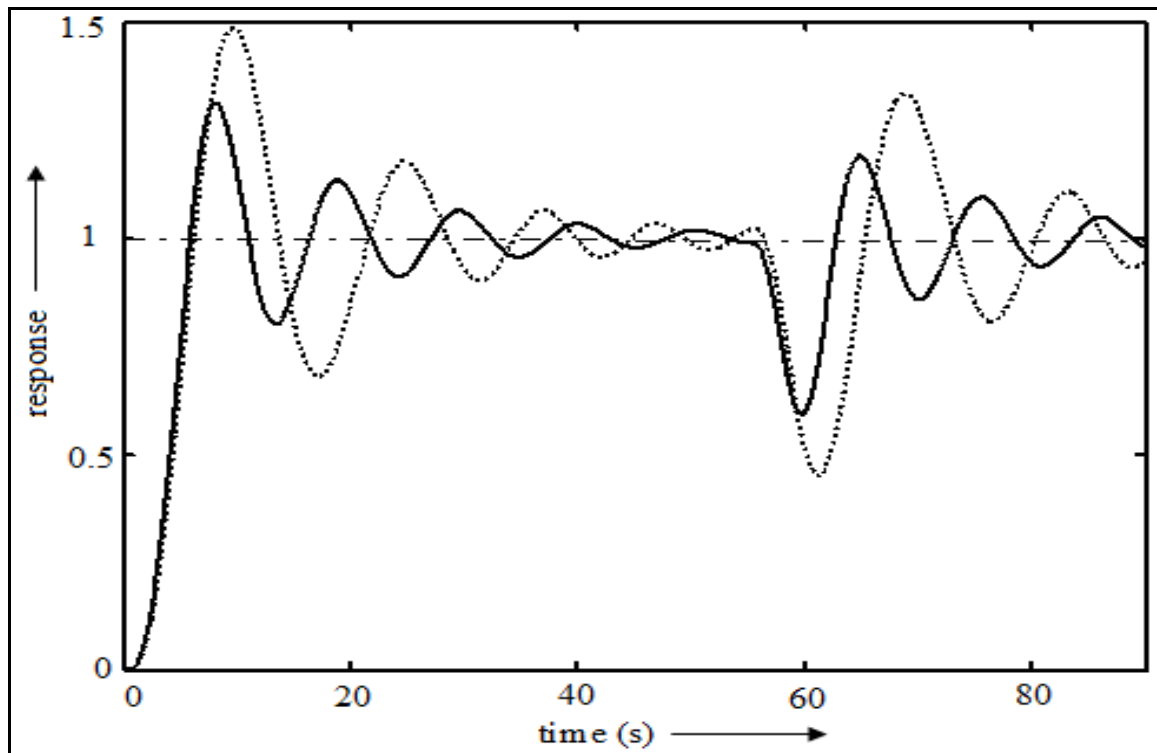


Figure 3.12: Responses of process (3.10) with $L=0.2$ with load disturbance at $t=55$ s (FPIC-dotted and STFPIC α -solid).

Table 3.4: Performance analysis for the marginally stable 2nd order process (3.10)

L	FLC	G _u	%OS	t _r (s)	t _s (s)	IAE	ITAE	ISE
0.0	FPIC	0.016	40.85	6.1	31.9	7.23	64.86	3.43
	STFPIC	0.048	21.36	5.9	28.0	5.07	32.69	2.70
	STFPIC _α	0.060	22.48	5.6	24.8	4.91	29.30	2.63
0.1	FPIC	0.016	44.62	6.1	41.2	8.10	87.15	3.71
	STFPIC	0.048	24.38	5.8	28.4	5.50	43.26	2.83
	STFPIC _α	0.059	26.71	5.6	30.1	5.55	42.63	2.79
0.3	FPIC	0.012	54.79	6.9	64.3	12.54	220.20	5.33
	STFPIC	0.036	34.73	6.5	47.1	7.69	96.93	3.52
	STFPIC _α	0.041	36.00	6.5	48.9	8.05	100.75	3.58

Second Order Nonlinear Process

We have tested the performance of STFPIC_α and other controllers in a nonlinear process with different dead-times. Consider the nonlinear process:

$$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.25y^2 = u(t-L) \quad (3.11)$$

Performance of this nonlinear process for $L=0.1$ and 0.4 is observed. Fig. 3.13, Fig. 3.14 and Table 3.5 reveal that in this case also STFPIC_α exhibits better performance over FPIC. STFPIC_α reduces the *ITAE* value for different dead-times. The load disturbance is applied at $t=30s$ and the corresponding response is plotted at Fig. 3.14. Table 3.6 clearly indicates that STFPIC_α have outperformed others, even at load disturbances. From Table 3.6, it is observed that at different dead-times STFPIC and STFPIC_α show almost identical performances.

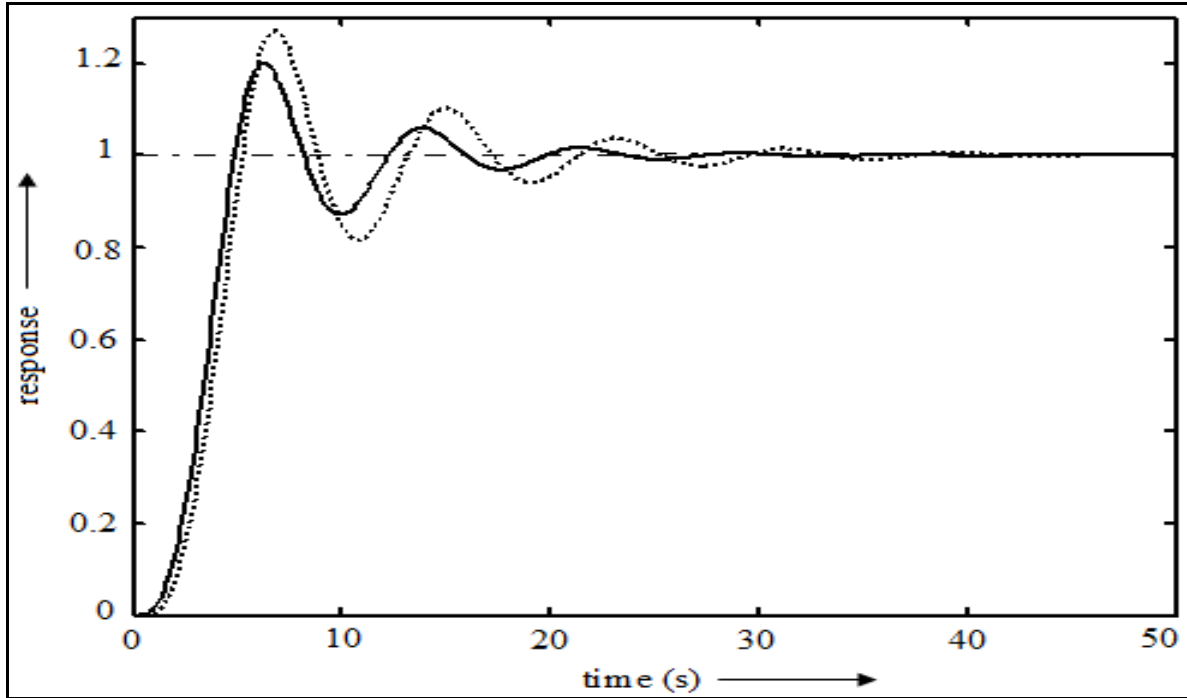


Figure 3.13: Responses of process (3.11) with STFPICa for $L=0.1$ (solid) and $L=0.4$ (dotted).

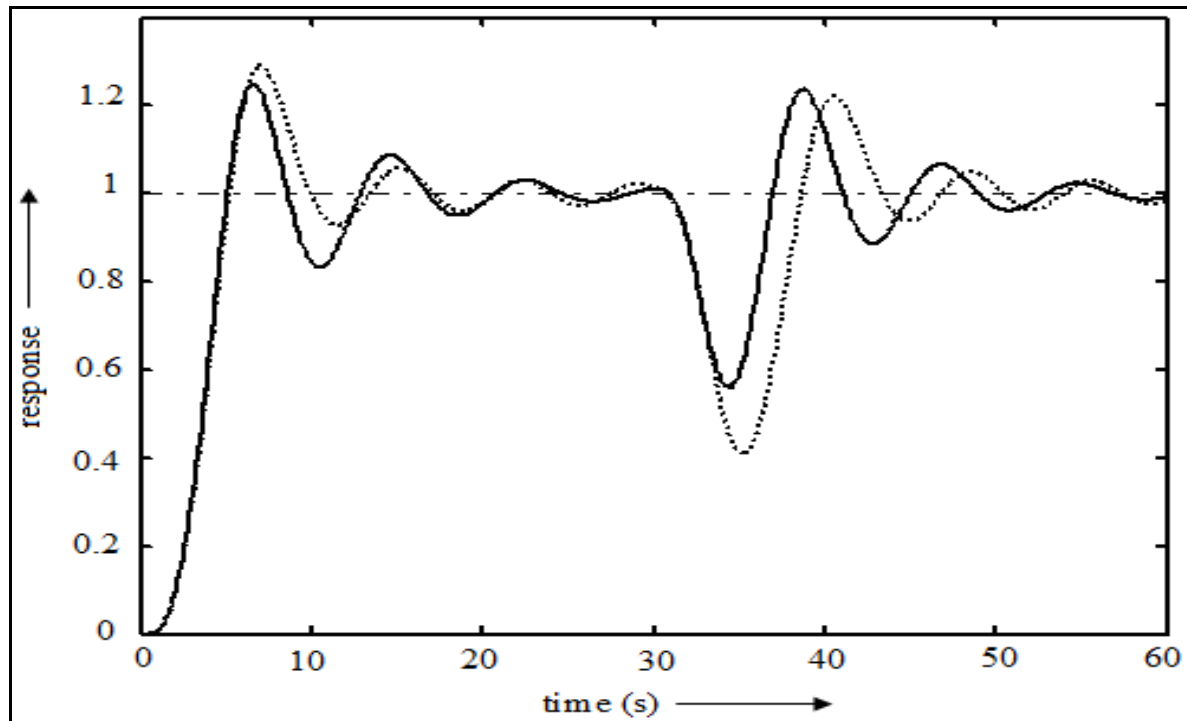


Figure 3.14: Responses of process (3.11) with $L=0.3$ with load disturbance at $t=30$ s (FPIC-dotted and STFPICa-solid).

Table 3.5: Performance analysis for the nonlinear 2nd order process (3.11)

L	FLC	G _u	%OS	t _r (s)	t _s (s)	IAE	ITAE	ISE
0.0	FPIC	0.025	24.39	5.0	15.6	4.07	17.50	2.41
	STFPIC	0.075	19.25	4.7	15.7	3.78	15.70	2.23
	STFPIC _α	0.084	20.25	4.7	18.3	3.90	15.50	2.28
0.2	FPIC	0.025	26.67	5.0	19.0	4.42	26.57	2.51
	STFPIC	0.075	22.33	4.7	19.1	4.07	20.70	2.32
	STFPIC _α	0.081	22.61	4.8	19.2	4.20	19.00	2.40
0.4	FPIC	0.025	31.77	5.1	37.5	5.63	74.28	2.73
	STFPIC	0.075	29.13	4.8	24.2	4.94	35.88	2.59
	STFPIC _α	0.080	27.28	5.1	24.3	4.93	28.50	2.60

Table 3.6: Performance analysis for the 2nd order processes with load disturbance

Process	L	Load at t(s)	FLC	G _u	%OS	t _r (s)	t _s (s)	IAE	ITAE	ISE
Linear <i>equⁿ</i> . (3.9)	0.2	40	FPIC	0.02	29.68	5.9	28.8	11.52	325.21	5.84
			STFPIC _α	0.067	24.21	5.6	27.8	9.44	243.48	4.38
Marginally Stable <i>equⁿ</i> .(3.10)	0.2	55	FPIC	0.016	48.80	6.1	48.0	16.42	590.72	7.47
			STFPIC _α	0.056	31.41	5.8	41.0	11.08	354.55	5.09
Nonlinear <i>equⁿ</i> . (3.11)	0.3	30	FPIC	0.025	29.13	5.1	23.0	10.08	228.87	5.48
			STFPIC _α	0.078	24.98	5.0	23.2	8.74	172.18	4.64

The above simulation study justifies the use of α . The proposed self-tuning fuzzy controller with dynamic gain variation (STFPIC_α) and with limited number of fuzzy *if-then* rules shows better performance compared to fuzzy PI and self-tuning fuzzy PI controllers.

3.5 Real time experiments on DC motor

High performance DC motor drives are used extensively in industrial applications for its good starting and braking performance. The DC motor drive is a highly controllable electrical motor drive suitable for robotic manipulators, guided vehicles, steel mills, mining machines; mine hoist machines and electrical traction [5]. Usually, precise, fast, effective speed references tracking with minimum overshoot/undershoot and small steady state error are essential control objectives of such a drive system.

In the present speed control application as shown in Fig. 3.15 and 3.16, a small DC servo motor is used (Manufacturer: SHINKO ELECTRIC CO. LTD., Japan; Speed: 1500 *R.P.M.*). For speed measurement, a slotted opto-coupler is fitted with the wheel that contains an Infrared (IR) ray emitter at one end and a receiver with a photo sensor at the other end. If IR beam falls on the photo sensor, the output of the processing circuit becomes logic ‘0’ and it becomes logic ‘1’ when IR beam is blocked by some object. The slotted wheel rotates inside the slotted space of the opto-coupler and thus the IR beam cuts at regular interval of time as shown in Fig. 3.15. The processing circuit produces a series of pulse train, logic 0 and 1. The signal is then converted into voltage signal by F/V (*frequency to voltage*) converter that is found to be linear in nature as shown in Fig. 3.17. Voltage vs. rpm calibration curve is shown in Fig. 3.18.

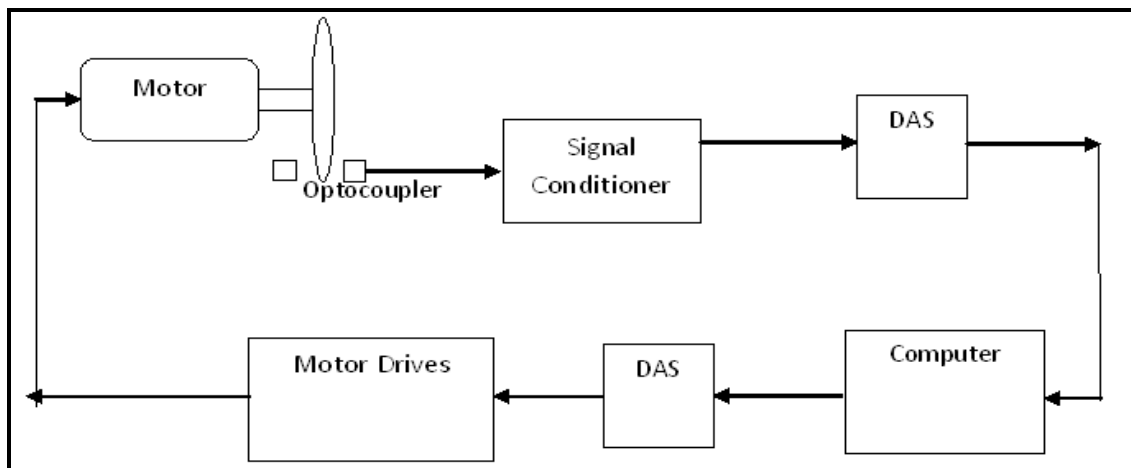


Figure 3.15: Block diagram for set-up of speed control of DC motor.



Figure 3.16: Experimental set-up for the speed control of DC motor.

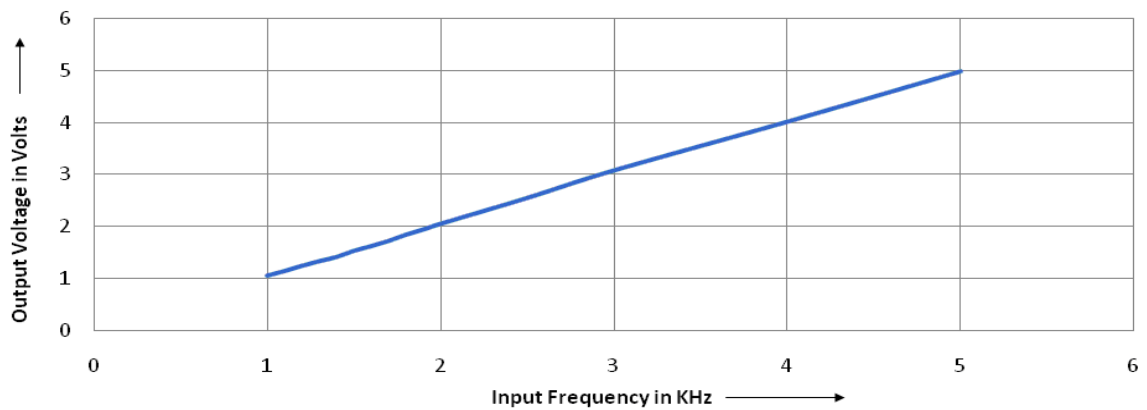


Figure 3.17: Frequency (proportional to motor speed) vs. voltage curve.

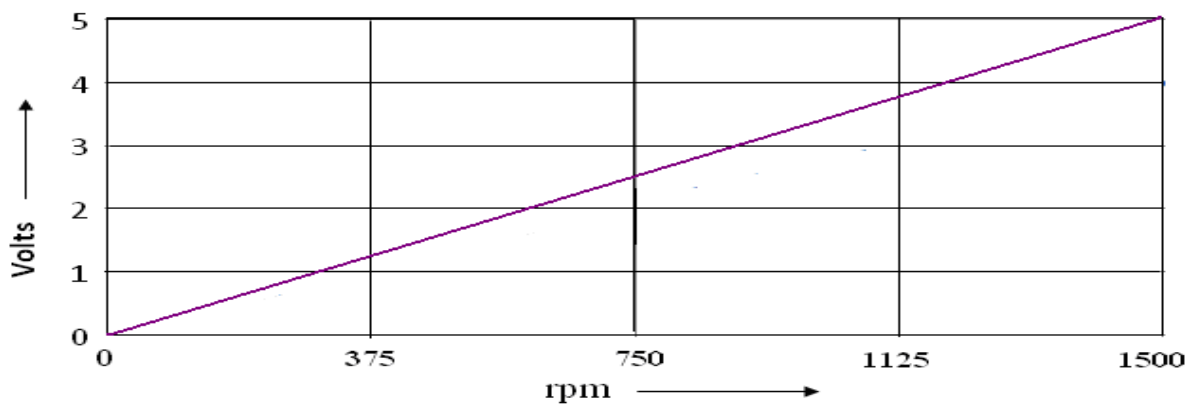


Figure 3.18: Voltage vs. rpm curve.

3.5.1 Relay feedback test

To generate the sustained oscillation, relay feedback test is performed on the motor as described in **section 3.2**. The ultimate gain (K_u) and ultimate period (P_u) are measured from the principal harmonic approximation shown in Fig. 3.19. The proposed STFPIC α is designed by determining the value of α from the relation $\alpha = (K_u.P_u)/2.5$. From the relay feedback experiment we get

$$K_u = 1.958; \text{ and } P_u = 5;$$

$$\text{Therefore, } \alpha = (K_u.P_u)/2.5 = 3.916;$$

To design the ZNPIC, its K_c and T_i are calculated from the relay feedback test, as follows:

$$K_c = K_u / 2.2 = 0.89;$$

$$T_i = P_u / 1.2 = 4.167;$$

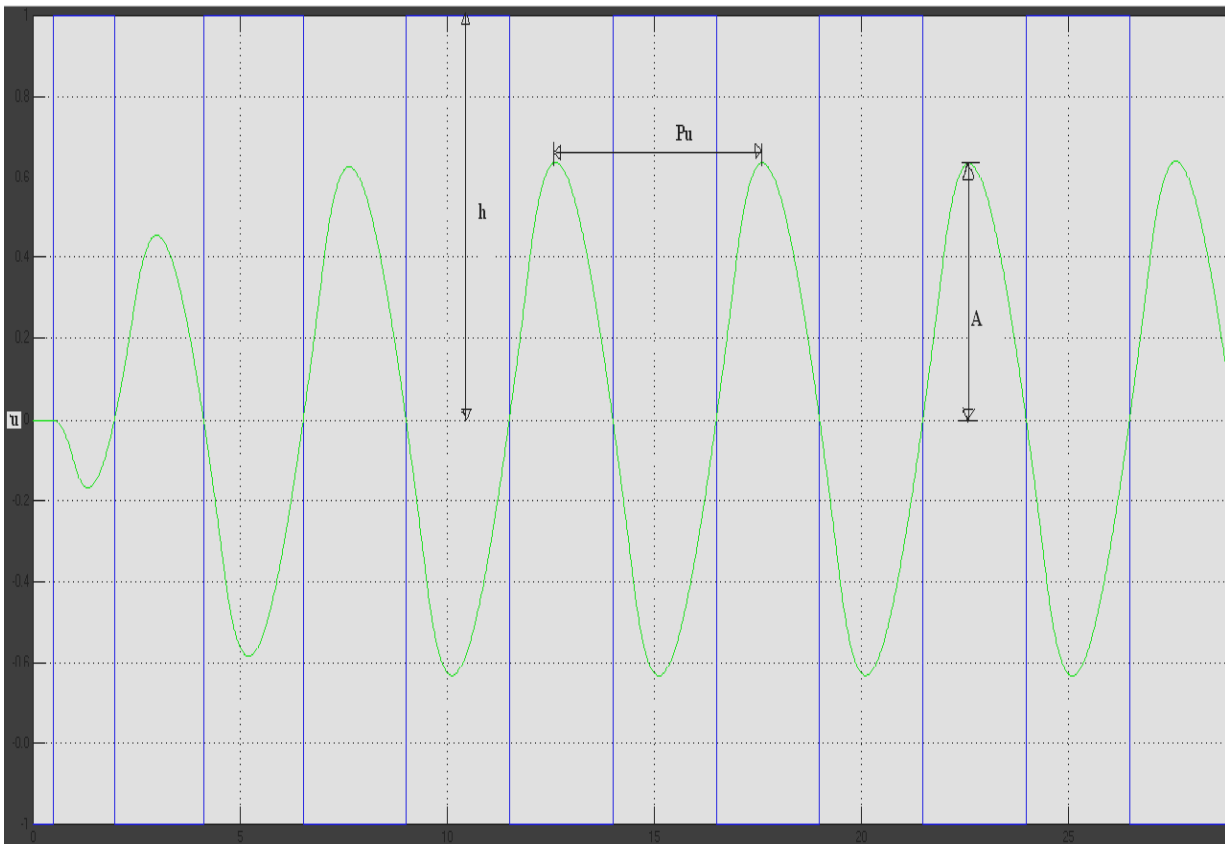


Figure 3.19: Relay feedback response of a DC motor in close loop.

3.5.2 Results and discussion

The speed of the DC motor in terms of voltage is observed by using proposed STFPICa. From Fig. 3.23 and Table 3.7, we find that STFPICa efficiently controls the motor speed under varying dead-time. Fig. 3.20, Fig. 3.21 and Fig. 3.22 represent speed vs. time response plots for applying ZNPIC, FPIC and STFPICa respectively. A comparative study has been done [156] as shown in Table 3.8 with FPIC (49 rules), STFPICa (98 rules) and with ZNPIC. From Table 3.8, we realize the overall improved performance of STFPICa compared to conventional fuzzy and non-fuzzy controllers. The controller provides a comparable rise time with respect to ZNPIC and FPIC. Also STFPICa gives very low peak time, peak overshoot and it settles within 34.75s. Due to its self-tuning mechanism, STFPICa always tracks the reference value of motor speed, thus gives zero steady state error and negligible IAE and ITAE. However, comparing Table 3.7 and Table 3.8, it is observed that STFPICa (50 rules) performs better than even STFPICa despite using lesser number of *if-then* rules.

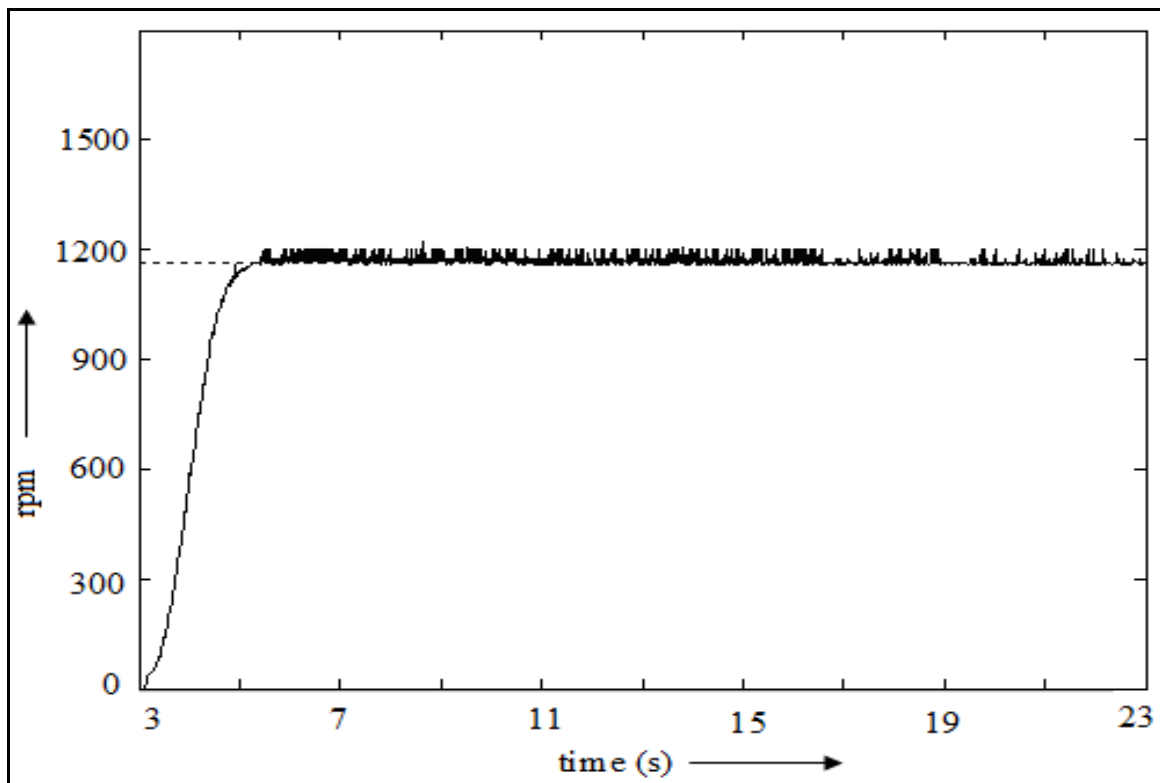


Figure 3.20: Step response of the DC motor for ZNPIC.

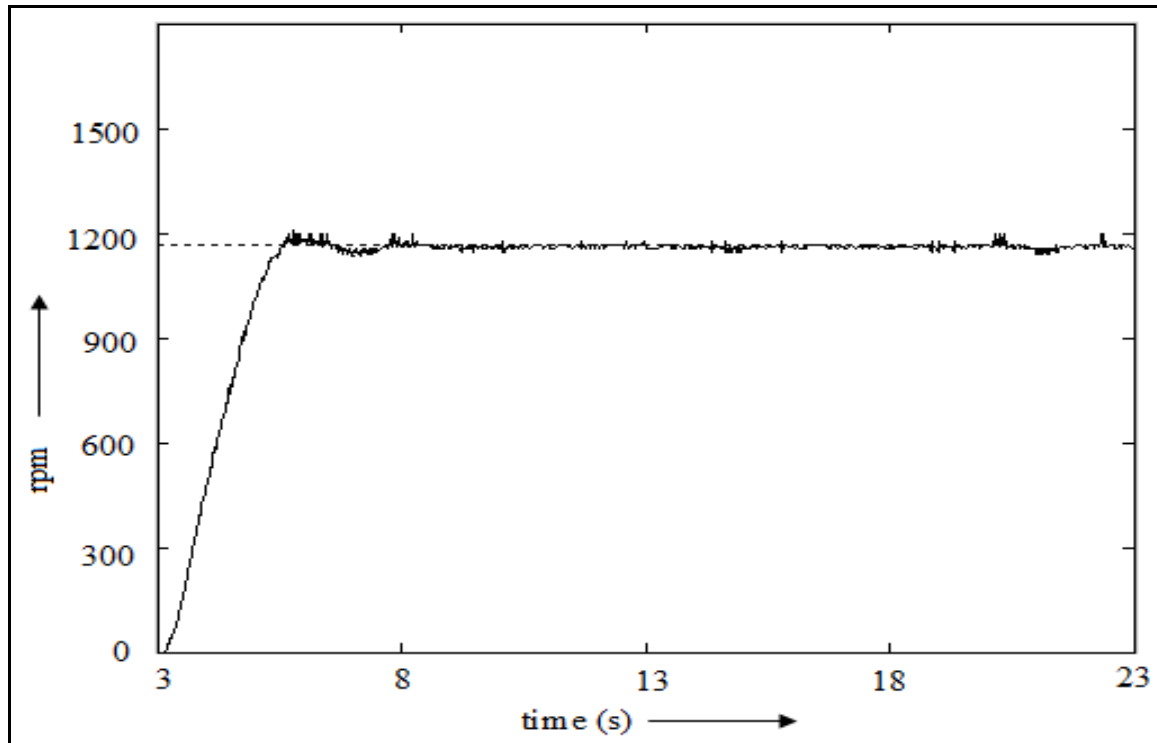


Figure 3.21: Step response of the DC motor for FPIC.

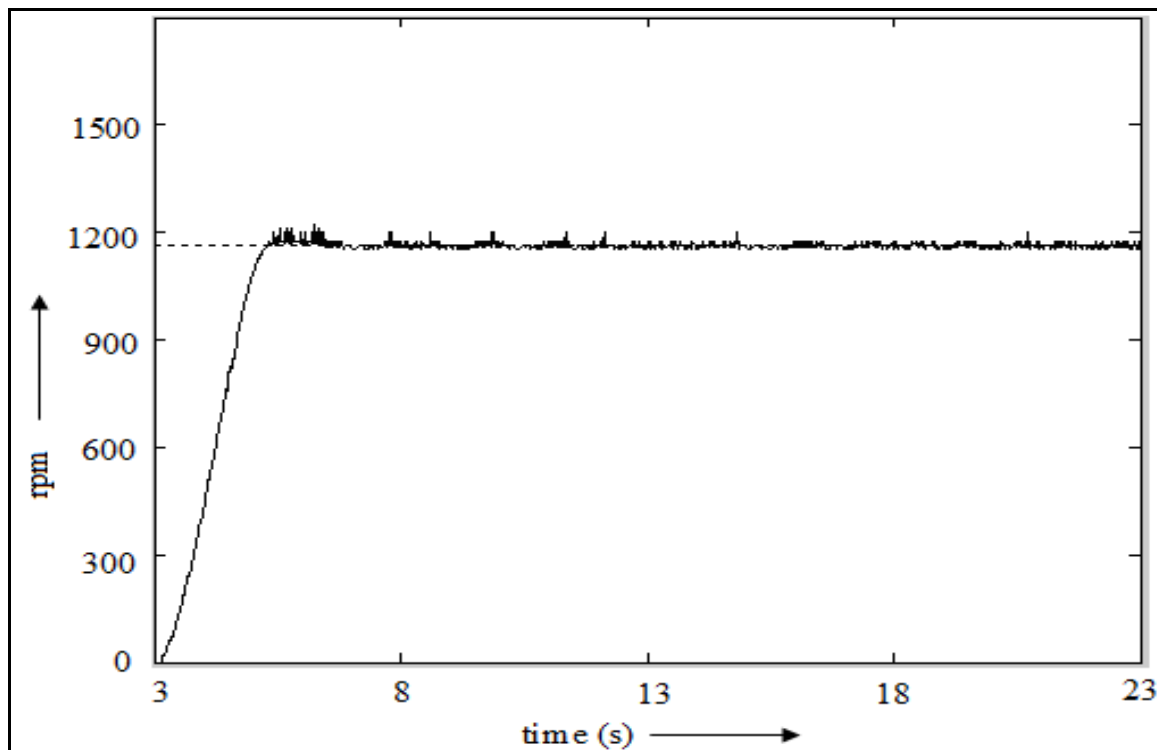


Figure 3.22: Step response of the DC motor for STFPIC.

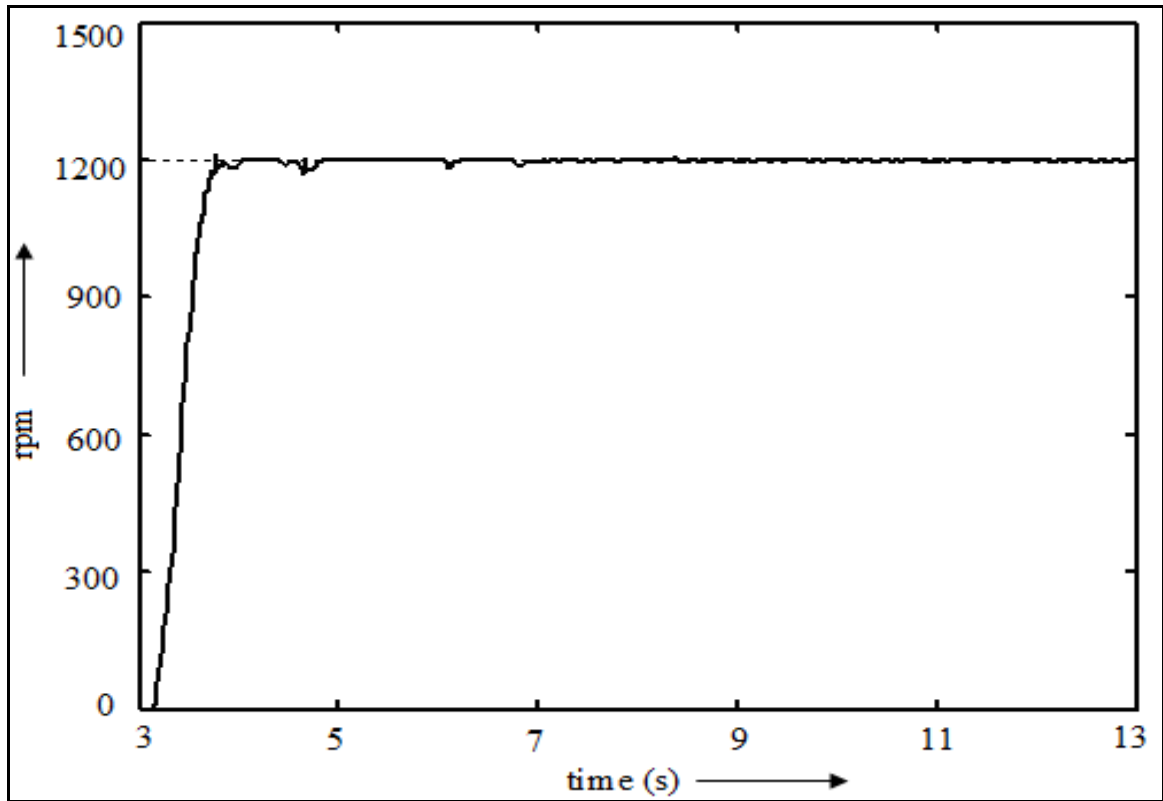


Figure 3.23: Step response of the DC motor for STFPICa.

Table 3.7: Performance analysis with STFPICa for the practical DC motor

Controller Type	L	$t_r(s)$	$t_s(s)$	%OS	IAE	ITAE
STFPICa	3.0	3.76	4.64	0.96	114	534

Table 3.8: Performance comparison for different controllers

Controller Type	$t_r(s)$	$t_p(s)$	$t_s(s)$	%OS
ZNPIC	5.26	8.66	61.28	5.1
FPIC	5.29	6.25	58.83	5.1
STFPIC	5.62	5.82	34.75	3.6

3.6 Conclusion

A simple self-tuning fuzzy PI controller with dynamic gain (STFPIC α) has been proposed in this chapter. The output SF of the STFPIC is updated on-line based on the process trend as well as the dynamics of the system. STFPIC α has shown significantly improved performance compared to its conventional fuzzy and non-fuzzy controllers for high-order and nonlinear systems with varying dead-time. Even STFPIC α with almost 50% reduced rules exhibited almost similar performances to that of STFPIC with fixed multiplicative factor. The performance comparison showed that the proposed STFPIC α also offers a better load regulation. One of the highlights of the proposed scheme is relay feedback tuning, which is performed on-line on the motor. The tuning method used here is rather simple to understand by the control engineer. The results revealed that the STFPIC α has improved the close loop performance by minimizing the steady state error and other performance criterions.

CHAPTER 4

Development of Adaptive fuzzy PD controller- AFPDC

4.1 Introduction

Over the past two decades, the application of knowledge-based systems in process control has been growing, especially in the field of fuzzy control, in which linguistic descriptions of human expertise are represented as fuzzy rules or relations [147]. This knowledge-base is used by an inference mechanism, in conjunction with some knowledge of the states of the process in order to determine control actions. Among the various types of hybrid controllers [157], PD-type fuzzy logic controller (FPDC) is very common and practical because its derivative control action, when added to a proportional controller, provides a means of obtaining a controller with high sensitivity. An advantage of using derivative control action is that it responds to the rate of change of the actuating error and can produce a significant correction before the magnitude of the actuating error becomes too large. Derivative control thus anticipates the actuating error, initiates an early corrective action, and tends to increase the stability of the system. PD-type FLCs are suitable for systems [127], like non-minimum phase systems, systems with integrating elements and few nonlinear systems.

PID controllers may be tuned in a variety of ways, including hand-tuning, Ziegler Nichols tuning, loop shaping, analytical methods, by optimization, pole placement, or auto tuning. When the control problem is to regulate the process output around a set-point, it is natural to consider error as an input, even to a fuzzy controller, and it follows that the integral of the error and the derivative of the error may be useful inputs as well. In a fuzzified PID controller, however, it is

difficult to tell the effect of each gain factor on the rise time, overshoot, and settling time, since it is most often nonlinear and has more tuning parameters than a PID controller. The objective in this section is to find a systematic tuning procedure for fuzzy controller. A systematic tuning procedure would make it easier to install fuzzy controllers, and it might pave the way for auto-tuning of fuzzy controllers. As discussed in **chapter-2**, Mudi *et al* [22, 23] proposed a robust self-tuning scheme for PD-type FLC. In their scheme of self-tuning, an on-line fuzzy gain modifier is determined by 49 fuzzy *if-then* rules based on operator's knowledge. However, in ***our proposed controller, a simple non-fuzzy adaptive scheme is used to update the controller gain continuously with the help of process error and change of error.***

Motivated by the encouraging results of [142], instead of a fuzzy rule based tuning scheme [22, 23], here a simple non-fuzzy adaptive technique is proposed. In the proposed adaptive fuzzy PD controller (AFPDC), output SF is continuously updated by a gain updating factor β , which is related to the normalized error (e_N) and normalized change of error (Δe_N) of the process under control. Here, the output of the controller is modified in accordance with the present situation of the process under control, thus, it is expected that the proposed AFPDC will improve the close-loop performance. Another important point of our scheme is that it uses significantly lesser number of rules compared to other FLCs [22, 155] reported in the literature. The proposed scheme is demonstrated on different second order models with variable dead-time and also its effectiveness is tested on a practical overhead crane set-up [158, 159].

The overhead cranes are commonly employed in the transport industry for the loading and unloading of freight, in construction industry for the movement of raw materials and in the manufacturing industry for the assembling of heavy equipment [160]. In such applications, external and internal excitations at the suspension point can produce in-plane and out-of plane pendulations as well as vertical oscillations of the payload. This problem of pendulations in the crane is aggravated due to its lightly damped nature that means any transient motion takes a long time to dampen out. Suppression of payload oscillations is especially important for offshore cranes due to wave-induced motions of the crane-platform. Onshore cranes may also experience base excitations, leading to a complex dynamic response of the free swinging load, due to variety of reasons, such as wave breaking on the shore and the interaction between the payload motion

and the platform support system. However, this problem is most pronounced in offshore cranes [161].

Anti-sway and position control have become the requirements as a core technology for automated crane system. The purpose of crane control is to reduce the pendulum type motion of the loads while moving the trolley to the desired position as fast as possible. Crane operators, often aided with automatic anti-sway systems, are always involved, and the resulting performance, in terms of swiftness and safety, heavily depends on their experience and capability. Thus, the need for faster cargo handling requires the precise control of crane motion so that its dynamic performance is improved [162, 163].

Various attempts have been made to solve the problem of swing of load [164-169]. Most of them focus the control on suppression of load swing without considering the position error in crane motion [170]. *Liu et al.* [171] investigated an adaptive sliding mode fuzzy control approach for a linearized two dimensional overhead crane system. But the methods based on the linearized crane dynamics may lose the sufficient accuracy of information about position and swing angle, so some uncertainty may arise that can reduce the performance of these crane control systems. Besides, several authors have considered optimization techniques to control the cranes. They have used minimal time control technique to minimize the load swing [172, 173]. Since the swing of load depends on the motion and acceleration of the trolley, minimizing the cycle time and minimizing the load swing are partially conflicting requirements.

In this study, we attempt to provide a practical solution for the anti-swing and precise position control of an overhead crane. The position of trolley, swing angle of load and their differentiations are applied to derive the proper control input of the trolley crane. Two PD-type fuzzy logic controllers are used to deal separately with the feedback signals, swing angle and trolley position and their differentiations [174]. The fuzzy rules can be designed according to the experience of crane workers. The main advantage of this separated approach is to greatly reduce the computational complexity of the crane control system. The total number of fuzzy rules for the complete control system is therefore less than the number of rules used by conventional fuzzy system. Thus, the proposed algorithm is very easy to implement.

4.2 Design of the proposed controller - AFPDC

A skilled operator always tries to eliminate the error within the shortest possible time by changing controller output. Considering the role of a human operator, the output scaling factor should be considered a very important parameter of the FLC since its function is similar to that of the controller gain. So the output SF should be determined very carefully for the successful implementation of a FLC. Thus, depending on the process trend, an expert operator or a control engineer tries to modify the control action of the controller. Control action should be modified by changing the output SF to improve the system performance maintaining the stability of the close loop system. Following such an operator’s policy, here, we suggest a simple adaptive scheme of Fuzzy PD Controller (FPDC) [142], where an on-line gain modifier β is determined from the relation $\beta = K(I+\alpha)$. The parameter α is obtained from the multiplication of e_N and Δe_N . The block diagram of the proposed AFPDC is shown in Fig. 4.2. The block diagram of Fig. 4.2, without the β function can be considered as a figure of PD type FLC as shown in Fig. 4.1. Other design considerations are discussed in the next subsections.

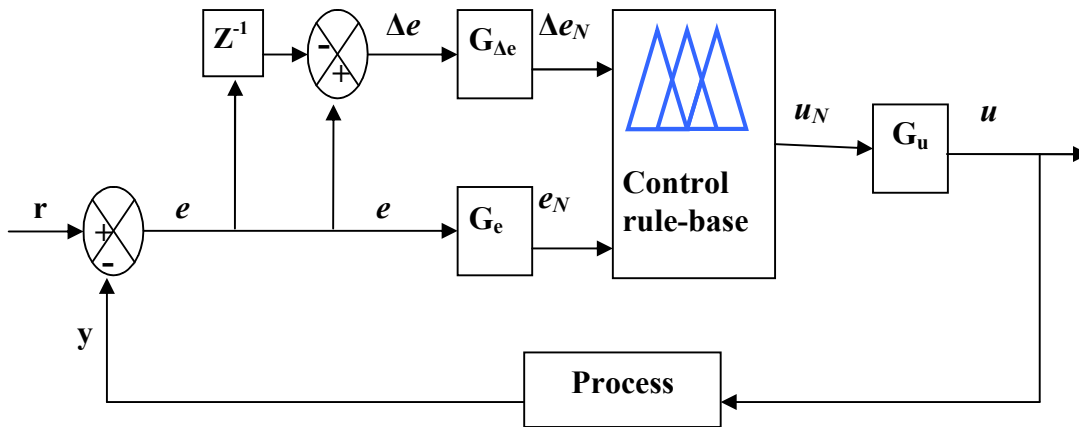


Figure 4.1: Block Diagram of FPDC.

4.2.1 Fuzzy membership functions

Membership functions (MFs) for controller inputs error (e), change of error (Δe) and controller output (u) are defined on a common normalized domain $[-1, 1]$, as shown in Fig. 4.3. Symmetric triangles with equal base-width and 50% overlap with neighboring MFs are used here due to its

natural and unbiased nature. The term sets of e , Δe and u for PD type FLC contain the same linguistic expressions for the magnitude part of the linguistic values, *i.e.*, $LE = L\Delta E = LU$ {NB- Negative Big, NM- Negative Medium, ZE- Zero, PM- Positive Medium, PB- Positive Big}.

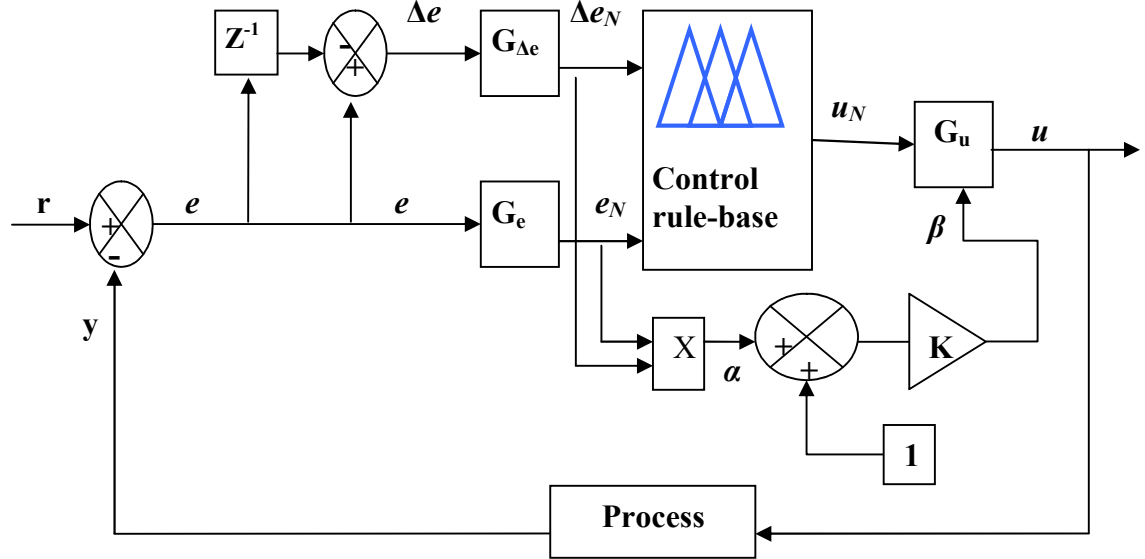


Figure 4.2: Block Diagram of AFPDC.

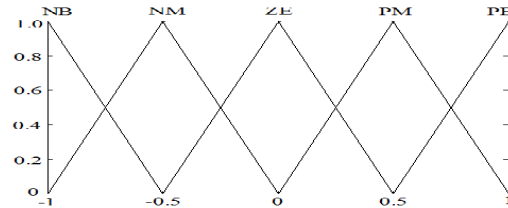


Figure 4.3: MFs of e , Δe and u .

4.2.2 Scaling factors

The normalized inputs (e_N and Δe_N) and normalized output (u_N) of the AFPDC as shown in Fig. 4.2 are defined on the normalized domain $[-1, 1]$. The input variables e and Δe are mapped onto normalized interval $[-1, 1]$ by the input SFs G_e and $G_{\Delta e}$, respectively. Normalized output (u_N) of FPDC is mapped onto actual output (u) by the output SF G_u . However, the actual output of the proposed AFPDC is obtained by applying effective SF ($\beta.G_u$) as shown in Fig. 4.2. Thus proper selections of these scaling factors are very important and are made based on the knowledge about the process to be controlled. The relationship between the SFs (G_e , $G_{\Delta e}$, G_u) and the input and output variables of the AFPDC are as follows:

$$e_N = G_e \cdot e$$

$$\Delta e_N = G_{\Delta e} \cdot \Delta e \text{ and}$$

$$u = (\beta \cdot G_u) \cdot u_N; \text{ where, } \beta = K(1+\alpha).$$

Here, β is the on-line calculated parameter for the output scaling factor G_u . Observe that the PD-type FLCs use output scaling factor G_u only, whereas the output SF of AFPDC is obtained by multiplying G_u with β .

4.2.3 Gain updating factor

The output scaling factor (G_u) is constant for a particular application of FLC, but it does not remain constant for the proposed adaptive fuzzy controller while in operation. The output SF of the AFPDC is modified in each sampling instant by β , which depends on the instantaneous process condition. The gain updating factor (β) as shown in Fig. 4.2 is calculated using the following relation:

$$\beta = K(1+\alpha) \tag{4.1}$$

$$\alpha = (e_N \cdot \Delta e_N) \tag{4.2}$$

In the above relation, α is derived from the product of e_N and Δe_N . In *equation (4.1)*, ‘ K ’ is a positive constant, used to provide the appropriate range of variation of β . *Equation (4.2)* illustrates the characteristic of α as a function of e_N and Δe_N . Thus functional relationship of β can be viewed as:

$$\beta(k) = f(e_N(k), \Delta e_N(k)) \tag{4.3}$$

Where f is a nonlinear function of e_N and Δe_N . The nonlinear surface of α (Fig. 4.4) indicates that, *if e is +ve and Δe is -ve or vice versa (i.e. α is -ve), then the system approaches towards set-point and hence the controller gain becomes very small which will help to avoid large overshoot. Similarly, when both the inputs of the controller are of the same sign (-ve, -ve or +ve, +ve), (i.e., α is +ve), that indicates the system is moving away from the set-point) then the controller gain becomes large, which will help to achieve a faster recovery of the system. These useful observations are shown in the table format below.*

e_N	Δe_N	α	β (for $K=1$)	Observations
-	+	-	<1	System approaches towards set-point and hence the controller gain becomes very small, which will help to avoid large overshoot.
+	-	-	<1	
-	-	+	>1	System is moving away from the set-point and hence the controller gain becomes large, which will help to achieve a faster recovery of the system.
+	+	+	>1	

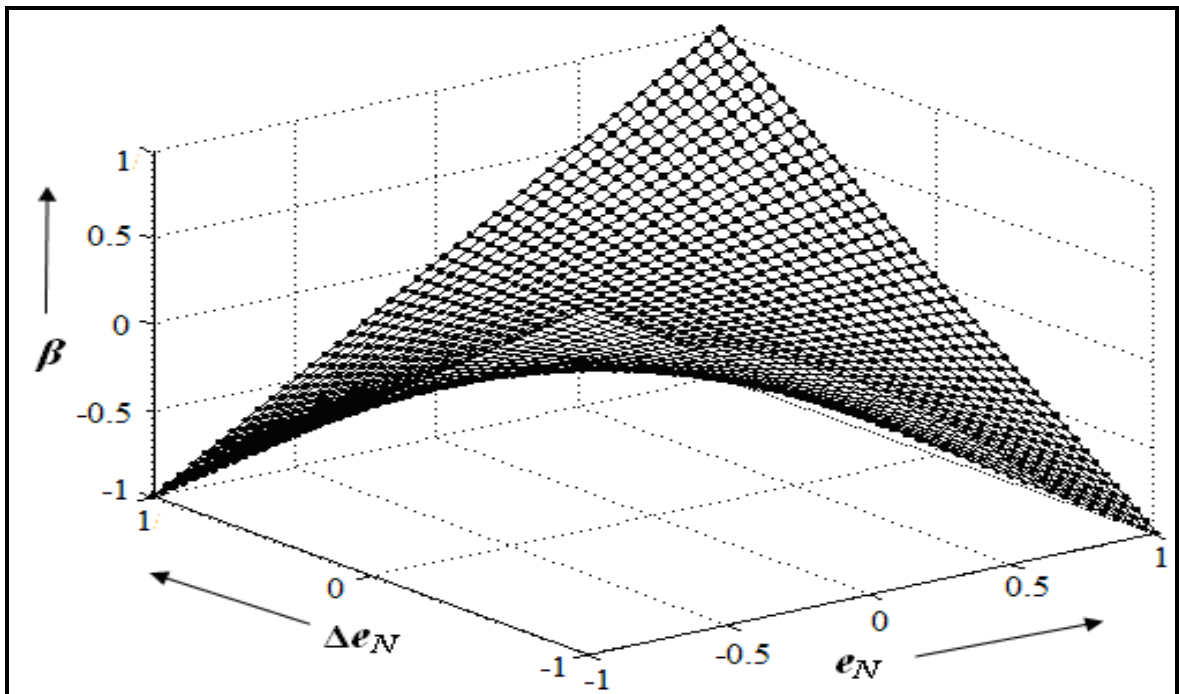


Figure 4.4: Variation of β with e_N and Δe_N .

4.2.4 The rule-bases

Table 4.1: Fuzzy control rules for computation of u

$\Delta e/e$	NB	NM	ZE	PM	PB
NB	NB	NB	NB	NM	ZE
NM	NB	NB	NM	ZE	PM
ZE	NB	NM	ZE	PM	PB
PM	NM	ZE	PM	PB	PB
PB	ZE	PM	PB	PB	PB

The fuzzy PD controller uses rules of the form:

If e is E and Δe is ΔE then u is U .

The rule-base shown in Table 4.1 is designed with a two-dimensional phase plane in mind where the FLC drives the system in sliding mode. The designed rule-base is flexible in nature and it can be modified according to the process requirement. The control surface, *i.e.*, controller output (u) vs. inputs e and Δe for the fuzzy PD controller (FPDC) is shown in Fig. 4.5. The control surface of FPDC is nonlinear and bumpy in nature. In this chapter to solve this problem, instead of using additional fuzzy rules [22] we propose an alternative method for gain adjustment. Control surface of AFPDC is shown in Fig. 4.6 with same number of fuzzy rules. After a careful inspection of the two surfaces it can be realized that the control surface of the proposed AFPDC is more nonlinear but smooth in nature than that of FPDC. This smoothness is very much essential for practical implementation.

Performance of the proposed AFPDC is compared with conventional PID and FPDC through simulation study on nonlinear and unstable systems. To ensure the robustness of the scheme same value of K ($K=2$) in the relation of β (equation 4.1), is used for all the examples.

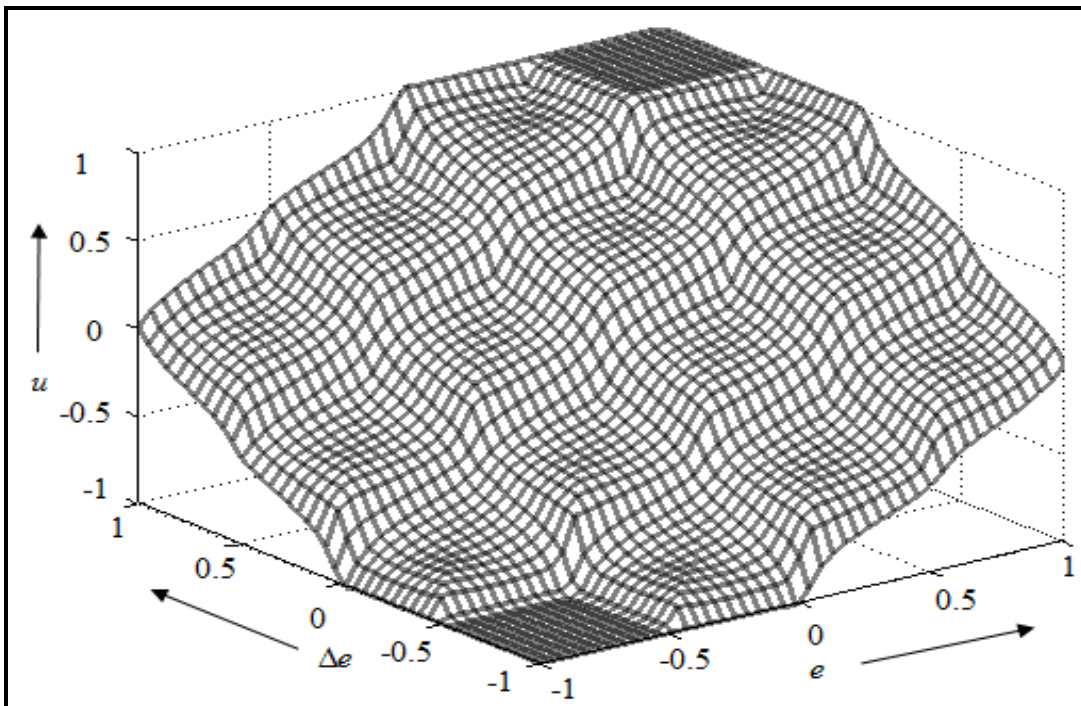


Figure 4.5: Control surface of FPDC.

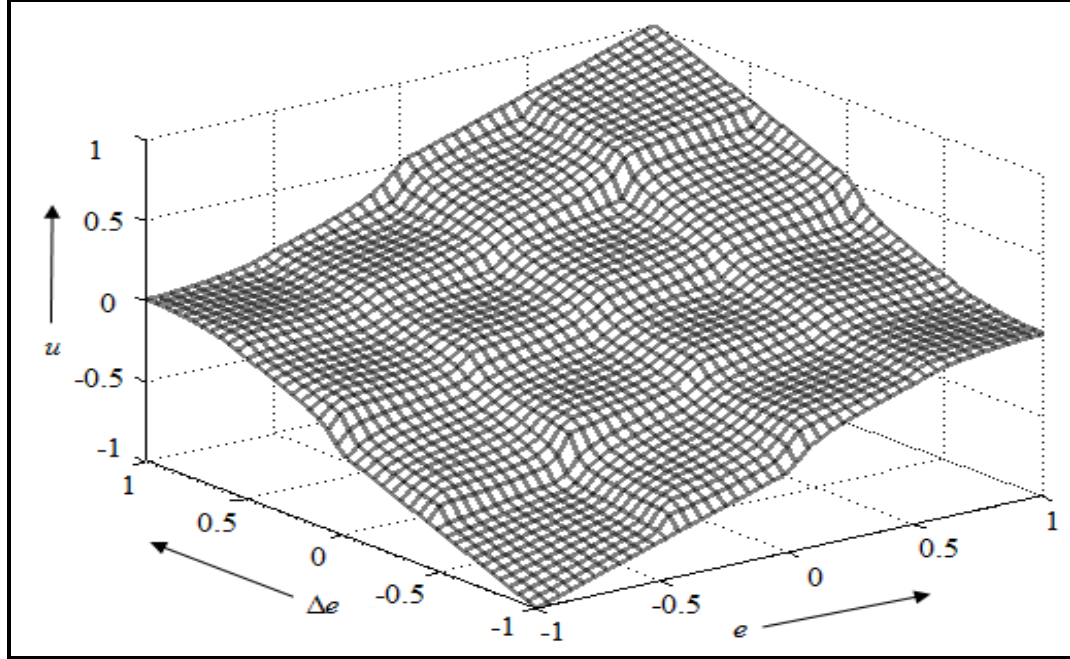


Figure 4.6: Control surface of AFPDC.

4.3 Simulation experiments

The performance of AFPDC is evaluated here with respect to conventional PID and FPDC through simulation study on wide variety of second order systems, *i.e.*, non-minimum phase, nonlinear, integrating and unstable systems with variable dead-time. To establish the robustness of the proposed scheme same MFs (Fig. 4.3) and same rule-base (Table 4.1) are used for all the processes. We have used Mamdani type interfacing and height method of defuzzification [42] in all the cases. In the proposed scheme the value of β is dependent on the process dynamics and it is not a fixed value. Therefore, the effective control action generated by AFPDC is varied in accordance with the β value as shown by the control surface of β in Fig. 4.4.

Non-minimum Phase System

We have considered a 2nd order unstable system described by:

$$\frac{d^2y}{dt^2} - y = u(t-L) \quad (4.4).$$

This simplified linear model of inverted pendulum is unstable in nature due to the presence of a non-minimum phase pole at right hand side of the s -plane. It is very difficult to control a non-

minimum phase system using conventional PID controller. To eliminate the effect of this right hand side pole and to make the system stable by pole-zero cancellation, it is better to choose a PD controller, which introduces a zero in the left hand side of s -plane.

Fig. 4.7 shows the response of *equation* (4.4) for an impulse input with PID controller. Corresponding results for PID controller with different dead-time is tabulated in Table 4.2. Fig. 4.8 and Fig. 4.9 depict the impulse responses of the system (4.4) for FPDC and AFPDC with scaling factors ($G_e=0.9$ and $G_{\Delta e}=2.7$). The effect of dead-times (0 and 0.1) and load disturbances (at 10s and 15s) in FPDC and AFPDC are effectively demonstrated in Fig. 4.8 and Fig. 4.9. Table 4.3 shows the performance analysis of the non-minimum phase system considering different performance criterion for different dead-times. Responses of the system indicate a remarkable improvement in the performance using AFPDC over PID and FPDC. The designed controller is also working satisfactorily in load disturbances as shown in Fig. 4.8 and Fig. 4.9. However, such types of systems are found to be uncontrollable even with self-tuning schemes for higher values of dead-time.

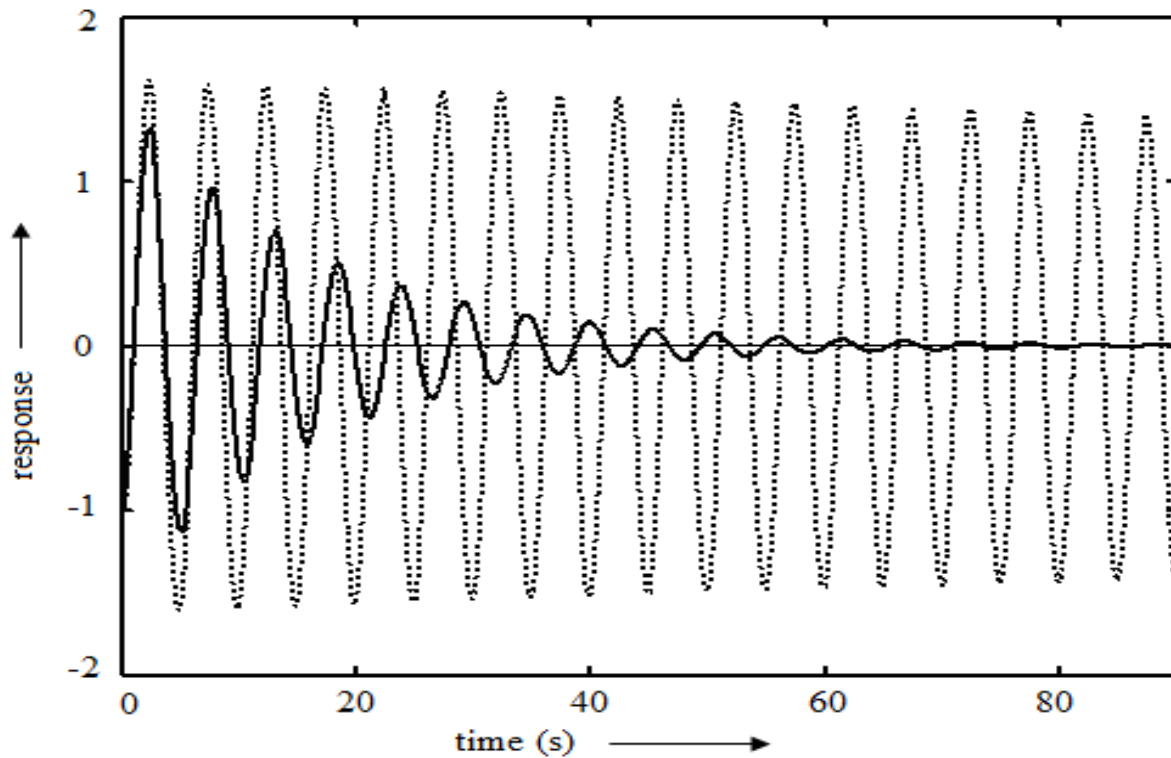


Figure 4.7: Responses of (4.4) with PID for $L= 0.0$ (solid) and $L=0.1$ (dotted).

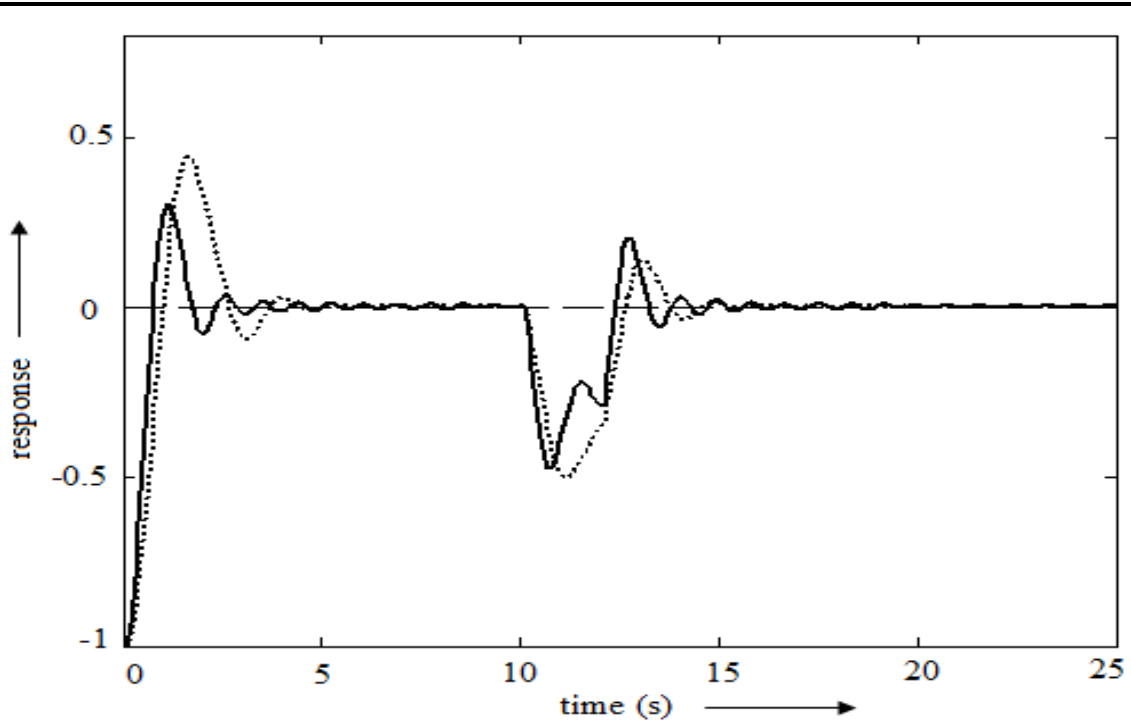


Figure 4.8: Responses of (4.4) with FPDC (dotted) and AFPDC (solid) for $L=0$ with load disturbance at 10s.

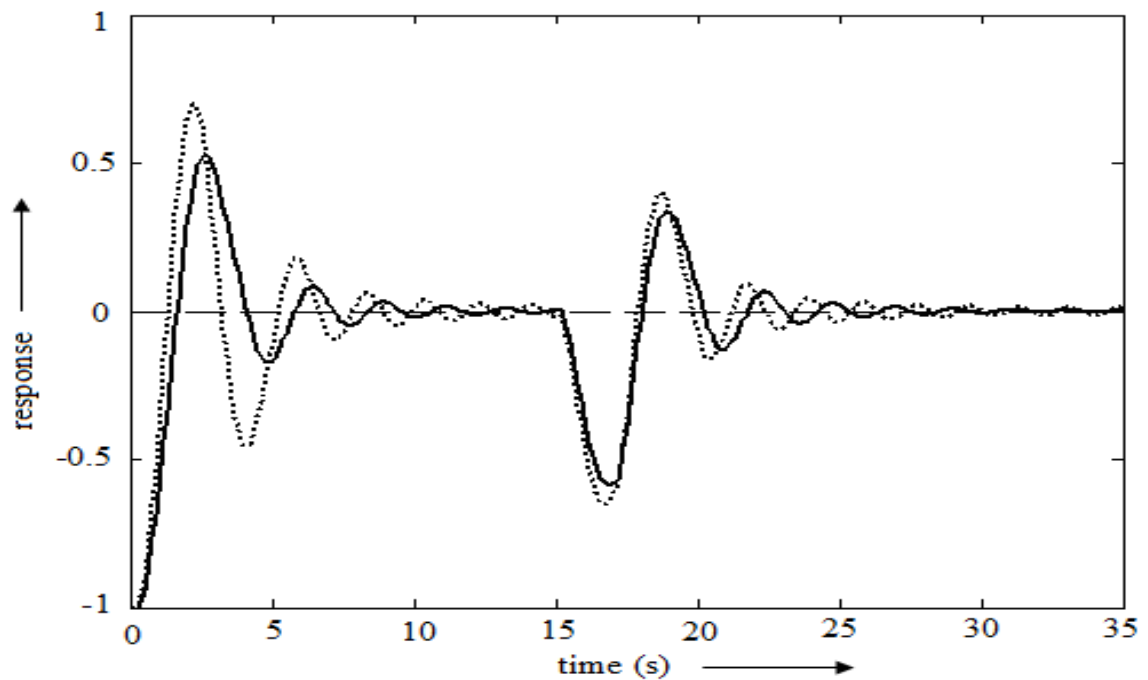


Figure 4.9: Responses of (4.4) with FPDC (dotted) and AFPDC (solid) for $L=0.1$ with load disturbance at 15s.

Table 4.2: Performance analysis of the non-minimum phase system (4.4) with PID

L	%OS	t _r (s)	t _s (s)	IAE	ITAE	ISE
0.0	133.26	1.1	67.1	14.44	240.49	7.51
0.1	161.56	1.1	-	77.59	3423.19	82.18

Table 4.3: Performance analysis of the non-minimum phase system (4.4) with FLCs

L	FLC	G _u	%OS	t _r (s)	t _s (s)	IAE	ITAE	ISE
0.0	FPDC	6	44.14	1.1	3.7	0.36	1.06	0.65
	AFPDC	6β	26.58	0.8	3.0	0.36	0.45	0.42
0.1	FPDC	4.2	70.16	1.4	25.3	0.61	1.70	1.59
	AFPDC	4.2β	52.80	1.7	8.3	0.57	1.05	1.39

Nonlinear Process

We have tested the performance of AFPDC for nonlinear process:

$$\frac{d^2y}{dt^2} + 0.3y \frac{dy}{dt} = u(t-L) \quad (4.5).$$

Note that widely used relay-feedback (Fig. 4.10) tuned PID controller exhibits very poor performance and produces very large overshoots for this nonlinear process as shown in Fig. 4.11 and Table 4.4. A comparative study is made between PID and AFPDC in Fig. 4.12 with load disturbance at $t=20s$. The performances of AFPDC and FPDC (with $G_e=0.9$ and $G_{\Delta e}=11$) are observed for nonlinear process (4.5) for various dead-time by applying a unit step input. Load disturbances are applied at $t=20s$ as shown in Fig. 4.13 and Fig. 4.14. Table 4.5 provides the quantitative performance analysis of AFPDC, FPDC and PID for different values of dead-time. From the analysis of nonlinear process (4.5), we find that the AFPDC with only 25 fuzzy rules always outperforms FPDC and PID controllers. This indicates that proposed on-line gain adjustment scheme through gain updating factor β is working satisfactorily for the nonlinear process also.

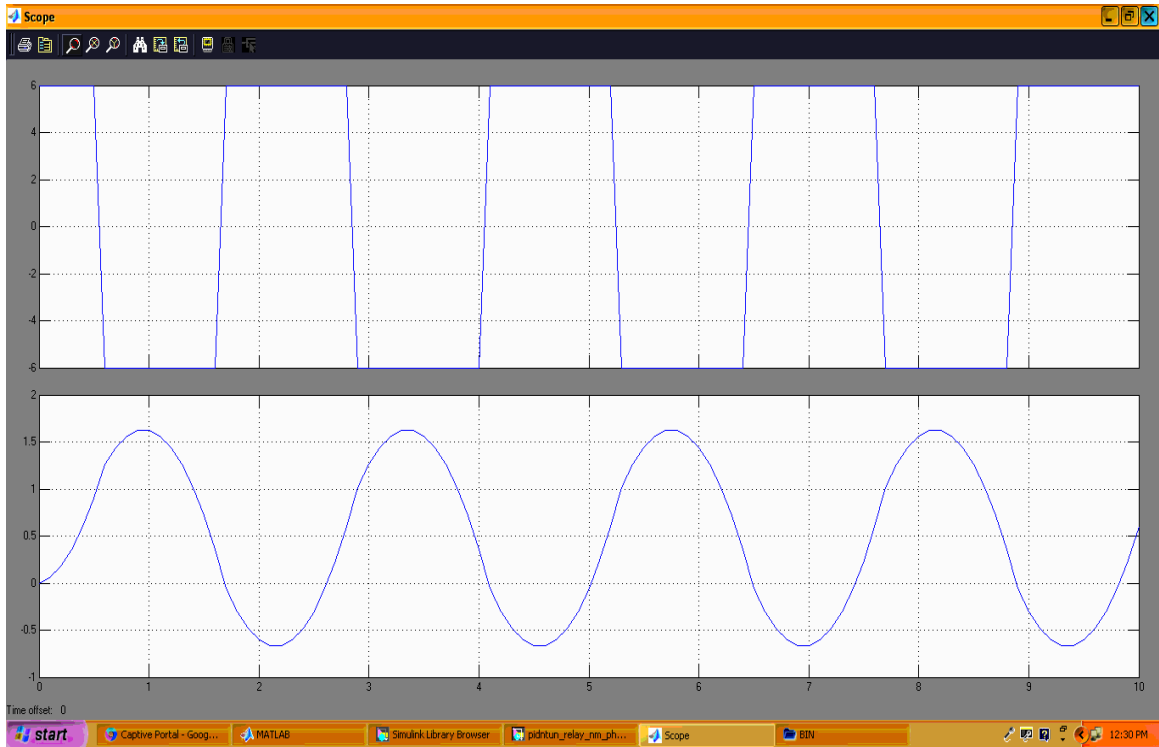


Figure 4.10: Relay feedback tuning graph of $d^2y/dt^2 + 0.3y.dy/dt = u(t-L)$.

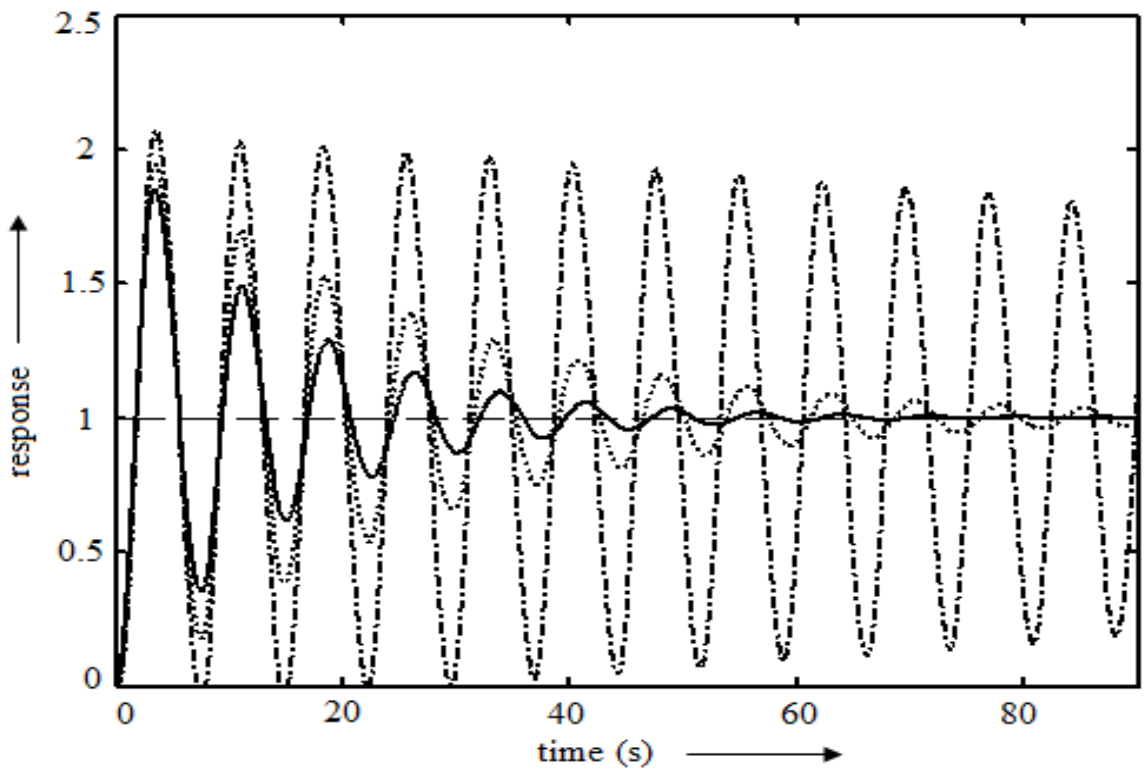


Figure 4.11: Responses of (4.5) with PID for $L=0$ (solid), $L=0.1$ (dotted) and $L=0.2$ (dash-dot).

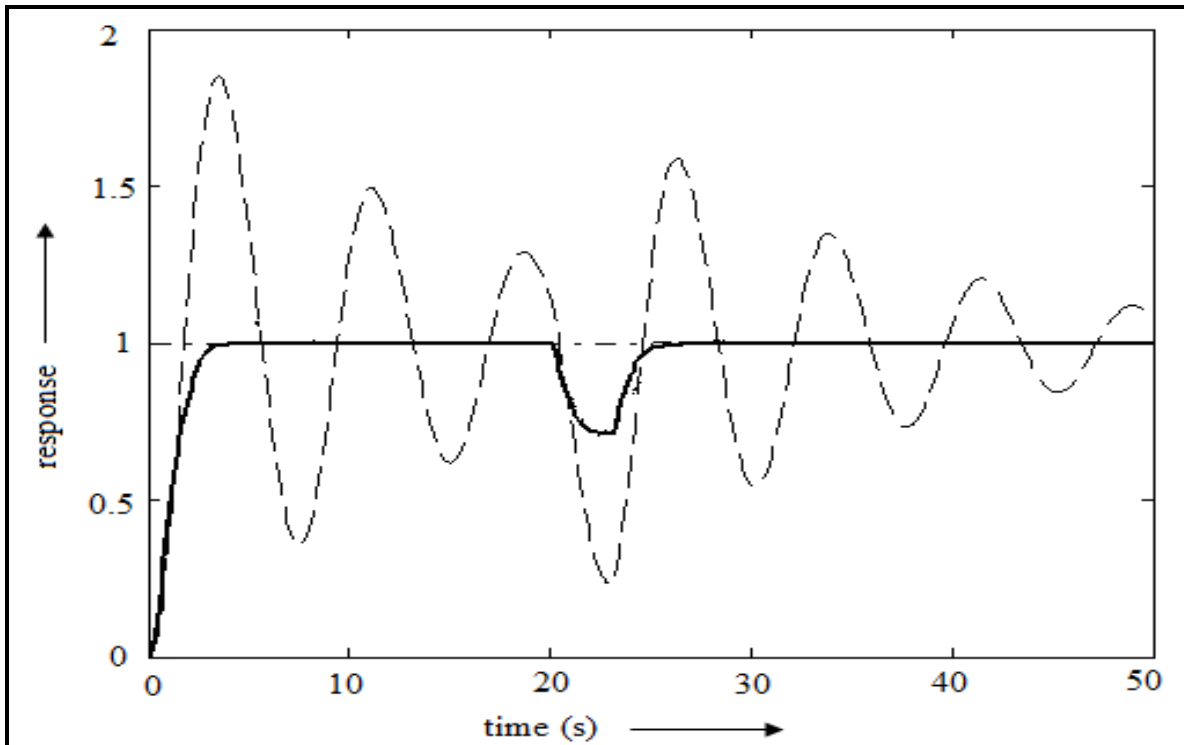


Figure 4.12: Responses of (4.5) with PID (dashed) and AFPDC (solid) for $L=0$ with load disturbance at 20s.

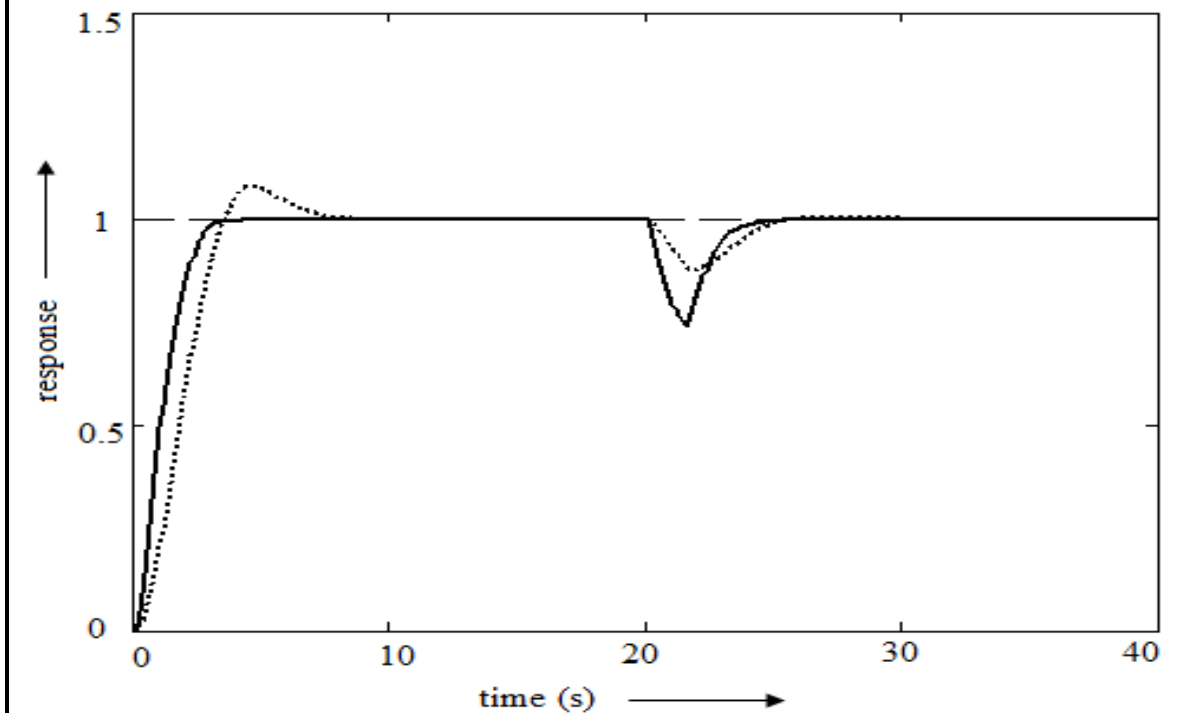


Figure 4.13: Responses of (4.5) with FPDC (dotted) and AFPDC (solid) for $L=0$ with load disturbance at 20s.

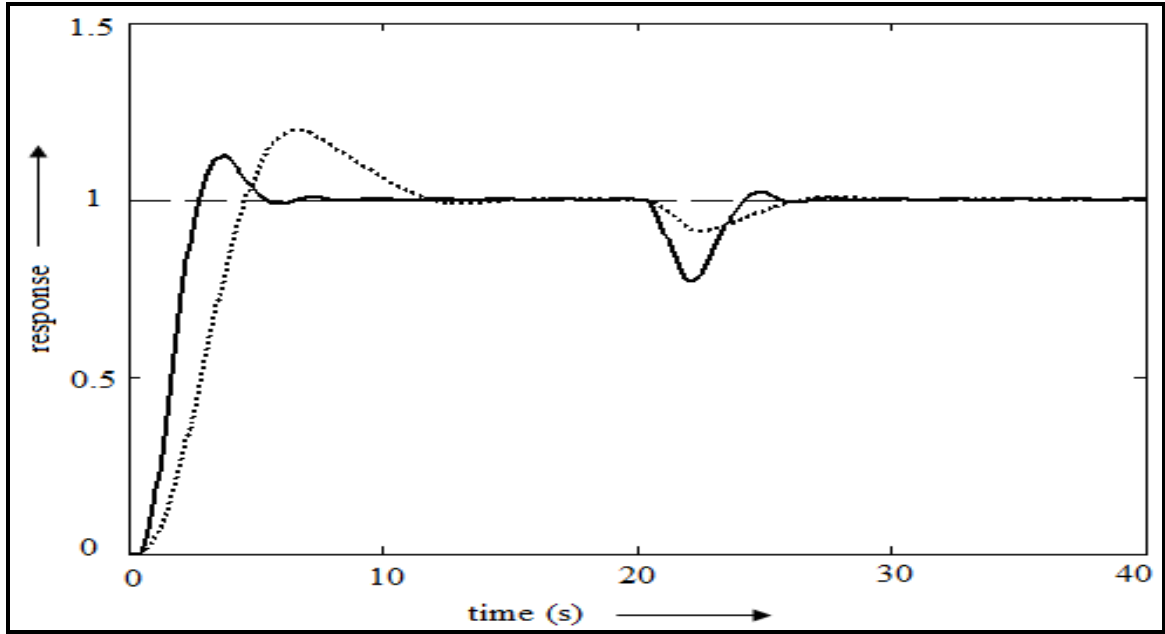


Figure 4.14: Responses of (4.5) with FPDC (dotted) and AFPDC (solid) for $L=0.3$ with load disturbance at 20s.

Table 4.4: Performance analysis of (4.5) with PID without load disturbance

L	%OS	$t_r(s)$	$t_s(s)$	IAE	ITAE	ISE
0.0	85.04	1.8	49.8	8.89	123.74	3.45
0.1	95.12	1.8	-	15.45	346.96	6.18
0.2	107.02	1.9	-	48.52	2071.79	32.54

Table 4.5: Performance analysis of the nonlinear process (4.5)

L	Controller Type	%OS	$t_r(s)$	$t_s(s)$	IAE	ITAE	ISE
0.0	PID	85.04	1.8	68.6	13.68	292.16	5.16
	FPDC	7.85	3.4	6.7	3.25	32.50	1.56
	AFPDC	0.0	2.7	3.0	1.71	12.50	0.82
0.1	PID	95.12	1.8	-	23.96	707.21	10.99
	FPDC	12.07	3.7	8.1	3.74	35.90	1.81
	AFPDC	2.17	2.6	2.8	1.88	14.24	0.96
0.3	FPDC	19.66	4.4	11.0	4.97	47.21	2.48
	AFPDC	12.42	2.7	5.1	2.25	13.75	1.22

Integrating and Unstable System

Let us consider a 2nd order integrating system:

$$\frac{d^2y}{dt^2} - \frac{dy}{dt} = u(t-L) \quad (4.6).$$

The process is unstable in nature because one of its poles is located at the right-half of the s -plane. Here, the PD controller plays an important role in converting this integrating and unstable process (4.6) into a stable one, because PD controller introduces a *zero* in the left-half s -plane. It is observed that conventional controllers are unable to provide satisfactory performance as indicated by Fig. 4.15 and Table 4.6. As shown in Fig. 4.16, AFPDC is found to be able to provide stable performance for increased dead-time, *i.e.*, $L=0.1$, but the PID and even FPDC fail to do so. Response characteristics of (4.6) are depicted in Fig. 4.15 and Fig. 4.16 with disturbance at $t=30s$. Table 4.6 provides a detail performance comparison of the system with different controllers. Like previous results, here also, AFPDC exhibits highly improved performance compared to others.

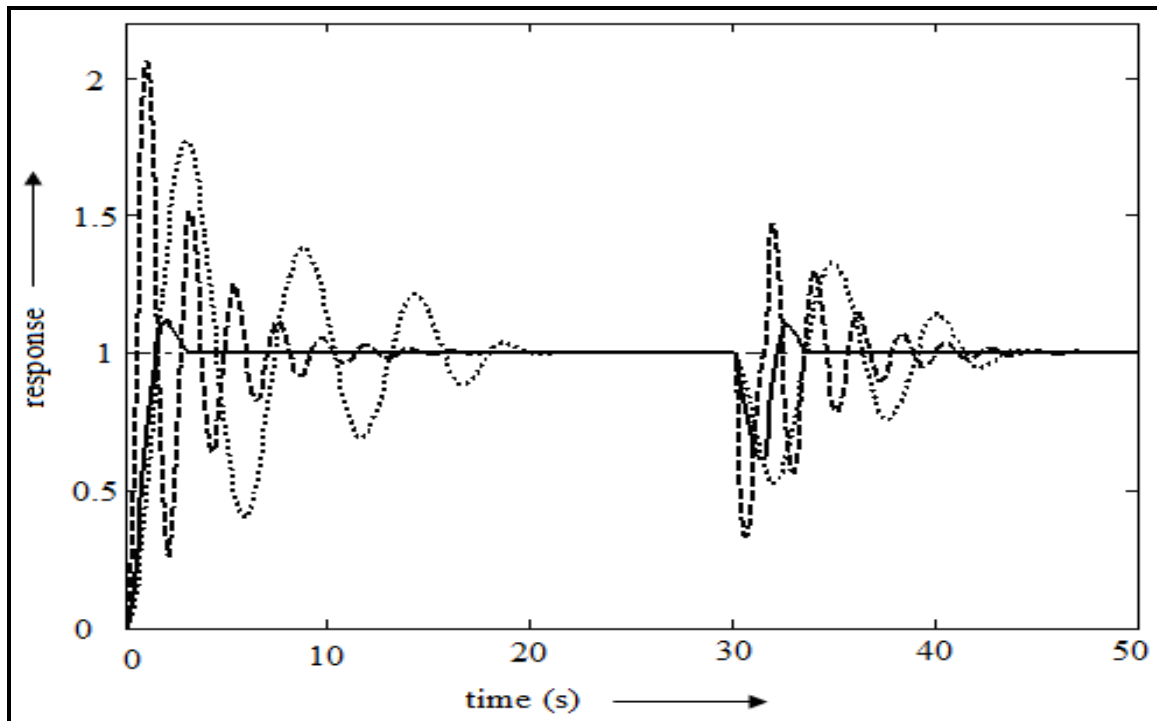


Figure 4.15: Responses of the system (4.6) with PID (dashed), FPDC (dotted) and AFPDC (solid) for $L=0$ with load disturbance at 30s.

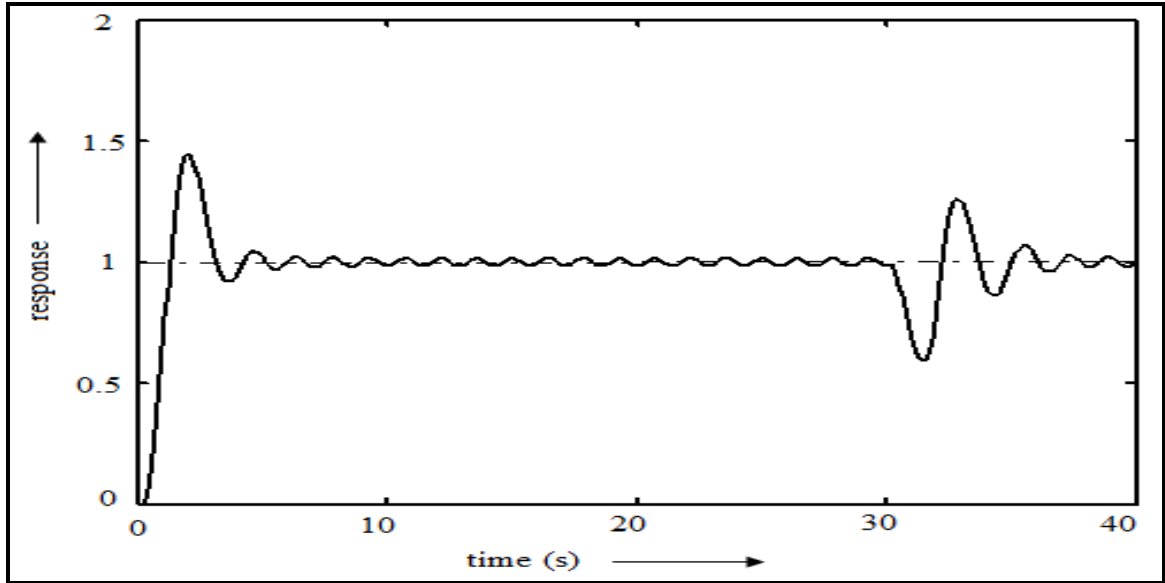


Figure 4.16: Response of the system (4.6) with AFPDC for $L=0.1$ with load disturbance at 30s.

Table 4.6: Performance analysis of the integrating process (4.6)

L	Controller Type	%OS	$t_r(s)$	$t_s(s)$	IAE	ITAE	ISE
0.0	PID	106.2	0.6	10.0	4.41	61.75	1.94
	FPDC	77.44	1.6	17.7	6.92	100.71	2.64
	AFPDC	11.84	1.5	2.8	1.31	13.65	0.64
0.1	AFPDC	44.90	1.3	4.1	2.89	65.04	0.90

4.4 Real time experiments on overhead crane

In this section, experimentation with an overhead crane is carried out to verify the performance of proposed non-fuzzy tuning scheme. The overhead cranes have been widely employed in many industrial fields such as harbors and factories, where they should be operated to transfer cargoes as quickly and safely as possible within a given time for high transportation efficiency [161, 163]. The overhead cranes have one control input (trolley driving force) and two output variables (horizontal trolley position and load swing angle). This property results in a coupling effect between the load swing and cart position. In addition, uncontrolled load sway dynamics causes safety problems in crane systems, which makes it much more challenging to control. These integrating and pendulum type control problems of crane systems have attracted much interest

among control engineers. Thus, the need for faster cargo handling requires the precise control of crane motion so that its dynamic performance is improved.

Overhead crane set-up

A laboratory scale crane set-up (FEEDBACK, UK) shown in Fig. 4.17 consists of a cart, moving along the 1m length track and a load is attached with the cart through shaft [142, 143]. The cart can move back and forth causing the load to swing. The movement of the cart is caused by pulling the belt in two directions by the DC motor attached at the end of the rail as shown in Fig. 4.18. By applying a voltage to the motor we control the force with which the cart is pulled. The value of the force depends on the value of the control voltage. The two variables that are read using optical encoders are the load angle and the cart position on the rail. The controller's task will be to change the DC motor voltage depending on these two variables, in such a way that the desired crane control task is fulfilled. An optical encoder consists of a light source, light detector and a slit disk placed between them. This way the relative position with respect to the initial point can be measured by counting the pulses on the light detector. Additionally, safety limit switches are mounted on either side of the track which cut power to the motor when the cart crosses them. Initially the control signal is set to $-2.5v$ to $2.5v$ and the generated force magnitude of around $-20N$ to $+20N$. The cart position is physically bounded by the rail length and is equal to $-0.5m$ to $+0.5m$.

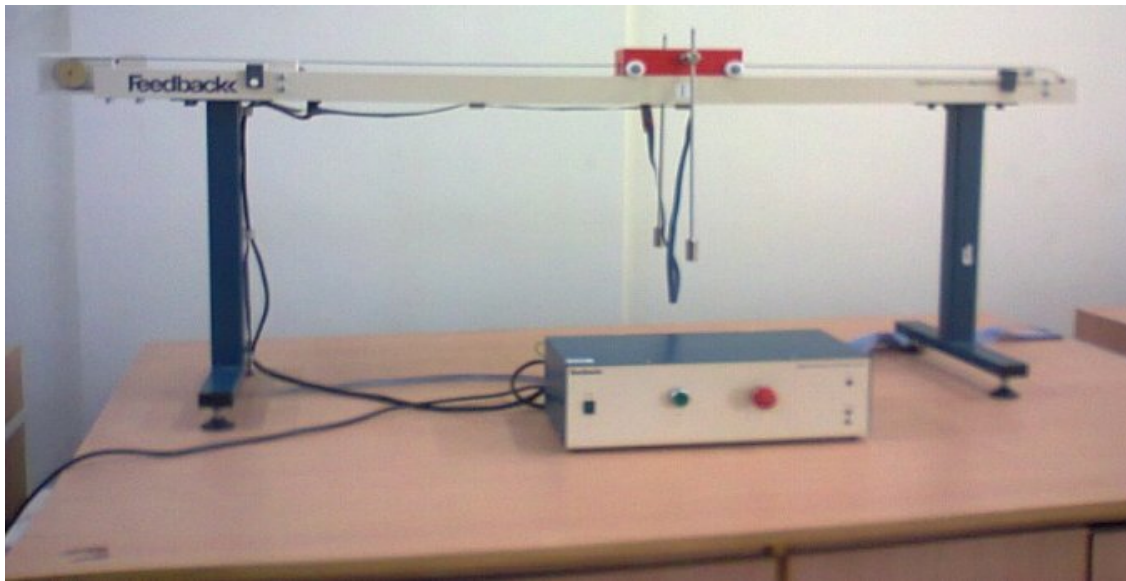


Figure 4.17: Overhead crane set-up.

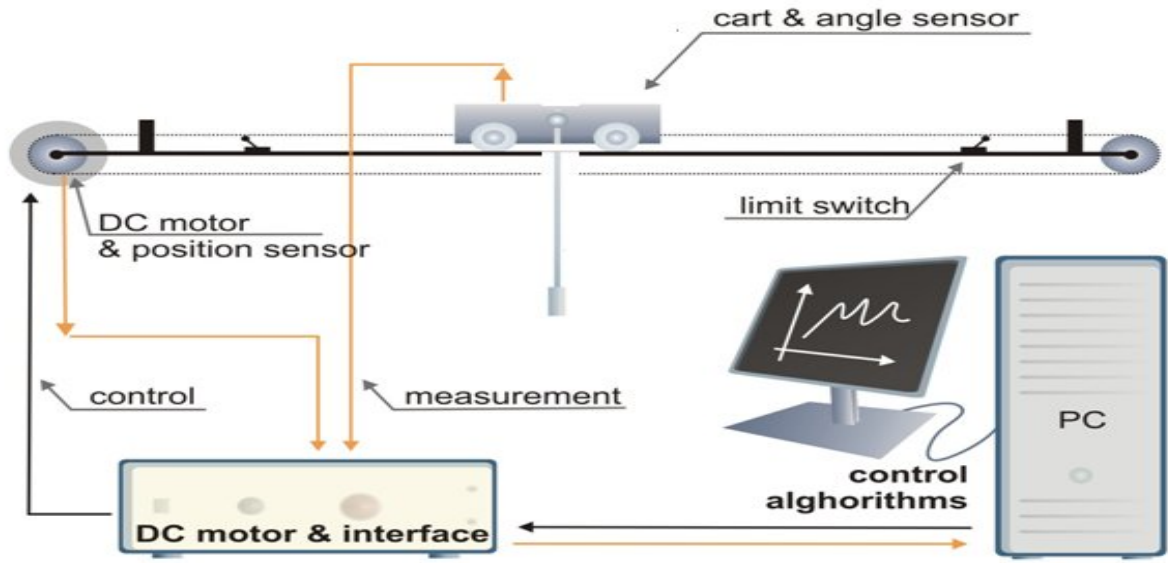


Figure 4.18: Crane control diagram.

Model of the overhead crane

Fig. 4.19 shows the schematic of an overhead crane traveling on a rail, where x , θ and u are the cart position, load swing angle and cart driving force respectively. The model is a SIMO plant – single-input multiple-outputs. The cart mass (M), load mass (m), pole length (l) and gravitational force (g) are given by $2.4kg$, $0.23kg$, $0.36m$ and $9.81m/s^2$ respectively. Here, the stiffness and mass of the rope are neglected and the load is considered as a point mass. The proposed scheme is focused on anti sway tracking control of an indoor overhead crane; therefore, the hoisting motion and the effects of wind disturbance are not considered. The phenomenological model of the overhead crane is nonlinear, meaning that at least one of the states (x and its derivative or θ and its derivative) is an argument of a nonlinear function [158].

We use Langrangian approach to derive the equations of motion. It follows from Fig. 4.19 that the load and cart position vector are given by

$$\vec{x}_l = \{x + l \sin \theta, -l \cos \theta\} \text{ and } \vec{x}_c = \{x, 0\} \quad (4.7)$$

Then the kinetic and potential energy of the granty crane system is given by K and P respectively:

$$K = \frac{1}{2} M \overline{\dot{x}_c \cdot \dot{x}_c} + \frac{1}{2} m \overline{\dot{x}_l \cdot \dot{x}_l} \quad (4.8)$$

$$\text{and } P = -mgl \cos \theta \quad (4.9)$$

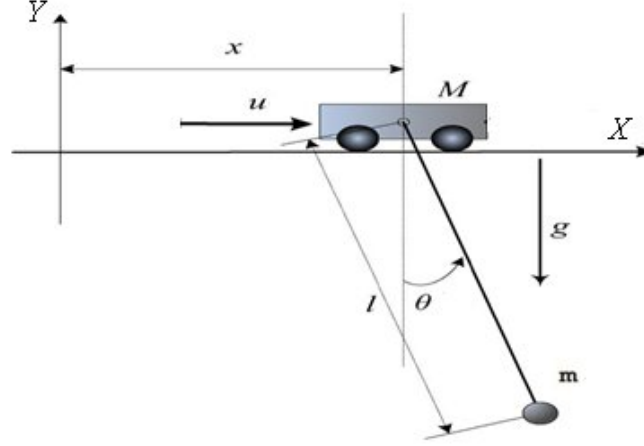


Figure 4.19: Model of the overhead crane system.

Let the generalized forces corresponding to the generalized displacements

$$\vec{q} = \{x, \theta\} \text{ be } \vec{u} = \{u_x, 0\}.$$

Then construct Lagrange's equations for $i=1, 2$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = u_i \quad (4.10)$$

Where, $L = K - P$ and u_i is the external force to the trolley.

Then, the equations of motion of the overhead crane system without uncertainty [157, 160] with respect to x and θ are obtained through equation (4.11) and (4.12) are:

$$(M + m)\ddot{x} + ml\ddot{\theta} \cos \theta + m\dot{l} \sin \theta + 2m\dot{l}\dot{\theta} \cos \theta - ml\dot{\theta}^2 \sin \theta = u_x \quad (4.11)$$

$$l\ddot{\theta} + g \sin \theta + 2\dot{l}\dot{\theta} + \dot{x} \cos \theta = 0 \quad (4.12)$$

The main difficulty in controlling the overhead crane system basically lies in the handling of the coupled nature between the sway angle and cart movement. The dynamic model represented in equations (4.11) and (4.12) are nonlinear in nature, that means the cart position and its derivative (\dot{x}) or swing angle and its derivative ($\dot{\theta}$) are nonlinear functions. To carry out analysis of the

model dynamics for open loop as well as closed loop system, the model has to be linearized for conventional controllers, which is not essential for our proposed scheme. Linearization of a given phenomenological model also can be carried out by using state-space model. For small angle of deviations and negligible change in rope length, the above dynamic model can be transformed to state space form as:

$$\dot{x} = Ax + Bu$$

$$y = Cx .$$

The state space model with $x = [x_1, x_2, x_3, x_4]^T$; where $x_1 = x$, $x_2 = \theta$ and x_3, x_4 are the derivatives of x_1, x_2 respectively. Then the matrices A , B and C are represented by

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{mg}{M} & 0 & 0 \\ 0 & -\frac{(M+m)g}{Ml} & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{M} \\ -\frac{1}{Ml} \end{bmatrix} \quad \text{and } C = [1 \ 0 \ 0 \ 0]$$

Controller design for overhead crane

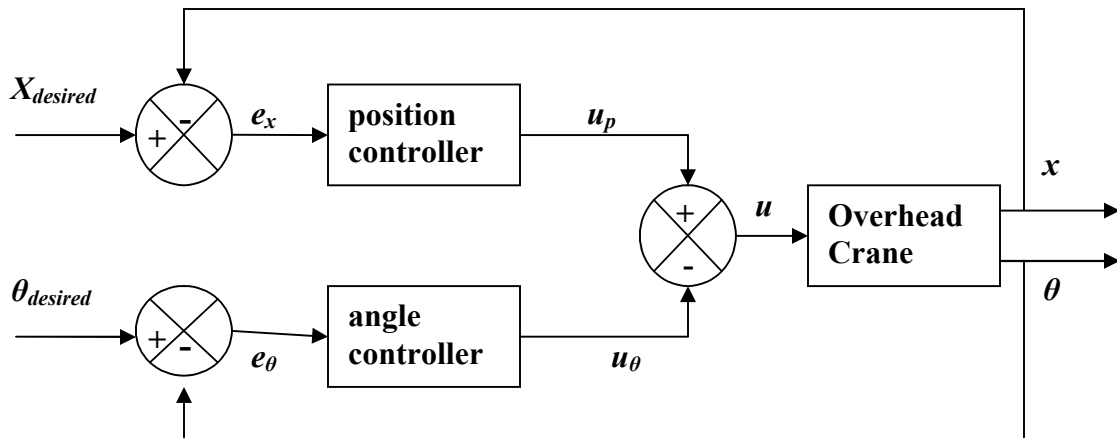


Figure 4.20: Dual control structure (position x and angle θ) for overhead crane control.

Fig. 4.20 shows the block diagram of dual control structure for the proposed overhead crane control. The feedback signals from the overhead crane act as the input variables of the controller as shown in the Fig. 4.20. Fig. 4.21 shows the two similar fuzzy logic controllers, which deal separately with the cart position and swing angle to drive the overhead crane. In this design, the normalized values of position error (e) and change of position error (Δe) are selected as the input linguistic variables of fuzzy position controller. The input linguistic variables of fuzzy angle controller are selected as normalized values of swing angle error (e_θ) and its derivative (Δe_θ).

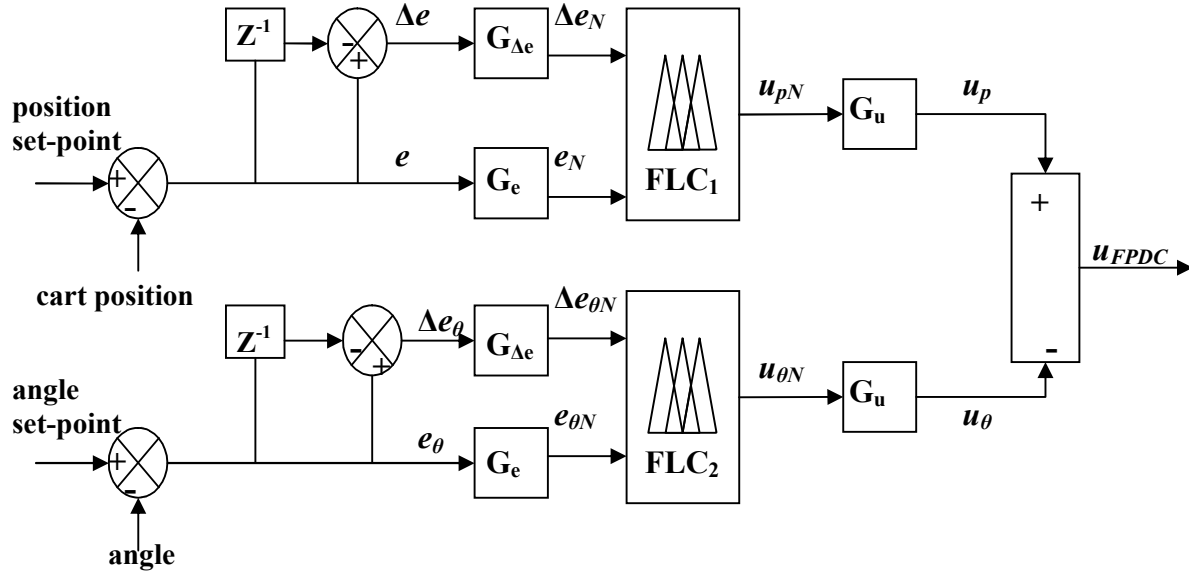


Figure 4.21: Diagram of FPDC for overhead crane control.

Our proposed adaptive scheme is applied in FPDC as shown in Fig. 4.22. In this scheme, the output of FPDC (Fig. 4.21) is further modified by an automatic gain updating factor (β) to achieve satisfactory control performance.

In our design, the left swing of the load is defined as positive swing, while the right swing of the load is negative swing. The output of the AFPDC for position and swing angle control are u_p and u_θ respectively as indicated in Fig. 4.22. For the overhead crane control using AFPDC, we incorporate the proposed gain adaptive scheme through an on-line gain modifier β , which is determined by the relation $\beta = K(1 + \alpha)$, where α is obtained from the multiplication of the normalized controller inputs. The actual control action to drive the cart by AFPDC is u_{AFPDC} , whereas, the actual control action to drive the cart using FPDC is u_{FPDC} . The controller output

u_{FPDC} and u_{AFPDC} of FPDC and AFPDC respectively are used to drive the DC motor of the overhead crane for position and angle control as shown in Fig. 4.21 and Fig. 4.22. Fig. 4.23 denotes the MFs of e , Δe and u_p for position controller and MFs of e_θ , Δe_θ and u_θ for angle controller. Error (e) due to position and error (e_θ) due to angle are obtained respectively from the cart position encoder and swing angle encoder. The ranges selected of input-output variables for position and angle controller are $[-0.5, +0.5]$ and $[-20^\circ, +20^\circ]$ respectively.

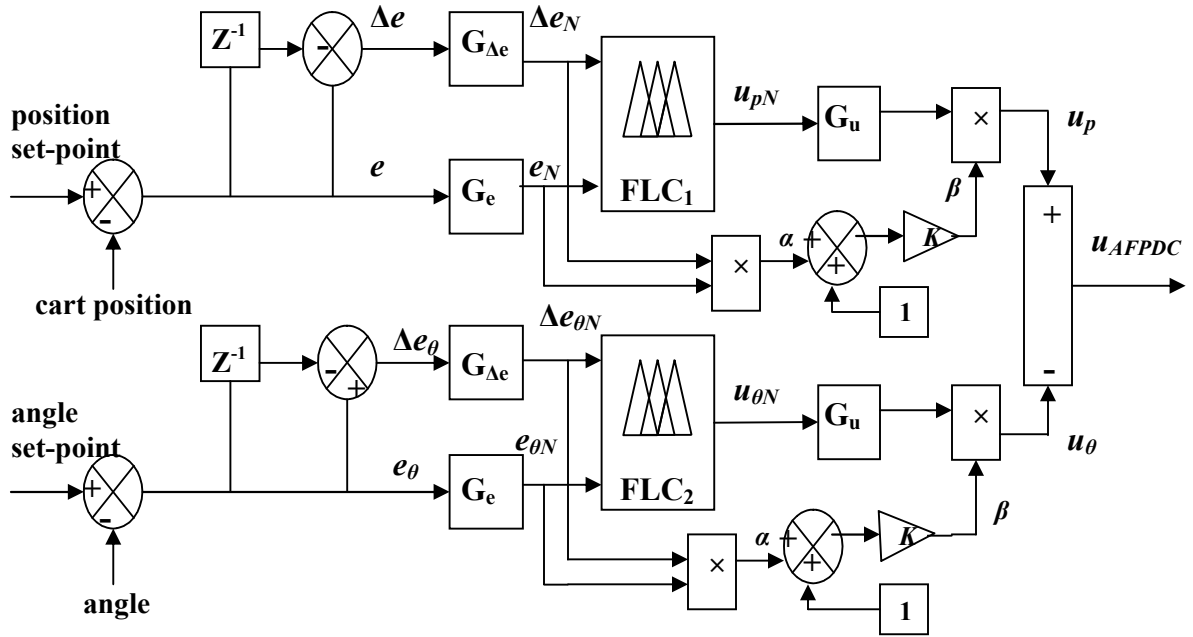


Figure 4.22: Diagram of AFPDC for overhead crane control.

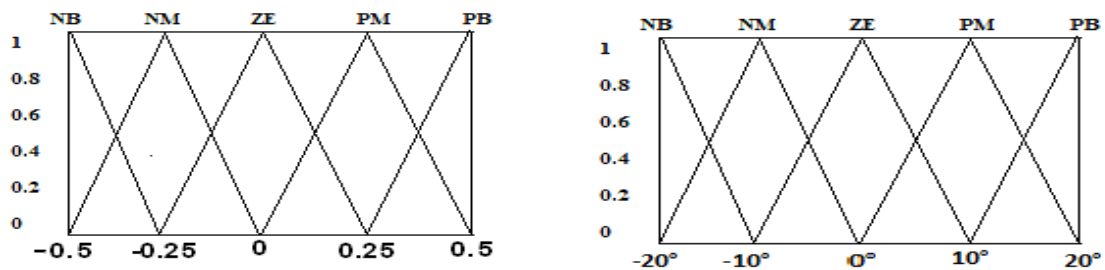


Figure 4.23: MFs for position $[-0.5, +0.5]$ and angle $[-20^\circ, +20^\circ]$ controller.

Each of the position and angle controllers consists of similar 25 fuzzy *if-then* rules as shown in Table 4.1. We have observed highly nonlinear but identical three dimensional control surfaces (Fig. 4.5 and Fig. 4.6) for position and angle controller both in FPDC and AFPDC.

Note that, here in dual control structure, both the controllers have only $i/2$ fuzzy atomic clauses in the antecedent part of each rule, where ‘ $i=4$ ’ refers 4 input linguistic variables (e_p , Δe_p , e_θ and Δe_θ). In FPDC or AFPDC, each input variable of position and angle controller has ‘ n ’ linguistic terms, here $n=5$ (Fig. 4.23), thus the number of control rules required is $2.n^{i/2} = 50$. However, a single conventional fuzzy controller for the same purpose may need n^i , i.e., $5^4=625$ rules, which is very much greater than the total number of rules required for the present dual control scheme (i.e., 50).

Results and discussion

The proposed gain adaptive scheme is tested on an overhead crane (Fig. 4.17) with square input and step input with different amplitudes [162, 171]. Mainly the step and square signals are used as reference because this type of signals matches the real situation, where the cart with payload is driven from home position to the desired position and back to the home position [175, 176]. The system is at first tested with PID controller. The cart position control response using PID for step and sine input are shown in Fig. 4.24 and Fig. 4.25 respectively. Fig. 4.26 and Fig. 4.27 show that the load sway for application of conventional PID controller is not smooth for different inputs, which is one of the most desirable parameters for overhead crane control in industry.

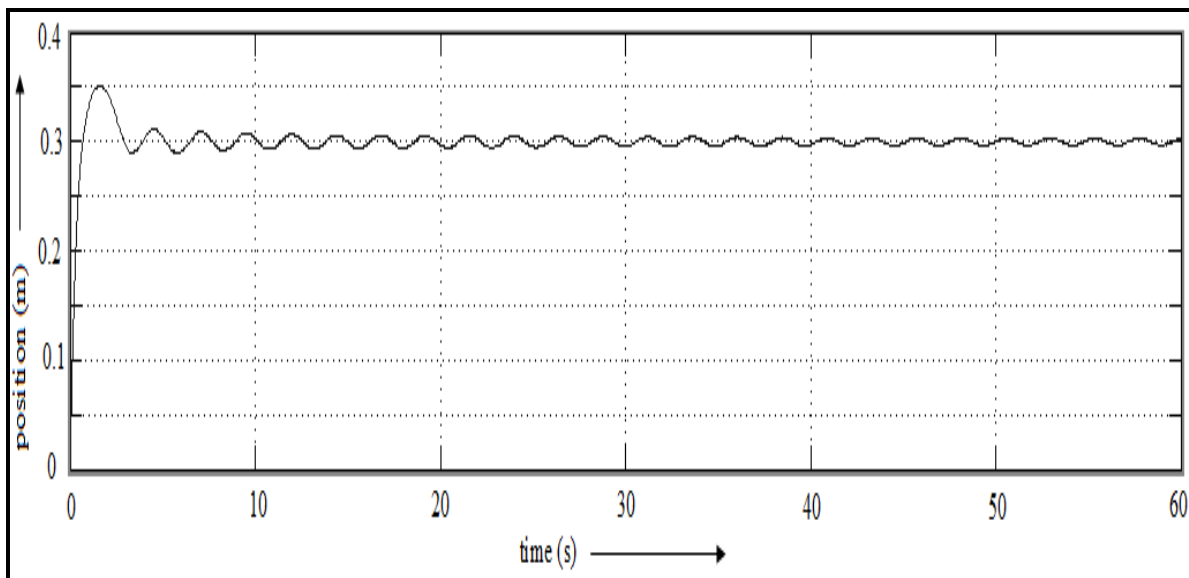


Figure 4.24: Overhead crane position control using PID controller for step input (0.3m).

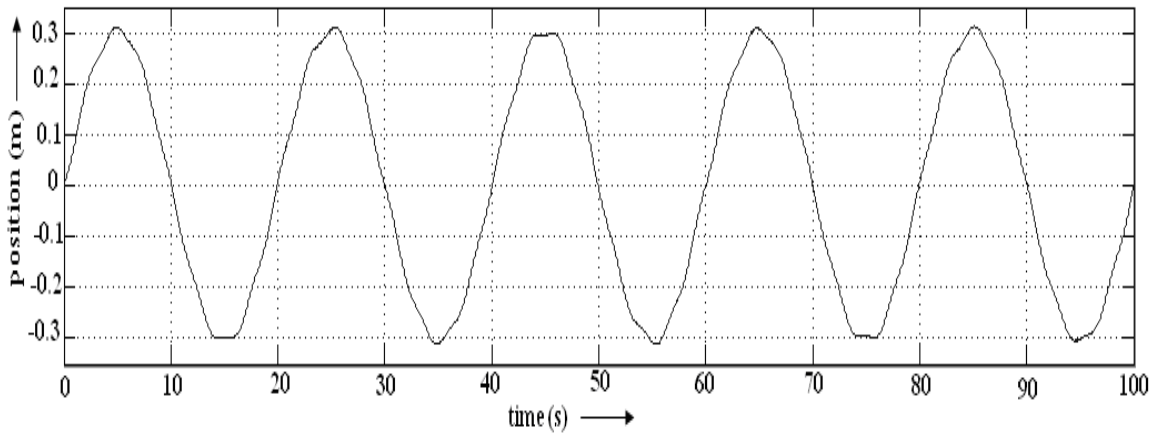


Figure 4.25: Overhead crane position control using PID controller for sine input ($\pm 0.3m$).

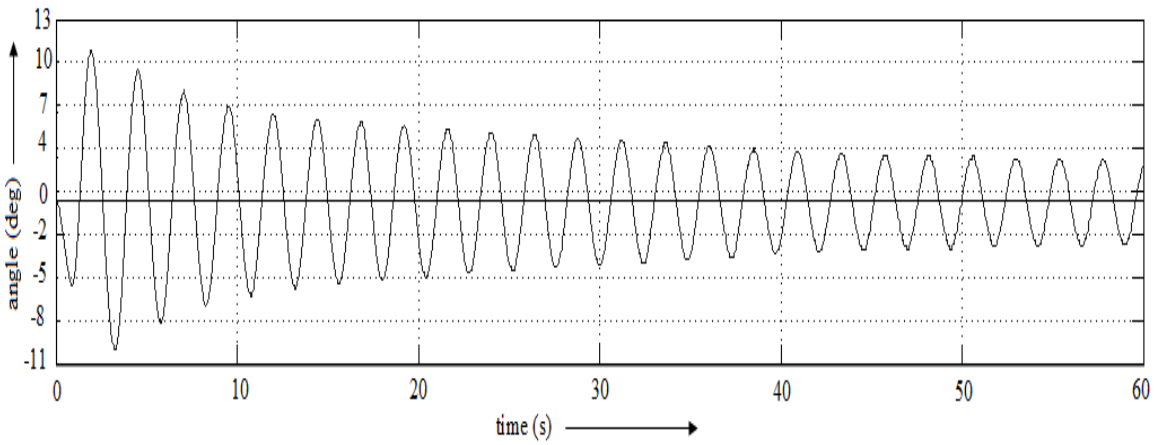


Figure 4.26: Overhead crane swing angle control using PID controller for step input ($0.3m$).

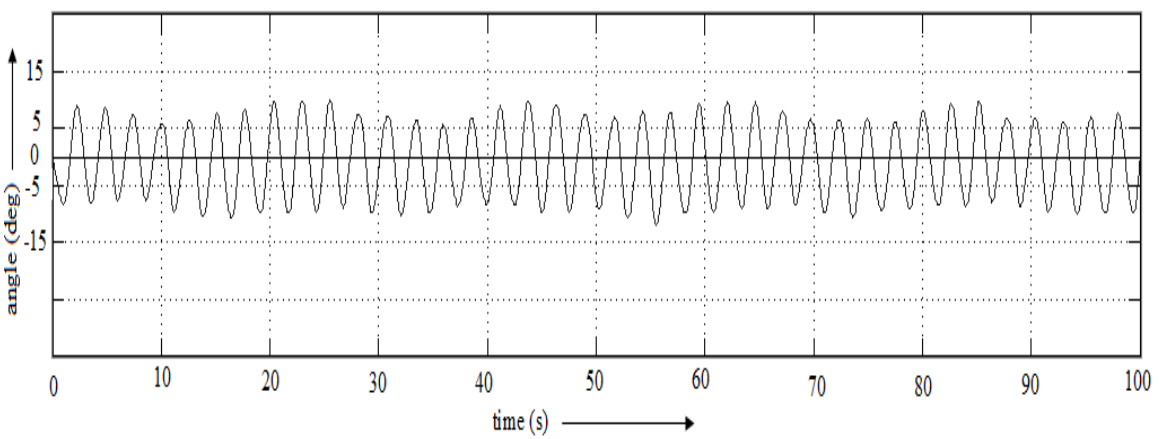


Figure 4.27: Overhead crane swing angle control using PID controller for sine input ($\pm 0.3m$).

The controller output (u_{FPDC} / u_{AFPDC}) of FPDC and AFPDC are separately applied to the overhead crane to control the crane position and as well as swing angle of the load attached. The AFPDC outperforms the PID and FPDC for different types of input applied as shown in Fig. 4.28 and Fig. 4.29. Real time experiments for position control of the overhead crane illustrate the advantages of proposed self-tuning scheme for various types of inputs. Fig. 4.28 and Fig. 4.29 show that proposed adaptive fuzzy controller tracks the set-point very efficiently and placed the trolley in desired position. From Table 4.7, we find that different performance parameters such as IAE , $ITAE$ and ISE are reduced by a large percentage when controlled by AFPDC compared to FPDC. In case of step response rise time is decreased to 1s compared to 1.6s for FLC.

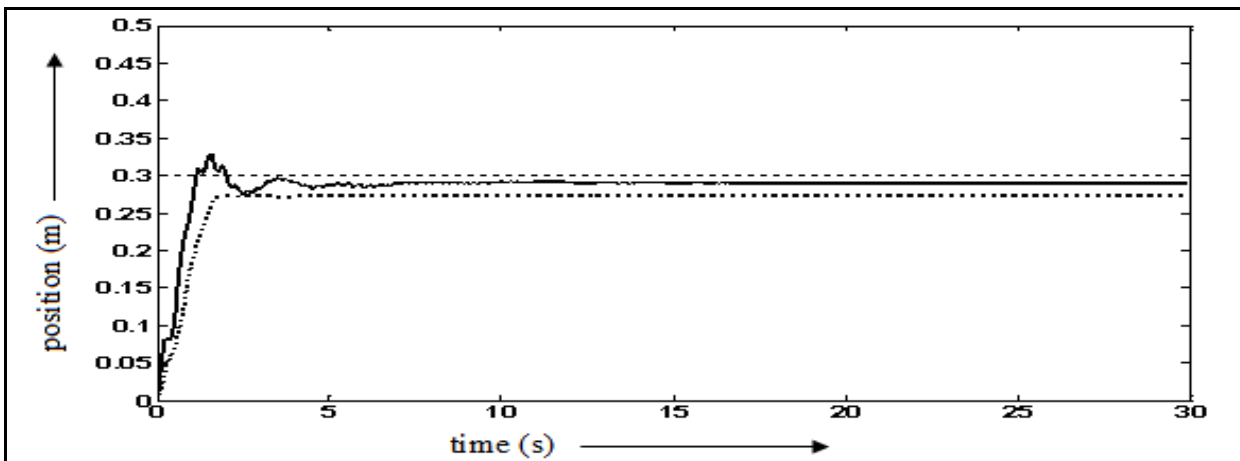


Figure 4.28: Position control for step input (0.3m) using FPDC (dotted line) and AFPDC (solid line).

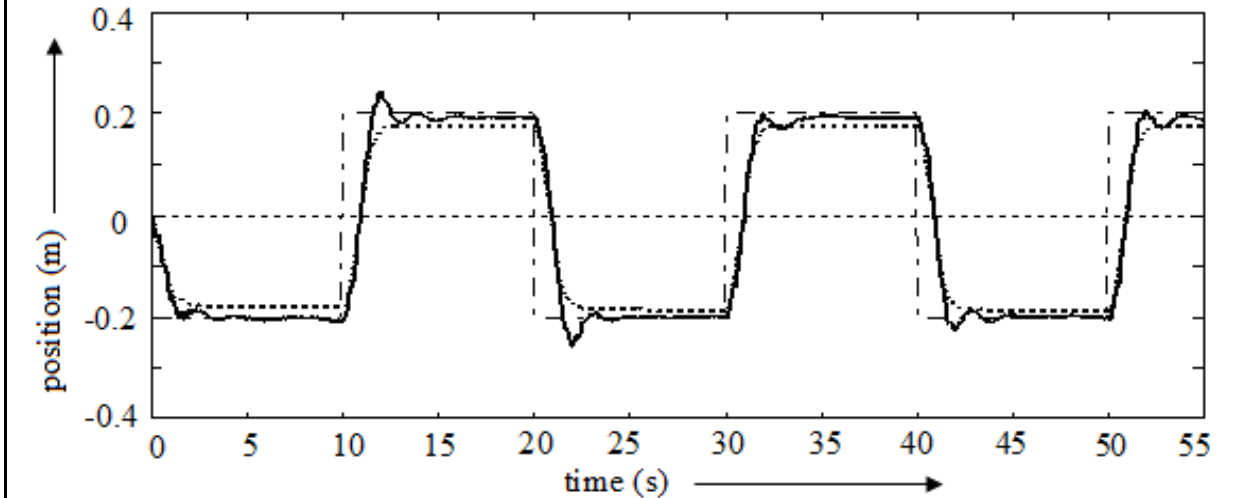


Figure 4.29: Position control for square input (dash-dot) using FPDC (dotted) and AFPDC (solid line).

Table 4.7: Performance analysis of the proposed controllers in overhead crane control

Reference input	Controller Type	IAE	ITAE	ISE
Step (amplitude $0.3m$)	FPDC	7.51	75.57	0.70
	AFPDC	3.67	26.62	0.39
Square (amplitude $\pm 0.2m$)	FPDC	31.10	847.40	6.77
	AFPDC	24.20	698.23	6.05

The proposed dual control scheme positions the cart in almost exact location with negligible offset as shown in Fig. 4.28 and Fig. 4.29. Responses due to step and pulse type inputs as presented in Fig. 4.30 and Fig. 4.31, illustrate that proposed AFPDC makes negligible sway angle for horizontal movement of the trolley crane. Also we find that swinging angle is very less (maximum around 5°) even with vertical movement of the crane trolley. We have also successfully demonstrated our proposed scheme for step input with different amplitudes as shown in Fig. 4.32 and in all the cases, we observe negligible sway angle [177].

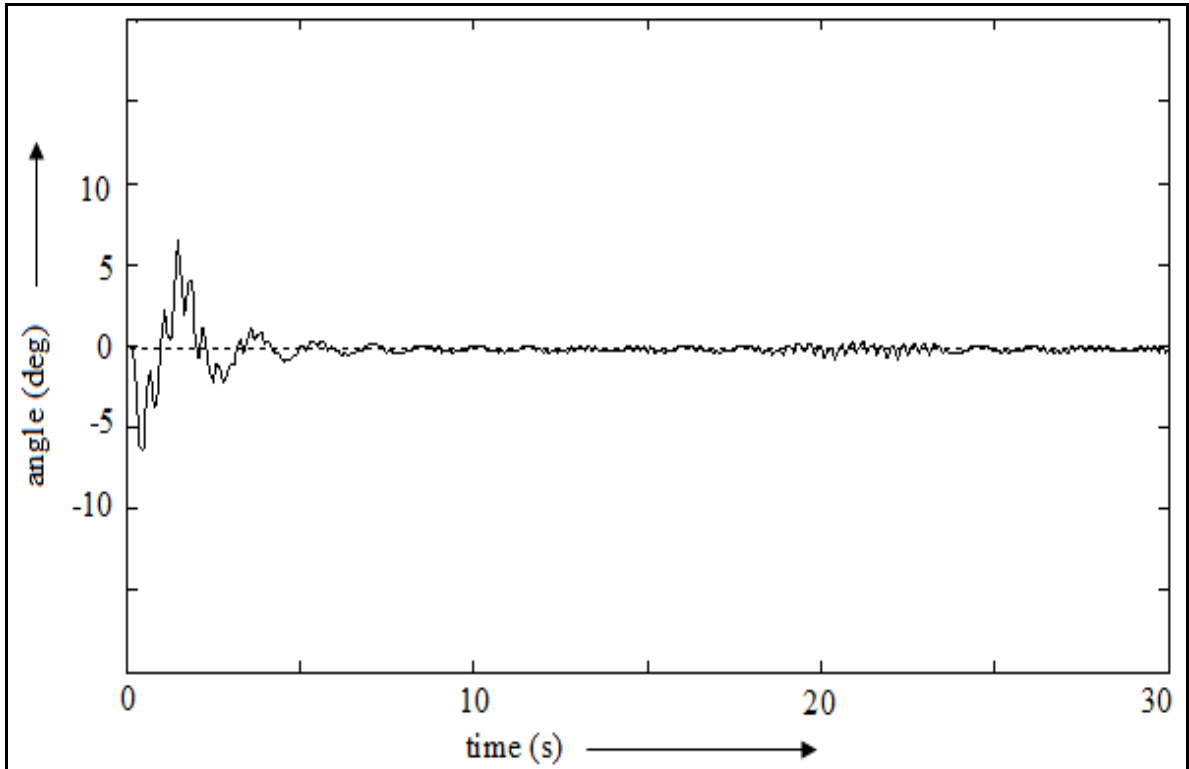


Figure 4.30: Overhead crane swing angle control for step input using AFPDC.

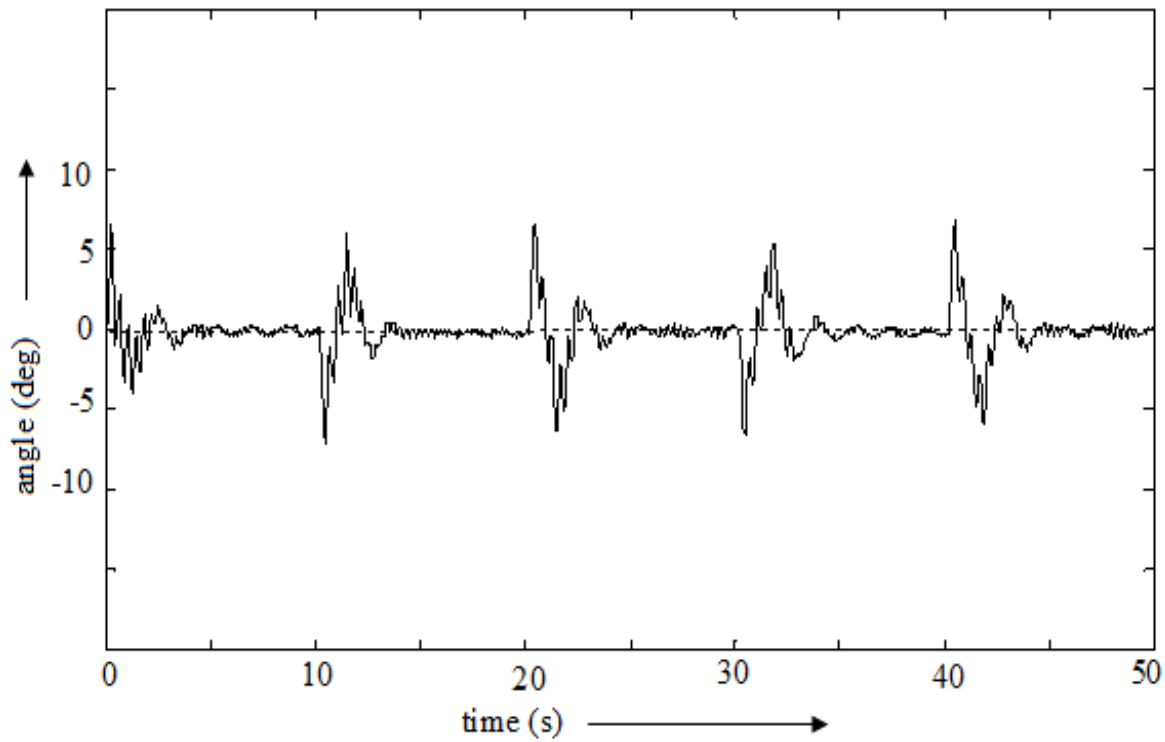


Figure 4.31: Overhead crane swing angle control for square input using AFPDC.

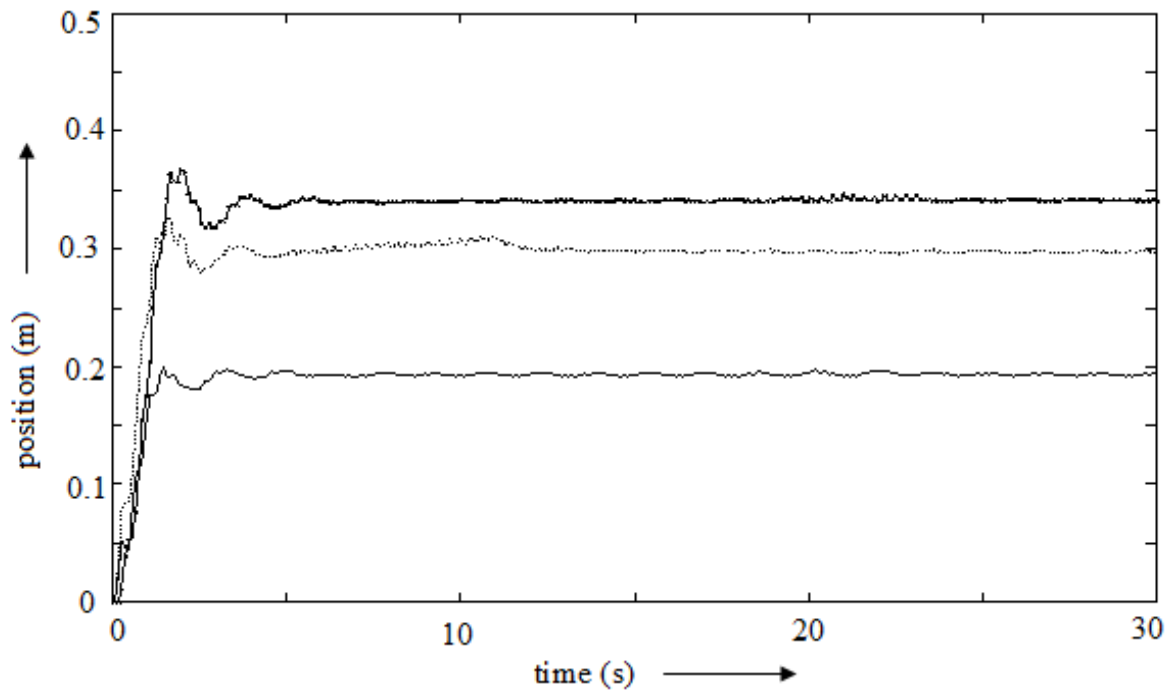
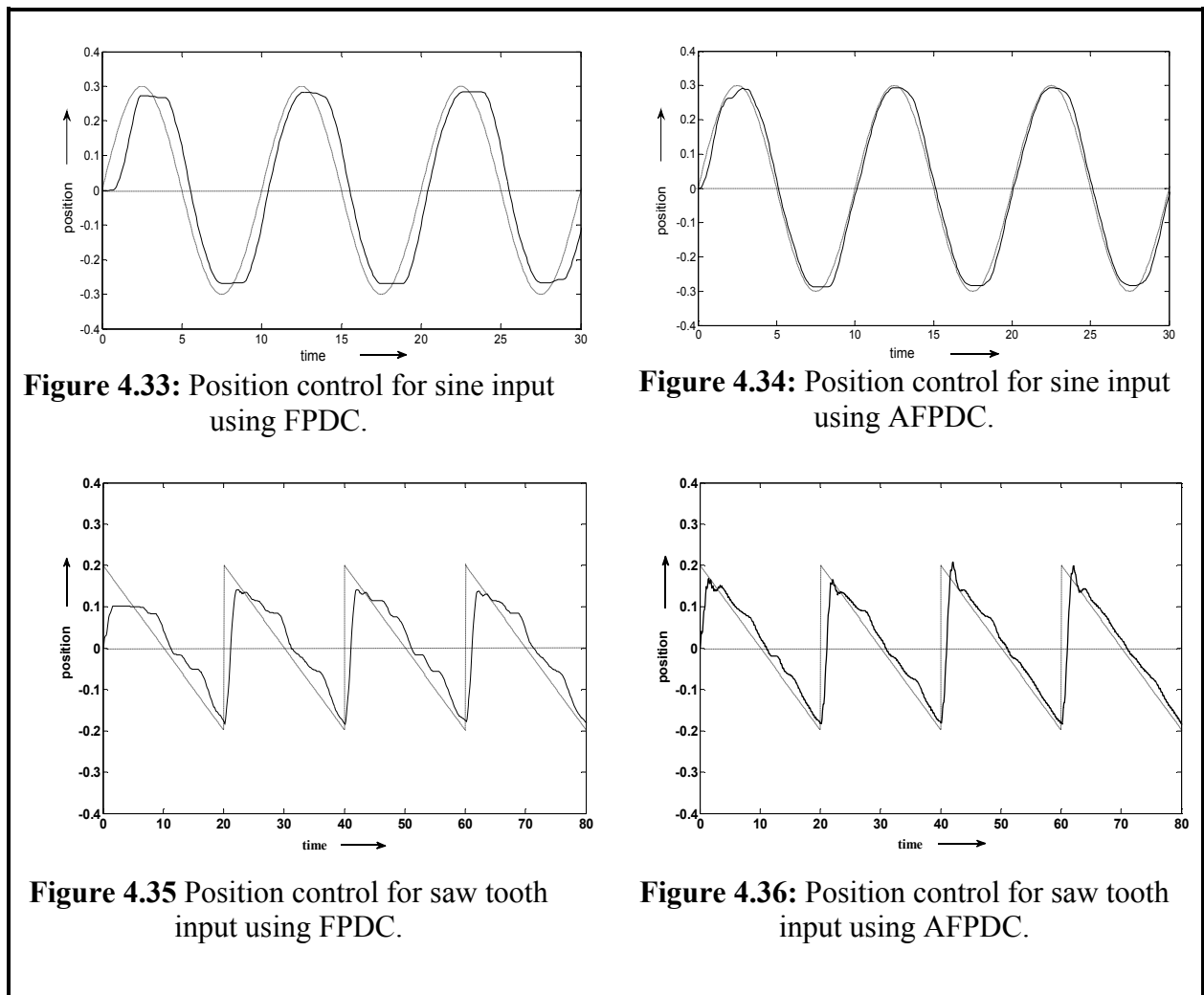


Figure 4.32: Overhead crane position control for step input using AFPDC for different amplitude ($0.2m$, $0.3m$ and $0.35m$) at constant speed.

We have also studied the effectiveness of our scheme with other inputs like sinusoidal input and saw tooth input. The performance of FPDC and AFPDC are well demonstrated in Fig. 4.33 to Fig. 4.40 for both position and angle control. The corresponding performance analysis is made in Table 4.8. The study proves the superiority of AFPDC over PID and FPDC. We have also successfully demonstrated our proposed scheme in Fig. 4.41 with sinusoidal input with different amplitudes and subsequently Fig. 4.42 indicates that even variation in speed does not hamper the quality of position control of the overhead crane when we apply our proposed non-fuzzy adaptive scheme.



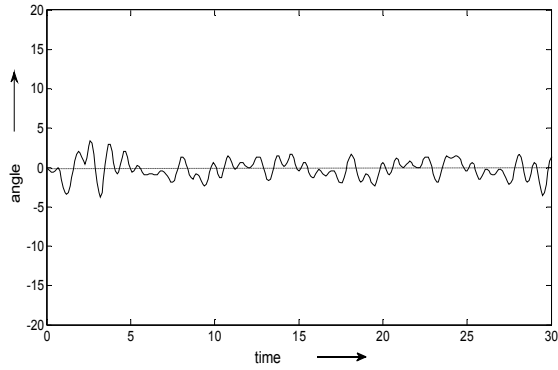


Figure 4.37: Overhead crane swing angle control for sine input using FPDC.

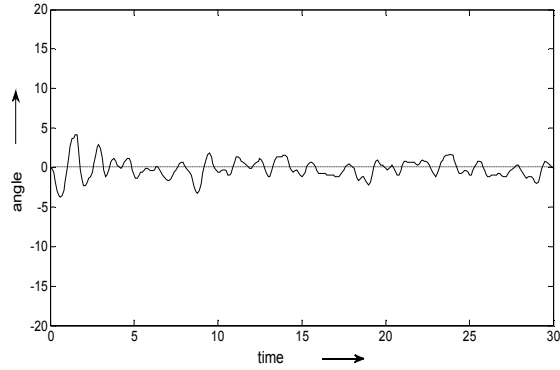


Figure 4.38: Overhead crane swing angle control for sine input using AFPDC.

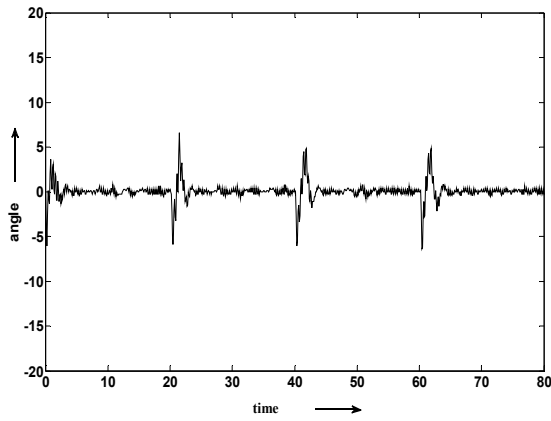


Figure 4.39: Overhead crane swing angle control for saw tooth input using FPDC.

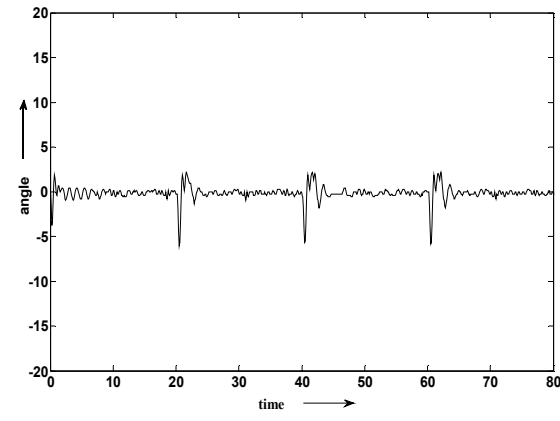


Figure 4.40: Overhead crane swing angle control for saw tooth input using AFPDC.

Table 4.8: Performance analysis of AFPDC for sine and saw tooth inputs

Reference input	Controller Type	IAE	ITAE	ISE
Sine (amplitude $\pm 0.3m$)	FPDC	32.64	799.22	2.89
	AFPDC	10.43	253.39	0.28
Saw tooth (amplitude $\pm 0.2m$)	FPDC	47.18	2236.3	6.08
	AFPDC	35.45	1714.1	4.92

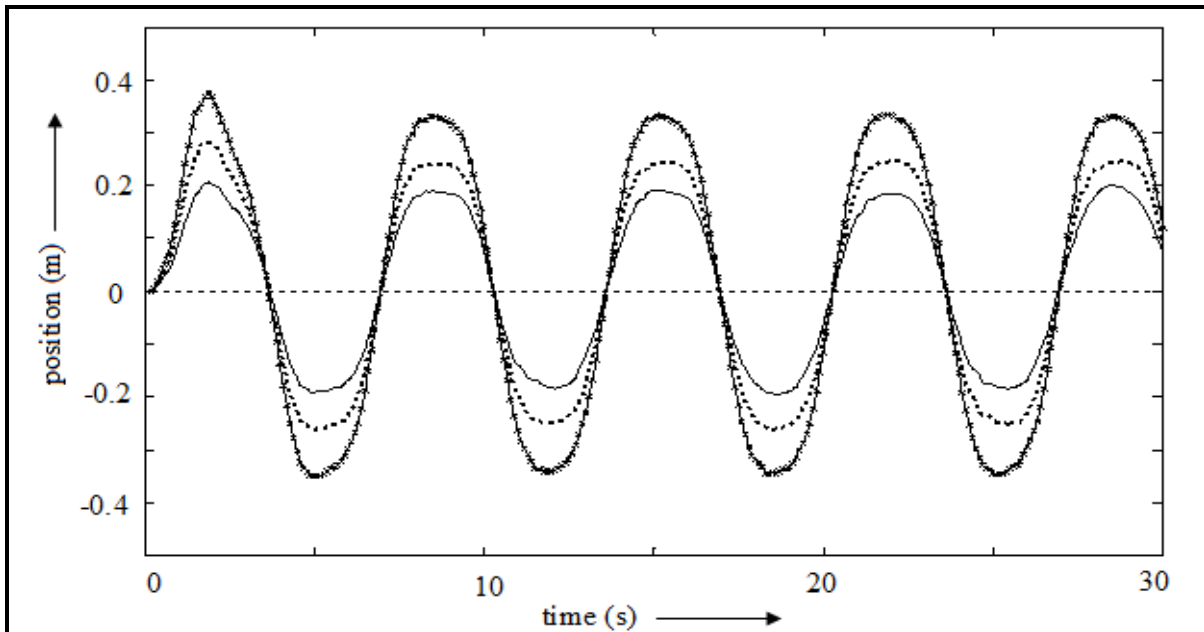


Figure 4.41: Overhead crane position control for sine input using AFPDC for different amplitude ($\pm 0.2m$, $\pm 0.3m$ and $\pm 0.35m$) at constant speed.

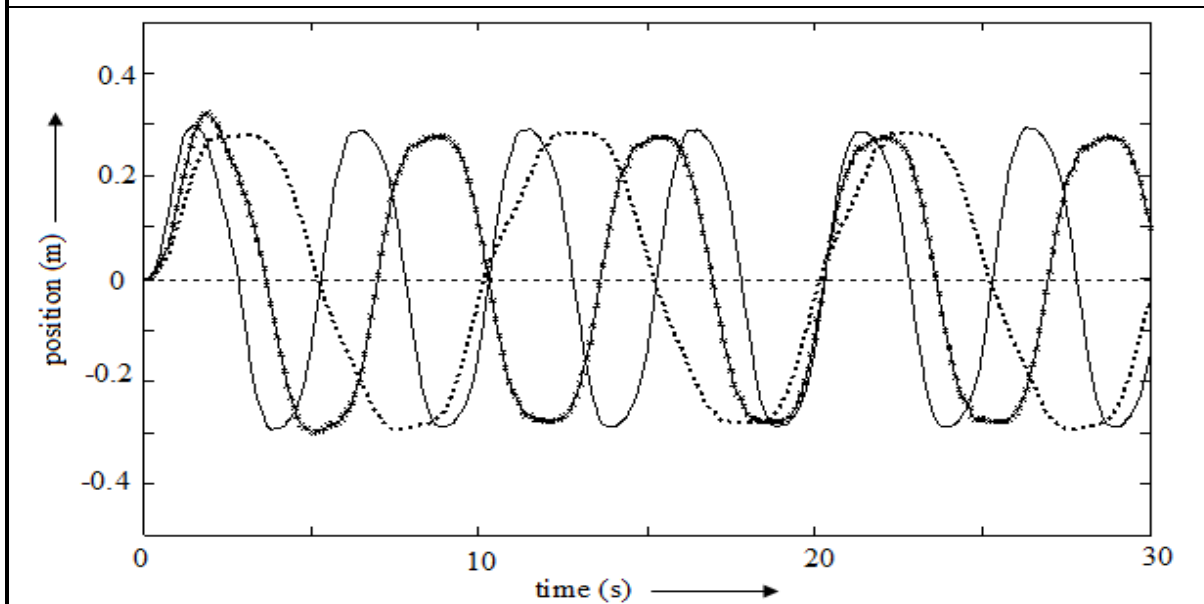


Figure 4.42: Overhead crane position control for sine input using AFPDC at different speed (with constant amplitude $\pm 0.3m$).

The above study reveals that the proposed gain adaptive scheme for fuzzy controller can fix the overhead crane in its desired position with negligible sway angle.

4.5 Conclusion

A simple gain adaptive or self-tuning scheme for a PD-type FLC has been proposed in this chapter [177]. In this study, the close loop gain of the proposed AFPDC is continuously updated by an on-line gain modifying parameter β defined on the normalized error (e_N) and change of error (Δe_N). The most important feature of the proposed scheme is that it depends neither on the process being controlled nor on the controller used. Even with significant reduction of rule-base, proposed AFPDC exhibited effective and improved performance compared to its conventional fuzzy and non-fuzzy controllers for wide variety of second order integrating, nonlinear and non-minimum phase systems with varying dead-time. This study also justified the usefulness of the dual control scheme to control the overhead crane. The proposed twin control scheme for overhead crane reduces the computational complexity and is easy to understand. By applying the proposed self-tuning method and dual control scheme the load swing angle of the crane comes to a minimum. Experimental results proved that the proposed AFPDC not only positioned the trolley in the desired location, it also significantly reduced the load swing during movement.

CHAPTER 5

Fuzzy rule-based system identification using self-organizing map

5.1 Introduction

In *chapter-2*, for self-tuning fuzzy controller development, we used 49 fuzzy rules for a conventional FLC design and further 49 rules were used for its tuning. But the question is, do we really need so much of fuzzy rules or can we realize the same level of controller performance even with much smaller set of rules? We have successfully investigated this issue in *chapter-3* and *chapter-4* by using smaller number of fuzzy rules and also by applying non-fuzzy adaptive scheme. In this chapter, and also in *chapter-6*, we investigate this issue and *propose a rule extraction scheme that can pick a smaller but adequate rule-base from a set of input-output data*.

Usually, any fuzzy model is developed by a number of fuzzy *if-then* rules and each of the rules has two parts: antecedent part and consequent part. Different methodologies have been proposed which generate fuzzy *if-then* rules to design an effective fuzzy model from the available input-output numerical data. However, the construction of fuzzy rules is not an easy task, especially for ill-defined, complex and unknown systems [14, 17, 105]. Though, this problem may easily be tackled by taking suggestions from the domain experts, but, unfortunately it is very difficult to locate such typical domain experts in reality. Every working system produces an output for any given input. Once the input-output behavior of a system is available, then it is possible to identify the system using fuzzy rules [27-29, 178], even in absence of domain expert. Given a set of input-output data, identification of a computational scheme capturing the relation between the input and output is an important problem and is known as system identification.

To understand this, let us consider a system having ‘ n ’ number of inputs and corresponding ‘ n ’ number of outputs, described by:

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\} \in \mathfrak{R}^p \text{ and } \mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\} \in \mathfrak{R}^q.$$

A convenient method for representing the input-output data \mathbf{X}^* is

$$\mathbf{X}^* = \begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix} = \left\{ \mathbf{x}_k^* = \begin{pmatrix} \mathbf{x}_k \\ \mathbf{y}_k \end{pmatrix} \in \mathfrak{R}^{p+q}, k = 1, 2, \dots, n \right\}, \mathbf{X} \in \mathfrak{R}^p, \mathbf{Y} \in \mathfrak{R}^q, \mathbf{X}^* \in \mathfrak{R}^{p+q}.$$

Here, $\mathbf{x}_k = \{x_{k1}, x_{k2}, \dots, x_{kp}\} \in \mathfrak{R}^p$ is the k^{th} input vector and $\mathbf{y}_k = \{y_{k1}, y_{k2}, \dots, y_{kq}\} \in \mathfrak{R}^q$ is the corresponding k^{th} output vector of a multiple-input multiple-output system. Thus the k^{th} input-output vector can be represented by $\mathbf{x}_k^* = \begin{pmatrix} \mathbf{x}_k \\ \mathbf{y}_k \end{pmatrix} \in \mathfrak{R}^{p+q}, k = 1, 2, \dots, n$. For simplicity

henceforth we will consider $p+q = m$ i.e. $\mathbf{X}^* = \{\mathbf{x}_k^* \in \mathfrak{R}^m, k = 1, 2, \dots, n\}$.

There exists an unknown system that transforms \mathbf{x} to \mathbf{y} . Let the system be denoted by S , thus $\mathbf{y} = S(\mathbf{x})$. Such a definition realizes systems like controllers to classifiers. The problem here is to find an explicit (mathematical) or implicit (computational schemes/algorithms) model for S . Such system can be described by a mathematical function (like regression), by neural networks or by fuzzy rule-based systems [179]. Here we consider only identification of fuzzy rule-based systems through exploratory data analysis.

Our objective is to develop an approximate fuzzy model from the input-output data that performs like S [180, 181]. Firstly, to design a fuzzy model, it is essential to extract suitable number of fuzzy rules from the input-output data using a suitable clustering method that can be able to describe the system properly [182, 183]. Several attempts have been made earlier to characterize the input-output relationship using various clustering concepts [82, 85]. We will briefly review some of the methods in this chapter.

In this chapter, first we review some existing methods for rule extraction using clustering. Then we explain our rule extraction scheme that uses self-organizing map (SOM) algorithm. In this study, we also propose three different methods for estimating peaks of the initial membership functions. The base-widths of the MFs are determined such that completeness of the rule-base is guaranteed. Finally, the initial rule-base thus extracted is tuned using gradient decent. In our

scheme, we guess the appropriate number of fuzzy rules required for identifying the unknown system. Usually, depending on the requirement of rules, user selects the number of clusters, although it is possible to select them by using error function [120]. The proposed scheme is first applied to illustrate how the gain rule-base with 49 rules for STFPIC [22, 23] can be drastically reduced maintaining almost the same level of performance. Similarly, we investigate the rule extraction scheme in STFPIC that contains 98 rules (49 gain rules and 49 control rules). The advantages of the scheme are also demonstrated to control a real time water pressure control loop and a laboratory based overhead crane [184] in the next chapter. Since most of the subjects that we discuss in this chapter are dependent on results of clustering the training data by SOM algorithm, we discuss it next.

5.2 Self-Organizing Map - SOM

5.2.1 Introduction to SOM

Learning (training) is a process in which the network adjusts its parameters (synaptic weights) in response to input stimuli so that the actual output response converges to the desired output response. When the actual output response is the same as the desired one, the network has completed the learning phase and the network has acquired knowledge. Learning methods in neural networks can be broadly classified into three basic types: *Supervised*, *Unsupervised* and *Reinforced*. There are two types of unsupervised learning: Hebbian and Competitive. When the learning is based on the input data and is independent of the desired output, in such cases the network may respond to several output categories on training. But only one of the several neurons has to respond. The mechanism by which only one unit of the network is chosen to make a decision to respond is called competition. The mostly used competition among group of neurons is *Winner-Takes-all*. Here only one neuron in the competing group will have a non-zero output signal when the competition is completed. In this procedure, during training of the network, the network selects the output unit that is the best match for the current input vector; the weight vector for the winner is then adjusted with respect to the network's learning algorithm.

Among the different competitive learning algorithm, here we discuss about Kohonen learning approach, called Self-Organizing Map (SOM). In this learning approach, the units update their weights by forming a new weight vector that is a linear combination of the old weight vector and

the current input vector. The unit whose weight vector is closest to the input vector is allowed to learn.

Kohonen Self-Organizing map can also be termed as topology preserving map [185]. Dimensionality reduction with preservation of topological information is common in normal human subconscious information processing. A good example is that of biological vision where three-dimensional visual images are routinely mapped into a two-dimensional retina and information is preserved in a way that permits perfect visualization of a three-dimensional world. As Kohonen pointed out, the purpose of intelligent information processing possibly lies in the creation of simplified internal representations of the external world at different levels of abstractions [116, 125]. In self-organizing map, large dimensional input vectors are projected down on the two-dimensional map in a way that maintains the natural order of the input vectors. The dimensionality reduction could allow us to visualize easily important relationships among the data that otherwise might go unnoticed. In short, SOM algorithm can be regarded as a fast, nonlinear, ordered, and smooth mapping of higher dimensional input space to lower dimensional output space.

A distinguish feature of SOM is that it enforces neighborhood relationships on the resulting cluster centroids. Because of this, clusters that are neighbors are more related to one another than clusters that are not. Such relationships facilitate the interpretation and visualization of the clustering results. Indeed, this aspect of SOM has been exploited in many areas of engineering and medical science.

Even though SOM is similar to K-means or other prototype-based approaches, there is a fundamental difference. Centroids used in SOM have a predetermined topographic ordering relationship. During the training process, SOM uses each data point to update the closest centroid and centroids that are nearby in the topographic ordering. In this way, SOM produces an order set of centroids for any given data set. In other words, the centroids that are close to each other in the SOM grid are more closely related to each other than to the centroids that are further away. Because of this constraint, the centroids of a two-dimensional SOM can be viewed as lying on a two-dimensional surface that tries to fit the n-dimensional data as well as possible.

5.2.2 The SOM algorithm

Clustering an unlabeled input-output data X^*

$$X^* = \{x_k^* \in \mathfrak{R}^m, k = 1, 2, \dots, n\}, X^* \in \mathfrak{R}^m \quad (5.1)$$

is a partitioning of X^* and hence, to the objects generating X^* , into c subgroups such that each subgroup represents a *natural* substructure present in X^* . In other words, clustering finds *homogeneous* groups in X^* . The definition of ‘*natural*’ substructures or ‘*homogeneous*’ groups often depends on the problem at hand. Formally, clustering can be described as the assignment of labels to the vector in X^* , and hence, to the objects generating X^* so that similar objects get similar labels [186, 101]. If the labels are hard (crisp), we hope they identify c natural subgroups in X^* .

There are many types of SOM neural networks, in a two-dimensional rectangular SOM, each node or neuron is assigned a pair of coordinates (i, j) as shown in Fig. 5.1. The architecture of Kohonen self-organizing map is shown in Fig.5.2, which has two layers; input layer and output layer.

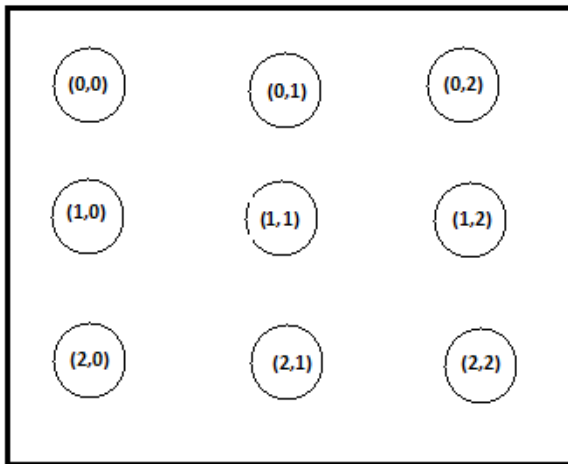


Figure 5.1: Two-Dimensional 3 by 3 rectangular SOM neural network.

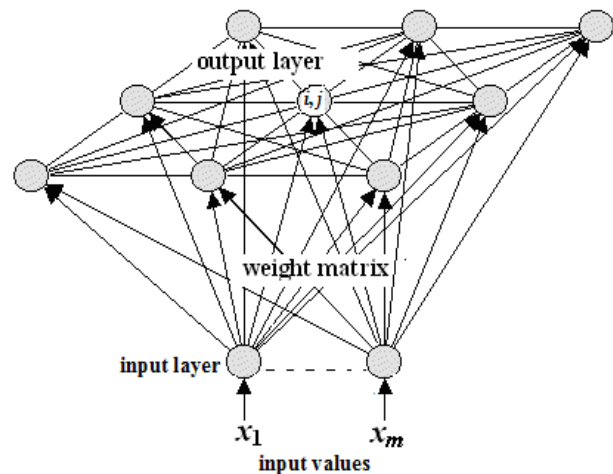


Figure 5.2: Kohonen Self-Organizing Map.

The SOM algorithm is primarily a competitive vector quantizer in which real valued patterns are presented sequentially to a linear or planar array of neurons which have *Mexican hat* kind of interactions. These interactions allow clusters of neurons to win the competition rather than one

single neuron. Then the weights of winning neurons are adjusted to bring about a better response to the current input. Iterative application of the competition – adaptation process to a sequence of input patterns eventually results in weights that specify clusters of network neurons that are topologically close, *being sensitive to clusters of inputs that are physically close in the input space*. The spatial location of a neuron in the array corresponds to a specific domain of inputs. In other words, we say that the map *preserves the topology of the input*. To generate the self-organizing map we require:

- *input neurons be exposed to a sufficient number of different inputs;*
- *for a given input, only the winning neuron and its neighbors adapt their connections;*
- *a similar weight update procedure be employed on many adjacent neurons which comprise topologically related subsets; and*
- *the resulting adjustment be such that it enhances the responses to the same or to a similar input that occurs subsequently.*

We now proceed to formalize the algorithm. Assume that the input-output data $\mathbf{X}^* = \{\mathbf{x}_k^* \in \mathfrak{R}^m, k = 1, 2, \dots, n\}$ is presented to a $(i \times j)$ field of neurons as shown in Figs. 5.1 and 5.2. Due to planar nature of the field, each neuron will be identified by the double row-column index ij . The ij^{th} neuron has an incoming weight vector $\mathbf{w}_{ij} = [w_{ij1}, w_{ij2}, \dots, w_{ijm}] \subset \mathfrak{R}^m$. The first step is to find the best matching weight vector $\mathbf{w}_{ij}, \forall i, j$ for the present input, and then to identify a neighborhood h_{ij} around the winning neuron c_{ij} .

One can find the best matching weight vector by two ways:

- Comparing the inner product $(\mathbf{x}_k^{*T} \mathbf{w}_{ij})$ of the impinging input \mathbf{x}_k^* with each weight vector $\mathbf{w}_{ij}, \forall i, j$. The winning neuron is the one that has the largest inner product.
- Equivalently, with normalized weight vectors we have seen that the maximum inner product criterion reduces to a minimum Euclidean distance criterion: the winning neuron is the one that minimizes the distance $\|\mathbf{x}_k^* - \mathbf{w}_{ij}\| \forall i, j$.

Kohonen suggested using the latter since it is more general and applies to natural signals in metric vector spaces. Therefore the winning neuron (c_{ij}) corresponding to the input vector \mathbf{x}_k^* is determined by

$$c_{ij} = \arg \min_{i,j} \{ \|\mathbf{x}_k^* - \mathbf{w}_{ij}\| \}$$

We define the topological neighborhood h_{ij} in a region surrounding the winning neuron with index ij . The shape of this neighborhood might be either rectangular or hexagonal, and the width of the region around the winning neuron c_{ij} is specified by a radius r measured discretely in terms of the number of neurons. The width of the neighborhood is a function of time: as epochs of training elapse, the neighborhood shrinks as shown in Fig. 5.3.

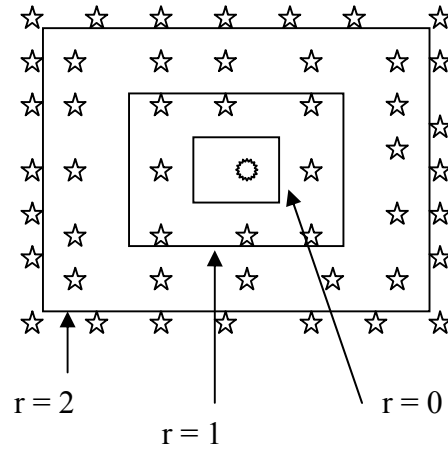


Figure 5.3: Neighborhood scheme for SOM (rectangular).

For time step t , let $\mathbf{x}_k^*(t)$ be the present input vector and has the weight vector $\mathbf{w}_{ij}(t)$, then, for time step $t+1$, the weight of the ij^{th} neuron is updated by using the following equation:

$$\mathbf{w}_{ij}(t+1) = \mathbf{w}_{ij}(t) + \eta(t)h_{ij}(t)[\mathbf{x}_k^*(t) - \mathbf{w}_{ij}(t)] \quad (5.2)$$

Thus, $\mathbf{w}_{ij}(t)$ is updated by adding a term $\eta(t)h_{ij}(t)[\mathbf{x}_k^*(t) - \mathbf{w}_{ij}(t)]$, which is proportional to $[\mathbf{x}_k^*(t) - \mathbf{w}_{ij}(t)]$, the difference between $\mathbf{w}_{ij}(t)$ and present input $\mathbf{x}_k^*(t)$. Here, $\eta(t) = \eta_0 \exp(-t/\tau_1)$ is the learning rate parameter in the range $0 < \eta(t) < 1$, which decreases monotonically with time and controls the rate of convergence depending on the value of time constant (τ_1). $h_{ij}(t)$ determines the effect that the difference $[\mathbf{x}_k^*(t) - \mathbf{w}_{ij}(t)]$ will have and is chosen so that

- (i) *it diminishes with time.*
- (ii) *typically h_{ij} is chosen as Gaussian function as shown in Fig. 5.4.*

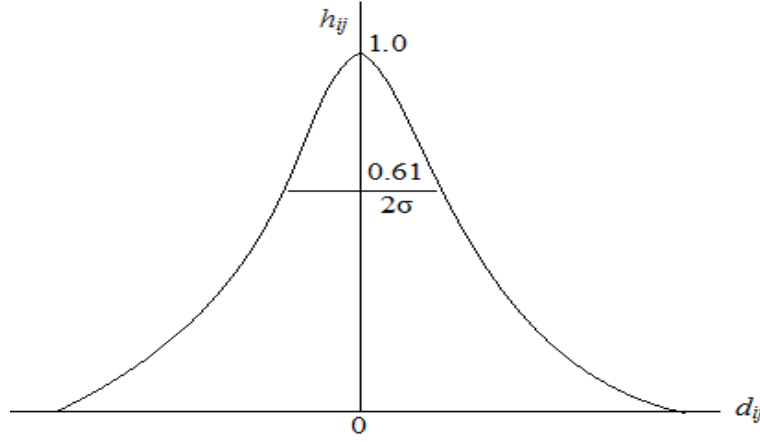


Figure 5.4: Gaussian neighborhood function.

$$h_{ij}(t) = \exp\left(-\frac{d_{ij}^2}{2\sigma^2(t)}\right) \quad (5.3)$$

For cooperation among neighboring neurons to hold, it is necessary that topological neighborhood h_{ij} as described in *equation 5.3*, be dependent on the lateral distance d_{ij} between the winning neuron c_{ij} and the excited neuron around c_{ij} in the output space, rather than some distance measure in the original input space. In the case of two-dimensional lattice, d_{ij} is defined by

$$d_{ij}^2 = \|\mathbf{r}_c - \mathbf{r}_{ij}\|^2$$

where the discrete vector \mathbf{r}_{ij} defines the position of excited neuron around c_{ij} and \mathbf{r}_c defines the position of the winning neuron c_{ij} , both of which are measured in the discrete output space. In *equation 5.3*, $\sigma(t)$ is defined by

$$\sigma(t) = \sigma_0 \exp(-t/\tau_2),$$

is the typical Gaussian variance parameter with time constant τ_2 that controls the width of the neighborhood, *i.e.*, a small σ will yield a small neighborhood, while a large σ will yield a wide neighborhood as shown in Fig. 5.4. Table below presents an operational summary of the self-organizing map algorithm.

Operational Summary of the SOM Algorithm

Given	For time step t , let $\mathbf{x}_k^*(t)$ be the current object
Initialize	<ul style="list-style-type: none"> ✓ The ij^{th} neuron with weight vector $\mathbf{w}_{ij} = [w_{ij1}, w_{ij2}, \dots, w_{ijm}] \subset \mathfrak{R}^m \forall i, j$ ✓ Value of neighborhood function $h_{ij}(t)$ ✓ Learning rate $\eta(t)$
Iterate	<p style="text-align: center;">Repeat</p> <p style="text-align: center;">{</p> <ul style="list-style-type: none"> ✚ Selection: pick dataset $\mathbf{x}_k^* \in \mathfrak{R}^m$ ✚ Similarity matching: Find the winning neuron c_{ij}, $c_{ij} = \arg \min_{i,j} \{d_{ij}(\mathbf{x}_k^*, \mathbf{w}_{ij})\}$ <p style="text-align: center;">where Euclidean distance $d_{ij}(\mathbf{x}_k^*, \mathbf{w}_{ij}) = \sqrt{\sum_{l=1}^m (x_{kl} - w_{ijl})^2}$</p> ✚ Adaptation: Update weights, $\mathbf{w}_{ij}(t+1) = \mathbf{w}_{ij}(t) + \eta(t)h_{ij}(t)[\mathbf{x}_k^*(t) - \mathbf{w}_{ij}(t)]$ $\mathbf{w}_{ij}(t) = \mathbf{w}_{ij}(t+1)$ ✚ Update $h_{ij}(t), \eta(t)$ <p style="text-align: center;">}</p> <p style="text-align: center;">until convergence criterion is satisfied</p>

In general, the learning rate $\eta(t)$ in the beginning should be close to unity. During this initial period, a general topological ordering of weight vectors takes place. Apart from careful adjustment of $\eta(t)$, the width of the neighborhood $\sigma(t)$ and its rate of contraction should be given due consideration in order for the network to converge. The learning scheme continues until sufficient number of iterations is completed or specified convergence criterion is satisfied [187, 188].

5.3 Rule extraction from clusters

In this section, first we review some existing techniques of rule-extraction before we explain our SOM-based rule extraction scheme. *Sugeno* and *Yasukawa* [14] was possibly the first to use clustering to determine an initial structure of the system. They used Fuzzy C-Means (FCM) algorithm [186] and clustered the output domain. The membership values of the input clusters were obtained by projecting the membership values of the extracted cluster on the X axis. The number of clusters, c , (*i.e.*, the number of rules) is determined by minimizing the following validity function suggested by *Fukuyama* and *Sugeno* [189]:

$$S(c) = \sum_{k=1}^n \sum_{i=1}^c (\mu_{ik})^m \left(\|\mathbf{y}_k - \mathbf{v}_i\|^2 - \|\mathbf{v}_i - \bar{\mathbf{y}}\|^2 \right)$$

Here, n = total number of data points to be clustered, c = number of clusters, μ_{ik} = membership value of the k^{th} data point, \mathbf{y}_k , to the i^{th} cluster, m = the fuzzy exponent used in the FCM algorithm, $\mathbf{v}_i = i^{th}$ cluster center and $\bar{\mathbf{y}}$ = the grand mean vector of all data points. The number of clusters is determined minimizing $S(c)$ with respect to c . The process starts with $c = 2$ and is continued up to some maximum $c = c_{max}$. For each c , the FCM centroids \mathbf{V} and the partition matrix \mathbf{U} are first obtained and then used them to compute $S(c)$. The value $c = c'$ at which $S(c)$ attains the minimum value is taken as the right number of rules.

Let the c clusters obtained from Y by the FCM be denoted by C_1, C_2, \dots, C_c and the associated set of centroids be $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_c\}$. Let μ_{ik} be the membership of \mathbf{y}_k to the i^{th} output cluster. In order to define the MFs on the input variables the following strategy is followed. The point \mathbf{y}_k is associated with \mathbf{x}_k . First \mathbf{x}_k is assigned the membership value of μ_{ik} . If there are several

$\mathbf{x}_k^* = \begin{pmatrix} \mathbf{x}_k \\ \mathbf{y}_k \end{pmatrix}$ such that the \mathbf{x}_k parts are the same but the \mathbf{y}_k parts are different, then such conflict can

be resolved. With this scheme each distinct $\mathbf{x}_k \in \mathfrak{R}^p$ will get a unique membership value. Now to define the MFs on each input feature, say the j^{th} feature, they used **max** as the aggregation operator O ; *i.e.*, $\mu(x' = x_{kj}) = \max_i \{\mu(\mathbf{x}_i) / x_{ij} = x'\}$. In this way, from each of the c clusters we can get a MF (and hence a fuzzy set) on each of the feature (linguistic variable).

Sugeno and Yasukawa [14] approximated these clusters by trapezoidal fuzzy sets. They used a heuristic method to adjust the parameters of different trapezoidal MFs with a view to achieving a set of parameters which provides the best model in terms of mean square error of the output. Their algorithm also provides a step for selection of input variables using a heuristic algorithm. To get a good subset of input variables, they use a regularity criterion, *RC*.

For a given I/O pair (x_i, y_i) with $x_i \in X, y_i \in Y$, where $i = 1, 2, \dots, n$, *Sin and de Figueiredo* [92] used FCM algorithm for clustering and suggested to use Xie-Beni index [93] for selecting the number of clusters. Each cluster obtained from X^* is presented by a TS-type rule. Let the membership and consequent functions for the i^{th} rule (cluster) be μ_i and u_i respectively. Here μ_i is not any fixed membership value, but is the FCM membership formula defined in terms of the i^{th} centroid \mathbf{v}_i^* . The consequent functions are estimated by minimizing the objective functions

$$E_k = \sum_{i=1}^n \mu_k \left(\mathbf{x}_i^* = \begin{pmatrix} \mathbf{x}_i \\ \mathbf{y}_i \end{pmatrix} \right) (u_k(\mathbf{x}_i) - y_i)^2; k=1,2,\dots,c.$$

For a given $\mathbf{x}_k \in \mathfrak{R}^p$, the firing strength, α_i , of the i^{th} rule is computed as $\alpha_i = \mu_k \left(\mathbf{x}^* = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} = \mathbf{u}_i(\mathbf{x}) \end{pmatrix} \right)$. This computation is a two step process. First they compute the output of the i^{th} rule as $\mathbf{y} = \mathbf{u}_i(\mathbf{x})$. This \mathbf{y}_i is then augmented to \mathbf{x} and the augmented vector is used to find the firing strength of the i^{th} rule. Thus, each rule uses a different $\left(\mathbf{x}^* = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} = \mathbf{u}_i(\mathbf{x}) \end{pmatrix} \right)$ to get the firing strength. The \mathbf{y}_i values are then combined, as in the TS model, to get the final

$$\text{predicted value } \mathbf{y} = \frac{\sum_{i=1}^c \alpha_i \mathbf{u}_i(\mathbf{x})}{\sum_{i=1}^c \alpha_i}.$$

Yager and Feliv [87] used the mountain clustering method (MCM) on X^* . The optimal number of clusters is chosen based on a user defined threshold on the mountain potential. Consider a multi-input single-output system with input $\mathbf{x}_k \in \mathfrak{R}^p$ and output $\mathbf{y} \in \mathfrak{R}$. A cluster

centroid $\mathbf{v}_i^* = \begin{pmatrix} \mathbf{v}_i^x \in \mathfrak{R}^p \\ \mathbf{v}_i^y \in \mathfrak{R} \end{pmatrix}$, $i = 1, 2 \dots c$ is then converted into a fuzzy rule of the form: *If \mathbf{x} is CLOSE to \mathbf{v}_i^x then \mathbf{y} is CLOSE to \mathbf{v}_i^y* . Writing $\mathbf{A}_i = \text{CLOSE to } \mathbf{v}_i^x$ and $\mathbf{B}_i = \text{CLOSE to } \mathbf{v}_i^y$, we get a set of c rules: *If \mathbf{x} is \mathbf{A}_i then \mathbf{y} is \mathbf{B}_i* ; $i = 1, 2 \dots c$. Each antecedent clause, '*If \mathbf{x} is*' then translated into p atomic clauses, \mathbf{x}_k is \mathbf{A}_{ik} ; $k = 1, \dots, p$, connected by AND. They used Gaussian type MF to model \mathbf{A}_{ij} and \mathbf{B}_{ij} :

$$\mathbf{A}_{ij}(\mathbf{x}_j) = \exp\left(\frac{1}{2\sigma_{ij}^2}(\mathbf{x}_j - \mathbf{v}_{ij}^x)^2\right) \text{ and } \mathbf{B}_i(\mathbf{y}) = \exp\left(\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{v}_i^y)^2\right).$$

Here σ_{ij} is the spread of the j^{th} antecedent fuzzy set of the i^{th} rule and σ is the spread of the consequents. They used the height method of defuzzification and *Mamdani-Assilian* (MA) model [40]. This can equivalently be viewed as the *Takagi-Sugeno* (TS) model of zero order [17]. The initial estimates of the parameters σ_{ij} are taken as $\sqrt{\frac{1}{2\beta}}$, where β is a parameter used in the mountain function for clustering. All parameters of the system $(\mathbf{v}_{ij}^x, \mathbf{v}_i^y, \sigma_{ij})$ are then further tuned with gradient descent to minimize the total square error. Although, MCM determines the number of clusters automatically, it is strongly influenced by the parameters of the mountain potential function and the threshold value used to stop the clustering process. Therefore, in absence of an appropriate choice of these parameters the number of rules may be over-determined.

Chiu [88] proposed a modified method of MCM known as *Subtractive Clustering Method* (SCM) for a group of n_i training data points $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\} \subset \mathfrak{R}^p$ representing a particular class. SCM considers each data point as a potential cluster centroid \mathbf{x}_i with potential $P_i = \sum_{j=1}^{n_i} \exp(-\alpha \|\mathbf{x}_i - \mathbf{x}_j\|^2)$. After calculating the potential of all data points, the point with the highest potential is selected as first cluster center. The potential of each data point is then revised by $P_i \Leftarrow P_i - P_1^* \exp(-\beta \|\mathbf{x}_i - \mathbf{x}_1^*\|^2)$ where P_1^* and \mathbf{x}_1^* are the potential value and location of the first cluster. After revision, the data point with the highest remaining potential is selected as second cluster center. For k^{th} such operations, the potential of each data point was

revised by $P_i \leftarrow P_i - P_k^* \exp\left(-\beta \|\mathbf{x}_i - \mathbf{x}_k^*\|^2\right)$ where, P_k^* and \mathbf{x}_k^* are the potential and location of the k^{th} cluster. If the i^{th} cluster center \mathbf{x}_i^* is found in the group of data for class $c1$, then the corresponding rule can be written as R_i : *If \mathbf{x} is CLOSE to \mathbf{x}_i^* then class is $c1$* . The fuzzy set CLOSE to \mathbf{x}_i^* is modeled by a Gaussian type MF. The rule R_i can be written in a more familiar form: *If x_1 is A_{i1}^k and x_2 is A_{i2}^k andthen the class is k* , where x_j is j^{th} input feature and A_{ij}^k is the MF in the i^{th} rule associated with the j^{th} input feature for class k . The MF A_{ij} is defined by

$$A_{ij}(\mathbf{x}_j) = \exp\left\{-\frac{1}{2}\left(\frac{\mathbf{x}_j - \mathbf{x}_{ij}^*}{\sigma_{ij}}\right)^2\right\} \text{ where } \mathbf{x}_{ij}^* \text{ is the } j^{th} \text{ element of } \mathbf{x}_i^*, \text{ and } \sigma_{ij}^2 = \frac{1}{2\alpha}. \text{ After obtaining}$$

the initial rule set, gradient descent technique is used to tune \mathbf{x}_{ij}^* and σ_{ij} with a view to minimizing the classification error measure $E = \frac{1}{2}(1 - \mu_{c,\max} + \mu_{-c,\max})^2$; where $\mu_{c,\max}$ is the maximum degree of fulfillment among all rules that infer the correct class c and $\mu_{-c,\max}$ denotes the maximum firing strength of all rules that infer the class c . No explicit cluster validity index is used here. The number of clusters SCM settles to is dependent on the parameters of the mountain function.

Babuska and Kaymak [90] used clustering in the *input-output* space for TS modeling. The identification process starts with clustering \mathbf{X}^* using *Gustafson-Kesel's* [96] fuzzy c-means (GKFCM) algorithm with a large value of c . Then the compatible cluster merging (CCM) criterion is used to merge compatible clusters. After this, GKFCM is again run with the reduced number of clusters. The process is repeated until no more clusters can be merged. The merging technique requires a user specified threshold. The fuzzy partition thus obtained is projected on to the input axes to generate MFs for the antecedent variables and hence the rules. Finally the consequent parameters are estimated using least square technique.

Delgado et al. [100] presented several methods for fuzzy modeling that use clustering. The first method clusters \mathbf{X}^* using FCM. Consider a p -input $\left(\mathbf{x} = (x_1, \dots, x_p)^T\right)$ and single output (\mathbf{y})

system. Let the centroids obtained by clustering be $\mathbf{v}_i^* = \begin{pmatrix} \mathbf{v}_i^x \\ \mathbf{v}_i^y \end{pmatrix} \in \mathfrak{R}^{p+1}; \mathbf{v}_i^x \in \mathfrak{R}^p, \mathbf{v}_i^y \in \mathfrak{R}; i = 1, \dots, c$.

Each such cluster is transformed into a rule of the form R_k : *If \mathbf{x} is A_k then \mathbf{y} is B_k* ; $k = 1, \dots, c$ where the membership values of fuzzy sets A_k and B_k are defined by the FCM membership formula [186] with $v_i = v_k^x$ and $v_i = v_k^y$, respectively. For the TS model the k^{th} rule takes the form R_k : *If \mathbf{x} is A_k then \mathbf{y} is v_i^y* . The firing strength of the k^{th} rule is computed using the FCM formula defined in terms of $v_i^x \in \mathfrak{R}^p$. *Sin and de Figueiredo* [92] also used the same multidimensional FCM membership functions for computing the firing strength but they used the FCM formula defined using $v_k^* \in \mathfrak{R}^{p+q}$. It requires getting an approximate value of the output y before the firing strength of any rule can be computed.

There are several other clustering methods that are used for rule generation. For example, *Huang and Chang* [190] used clustering for obtaining qualitative rules of the form *If x is A_i then y is B_j* , where A_i and B_j represent fuzzy sets of the type ‘CLOSE TO x ’ and ‘CLOSE TO y ’. *Wong and Chen* [103] used switching regression model for extracting of TS rules. *Chak et al.* [191] developed a fuzzy neural network hybrid algorithm with a hierarchical space partitioning method to generate rules for a MIMO system. They used the normalized root mean square error (NRMSE) as the performance index. The performance index should be less than some pre-assumed value.

Cho and Wang [192] proposed a neuro-fuzzy system called Radial Basis Function Network (RBFN) based Adaptive Fuzzy Systems (AFS) to learn fuzzy rules from input-output data. Three different architectures of RBF based AFS have been proposed to accommodate both TS and MA type models. They proposed a Hierarchally Structural Self-Organizing Learning (HiSSOL) method to train the RBF based AFS. Initially, the number of rules is unknown- using HiSSOL; it is determined by incrementally recruiting the RBF unit. The incremental addition of nodes in HiSSOL is based on the control of effective radius of an individual basis function and adjustment of its mean and variance vectors. The learning algorithm updates the network parameters using gradient decent method.

Kim et al. [193] suggested a new fuzzy modeling algorithm using fuzzy c-regression model (FCRM) clustering in the case of a MISO system and used coarse tuning to determine approximately the consequent parameters. For a set of n sample data $(\mathbf{x}_k, \mathbf{y}_k)$, for $\forall k$, they clustered the input data using FCM and used them as initial parameters for FCRM. For fine tuning they used gradient descent algorithm.

Wong et al. [194] proposed a technique to extract fuzzy rules directly from input-output pairs. They used a self-organizing neural network and association rules to construct the fuzzy rule base. The self-organizing neural network was first used to classify the output data by realizing the probability distribution of the output space. Association rules are then used to find the relationships between the input space and the output classification, which are subsequently converted to fuzzy rules. For a given data set with k inputs, the given input-output data pairs with n patterns are: $(x_1^i, x_2^i, \dots, x_k^i; y^i); i = 1, 2, \dots, n$. Then from the association rules and the membership functions calculated, they constructed the relation: $A_1(x_1^i) = A_2(x_2^i) = B(y^i)$.

From the above relation, a fuzzy rule can be constructed as:

If x_1 is A_1 and x_2 is A_2 , then y is B .

Barreto and Araújo [123] introduced a general modeling technique, called *vector-quantized temporal associative memory* (VQTAM), which used SOM as an alternative to multilayer perceptron (MLP) and radial basis function (RBF) neural models for dynamical system identification and control. They demonstrated that the estimation errors decrease as the SOM training proceeds, allowing the VQTAM scheme to be understood as a self-supervised gradient-based error reduction method. The model accuracy is evaluated through root mean square error. The performance of the proposed approach is evaluated on a variety of complex tasks, namely: (i) time series prediction; (ii) identification of SISO/MIMO systems; and (iii) nonlinear predictive control. For all tasks, the simulation results produced by the SOM are as accurate as those produced by the MLP network and better than those produced by the RBF network. The SOM has also shown to be less sensitive to weight initialization than MLP networks.

Moreno et al. [195] presents clustering techniques (K-means, Fuzzy C-means, and Subtractive) applied on specific databases and extracted production rules using *Mamdani* as well as *Takagi-*

Sugeno-Kang fuzzy logic inference systems. *Chen et al.* [181] clustered the training data from the j^{th} class, $X_j \subseteq \mathfrak{R}^p$ into n_j clusters. They used the K-means clustering algorithm. Since their main objective was to investigate the simultaneous feature selection and rule extraction, therefore they did not address the issue of choice of optimal number of rules for a class. Instead, they assumed a fixed number of rules for a class and demonstrate the effectiveness. They converted each such cluster into a fuzzy rule. For example, if the center of the i^{th} cluster in X_j is $\mathbf{v}_j \in \mathfrak{R}^p$, then this cluster is converted into the rule R_i : *if x is CLOSE TO \mathbf{v}_i , then the class is j* . The fuzzy set “CLOSE TO” is modeled by a multidimensional membership function such as $\mu_{\text{CLOSE TO } \mathbf{v}_i}(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{v}_i\|^2}{\sigma_i^2}\right)$. Such a multidimensional membership function is difficult to interpret (less readability) and may not perform always well, particularly when different features have considerably different variances. Thus, such a rule is expanded as a conjunction of p atomic clauses

R_i: If x_1 is CLOSE TO v_{i1} AND AND x_p is CLOSE TO v_{ip} , then class is j .

Liu et al. [196] introduced a new type of coherence MF to describe fuzzy concepts, which builds upon the theoretical findings of the Axiomatic Fuzzy Set theory. The proposed algorithm consists of three major steps: (a) generating fuzzy decision trees by assuming some level of specificity quantified in terms of threshold; (b) pruning the obtained rule-base; and (c) determining the optimal threshold resulting in a final tree. Each path starting from the root traversing down to a classification node (terminal node) is converted to a rule. The rules are directly extracted from the axiomatic fuzzy set decision tree. However, they may include redundant structures as well as poorly performing rules, and therefore they pruned the rule-base.

Gao et al. [197] proposed a fuzzy-based support vector machine (SVM) classification algorithm for blast furnace black-box models. *Hashimoto et al.* [122] approached an identification procedure that is divided into three steps, which are (i) Self-Organizing Map (SOM) based clustering of the regression vectors consisting of observed input and output signals, (ii) Local system identification by using genetic programming, and (iii) Model fusion of local models by fuzzy inference to provide the global model.

Yang and Bose [121] generated fuzzy membership function *via* SOM. Instead of two step procedure, they showed that it was possible to integrate the two-step procedure and generate the fuzzy membership function directly during the learning phase. A key step in the proposed technique is to combine the input feature vector $\mathbf{x}_n = [\mathbf{x}_{n1}, \mathbf{x}_{n2}, \dots, \mathbf{x}_{nd}]^T$ with the vector $\mathbf{y}_n = [\mathbf{y}_{n1}, \mathbf{y}_{n2}, \dots, \mathbf{y}_{nc}]^T$ coding the class labeling information. The dimensions of \mathbf{x}_n and \mathbf{y}_n are respectively, the number of input features d and the number of class labels c . That is, a new vector \mathbf{z}_n of dimension $c+d$ is constructed according to $\mathbf{z}_n = [\mathbf{x}_n \mathbf{y}_n]^T = [\mathbf{x}_n \mathbf{0}]^T + [\mathbf{0} \mathbf{y}_n]^T$. In the learning phase, the newly constructed \mathbf{z}_n will be the input feature vector to SOM. After the learning phase, the SOM can be considered as a membership generation network just like its counterpart, the feedforward multilayer neural network trained with a supervised learning algorithm. However, in the retrieving phase, it is not as straightforward as in the case of the feedforward multilayer neural network. In the retrieving phase, the input feature vector is only \mathbf{x}_n . Therefore, the input feature vector will find the best matching neuron q by considering only the weight sub-vector $\mathbf{w}_{jd} = [\mathbf{w}_{j1}, \dots, \mathbf{w}_{jd}]^T$ related to input features, that is, $q(\mathbf{x}_n) = \min_j \|\mathbf{x}_n - \mathbf{w}_{jd}\|$. After finding the winning neuron q , the output of SOM is the weight sub-vector $\mathbf{w}_{qc} = [\mathbf{w}_{q(d+1)}, \dots, \mathbf{w}_{q(d+c)}]^T$, associated with the labeling information. Also, it is the fuzzy membership generated by SOM.

Among the different clustering models, the SOM has been selected since it operates in an unsupervised manner, thus minimizing the requirement for human guidance in labeling the nodes following training. In SOM, the projected data preserves the topological relationship of the original data; therefore, this ordered grid could be used as a convenient visualization surface for showing various features of the training data. In our proposed scheme, SOM is first used to cluster the data and after that, some computational steps have been suggested to extract fuzzy rules from each cluster.

It's important to select a proper clustering domain. There are choices like: individual clustering of X or Y and clustering of X^* . Here, clustering of X^* have been used to generate the initial rule-base because it is better to consider X^* (the input-output data) in place of individual X or Y . In

individual clustering, it is difficult to establish correspondence between clusters of X and clusters of Y .

Clustering algorithm generate a set of centroids

$$V^* = \left\{ \mathbf{v}_i^* = \begin{pmatrix} \mathbf{v}_i^x \in \mathfrak{R}^p \\ \mathbf{v}_i^y \in \mathfrak{R}^q \end{pmatrix}; i = 1, \dots, c \right\}.$$

These clustering results can be used to extract *if-then* rules. If there is a cluster in the input space with centroid \mathbf{v}_i^x and we assume a smooth relationship between the input and output, then the points in the output space corresponding to the input cluster are likely to form a cluster around \mathbf{v}_i^y . This local input-output relationship can be represented by a fuzzy *if-then* rule of the form,

$$R_i: \text{If } x \text{ is } \mathbf{A}_i \text{ then } y \text{ is } \mathbf{B}_i,$$

where the membership functions (MFs) \mathbf{A}_i and \mathbf{B}_i are defined using \mathbf{v}_i^x and \mathbf{v}_i^y respectively [104]. The number of clusters is usually predefined, but it can also be a part of the error function [120]. Refining of the extracted rule-based model may be done through various parameter adjustments or optimization schemes.

5.3.1 The proposed rule extraction scheme

In this study, we propose a novel approach for fuzzy rule-based system identification that automatically generates fuzzy rules as well as the MFs for the antecedent and consequent parts of the rule [198]. A Self-Organizing Map (SOM) based clustering technique is used for structure identification and prototype generation with initial MFs picked up from the clustering results. This identified prototype model is fine tuned using gradient descent technique. The proposed method has a provision of selecting essential number of rules from any system, if its input-output data are available. Outline of the proposed method is presented in Fig.5.5. Next we describe the computational steps of the different layers associated with the proposed fuzzy rule-based modeling of an unknown system having its input-output data.

Layer 1:

The computational steps to identify the centroid by finding the winning neuron (c_{ij}) using the self-organizing map (SOM) algorithm have already been elaborated in **section 5.2.2**. By this

process, weight vectors (w_{ij}) move towards the input vectors (x_k^*) and tend to follow the distribution of input vectors [188]. The Euclidean distance $d_{ij}(x_k^*, w_{ij})$ between the input vector x_k^* and the updated weight w_{ij} in the network is found out from the equation:

$$d_{ij} = d_{ij}(x_k^*, w_{ij}) = \sqrt{\sum_{l=1}^m (x_{kl} - w_{ijl})^2}$$

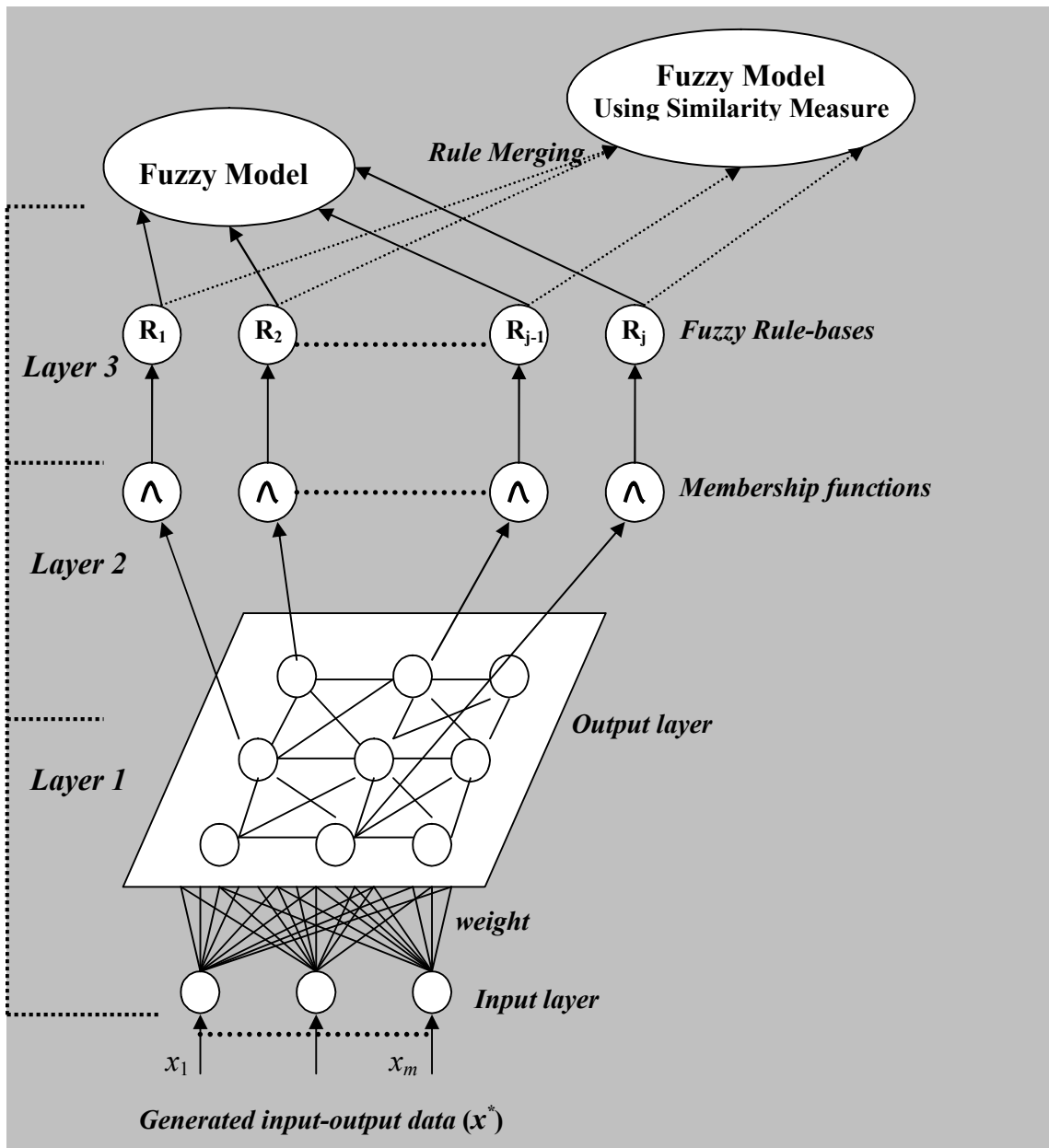
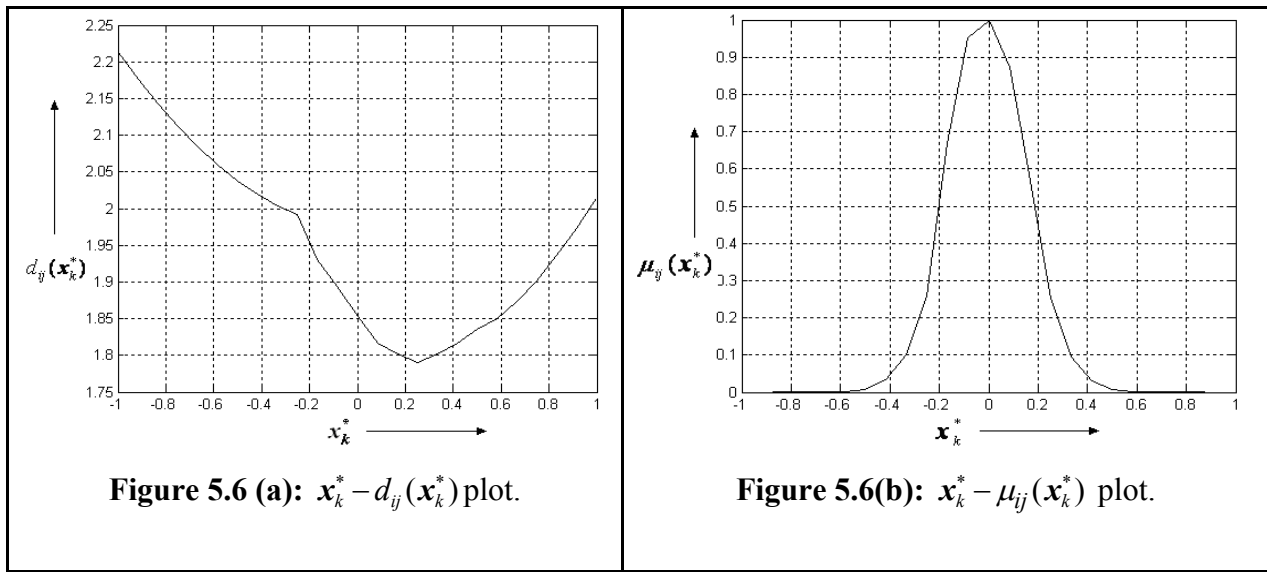


Figure 5.5: Outline flow diagram of the proposed scheme

Layer 2:

In order to get fuzzy model from the clustered data we convert each node into a fuzzy rule. This needs appropriate MFs of each input-output feature [104, 105]. The most important information that can be extracted from the clustered data is the location of their centroids and the distribution of the data around them.

The $x_k^* - d_{ij}(x_k^*)$ plot shown in Fig. 5.6(a) does not give direct information about the shape of the MFs. Moreover, from this plot we are unable to get the highest membership value of a MF. However, the clustering information may be used to define approximate but simple MFs (symmetric triangle or Gaussian with equal base-width) for the extracted / identified fuzzy model. In order to define such MFs we propose the following steps:



- ✚ After clustering the input-output data using SOM algorithm, we propose to form Gaussian type MFs from the clustered information. The multidimensional MF can be modeled from the relation,

$$\mu_{ij}(x_k^*) = \exp\left[-\left(\frac{d_{ij}-c_{ij}}{\sigma}\right)^2\right].$$

Where, μ_{ij} is the membership value of x_k^* in the ij^{th} node, and

$$c_{ij} = \min_{i,j} \{d_{ij}\}, \text{ where } d_{ij} = d_{ij}(\mathbf{x}_k^*, \mathbf{w}_{ij}) = \sqrt{\sum_{l=1}^m (x_{kl} - w_{ijl})^2}.$$

Therefore, at ij^{th} node,


when $d_{ij} = c_{ij}$, we get the peak of the ij^{th} MF (*i.e.*, $\mu_{ij} = 1$).

A typical $\mathbf{x}_k^* - \mu_{ij}(\mathbf{x}_k^*)$ plot is shown in Fig. 5.6(b). After we obtain the peak of different nodes, the next step is to get an initial estimate of the base width (σ) of each MF, so that the rule-base becomes complete (*i.e.*, every \mathbf{X}^* fires at least one rule). Also, observe that the spread of the base-width (σ) will depend on the number of extracted rules and range of \mathbf{X}^* .

- ✚ Assign the MF $\mu_{ij}(\mathbf{x}_k^*)$ for each of the individual data to form data pairs in the form of $\mathbf{x}_k^* - \mu_{ij}(\mathbf{x}_k^*)$ in all the respective nodes.
- ✚ After assigning MF, it is observed that there are several \mathbf{x}_k^* , where \mathbf{x}_k (input) parts are the same but the \mathbf{y}_k (output) parts are different. This conflict can be resolved by applying **max** as the aggregation operator O ; then say for the j^{th} node it will be,

$$\mu(\mathbf{x}' = \mathbf{x}_{kj}) = O_i \{ \mu(\mathbf{x}_k^*) / \mathbf{x}_{ij} = \mathbf{x}' \}$$

Membership Function Generation

<p>Computation al steps to MF generation</p> 	<ul style="list-style-type: none"> ❖ <i>Compute Euclidian distance:</i> $d_{ij} = d_{ij}(\mathbf{x}_k^*, \mathbf{w}_{ij}) = \sqrt{\sum_{l=1}^m (x_{kl} - w_{ijl})^2}$ <ul style="list-style-type: none"> ❖ <i>Model multi-dimensional MF:</i> $\mu_{ij}(\mathbf{x}_k^*) = \exp\left[-\frac{(d_{ij} - c_{ij})^2}{\sigma}\right];$ <p>$c_{ij} = \min_{i,j} \{d_{ij}\}$ and determine peak at $d_{ij} = c_{ij}$.</p>
---	---

	<ul style="list-style-type: none"> ❖ Assign MF: $\mu_{ij}(\mathbf{x}_k^*)$ ❖ Arrange MF in ascending order and form pairs $\mathbf{x}_k^* - \mu_{ij}(\mathbf{x}_k^*)$ ❖ In \mathbf{x}_k^*, if \mathbf{x}_k parts are same but \mathbf{y}_k parts are different, this conflict is resolved by applying max as the aggregation operator O, $\mu(\mathbf{x}' = \mathbf{x}_{kj}) = O_l \left\{ \mu(\mathbf{x}_k^*) / \mathbf{x}_{ij} = \mathbf{x}' \right\}$
--	---


Layer 3:

The above information can be used to extract fuzzy rules [179, 198]. Use of clustering results for fuzzy rule extraction is motivated by the fact that if there is a cluster in the input space with centroid c_{ij}^x and we assume a smooth relationship between the input and the output, then the points in the output space corresponding to the input cluster are likely to form a cluster centroid c_{ij}^y . This local input-output relation can be represented by a fuzzy *if-then* rule of the form:

$$R_{ij}: \text{If } x \text{ is } \mu_{ij}(x) \text{ then } y \text{ is } \mu_{ij}(y).$$

Since we use *Height method of defuzzification* and consider symmetric MF [22, 104], peaks of the MFs for antecedent and consequent of the ij^{th} rule are computed as follows:

✚	<p>Peaks can be determined by taking weighted average of \mathbf{x}_k and their corresponding MF, $\mu_{ij}(\mathbf{x}_k)$. μ_{ij} is the membership of \mathbf{x}_k in the ij^{th} node. We denote these MFs by MF₁.</p>
✚	<p>However, occasionally throughout the spread, all the data might not get enough support. For efficient computing, it is wiser to ignore such data, which do not have adequate support for the node.</p> <p>Calculate peaks by taking weighted average of those data having membership values, $\mu_{ij}(\mathbf{x}_k) \geq 0.3$. We call such MFs as MF₂.</p>

	<p>Lastly, centroids (c_{ij}) as extracted by the SOM algorithm are taken as peak.</p> <p>This type of MFs is referred to as MF₃.</p>
---	--

After getting the peak, it is required to decide the base-width, so that the every input fires at least one rule. The choice of the base-width largely depends on the number of MFs for a particular linguistic variable. To check our initial guess as well as to boost our confidence regarding rule completeness, we may go through the following steps:

StepI	<i>Uniformly quantize with high resolution, each domain of input-output variables within its range.</i>
StepII	<i>Calculate the firing strength for every quantized x_k. Existence of at least one rule with non-zero firing strength for every x_k, ensures completeness of the rule-base.</i>
StepIII	<i>Increase the width of all MFs by a small percentage (say 5%), if condition of completeness is not satisfied. Again the rule completeness is checked for this new base-width and this process will continue until the completeness is achieved.</i>

Our main objective is to describe or model the original system by extracting proper rules from the input-output data. For successful system identification, tuning or parameter adjustment of the prototype model is very important. In the next section, the extracted rule-base model is tuned using *gradient decent tuning* algorithm [104].

5.3.2 Fine tuning of extracted fuzzy rule-base model

If x is the p -dimensional input variables and y is the output variable of a system, then j^{th} rule is represented by:

$$R_j : \text{if } x_1 \text{ is } \mu_j(x_1), x_2 \text{ is } \mu_j(x_2), \dots, \text{and } x_p \text{ is } \mu_j(x_p) \text{ then } y \text{ is } \mu_j(y);$$

where, $j = 1, 2, \dots, n$ and $i = 1, 2, \dots, p$.

The firing strength or MF (μ_j) of the antecedent part of j^{th} rule is computed as:

$$\mu_j = \prod_{i=1}^p \mu_j(x_i)$$

$$\text{for, } \mu_j(x_i) = 1 - \frac{2(x_i - a_{ji})}{b_{ji}}.$$

Where a_{ji} and b_{ji} represent the peak and corresponding support of a symmetric triangular MF in the universe of discourse and x_i is the crisp input. If output MF is a fuzzy singleton set defined by

real number y_j , then the final non-fuzzy output realized from the j^{th} rule is:
$$y_j = \frac{\sum_{j=1}^n \mu_j \cdot y_j}{\sum_{j=1}^n \mu_j}.$$

If different input variables are $x_1^r, x_2^r, \dots, x_p^r$, then for a desired control output of y^r , it is required to optimize the parameters (a_{ji}, b_{ji}, y_j) associated with the j^{th} rule.

The steepest descent algorithm [42, 70] is used to minimize the objective function E , given by *equation 5.4*.

$$E = \sum_{j=1}^n (y - y^r)^2 \tag{5.4}$$

The applied gradient decent algorithm always intends to decrease the value of the objective function (E) by updating the associated parameters a_{ji}, b_{ji}, y_j through *equation 5.5* to *5.7*.

$$a_{ji}(t+1) = a_{ji}(t) - \lambda_1 \frac{\partial E}{\partial a_{ji}} \tag{5.5}$$

$$b_{ji}(t+1) = b_{ji}(t) - \lambda_2 \frac{\partial E}{\partial b_{ji}} \tag{5.6}$$

$$y_j(t+1) = y_j(t) - \lambda_3 \frac{\partial E}{\partial y_j} \tag{5.7}$$

Where λ_1, λ_2 and λ_3 are respective learning coefficients for $a_{ji}(t), b_{ji}(t)$ and $y_j(t)$.

This process of optimization is continued till the change of error is suitably small or zero, as a result the modified values of peak, width and control output are obtained.

The identified rules after tuning may not always be same as the actual rules due to lack of knowledge of the investigated system. But the closeness can be verified by checking *mean square error* (MSE). One of the important applications of system identification is controller design. The controller works in tandem with the system and is designed to modify the response of the system to meet the overall specification. Here, we generate the input-output data X^* from a highly nonlinear self-tuning fuzzy PI controller (STFPIC) [22]. Our proposed scheme is illustrated to identify the fuzzy rules required for updating the gain factor (β) of STFPIC. Finally, the effectiveness of our proposed scheme is demonstrated in different linear, nonlinear and marginally stable systems.

5.4 Study of the gain surface of STFPIC

In this section, the utility of the proposed rule extraction scheme is demonstrated in a gain surface of self-tuning fuzzy PI controller (STFPIC), which is highly nonlinear in nature.

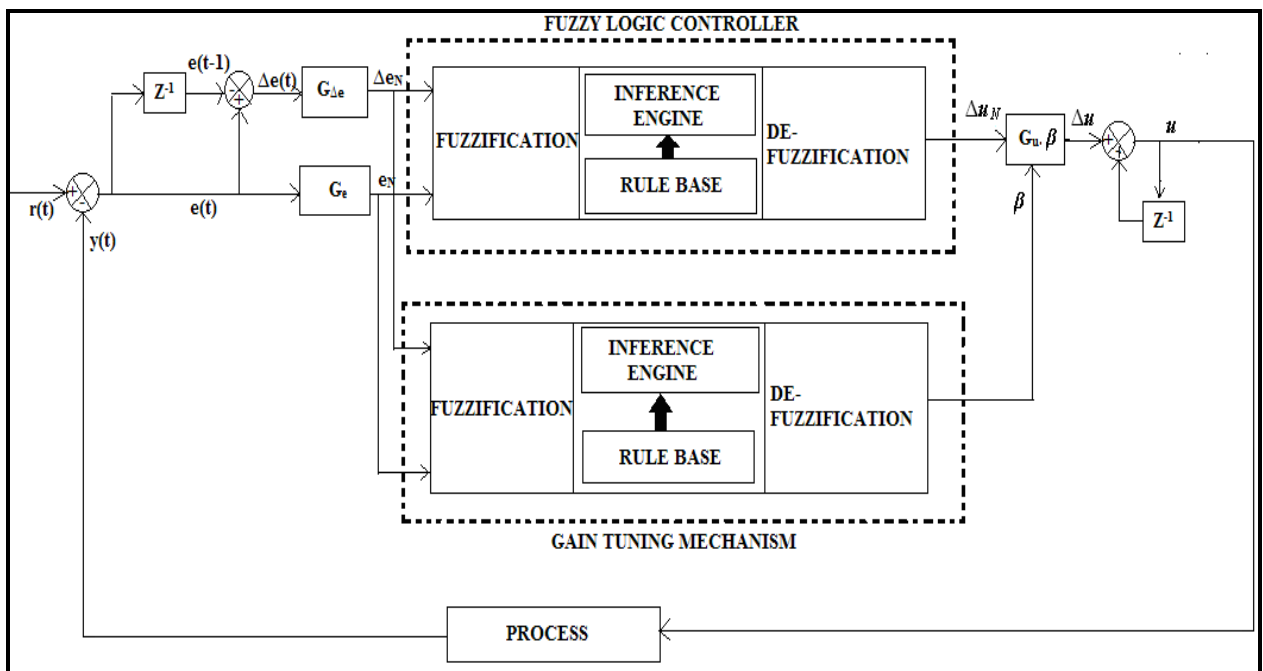


Figure 5.7: Block Diagram of STFPIC.

The block diagram of the STFPIC is shown in Fig. 5.7, where e_N (normalized error) and Δe_N (normalized change of error) are two inputs and u is the output of the controller, and β is the gain updating factor in this self-tuning mechanism [23]. The relationships between the scaling factors (G_e , $G_{\Delta e}$ and G_u), and the input-output variables of the STFPIC are as follows:

$$\begin{aligned} e_N &= G_e \cdot e, \\ \Delta e_N &= G_{\Delta e} \cdot \Delta e, \\ \text{and } \Delta u &= (\beta G_u) \cdot \Delta u_N \end{aligned}$$

The rule-bases for computing Δu_N and β are already shown in Table 2.3 and Table 2.4 respectively. Steps for rule extraction scheme of β using the proposed scheme are summarized below:

Rule extraction scheme steps	
Step 1	Generate input-output data.
Step 2	Cluster the data by SOM algorithm.
Step 3	Obtain centroids, find MFs and translate each cluster into a rule.
Step 4	Tune the MFs using gradient descent technique.

Mamdani type interfacing and Height method of defuzzification for generation of input-output data are used here. We have generated data of 625 triplets $\{e, \Delta e, \beta\}$ and $\{e, \Delta e, u\}$, when e and Δe are uniformly quantized using *equation 5.8* within their normalized domain $[-1, 1]$ as follows:

$$\forall i(0 \leq i \leq 24) : \begin{cases} e = -1 + i \times 0.0833 \\ \Delta e = -1 + i \times 0.0833 \end{cases} \quad (5.8)$$

The value of β (gain updating factor) and u (control output) for every $(e, \Delta e)$ pair is determined and the respective gain surface and control surface of STFPIC are plotted in Fig. 5.8(a) and Fig. 5.8(b).

Initially input-output data $\{e, \Delta e, \beta\}$ of the gain surface of STFPIC are clustered in a two dimensional (5×5) plane using self-organizing map. Then we extract MFs for e , Δe and β from

each of the 25 nodes according to the steps described in *section 5.3.1*. MFs thus extracted are shown in Fig 5.9.

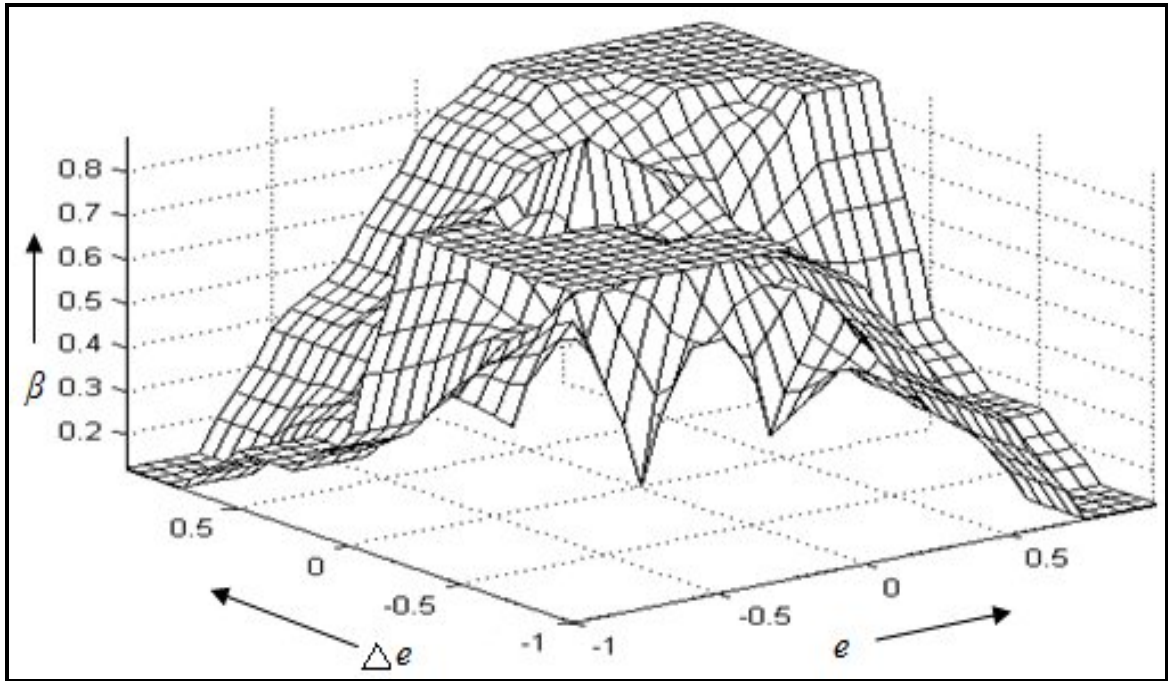


Figure 5.8(a): Gain surface of STFPIC with 49 rules.

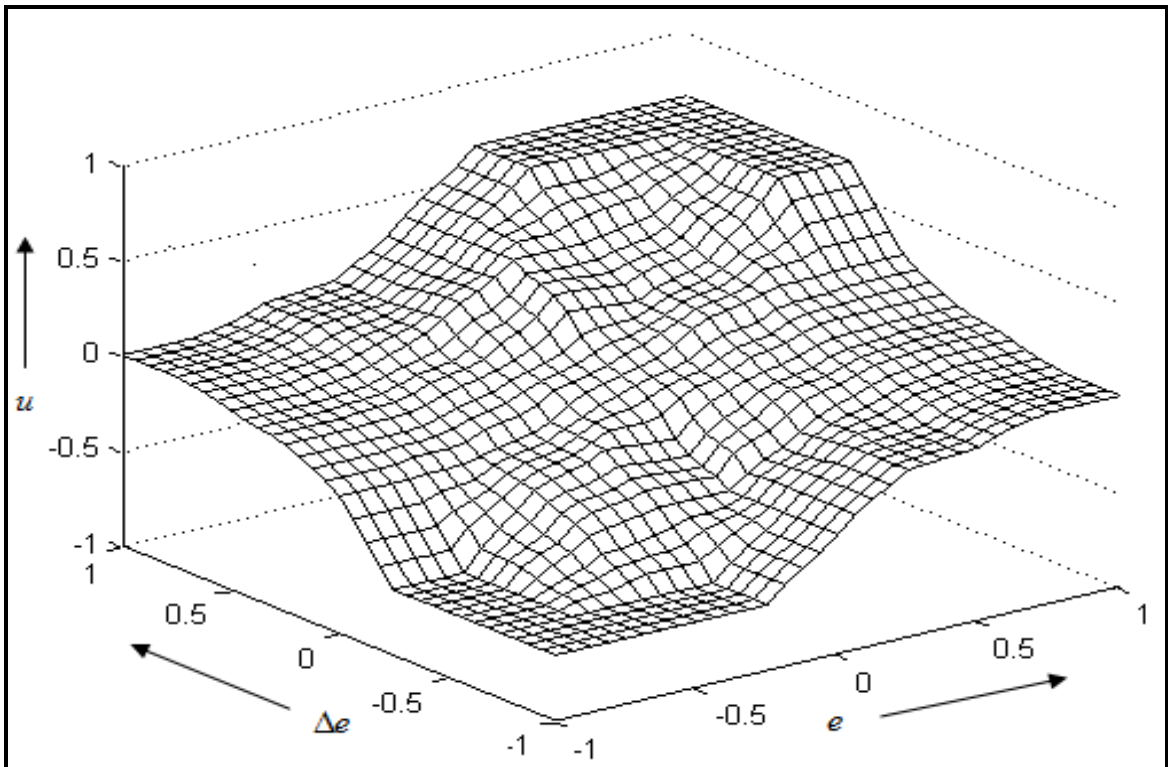
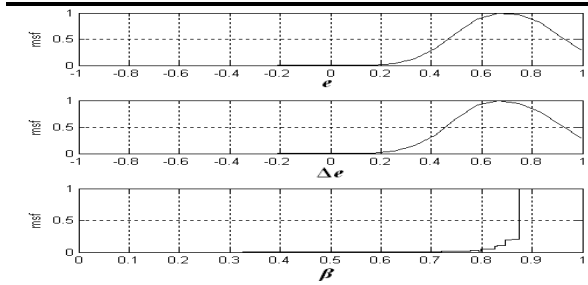
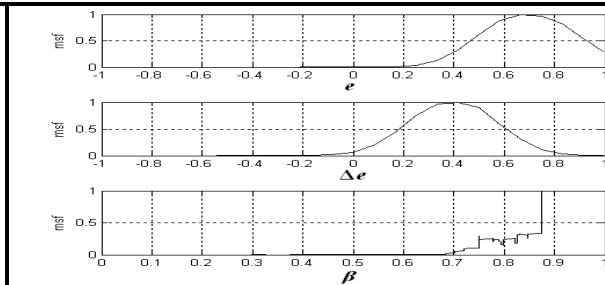


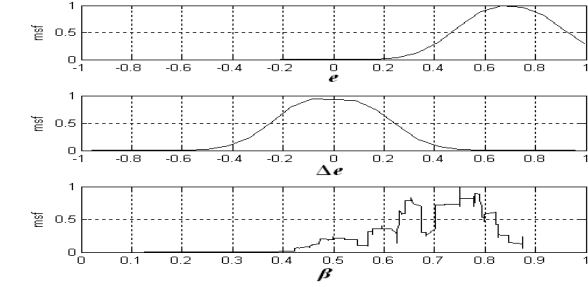
Figure 5.8(b): Control surface of STFPIC with 98 rules.



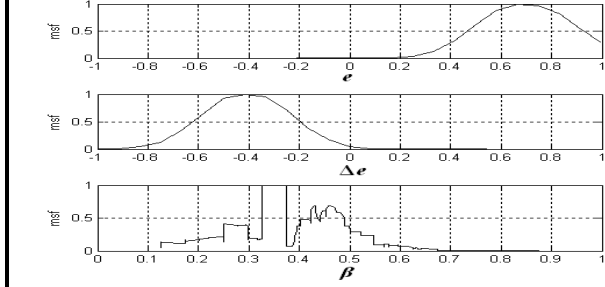
Node-11



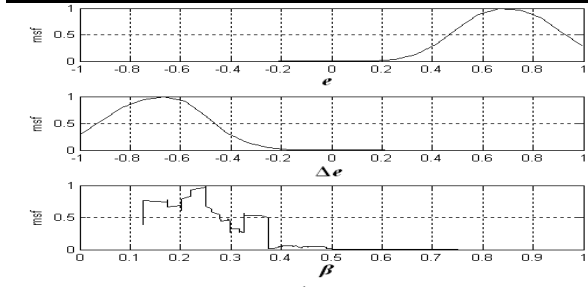
Node-12



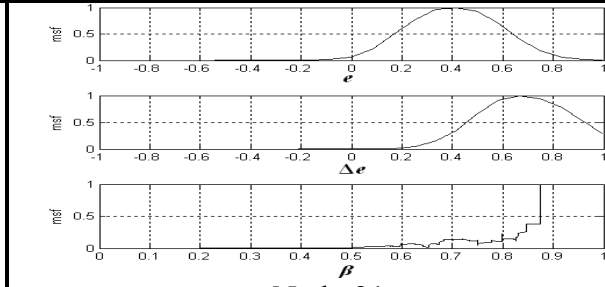
Node-13



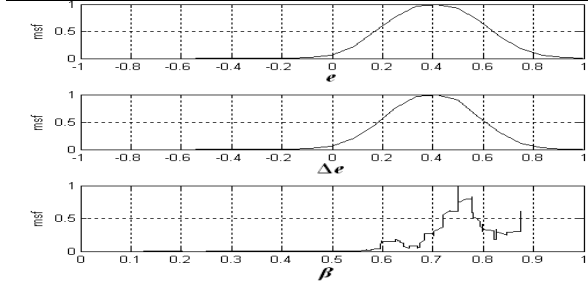
Node-14



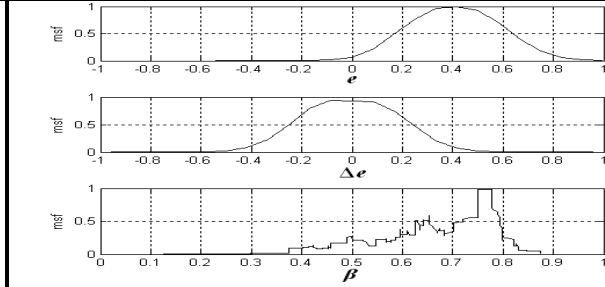
Node-15



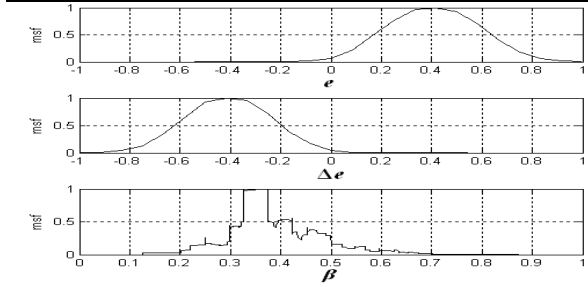
Node-21



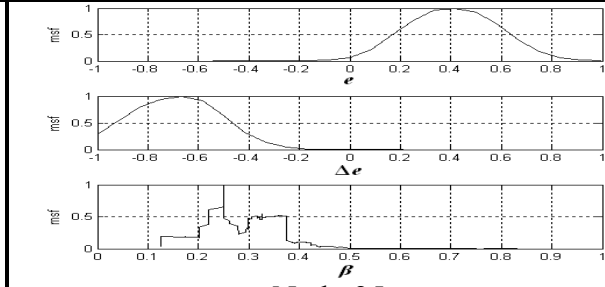
Node-22



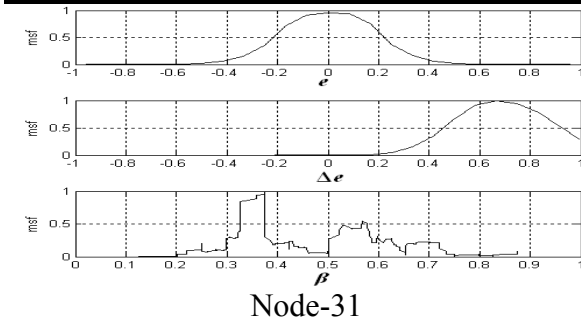
Node-23



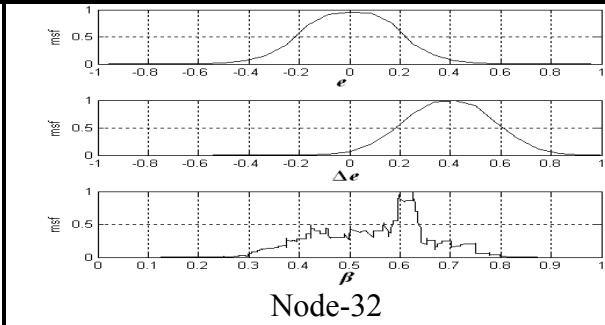
Node-24



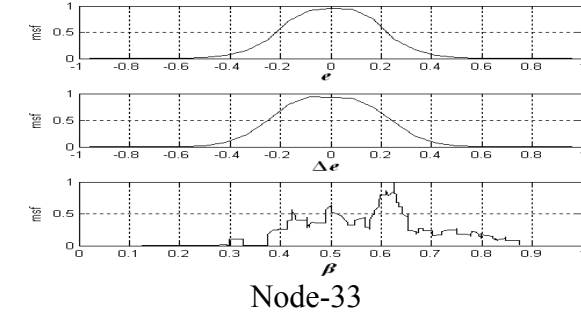
Node-25



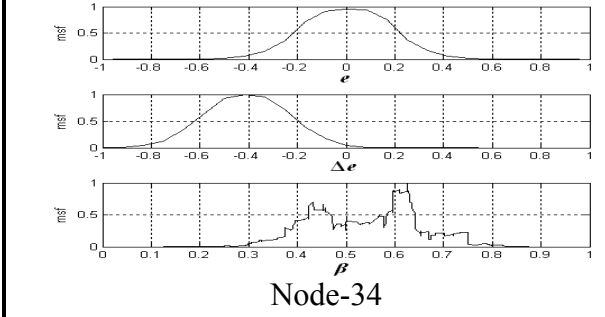
Node-31



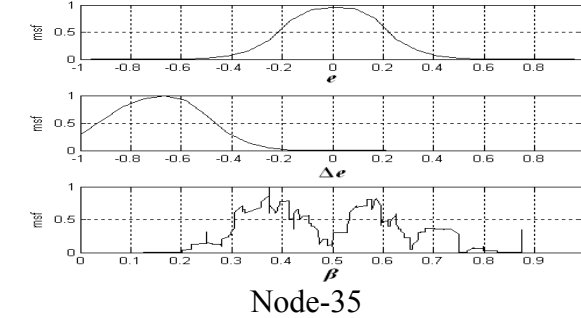
Node-32



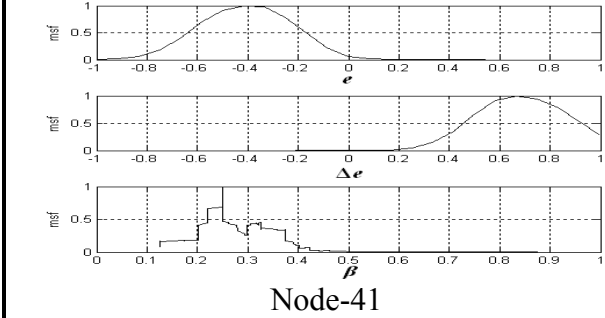
Node-33



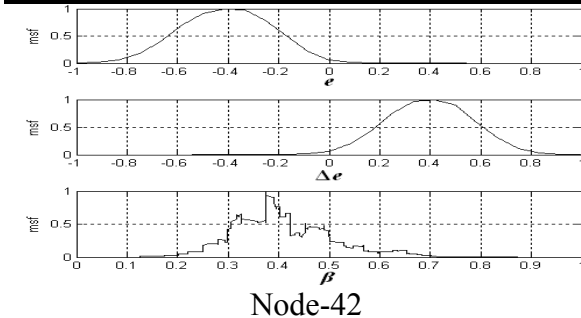
Node-34



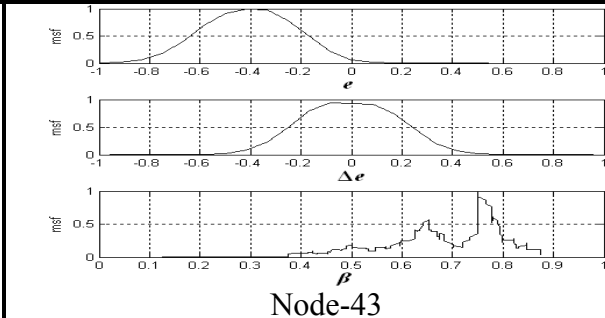
Node-35



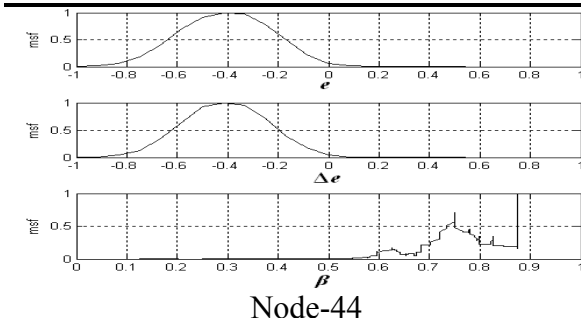
Node-41



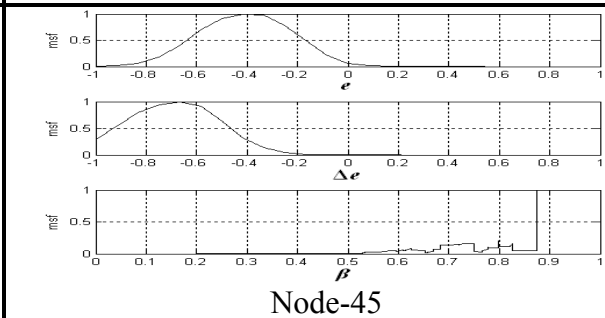
Node-42



Node-43



Node-44



Node-45

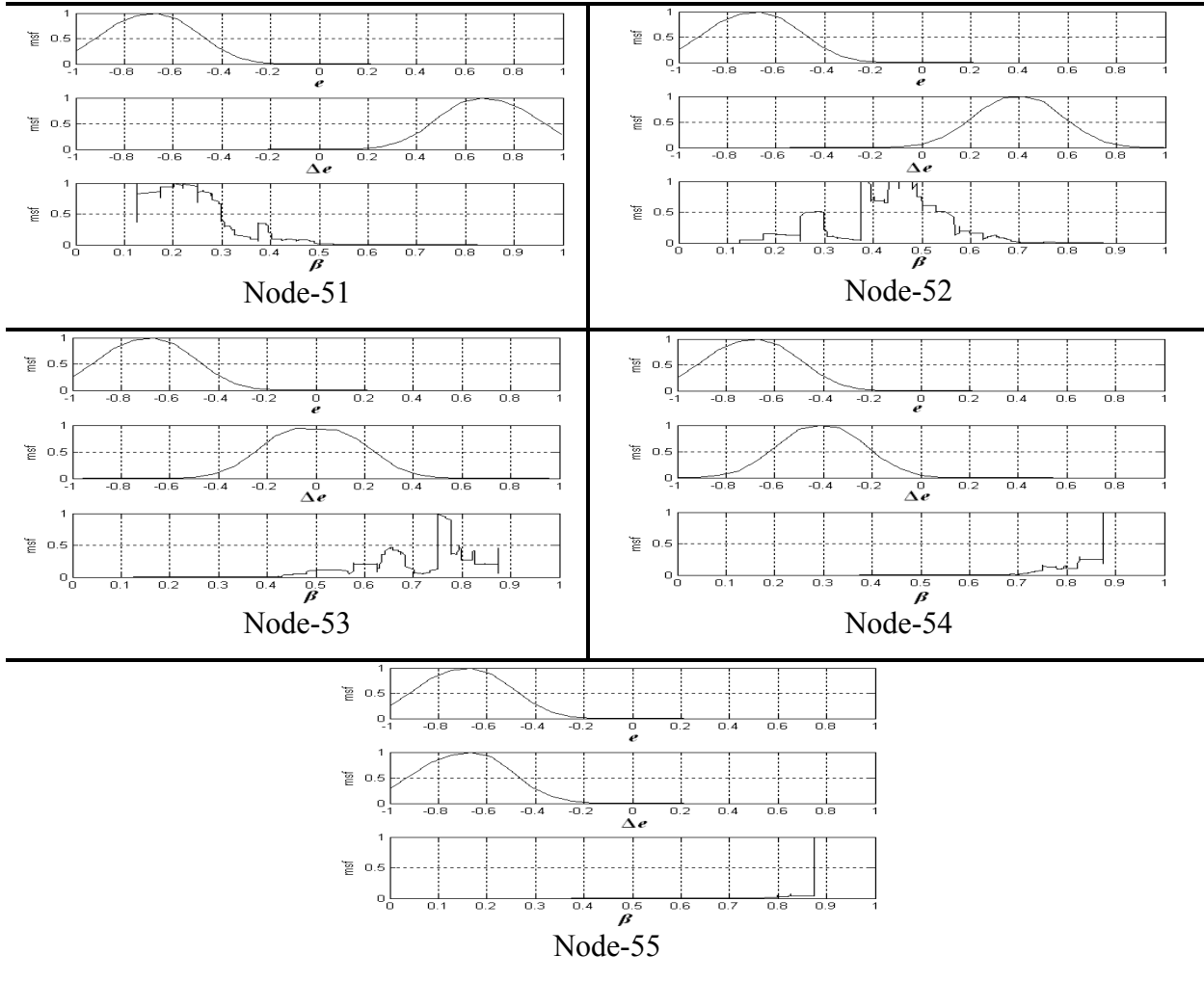


Figure 5.9: MF plots of 25 nodes for $\{e, \Delta e, \beta\}$.



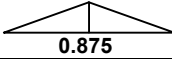
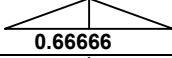
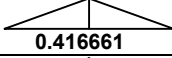
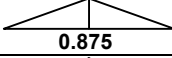

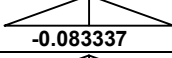
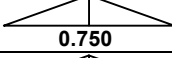

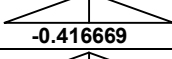
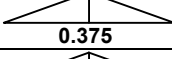
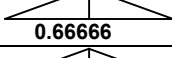
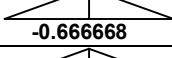
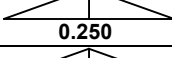
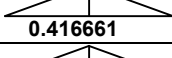
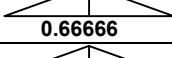
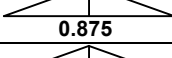
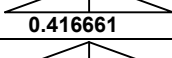
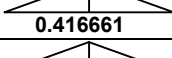
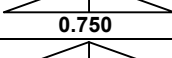
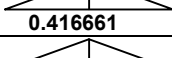
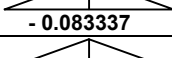
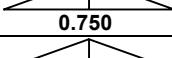
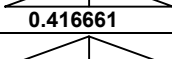
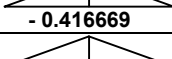
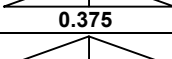
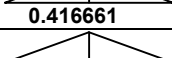
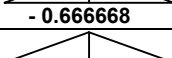
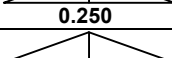
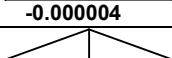

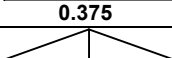
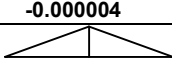
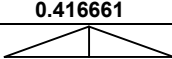
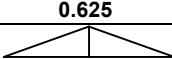
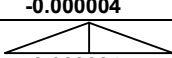
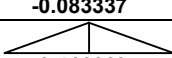
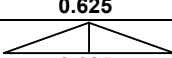
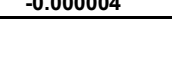
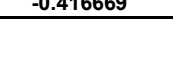
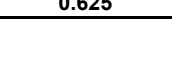
The plots for input data (e and Δe) have smooth shapes and these can be modeled easily, but it is difficult to model gain factor (β) as shown in Fig. 5.9. For extracting MFs, it is important to identify their centroids (or peak) and to choose proper base-width from the cluster formation as discussed in *section 5.3.1*. After extracting MFs for all the 25 clusters, we obtain 25 rules of the form:

$$R_k: \text{if } e \text{ is } \mu_k(e) \text{ and } \Delta e \text{ is } \mu_k(\Delta e) \text{ then } \beta \text{ is } \mu_k(\beta).$$

In our scheme, it is possible to identify 25 distinct MFs for each input (e and Δe) of 25 rules. The study of the plots (Fig. 5.9), corresponding to 25 nodes reveals that many of the MFs are almost the same. For example, the MF of e for *node-11*, *node-12*, *node-13*, *node-14* and *node-15* is

almost same. Similarly, the MF of Δe for *node-11*, *node-21*, *node-31*, *node-41* and *node-51* is almost identical, though each of the clusters represents a separate rule. From all the 25 clusters we observe that there is a possibility of 5 distinct MFs for each of the input variables e and Δe .

Each domain of e and Δe is uniformly quantized for 5 MFs within the range $[-1, 1]$ as we did for training data generation. If we consider MFs are symmetric triangles with equal base-width and having nearly 50% overlap between neighboring MFs for each of e and Δe , then the average base-width of each MF comes around 0.8. As discussed in *section 5.3.1*, for every quantized pair $\{e, \Delta e\}$ the rule completeness is checked. If the condition of completeness is not satisfied the base-width of all MFs is increased by a small percentage (say 5%) of its previous value and again the rule completeness is checked with the new base-width and this iterative process is continued until the completeness is achieved. In this way we transform the 25 rule-bases into a complete one as shown in Fig. 5.10.

<i>Rules(i,j)</i>	<i>E</i>	Δe	β
R_{11}	 0.66666	 0.66666	 0.875
R_{12}	 0.66666	 0.416661	 0.875
R_{13}	 0.66666	 -0.083337	 0.750
R_{14}	 0.66666	 -0.416669	 0.375
R_{15}	 0.66666	 -0.666668	 0.250
R_{21}	 0.416661	 0.66666	 0.875
R_{22}	 0.416661	 0.416661	 0.750
R_{23}	 0.416661	 -0.083337	 0.750
R_{24}	 0.416661	 -0.416669	 0.375
R_{25}	 0.416661	 -0.666668	 0.250
R_{31}	 -0.000004	 0.66666	 0.375
R_{32}	 -0.000004	 0.416661	 0.625
R_{33}	 -0.000004	 -0.083337	 0.625
R_{34}	 -0.000004	 -0.416669	 0.625

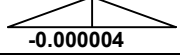
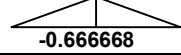
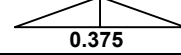
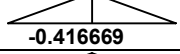
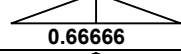
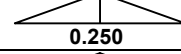
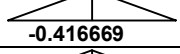
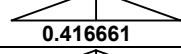
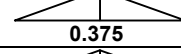
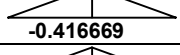
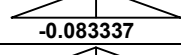
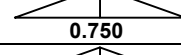
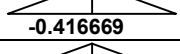
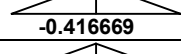
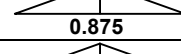
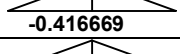
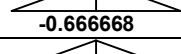
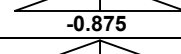
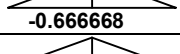
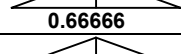
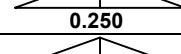
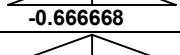
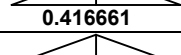
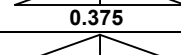
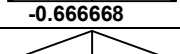
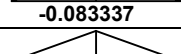
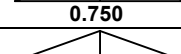
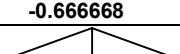
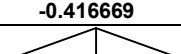
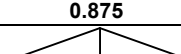
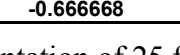
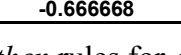
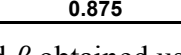
R_{35}	 -0.000004	 -0.666668	 0.375
R_{41}	 -0.416669	 0.666666	 0.250
R_{42}	 -0.416669	 0.416661	 0.375
R_{43}	 -0.416669	 -0.083337	 0.750
R_{44}	 -0.416669	 -0.416669	 0.875
R_{45}	 -0.416669	 -0.666668	 -0.875
R_{51}	 -0.666668	 0.666666	 0.250
R_{52}	 -0.666668	 0.416661	 0.375
R_{53}	 -0.666668	 -0.083337	 0.750
R_{54}	 -0.666668	 -0.416669	 0.875
R_{55}	 -0.666668	 -0.666668	 0.875

Figure 5.10: Representation of 25 fuzzy *if-then* rules for e , Δe and β obtained using rule-extraction scheme.

Observe that the gain modifying scheme of STFPIIC having 49 rules is modeled by the proposed SOM based algorithm with only 25 fuzzy *if-then* rules out of 625 input-output data. The corresponding gain surface of the model is shown in Fig. 5.11(a). To find out the accuracy of the identified model, a comparison is made between the original (Fig. 5.8a) and the model gain surface (Fig. 5.11a) using mean square error (MSE) as stated in *equation (5.9)*.

$$MSE = \sum_{i=1}^n (y_a^i - y_m^i)^2 / n \quad (5.9)$$

Where n is the number of data, y_a is the actual output and y_m is the model output. Though the output MF is not smooth in shape, yet we have obtained similar type of gain surface and the ***MSE is calculated 0.0274***. The closeness between original gain surface and identified gain surface is reflected in error surface (Fig. 5.11b), which is almost flat in nature.

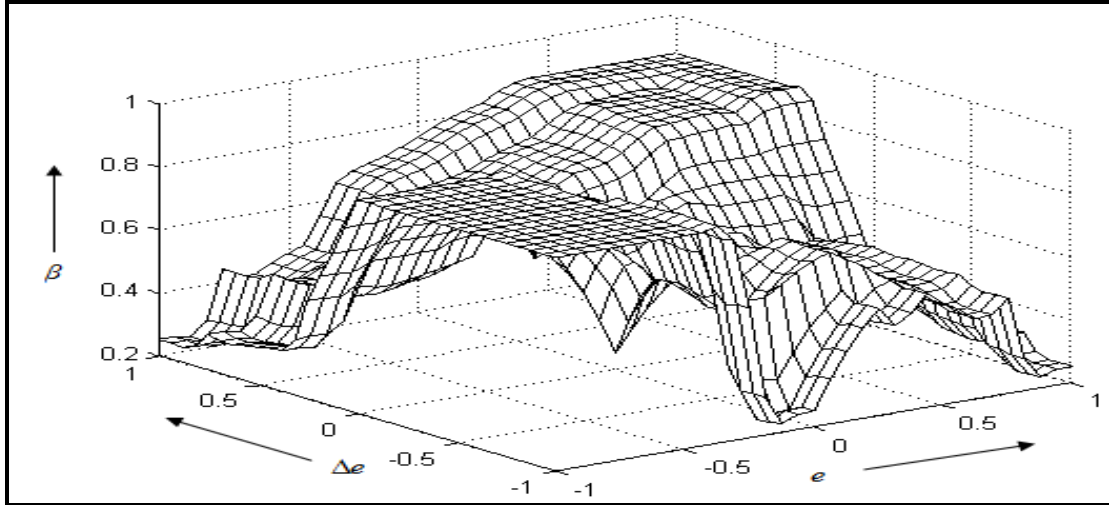


Figure 5.11(a): Gain surface of STFPIC with 25 identified rules.

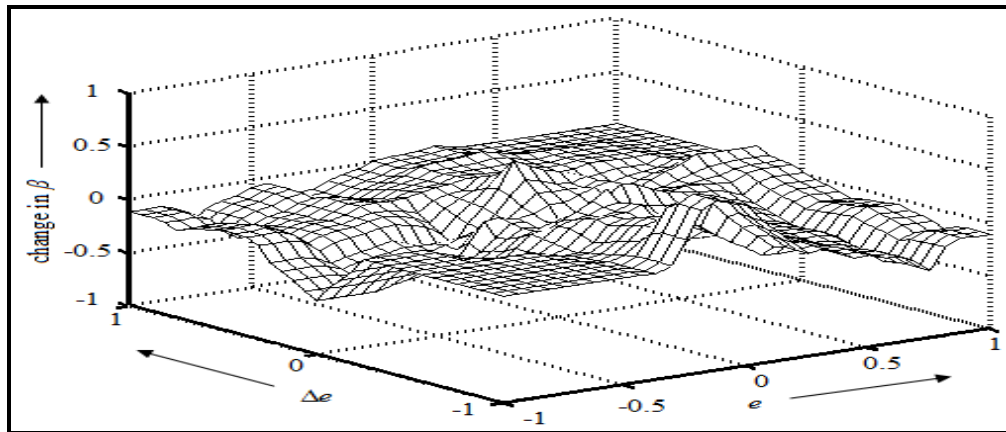


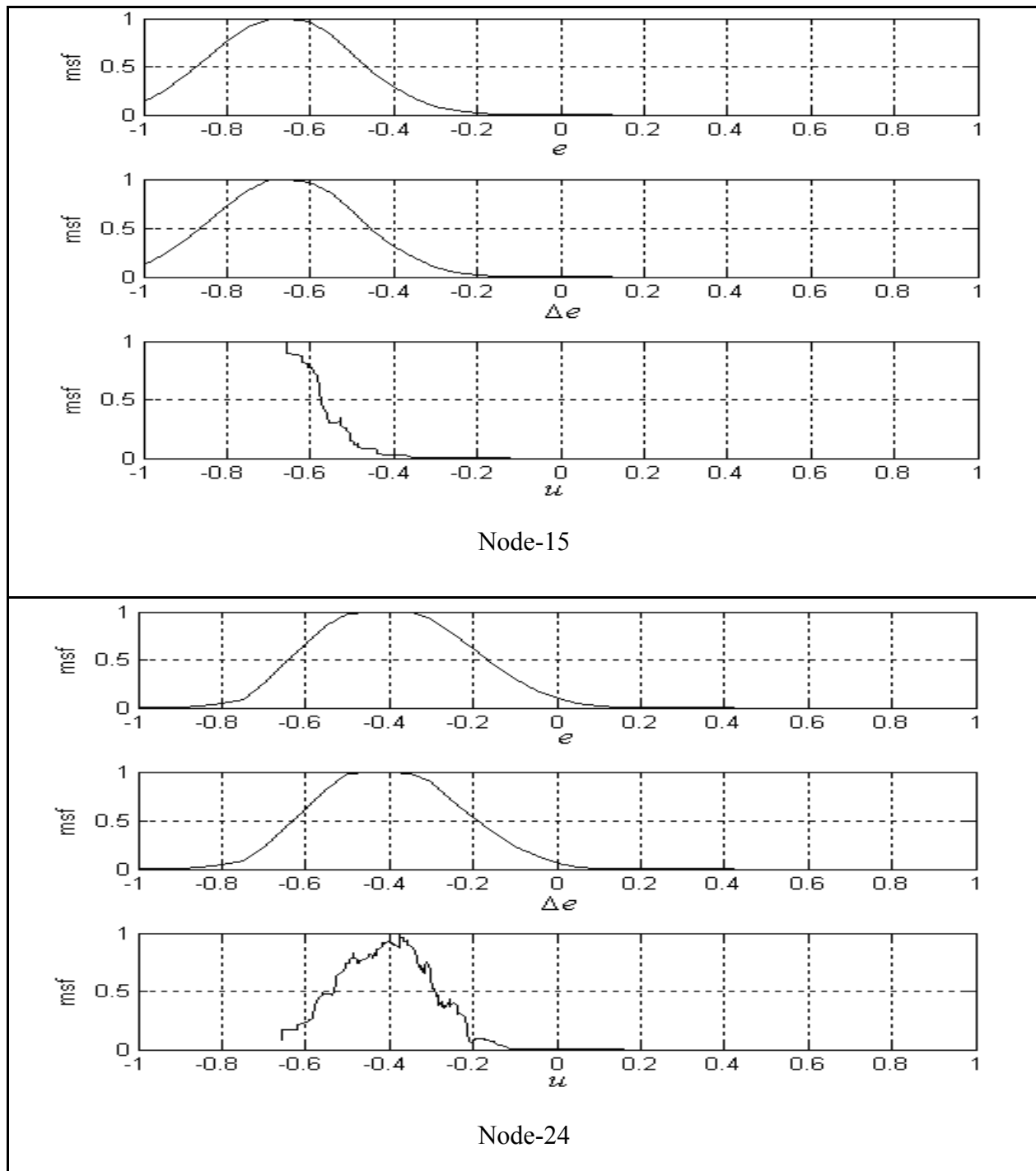
Figure 5.11(b): Error surface (difference between original and identified gain surface).

5.5 Study of the control surface of STFPIC

Like the gain surface of STFPIC, the validity of the proposed scheme is established through the study of control surface of STFPIC. In this section, input-output data $\{e, \Delta e, u\}$ are clustered in a two dimensional (5×5) plane using self-organizing map, then we determine peak and corresponding MFs for all the 25 nodes.

Few plots are shown in Fig.5.12. Like gain surface of STFPIC, its control surface is also modeled with only 25 fuzzy *if-then* rules from 625 input-output data. The validity of the

proposed scheme is established through the study of control surfaces as shown in Fig. 5.8b (which is reproduced here for close comparison) and Fig. 5.13. Study reveals that the closeness between the original control surface (Fig. 5.8b) and the identified control surface (Fig. 5.13) increases as the number of iteration in SOM clustering increases.



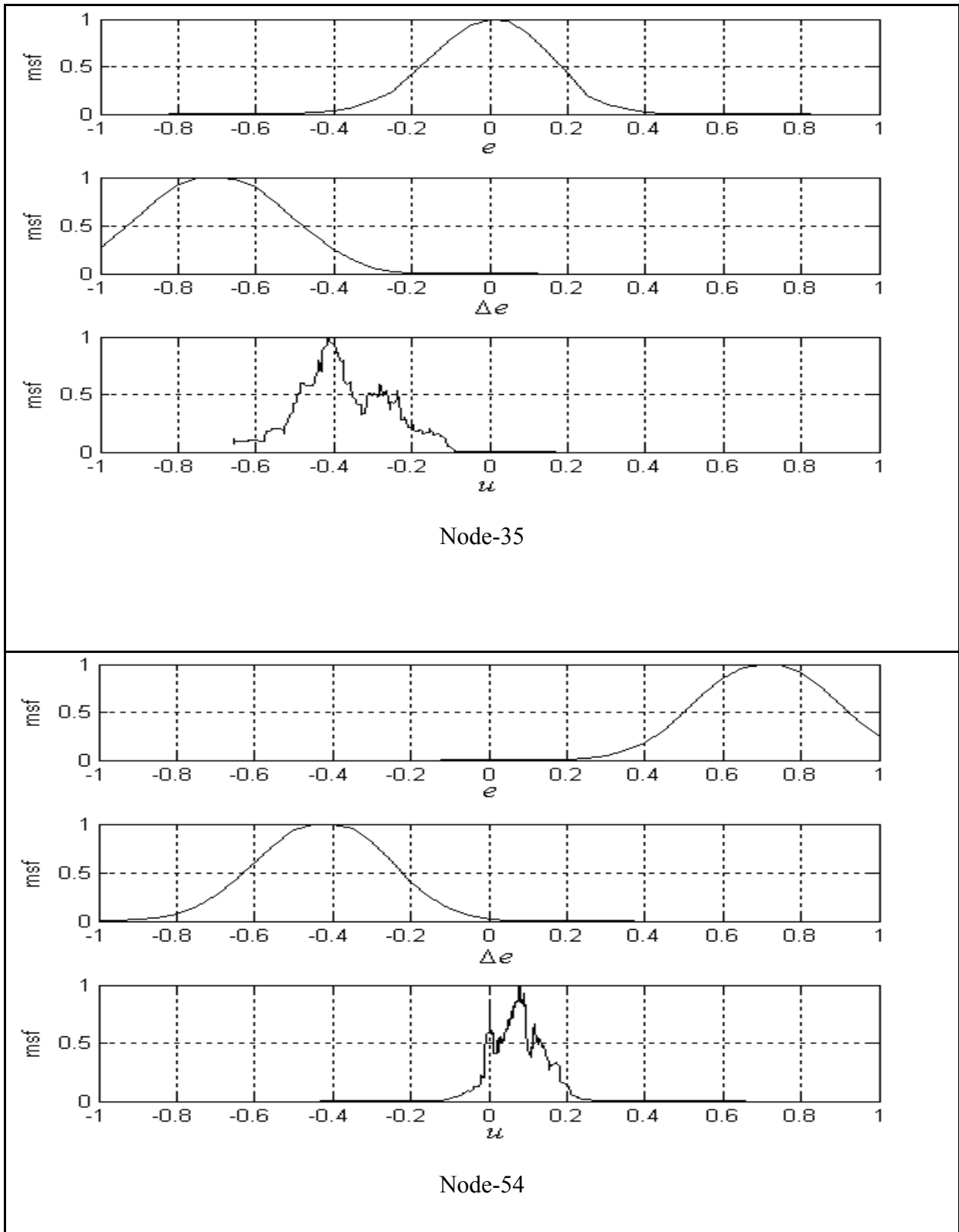


Figure 5.12: MF plots of 4 nodes out of 25 for e , Δe and u after 1000 iteration.

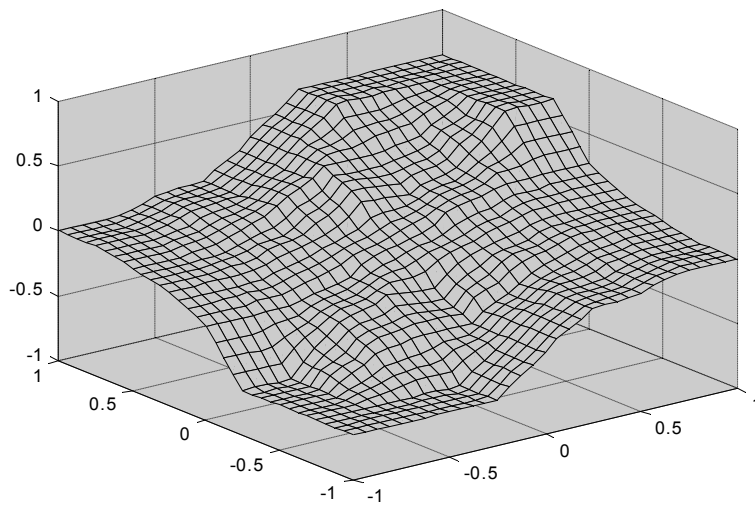
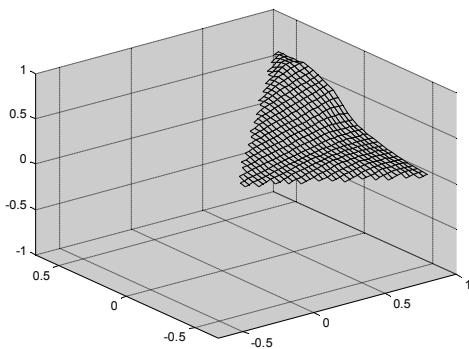
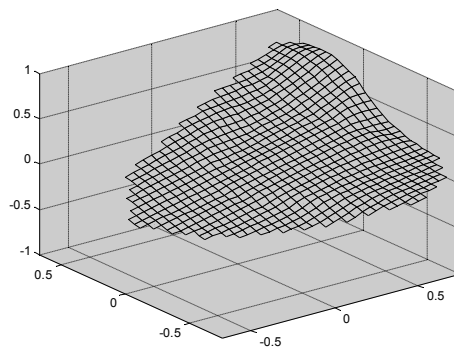


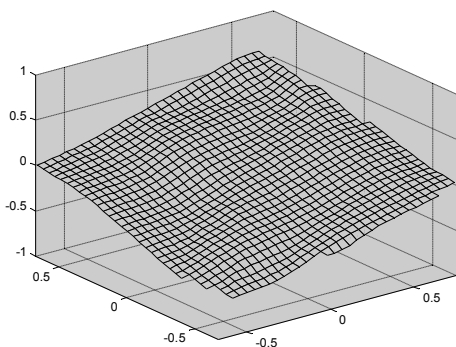
Figure 5.8(b): Original control surface of STFPIC with 98 rules.



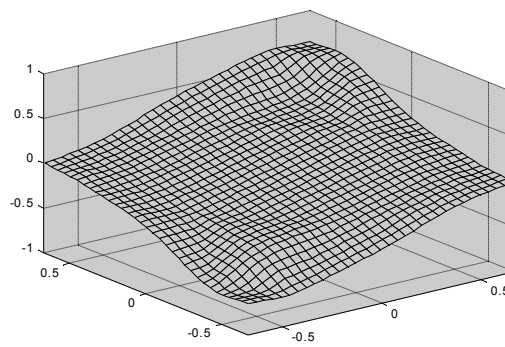
(a) Control surface (after 50 epochs).



(b) Control surface (after 100 epochs).



(c) Control surface (after 200 epochs)



(d) Control surface (after 1000 epochs)

Figure 5.13(a, b, c, d): Control surface of identified FLC with 25 rules after different number of iterations.

5.6 Simulation study with the identified STFPIIC

5.6.1 With reduced gain rules

Usefulness of the identified model of the gain surface is judged by comparative study between original (with 49 gain rules) and identified model (with 25 gain rules) in different systems. By using the reduced gain rules of STFPIIC, the responses of linear, nonlinear and marginally stable systems with dead-time are observed. As discussed in *section 5.3.1*, 25 gain rules are identified for STFPIIC and they are termed as, SOM-STFPIIC₁, SOM-STFPIIC₂ and SOM-STFPIIC₃ for three types of membership functions MF_1 , MF_2 and MF_3 respectively.

Second Order Linear Process

Response characteristics of the linear process

$$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.2y = u(t - L) \quad (5.10)$$

is shown in Fig. 5.14 for $L=0.3$, when disturbance is applied at 40s. Comparative study and tabulation of various performance indices of the process for STFPIIC, SOM-STFPIIC₁, SOM-STFPIIC₂ and SOM-STFPIIC₃ are shown in Fig. 5.14 and in Table 5.1. Fig. 5.14 shows satisfactory performances of SOM-STFPIIC₁, SOM-STFPIIC₂ and SOM-STFPIIC₃ using only 25 extracted rules for β . Table 5.1 shows that the performance indices of SOM-STFPIIC₁, SOM-STFPIIC₂ and SOM-STFPIIC₃ are almost same and very close to STFPIIC, except settling time t_s .

Table 5.1: Performance analysis of linear process (5.10)

Controller Type	$t_r(s)$	$t_s(s)$	%OS	IAE	ITAE	ISE
STFPIIC	5.9	29.9	24.55	11.46	365.12	5.04
SOM-STFPIIC ₁	6.2	17.8	23.05	10.61	317.91	5.14
SOM-STFPIIC ₂	6.0	21.1	23.69	10.63	320.10	5.14
SOM-STFPIIC ₃	6.1	21.1	23.47	10.67	321.40	5.17

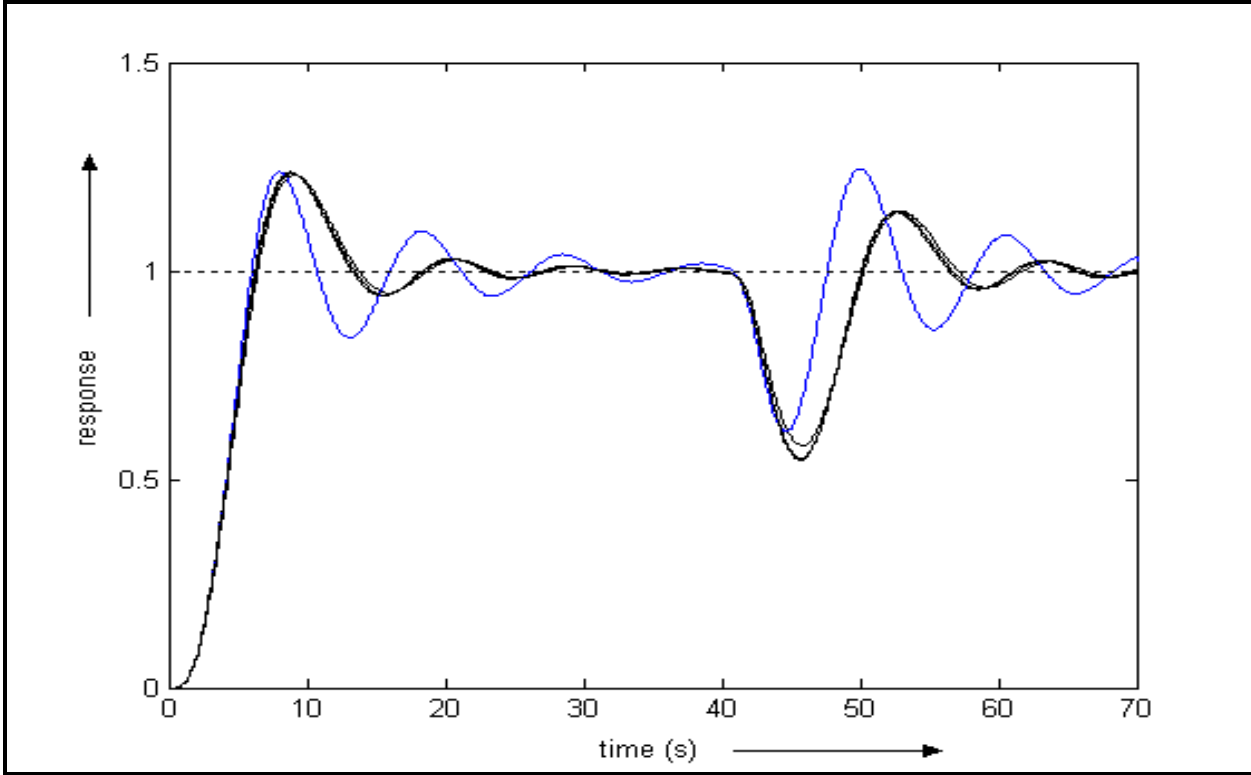


Figure 5.14: Responses of the linear process (5.10) with load disturbance for STFPIC (blue) and SOM-STFPIC₁, SOM-STFPIC₂ and SOM-STFPIC₃ (black).

Second Order Nonlinear Process

The effectiveness of the proposed scheme is studied on a nonlinear process (5.11) with different dead-time (*i.e.*, $L=0.3, 0.5$ and 0.1).

$$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.25y^2 = u(t-L) \quad (5.11)$$

Fig. 5.15 and Table 5.2 provide relative performances of the controllers with the extracted gain rules. In Fig. 5.15, we observe the comparable and satisfactory performance of SOM-STFPIC₁, SOM-STFPIC₂ and SOM-STFPIC₃ for dead-time of 0.3 and 0.5. We also analyze the performance of nonlinear system (5.11) in Fig. 5.16 with dead-time 0.1 and for a load variation at 28s. Table 5.2 provides relative performances of the controllers and we find that the FLCs that model with extracted gain rules even perform better than the original STFPIC, though performance enhancement is not our objective in this present study.

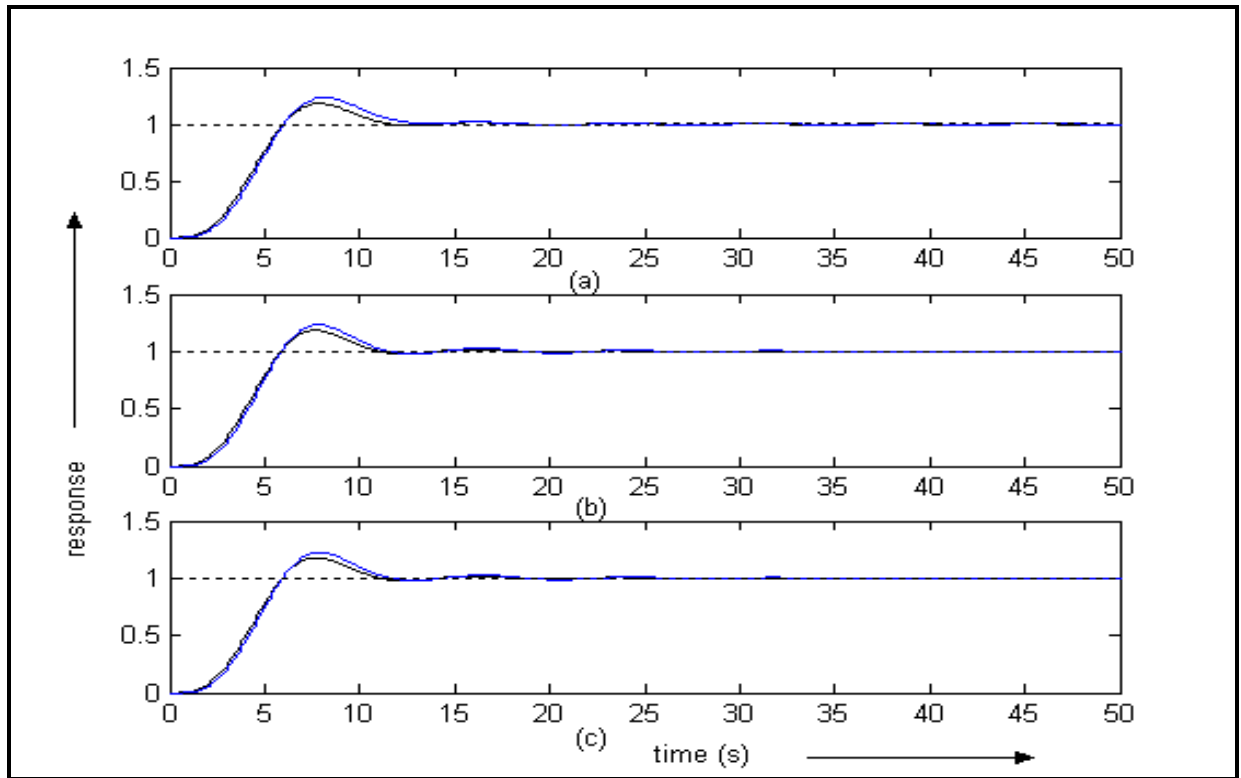


Figure 5.15 (a, b, c): Responses of (5.11) for SOM-STFPIC₁, SOM-STFPIC₂ and SOM-STFPIC₃ respectively (blue for $L=0.5$ and black for $L=0.3$).

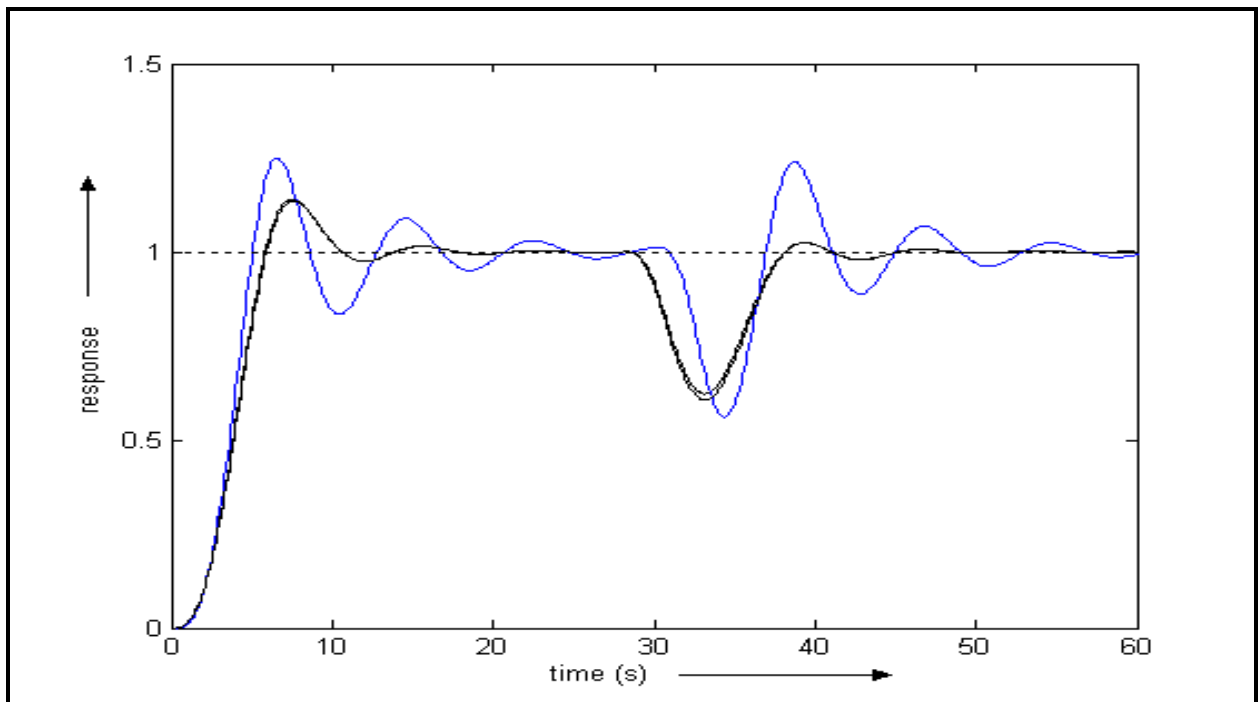


Figure 5.16: Responses of (5.11) with load disturbance at 28s for STFPIIC (blue) and SOM-STFPIC₁, SOM-STFPIC₂ and SOM-STFPIC₃ (black).

Table 5.2: Performance analysis of nonlinear process (5.11) at different dead-time

L(s)	Controller Type	t_r(s)	t_s(s)	%OS	IAE	ITAE	ISE
0.3	STFPIC	5.5	13.3	20.3	4.24	16.39	2.56
	SOM-STFPIC₁	5.8	11.0	18.05	4.23	14.13	2.74
	SOM-STFPIC₂	5.7	10.6	18.50	4.20	14.30	2.70
	SOM-STFPIC₃	5.7	10.7	18.00	4.20	14.20	2.70
0.5	STFPIC	5.5	17.1	25.2	4.83	23.40	2.78
	SOM-STFPIC₁	5.9	12.2	23.42	4.67	18.22	2.95
	SOM-STFPIC₂	5.7	11.1	23.30	4.59	19.02	2.89
	SOM-STFPIC₃	5.8	11.2	22.70	4.58	18.59	2.90
<i>With load variation at 28s</i>							
0.1	STFPIC	5.2	13.1	16.58	7.68	146.68	4.33
	SOM-STFPIC₁	5.8	10.2	13.14	7.39	120.80	4.39
	SOM-STFPIC₂	5.6	10.1	14.03	7.41	122.75	4.53
	SOM-STFPIC₃	5.7	10.1	13.58	7.52	125.80	4.59

Table 5.3: Performance analysis of (5.11) with the controllers developed by *Pal et al.*[105]

L(s)	Controller Type	t_r(s)	t_s(s)	%OS
0.3	TFPIC_{wav1}	5.8	13.8	19.50
	TFPIC_{wav2}	5.8	13.3	19.60
	TFPIC_{cent}	5.8	13.3	19.60
0.5	TFPIC_{wav1}	5.9	14.2	23.90
	TFPIC_{wav2}	6.0	17.6	23.90
	TFPIC_{cent}	6.0	14.0	19.60

Pal et al.[105] applied FCM for extraction of a small but adequate set of rules to realize the self-tuning mechanism and they demonstrated their scheme in nonlinear process (5.11). They denoted their controllers by TFPIC_{wav1}, TFPIC_{wav2} and TFPIC_{cent}. The system was tested with variable dead-time (*i.e.*, $L=0.3s$ and $L=0.5s$) and load variation (at 50s). We presented their findings in Table 5.3 for reference. Comparative study of Table 5.2 and Table 5.3 reveals that the SOM-

based rule extraction scheme performs better than the FCM-based rule extraction scheme [105] for this nonlinear example.

Second Order Marginally Stable System

The proposed rule extraction scheme is demonstrated on a marginally stable system (5.12) with a pole at origin and having dead-time (L) of 0.1 and 0.2.

$$\frac{d^2 y}{dt^2} + \frac{dy}{dt} = u(t-L) \quad (5.12)$$

The comparative study of the responses with proposed controllers is shown in Fig. 5.17 and Table 5.4 with variable dead-time and load disturbance (at 30s). From the study, we do not find any distinct differences between FLCs with reduced gain rules and original STFPIIC with larger number of rules. We present a comparative study between STFPIIC and the proposed controllers in Fig. 5.18 and Table 5.5 for a load disturbance at 25s.

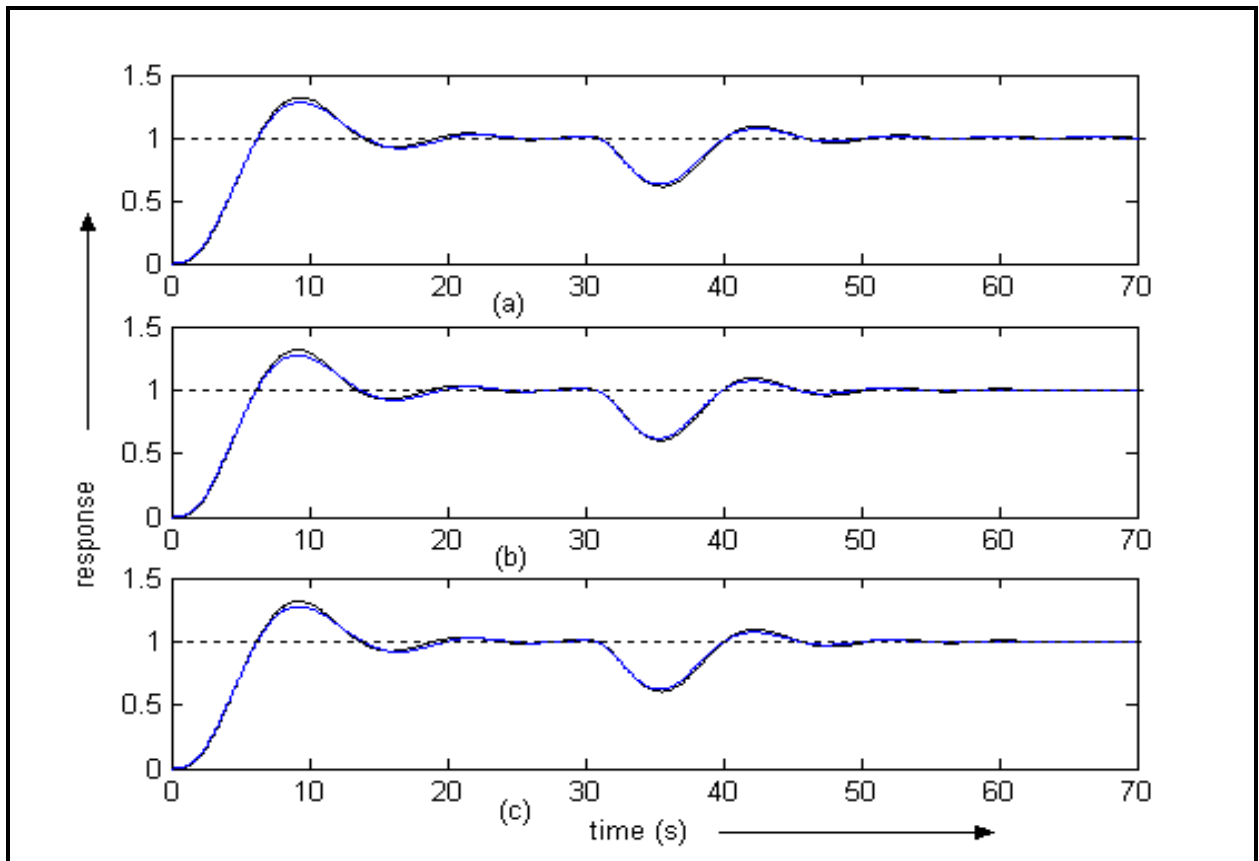


Figure 5.17 (a, b, c): Responses of system (5.12) for SOM-STFPIIC₁, SOM-STFPIIC₂ and SOM-STFPIIC₃ (blue for $L=0.1$ and black for $L=0.2$).

Table 5.4: Performance analysis of marginally stable process (5.12)

L(s)	Controller Type	$t_r(s)$	$t_s(s)$	%OS	IAE	ITAE	ISE
0.1	STFPIC	6.3	32.0	27.52	10.5	210.00	4.69
	SOM-STFPIC ₁	6.1	19.4	27.96	9.09	177.08	4.57
	SOM-STFPIC ₂	6.0	18.9	27.10	8.95	175.21	4.52
	SOM-STFPIC ₃	6.0	19.2	27.53	9.03	176.38	4.55
0.2	STFPIC	6.3	37.8	30.02	11.50	273.15	4.94
	SOM-STFPIC ₁	6.0	22.6	31.97	10.14	238.78	4.82
	SOM-STFPIC ₂	6.0	21.9	31.41	9.90	235.91	4.77
	SOM-STFPIC ₃	6.0	22.3	31.61	10.07	237.54	4.80

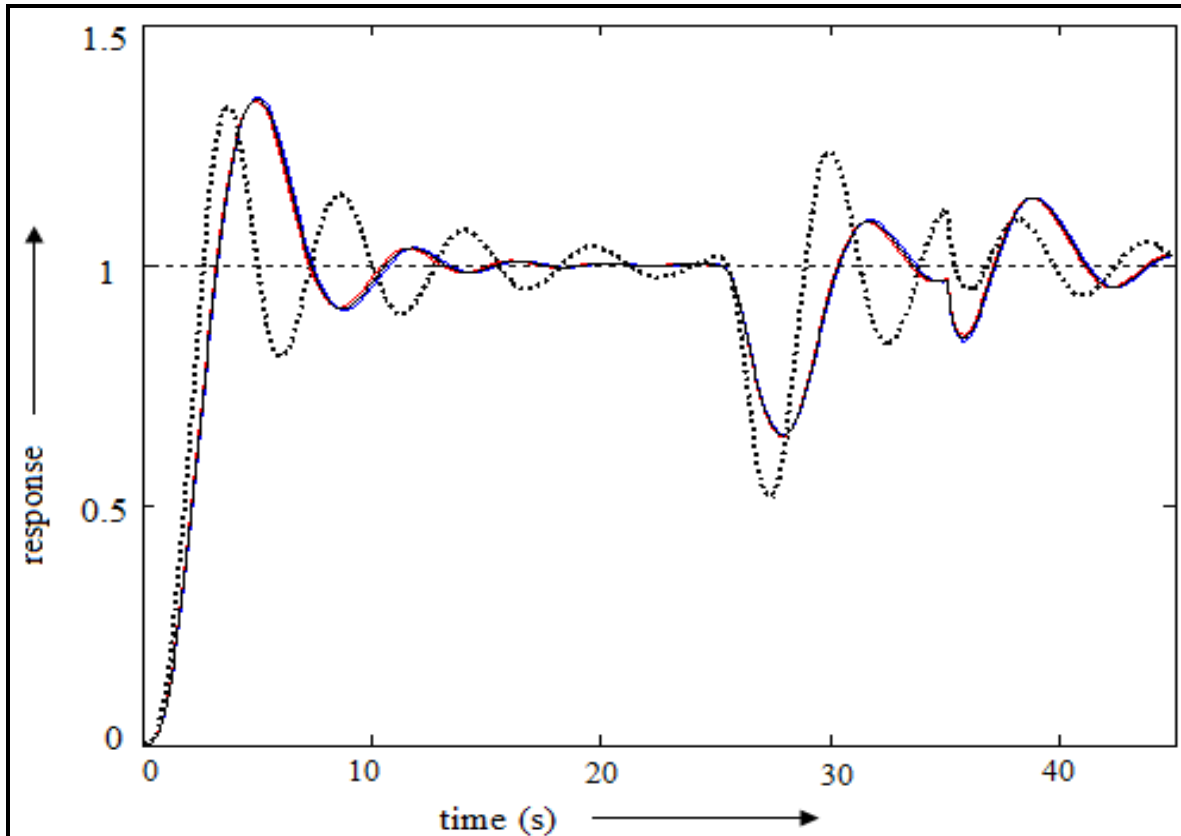


Figure 5.18: Responses of system (5.12) with load disturbance for STFPIC (dotted-black) and SOM-STFPIC₁, SOM-STFPIC₂ and SOM-STFPIC₃ (solid-colour).

Table 5.5: Performance analysis of marginally stable process (5.12) with load disturbance

Controller Type	$t_r(s)$	$t_s(s)$	%OS	IAE	ITAE	ISE
STFPIC	5.3	40.8	32.92	11.60	381.11	5.18
SOM-STFPIC ₁	6.4	25.1	34.86	10.80	325.28	5.04
SOM-STFPIC ₂	6.3	24.4	34.11	10.52	316.14	4.95
SOM-STFPIC ₃	6.3	24.8	34.45	10.66	320.63	4.99

5.6.2 With reduced control rules

In this section, the performance of identified FLC with only 25 rules in place of 98 rules of STFPIC, is tested with both linear and nonlinear systems. Comparative study between STFPIC with original 98 rules (49 control rules and 49 gain rules) and FLC with 25 identified overall rules (SOM-FLC₁) are investigated.

From gain surface study in previous section (*section 5.6.1*), we observe almost similar performance for all the three types identified controllers (SOM-STFPIC₁, SOM-STFPIC₂ and SOM-STFPIC₃) corresponding to their different types MFs (MF_1 , MF_2 and MF_3) as termed in *section 5.3.1*. Therefore *in this section, instead of three types, we only consider the first type of MF (MF_1) for controller (SOM-FLC₁) design.*

Linear System

Response characteristics of the linear process

$$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.8y = u(t) \quad (5.13)$$

is studied and comparative study of the system is provided in Fig. 5.19 and Table 5.6. Responses and detail performance analysis with original STFPIC (98 rules) and FLC with only 25 identified fuzzy rules (SOM-FLC₁) are presented in Fig. 5.19 and Table 5.6 respectively. Study reveals SOM-FLC₁ shows a comparable performance with respect to different performance criteria.

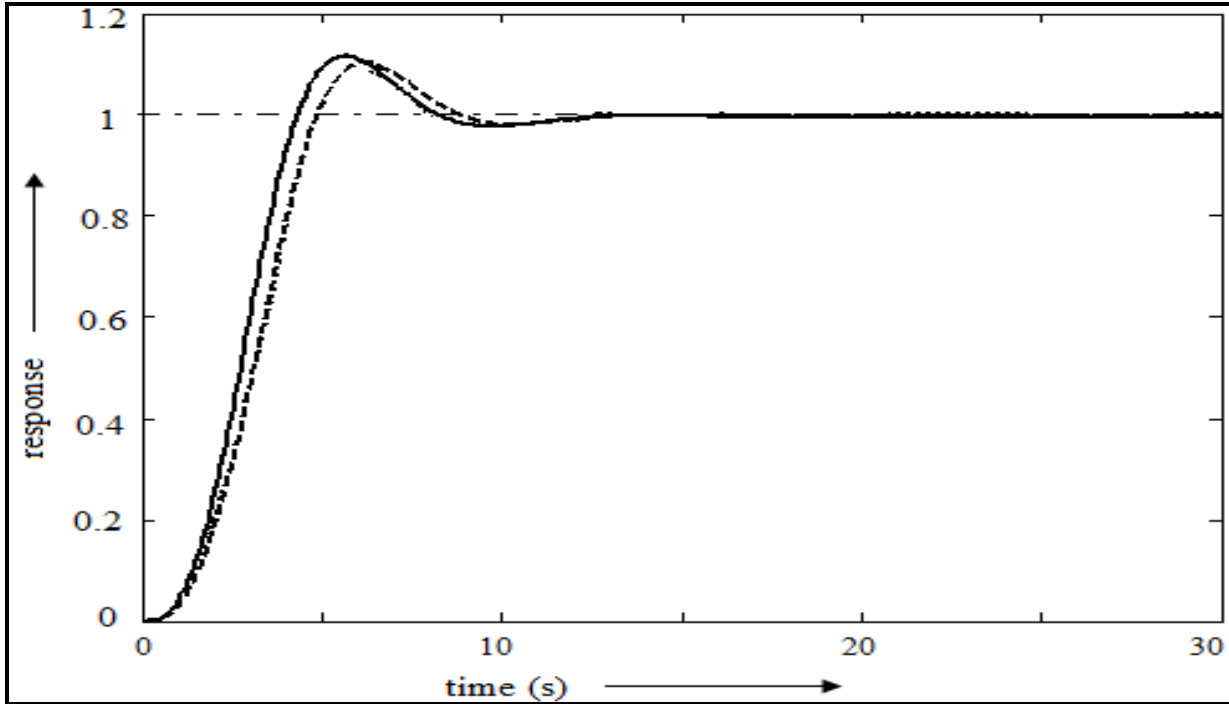


Figure 5.19: Responses of (5.13); dashed line for STFPIC (98 rules) and solid line for SOM-FLC₁ (identified 25 rules).

Nonlinear System

The performance of our rule extraction scheme is investigated on the below nonlinear process:

$$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.25y^2 = u(t - L) \quad (5.14).$$

From Fig. 5.20 and Table 5.6, we observe that the SOM-FLC₁ performs satisfactorily for nonlinear system (5.14) with dead-time (L) of 0.3.

Table 5.6: Performance analysis of linear (5.13) and nonlinear (5.14) system

System	Controller Type	%OS	t_r (s)	t_s (s)	IAE	ITAE
$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.8y = u(t)$	SOM-FLC ₁ (with 25 identified rules)	10.39	4.7	8.3	3.15	11.13
	STFPIC (98 rules)	8.15	4.9	11.8	3.17	8.41
$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.25y^2 = u(t - L)$	SOM-FLC ₁ (with 25 identified rules)	7.08	6.6	11	4.43	22.48
	STFPIC (98 rules)	14.7	6.6	11.7	4.58	15.13

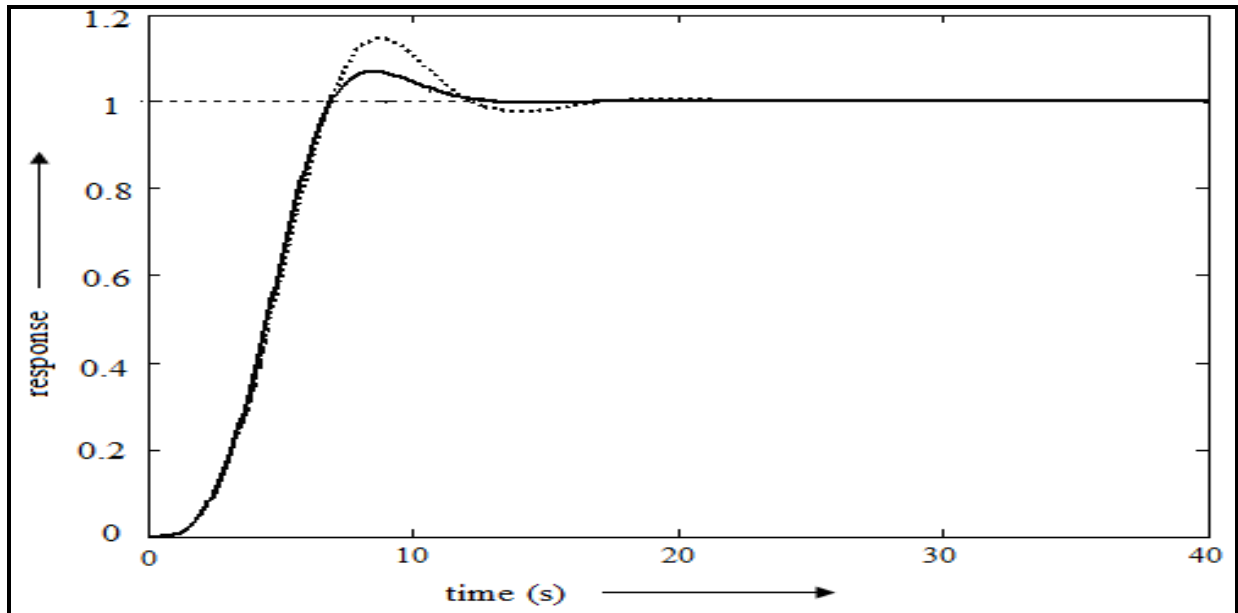


Figure 5.20: Responses of (5.14); dashed line for STFPIC (98 rules) and solid line for SOM-FLC₁ (25 rules).

5.7 Rule merging

After rule extraction as proposed through Fig. 5.5 in *section 5.3.1*, we observed many of the fuzzy sets are similar or overlapping in nature. So, there is a further scope for optimizing the size of the rule-base. Various methods, like interpolation approach have been adopted for fuzzy rule-base reduction [199-202]. Efforts in rule reduction have been made in this field using orthogonal transformation [203, 204]. *Setnes et al.* [205, 206] applied similarity measure scheme between fuzzy sets for simplification and reduction of rule-base. Several neural network and GA-based techniques are also implemented for this purpose [208].

5.7.1 Similarity measurement and fuzzy rule minimization

The validity of the rule extraction scheme is already established on STFPIC in close loop (*section 5.5*) by identifying 25 fuzzy *if-then* rules. In this section, a similarity measure scheme is introduced to further reduce the number of rules by merging similar fuzzy sets to model the system more simple [204, 205]. The technique is implemented on the control surface of STFPIC and the system is modeled by triangular type MFs with equal base-width as shown in Fig. 5.21. The study of the fuzzy sets reveals that some of the fuzzy sets are almost identical while many are very close to each other.

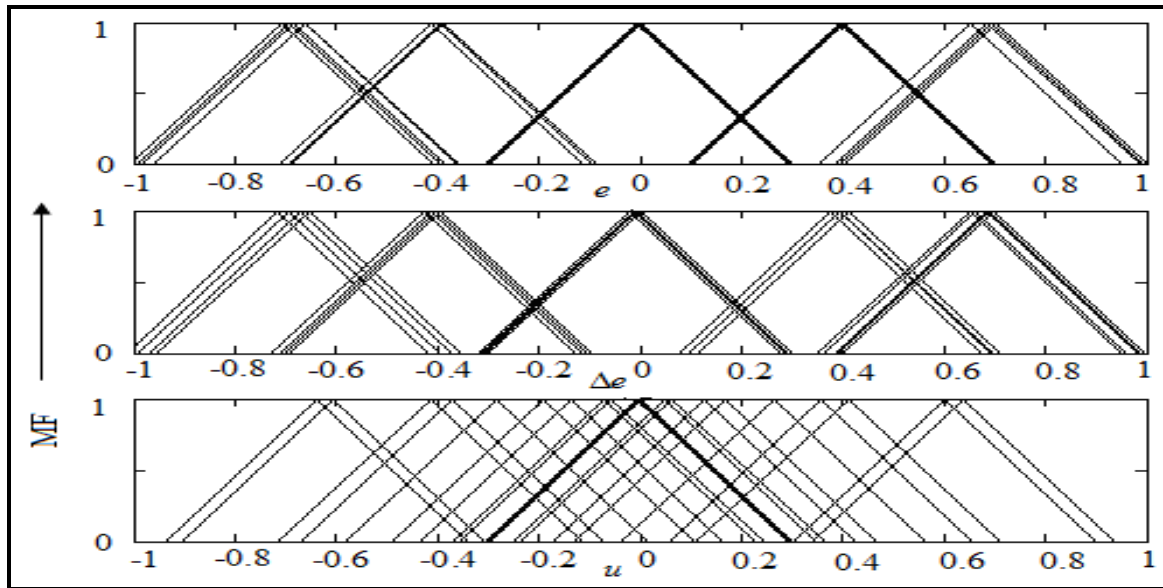


Figure 5.21: MF plots of 25 identified fuzzy sets for $\{e, \Delta e, u\}$.

Concept of similarity measurement

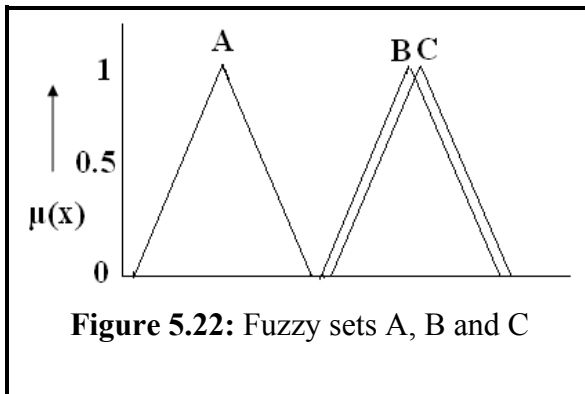


Figure 5.22: Fuzzy sets A, B and C

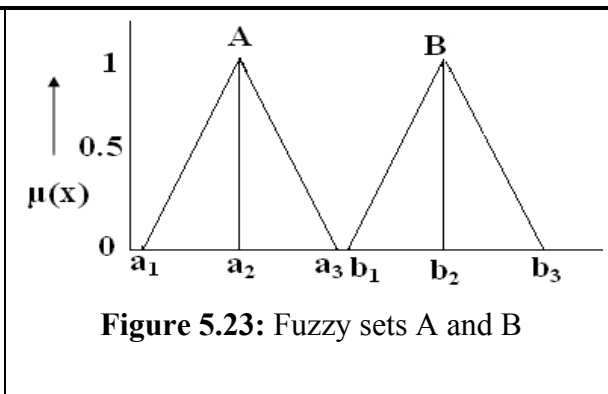


Figure 5.23: Fuzzy sets A and B

In Fig. 5.22, A, B and C are three similar triangular fuzzy sets. If we compare fuzzy sets A and B, they are similar in shape but conceptually different in nature. But B and C fuzzy sets are not only similar in shape but also similar in nature due to their closeness. It can be said that B and C fuzzy sets have high degree of equality. Using this concept of equality in Fig. 5.21, it is possible to reduce the effective numbers of fuzzy sets. If two triangular normal fuzzy sets A and B, as shown in Fig. 5.23, are described by $A=(a_1, a_2, a_3)$ and $B=(b_1, b_2, b_3)$, then the Degree of Similarity between Fuzzy Sets (*DSFS*) can be determined using *equation 5.15*. From the equation, it is observed that if $a_1=b_1, a_2=b_2$ and $a_3=b_3$, then $DSFS=1$, means complete matching; and mismatch of these points indicate dissimilarity [194, 205, 207].

$$DSFS = \frac{1}{1 + \sum_1^3 |a_i - b_i|/3}, \quad DSFS \in (0,1) \quad (5.15)$$

Using this principal of similarity, declare the sets are similar, if $DSFS$ is greater than any predefined value ξ , where $0 < \xi \leq 1$. In this process of similarity measurement, if A and B are two fuzzy sets found similar in nature, e.g. $DSFS(A, B) = 0.95$ or more, then they may be replaced by a new fuzzy set C,

$$\text{where, } C = (c_1, c_2, c_3) = \left(\frac{a_1 + b_1}{2}, \frac{a_2 + b_2}{2}, \frac{a_3 + b_3}{2} \right).$$

Rule- base reduction

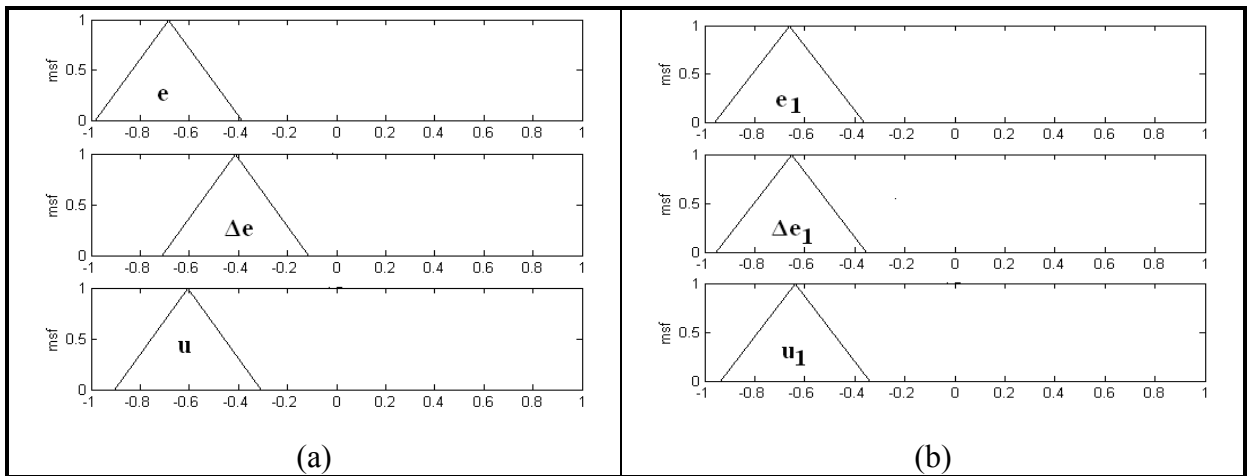


Figure 5.24: Two Fuzzy rules in the form of $\{e, \Delta e, u\}$.

In case of multi input-single output (MISO) system, as shown in Fig. 5.24 (a and b), it is found that inputs e and e_1 are similar, but other inputs Δe and Δe_1 are not very similar, still their outputs u and u_1 are almost similar. In such cases, we can merge the fuzzy sets by planning the following process:

Consider two rules (R_1 and R_2):

R_1 : if x_1 is A_1 and x_2 is B_1 then y is D_1

R_2 : if x_1 is A_1 and x_2 is C_1 then y is D_1

Then the Degree of Similarity between two Fuzzy Rules R_i and R_j ($DSFR_{ij}$) in terms of $DSFS$ can be computed using *equation* (5.16) [207].

$$DSFR_{ij} = \frac{\sum_{k=1}^n DSFS(R_{ik}, R_{jk})}{n} \quad (5.16)$$

where, $\sum_{k=1}^n DSFS(R_{ik}, R_{jk})$ is the summation of $DSFS$ of two fuzzy rules (R_i and R_j) and n is the total number of input and output parameters in a rule. In the above example, there are two inputs and one output, therefore $n = 3$. Using *equation* 5.16, $DSFR$ values are computed between all 25 rules as presented in Fig. 5.21. Different considerations of $DSFR$ value yields different number of fuzzy rules, e.g. $DSFR = 0.8$ yields 14 rules as shown in Fig. 5.25.

Rules	e	Δe	u
R ₁			
R ₂			
R ₃			
R ₄			
R ₅			
R ₆			
R ₇			
R ₈			
R ₉			
R ₁₀			
R ₁₁			
R ₁₂			
R ₁₃			
R ₁₄			

Figure 5.25: Conversion of 25 to 14 rules $\{e, \Delta e, u\}$ after similarity measurement.

5.7.2 Results

We examine this rule merging scheme on linear, nonlinear and on a numerical example.

Linear System

Let us consider a linear system

$$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.8y = u(t) \quad (5.17).$$

We study the linear system (5.17) with original STFPIC (with 98 rules), SOM-FLC₁ (with 25 rules) and Reduced SOM-FLC₁ (14 rules). Results are presented in Fig. 5.26 and Table 5.7. The results indicate that the system gives satisfactory performance only with 14 rules.

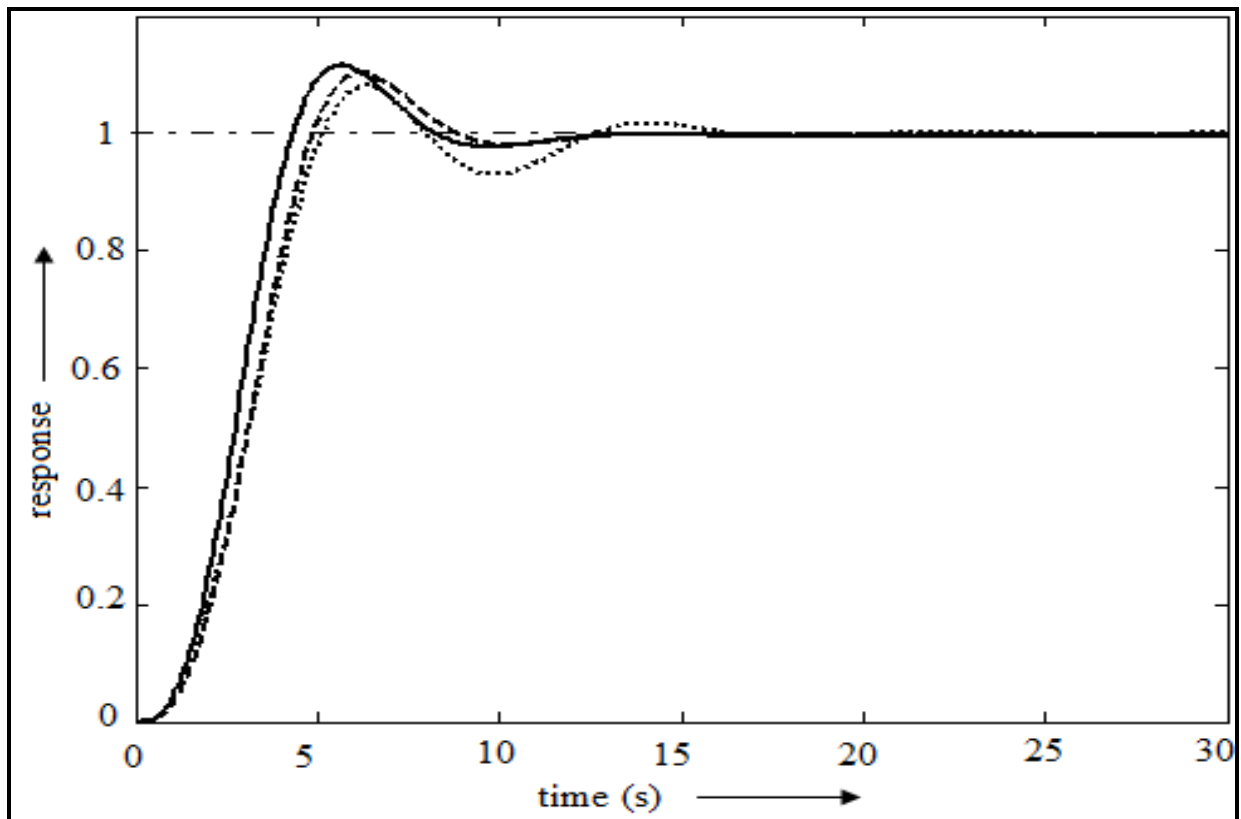


Figure 5.26: Responses of (5.17); dotted line for STFPIC, dashed line for SOM-FLC₁ and solid line for Reduced SOM-FLC₁.

Nonlinear System

From Fig. 5.27 and Table 5.7, we observe that the proposed similarity measure scheme works satisfactorily for the nonlinear system

$$\frac{d^2 y}{dt^2} + \frac{dy}{dt} + 0.25y^2 = u(t - L) \quad (5.18).$$

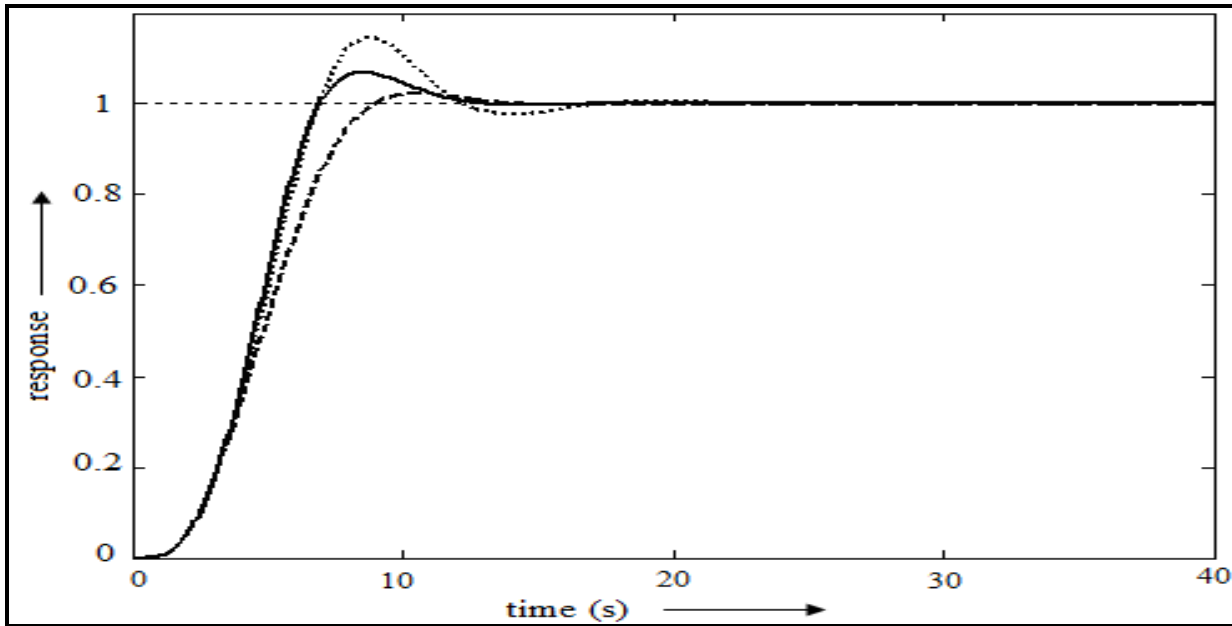


Figure 5.27: Responses of (5.18) for $L=0.3$; dotted line for STFPIC, dashed line for SOM-FLC₁ and solid line for Reduced SOM-FLC₁.

Table 5.7: Performance analysis of linear and nonlinear system

System	Controller Type	%OS	$t_r(s)$	$t_s(s)$	IAE	ITAE
$\frac{d^2 y}{dt^2} + \frac{dy}{dt} + 0.8y = u(t)$	Reduced SOM-FLC ₁ (14 rules)	11.65	4.1	7.8	3.05	20.92
	SOM-FLC ₁ (25 rules)	10.39	4.7	8.3	3.15	11.13
	STFPIC (98 rules)	8.15	4.9	11.8	3.17	8.41
$\frac{d^2 y}{dt^2} + \frac{dy}{dt} + 0.25y^2 = u(t - L)$	Reduced SOM-FLC ₁ (14 rules)	7.08	6.6	11.0	4.43	22.48
	SOM-FLC ₁ (25 rules)	2.32	8.0	8.5	4.55	14.68
	STFPIC (98 rules)	14.7	6.6	11.7	4.58	15.13

5.8 Generalization of the proposed scheme

Though the proposed scheme has been studied with STFPIC, it is applicable for other problems also. To generalize the scheme we have studied a function approximation problem [14] and to substantiate the scheme as a general one, next we apply this scheme for a function approximation problem. The nonlinear numerical example has two inputs (x and y) and one output (z). The nonlinear function is represented by *equation* (5.19).

$$z = (1 + x^{-2} + y^{-1.5})^2, \quad 1 \leq x, y \leq 5 \quad (5.19)$$

Initially, 50 datasets (x, y, z) in the range (0 - 6, 0 - 6, and 0 - 10) are generated using *equation* (5.19) to model the nonlinear system using our proposed rule extraction technique. The corresponding surface plot for the *equation* (5.19) is shown in Fig. 5.28.

We have applied our proposed rule extraction scheme as described in *section 5.3.1* to the above numerical example to extract 25 fuzzy *if-then* rules. The triangular MFs with equal base-width are presented in Fig. 5.29. The surface plot of *equation* (5.19) by extracted 25 fuzzy *if-then* rules is depicted in Fig. 5.30.

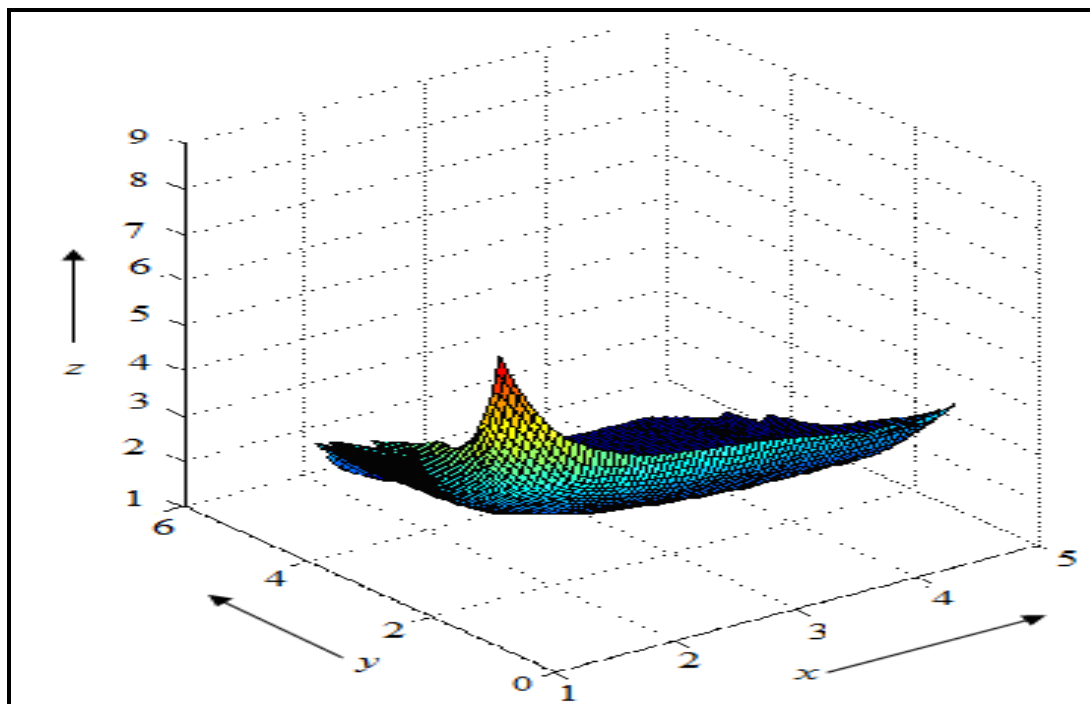


Figure 5.28: Surface plot of the nonlinear *equation* 5.19.

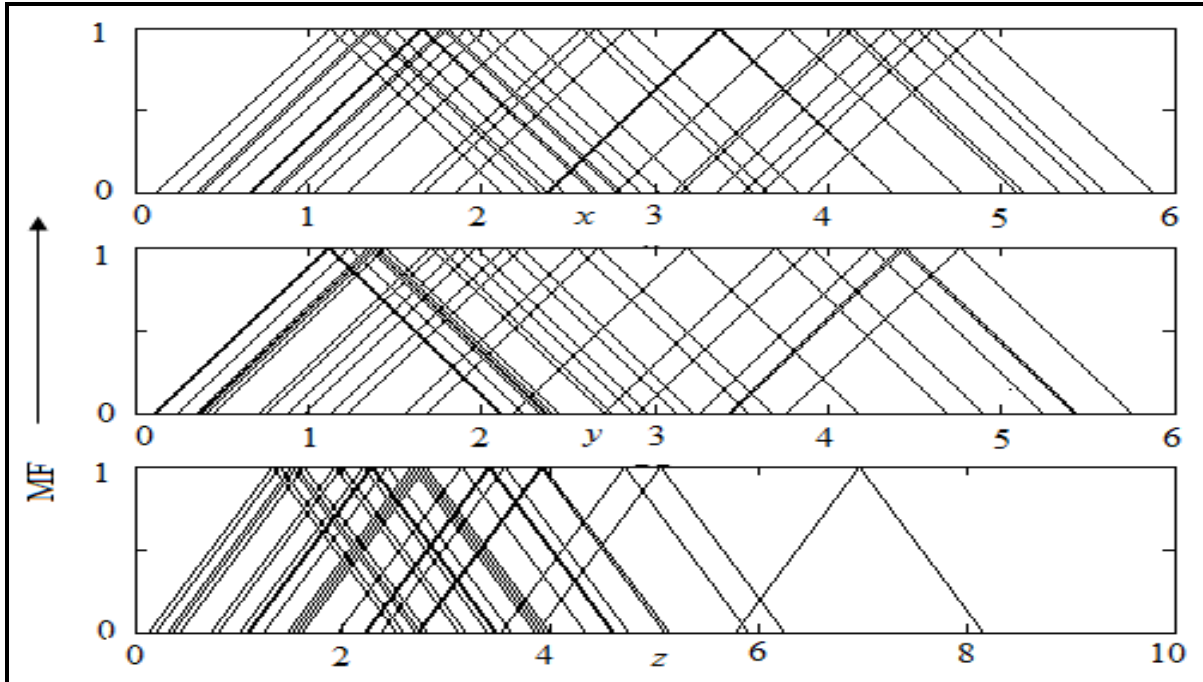


Figure 5.29: MFs for the identified fuzzy model of the nonlinear system (equation 5.19).

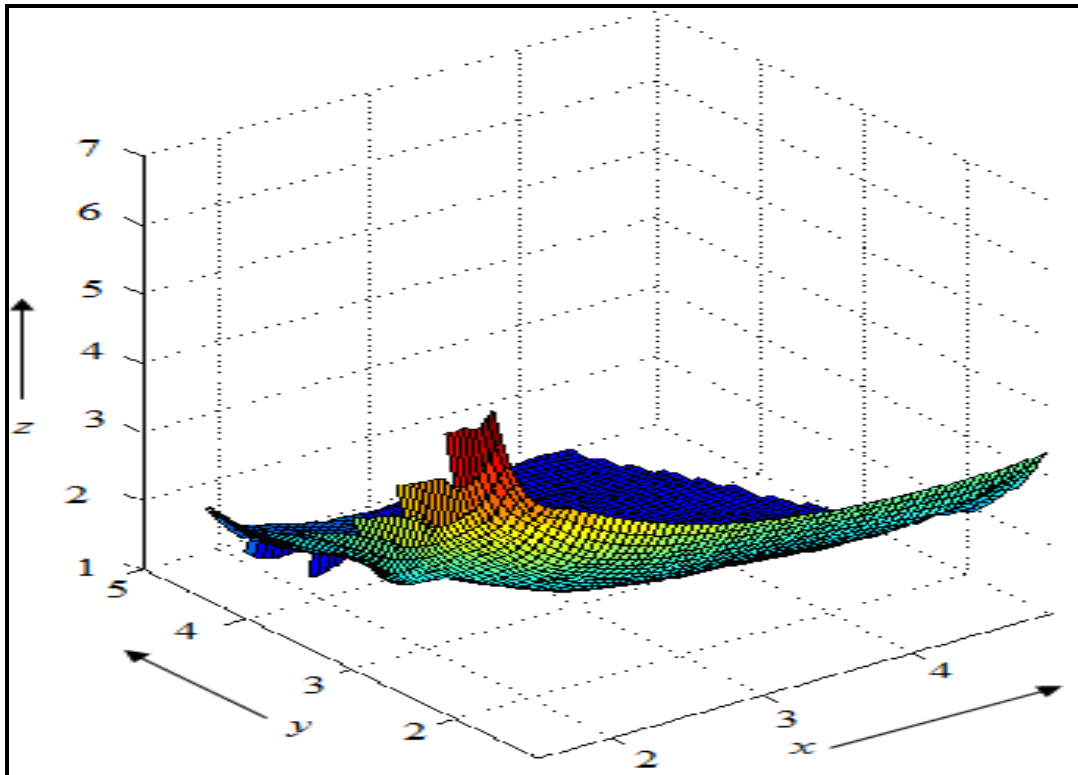


Figure 5.30: Surface plot of the nonlinear equation (5.19) with extracted 25 fuzzy rules.

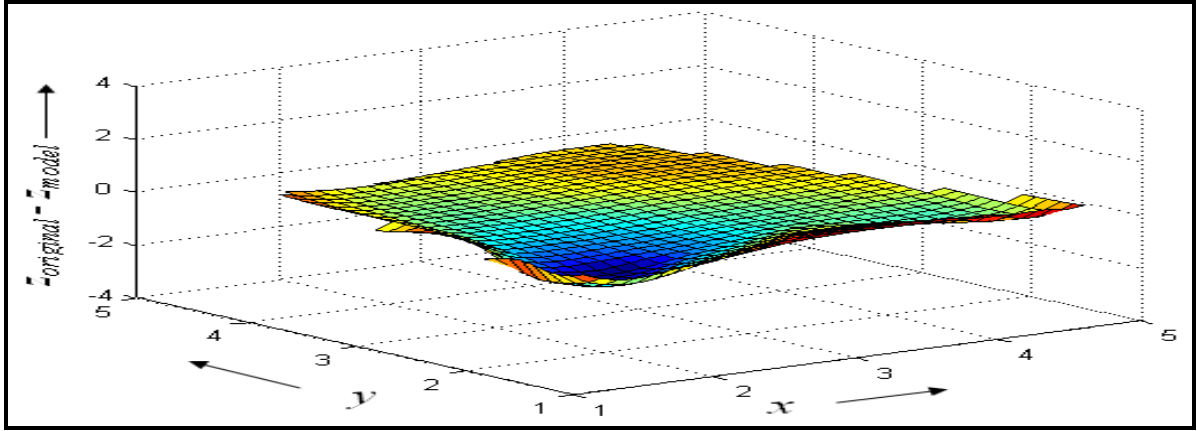


Figure 5.31: Error surface [difference between original surface (Fig. 5.28) and model surface (Fig. 5.30)].

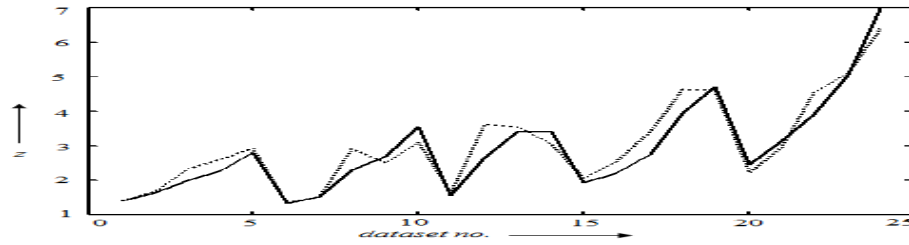


Figure 5.32: Comparative plot of *equation* (5.19) for actual output (solid line) and model output with 25 identified fuzzy rules (dotted line).

Study of Fig. 5.30 reveals that only 25 rules are sufficient to model a highly nonlinear *equation* (5.19) using our proposed scheme. We observe the error surface [difference between original output ($z_{original}$) and model output (z_{model}) with 25 rules] for the numerical example in Fig. 5.31. Using *equation* (5.9), ***MSE is calculated and the result is found 0.1582 only. Wong et al. [194] extracted 14 rules and the corresponding MSE was 0.253 for their set-up.*** The actual output and model output (using 25 rules) of *equation* (5.19) for same input data are plotted graphically in Fig. 5.32 and we observe a close resemblance between the two outputs. The identified model of the nonlinear system of (5.19) is found to be a satisfactory model considering the plots (Fig. 5.28, Fig. 5.30, Fig. 5.31 and Fig. 5.32) and MSE values.

Rule Merging

To generalize the rule merging scheme proposed in *section 5.7* is illustrated here in the function approximation problem (*equation 5.19*). The extracted fuzzy *if-then* rules from the input-output data (x, y, z) of *equation (5.19)* are already presented in Fig. 5.29. For $DSFR=0.8$ in *equation 5.16*, a total of 12 rules are developed using rule minimization scheme as represented in Fig. 5.33 and the corresponding surface plot of the identified model of the function approximation problem is depicted in Fig. 5.34(c).

Rules	x	y	z
R ₁			
	4.35	3.90	1.40
R ₂			
	3.37	2.13	1.99
R ₃			
	4.51	1.37	2.80
R ₄			
	4.14	4.76	1.33
R ₅			
	2.58	1.97	2.29
R ₆			
	4.88	1.12	3.56
R ₇			
	2.66	4.42	1.56
R ₈			
	2.03	1.88	2.66
R ₉			
	1.78	1.11	4.71
R ₁₀			
	1.48	4.44	2.44
R ₁₁			
	1.35	1.76	3.91
R ₁₂			
	1.12	1.12	6.97

Figure 5.33: Representation of 12 fuzzy *if-then* rules obtained using rule-extraction and similarity measure scheme.

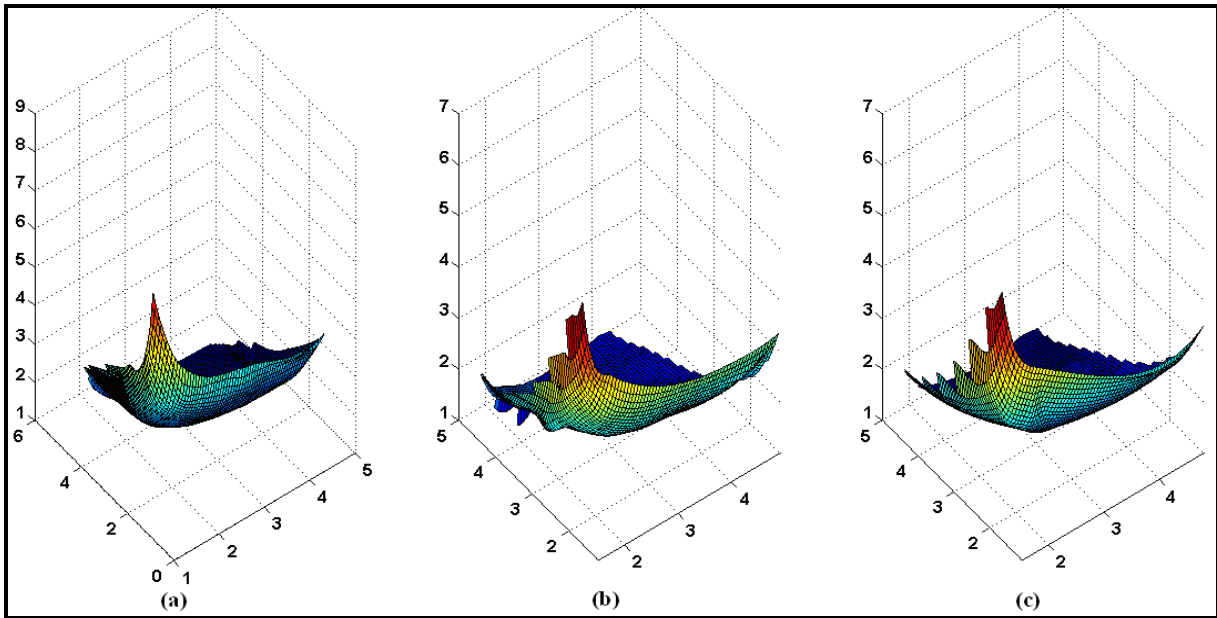


Figure 5.34: Surface plots of (a) original model, (b) model with 25 fuzzy rules, and (c) model with 12 rules.

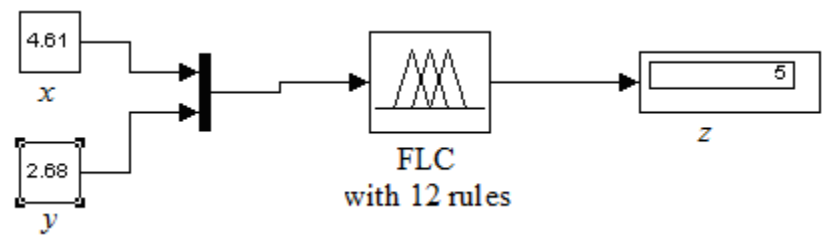


Figure 5.35: Output determination (z) for given inputs (x and y) from Model FLC designed using 12 fuzzy *if-then* rules.

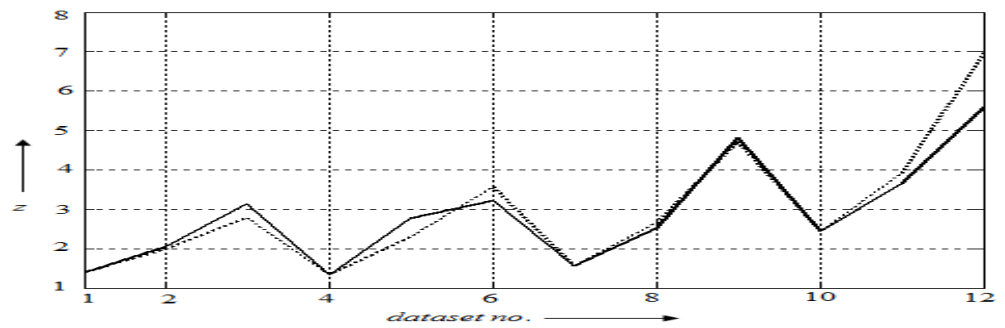


Figure 5.36: Comparative plot of actual output (dotted line) and model output (solid line)

The highly nonlinear equation (5.19) is effectively modeled with only 12 identified rules and MSE is found 0.2064, whereas Wong et al. [194] extracted 14 rules from the same equation and their corresponding MSE was 0.253. The model FLC shown in Fig. 5.35 is used to calculate output (z) from any input data (x, y) within the modeling range. The identified model quality and closeness is also verified by plotting the actual and model output for same set of inputs in Fig. 5.36.

5.9 Conclusion

In this Chapter, we proposed a SOM based technique for fuzzy rule extraction. The proposed method can extract required number of rules to model an unknown system having its input-output data. The scheme has been tested successfully in controller design as well as function approximations, which were highly nonlinear in nature. Even with significant reduction of rule-base, controllers designed by using our rule-extraction scheme exhibited close performance compared to its original fuzzy counterpart. The proposed scheme has been found to be equally well for function approximation problem. Comparative study with experimental results, control surfaces and MSE values ensured that the proposed rule extraction and fuzzy modeling technique can be used to model any complex system. In this study, we also suggested a rule merging scheme in order to further simplify the identified fuzzy model. The effectiveness of the rule merging scheme using similarity measure between fuzzy sets has also been established through different simulated examples.

CHAPTER 6

Real time implementation of SOM-based self-tuning fuzzy controller

6.1 Introduction

It is a challenging task to generate efficient fuzzy rules from the input-output data of ill-defined systems. In this chapter, the novel SOM-based approach for system identification which is integrated with rule extraction method presented in *chapter-5* is demonstrated in real time systems. The Self-Organizing Map based clustering technique already discussed in previous chapter is used here. The effectiveness of the proposed scheme is successfully tested on different real time processes like overhead crane control and water pressure control.

We already discussed about overhead crane control in *section 4.4* of *chapter-4*, where anti-sway and position control have become the requirements as a core technology. The overhead crane has one control input (trolley driving force) and two output variables (horizontal trolley position and load swing angle). This property results in a coupling effect between the load swing and cart position. In addition, uncontrolled load sway dynamics causes safety problems in crane systems, which makes it much more challenging to control. Thus the purpose of crane control is to reduce the pendulum type motion of the loads while moving the trolley to the desired position as fast as possible [163, 164]. Initially, two PD-type self-tuning fuzzy controllers (with 98 rules each) are used to deal separately with the feedback signals, swing angle and cart position and their derivatives.

The advantage of the scheme is also demonstrated to control a real time water pressure control loop [184]. The proposed rule extraction scheme is well tested in a real water pressure control loop for set-point variation and load disturbance.

In the next sections two practical systems (*i.e.*, **Overhead Crane and Water Pressure Loop**) are addressed. In each case, first we provide a brief statement about the system description. Then we illustrate the effectiveness of the proposed SOM-STFPDC₁/SOM-STFPIC₁ for such systems.

6.2 Real time systems

6.2.1 Demonstration on an overhead crane

In this section, experiments on an overhead crane are carried out to verify the performance of proposed rule extraction scheme. The laboratory scale crane set-up (FEEDBACK, UK) is shown in Fig. 6.1. The set-up is already discussed in *chapter-4*.

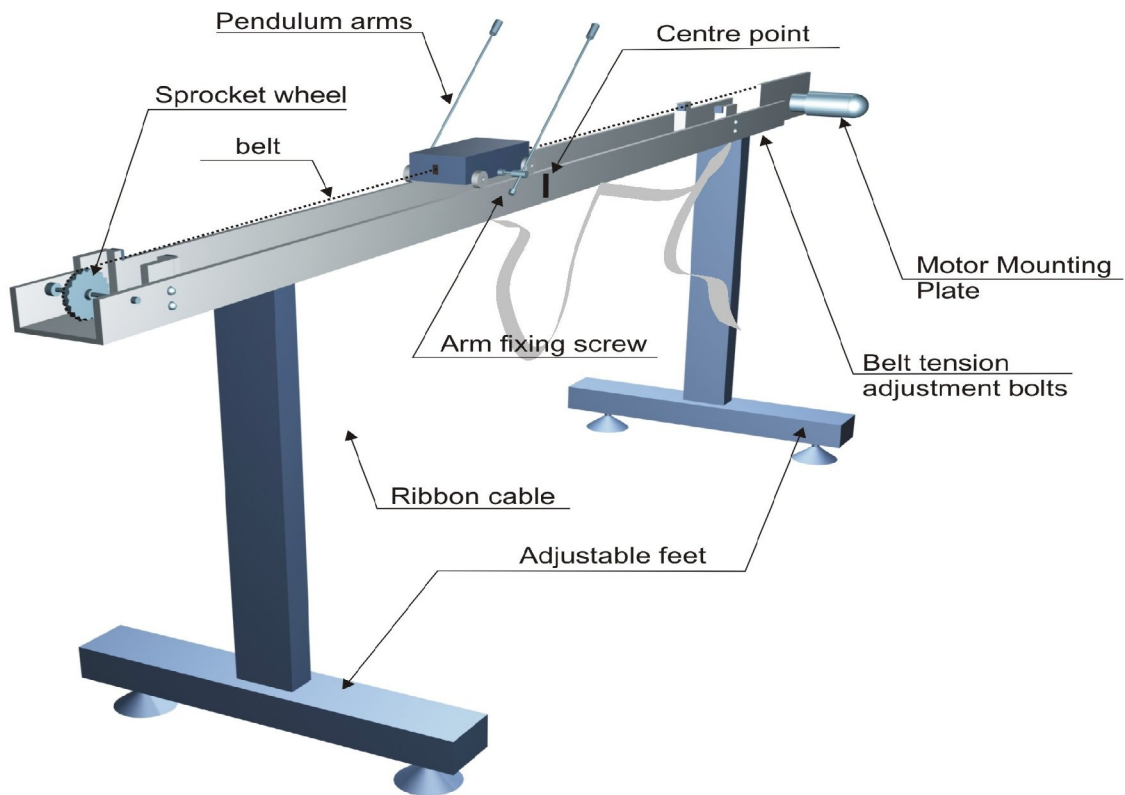


Figure 6.1: Mechanical unit of overhead crane set-up.

The equations of motion of the overhead crane system are already derived in *chapter-4* with respect to x and θ , are presented through *equation* (4.11) and (4.12).

$$(M + m)\ddot{x} + ml\ddot{\theta} \cos \theta + m\dot{l} \dot{\theta} \sin \theta + 2ml\dot{\theta} \cos \theta - ml\dot{\theta}^2 \sin \theta = u_x \quad (4.11)$$

$$l\ddot{\theta} + g \sin \theta + 2\dot{l}\dot{\theta} + \dot{x} \cos \theta = 0 \quad (4.12)$$

The different parameters [143] of the SIMO system are provided in Table 6.1.

Table 6.1: Parameters and their values of overhead crane

<i>Parameter</i>	<i>Value</i>
g - gravity	9.81 m/s ²
l – pole length	0.36 to 0.4 m (depending on the configuration)
M – cart mass	2.4 kg
m – pole mass	0.23 kg
I – moment of inertia of the pole	about 0.099kg.m ² (depends on the configuration)
b – cart friction coefficient	0.05 Ns/m
d – pendulum damping coefficient	Although negligible, necessary in the model–0.005 Nms/rad

It is difficult to control such system, which is oscillating in nature during its movement. It is essential to linearize such system for application of conventional controllers, which is not requiring for our proposed scheme. The fuzzy PD controller (FPDC), self-tuning fuzzy PD controller (STFPDC) and SOM based self-tuning fuzzy PD controller (SOM-STFPDC₁) are used to control the position and swing angle of the overhead crane. The advantage of derivative action is utilized here as derivative control anticipates the actuating error, initiates an early corrective action, and tends to increase the stability of the system.

The position controller and angle controller as shown in Fig. 6.2 deal separately with the cart position and swing angle to drive the overhead crane by using two similar STFPDC (with 49 gain rules and 49 control rules) or SOM-STFPDC₁ (with 25 gain rules and 49 control rules). The

presented scheme of Fig. 6.2 can also be used as FPDC by eliminating the automatic gain updating factor (β) portion.

In our design, the left swing of the load is defined as positive swing, while the right swing of the load is negative swing. The output of the controller for position and swing angle control are u_p and u_θ respectively as indicated in Fig. 6.2. Thus the actual control action ($u_p - u_\theta$) to drive the cart by FPDC, STFPDC and SOM-STFPDC₁ are u_{FPDC} , u_{STFPDC} and $u_{SOM-STFPDC1}$ respectively. The controllers output are used to drive the DC motor of the overhead crane for position and angle control. Error (e) due to position and error (e_θ) due to angle are obtained respectively from the cart position encoder and swing angle encoder. The ranges selected of input-output variables for position and angle controller are $[-1, +1]$ and $[-20^\circ, +20^\circ]$ respectively. Each SOM-STFPDC₁ is used only 25 gain rules instead of 49 gain rules used by STFPDC. The proposed dual control structure for crane control divides the input antecedents of fuzzy rules into two parts. Therefore, the present control scheme makes the system easier to understand. Also both the controllers are used identical control rules.

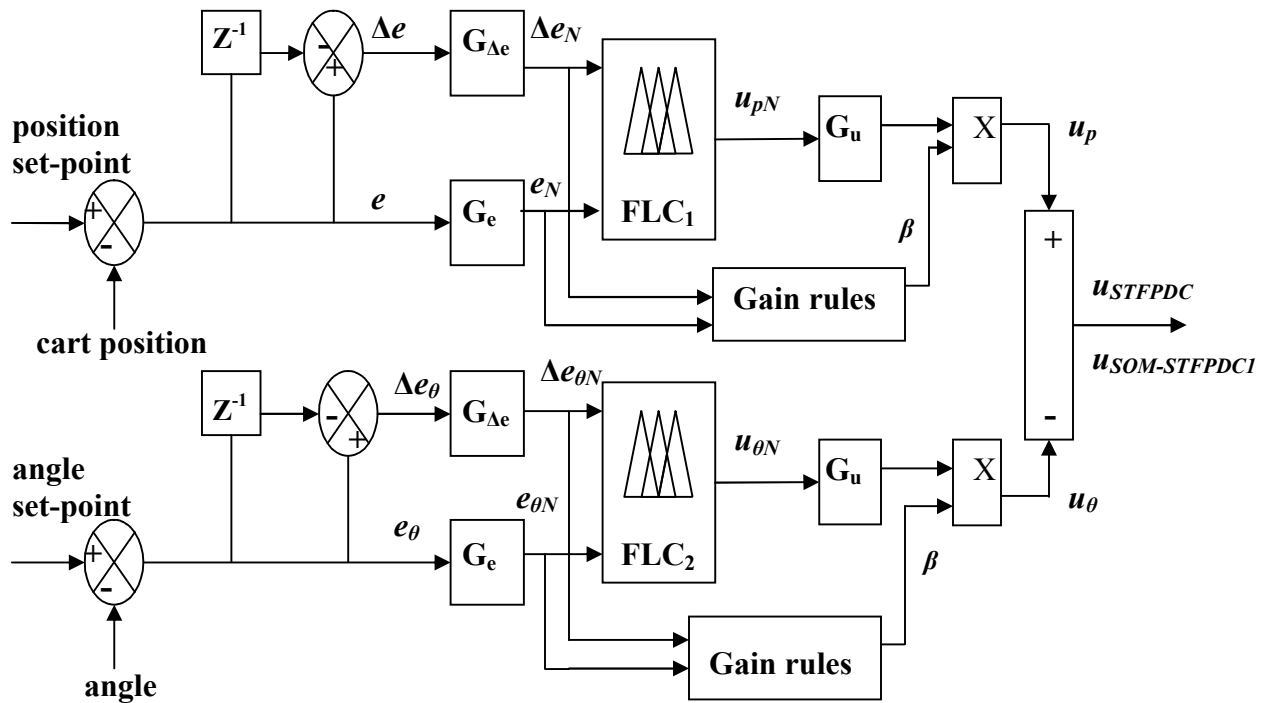


Figure 6.2: Diagram of STFPDC/SOM-STFPDC₁ for overhead crane control.

The proposed scheme is tested with square input and step input, where the load is transferred from initial position to the destination and back to the starting position. The control actions ($u_{FPDC} / u_{STFPDC} / u_{SOM-STFPDC1}$) are applied to the overhead crane to control the crane position and as well as swing angle of the load attached. The STFPDC and SOM-STFPDC₁ outperform the FPDC for different types of input applied as shown in Fig. 6.3 and Fig. 6.4. SOM-STFPDC₁ tracks the set-point very efficiently to place the trolley in its desired position. From Table 6.2, we find that different performance parameters such as *IAE* and *ITAE* are reduced by significant percentage in case of STFPDC or SOM-STFPDC₁ compared to FPDC. The performance of PID controller on overhead crane is already evaluated in *chapter-4*.

Table 6.2: Performance analysis of the controllers in overhead crane control

Reference input	Controller Type	IAE	ITAE
Step (amplitude +0.3m)	FPDC	16.79	220.05
	STFPDC	5.11	54.86
	SOM-STFPDC ₁	6.66	79.61
Square (amplitude ±0.3m)	FPDC	47.15	1283.00
	STFPDC	35.30	993.40
	SOM-STFPDC ₁	29.62	837.30

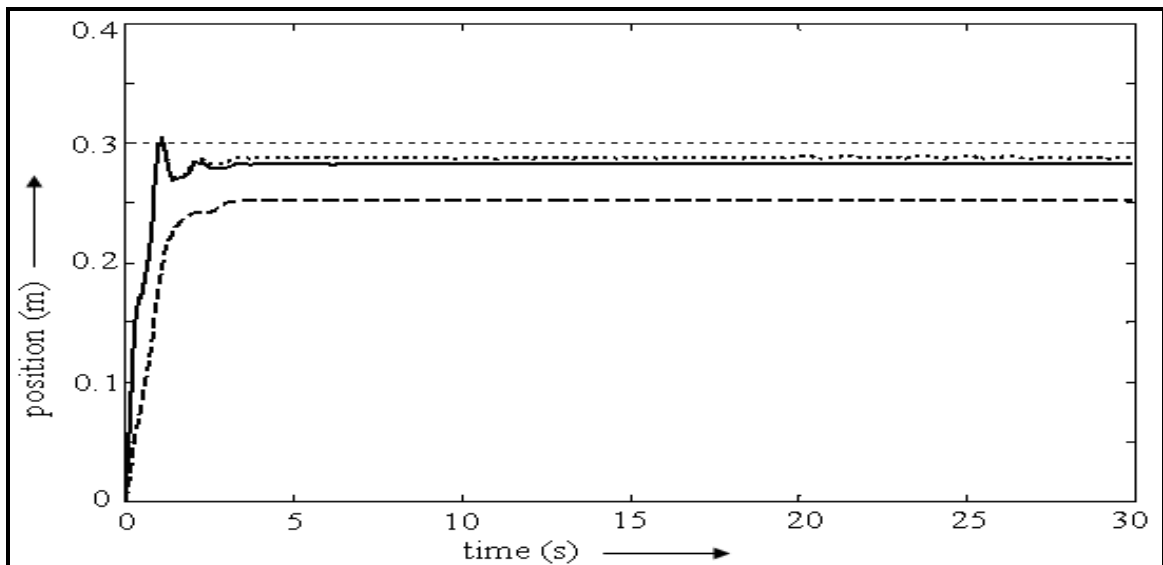


Figure 6.3: Position control for step input (0.3m) using FPDC (dashed line), STFPDC (dotted line) and SOM-STFPDC₁ (solid line).

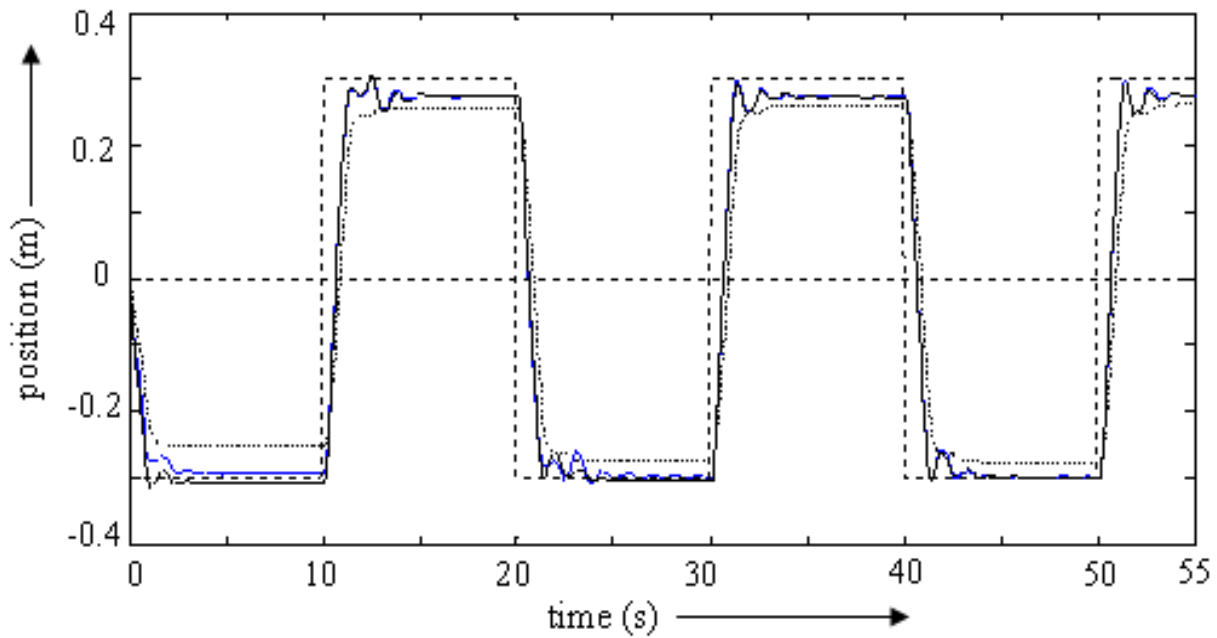


Figure 6.4: Position control for square input ($\pm 0.3m$) using FPDC (dotted-black), STFPDC (blue line) and SOM-STFPDC₁ (solid-black).

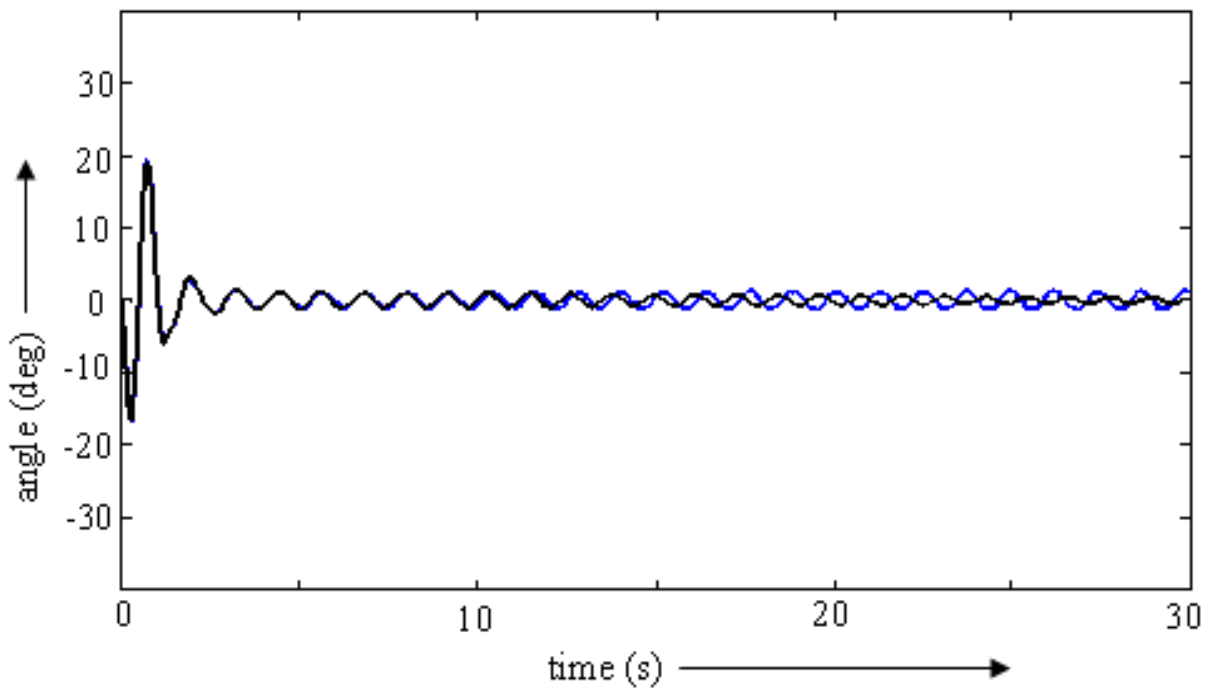
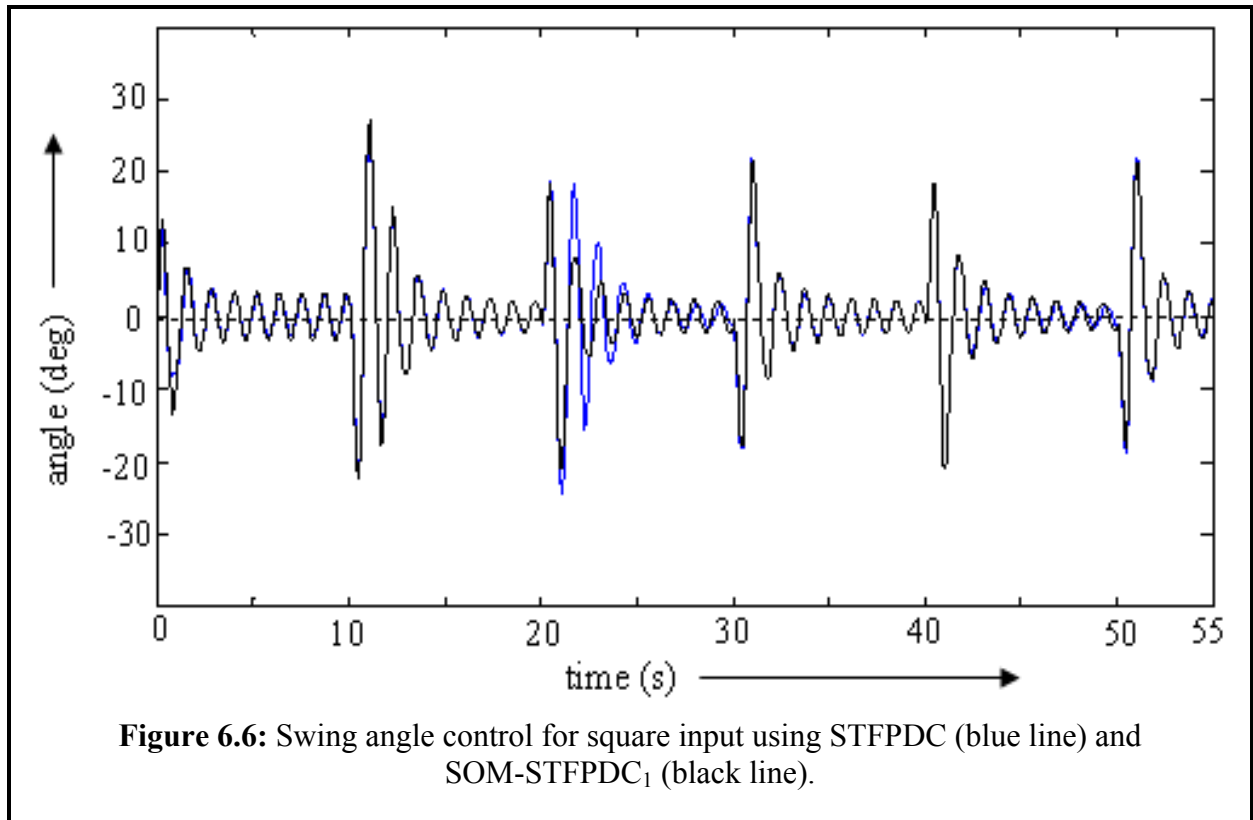


Figure 6.5: Swing angle control for step input using STFPDC (blue line) and SOM-STFPDC₁ (black line).



From Fig. 6.5 and Fig. 6.6, we observe that the load swing is minimum and especially in case of step input the swing angle approaches to almost zero for our proposed rule extraction scheme. Fig. 6.5 and Fig. 6.6 illustrate that SOM-STFPDC₁ makes negligible sway angle for horizontal movement of the trolley crane for application of step and pulse type input. The study reveals that the proposed rule extraction scheme for fuzzy controller works well and it can fix the overhead crane in its desired location with negligible sway angle.

6.2.2 Demonstration on water pressure control loop

Pressure control is an important parameter in most of the process plants. Different portions of a pressure and flow control loop are shown in Fig. 6.7 to Fig. 6.9. As shown in Fig. 6.10, the water pressure control loop consists of:

- 1) Water reservoir
- 2) Pump
- 3) Process pipe
- 4) Orifice plate

- 5) Control valve with electro-pneumatic positioner
- 6) Pressure header
- 7) Manual Valve
- 8) 3-way manifold
- 9) Compressor
- 10) Flow and pressure transmitters
- 11) Controllers, etc.

Pressure Header

The important portion of the system, pressure header is presented in Fig. 6.11, whose pressure is to be controlled using our scheme. Pressure head is a term used in fluid mechanics to represent the internal energy of a fluid due to the pressure exerted on its container. It is mathematically expressed as:

$$\psi = p/\gamma = p/\rho g \quad (6.1)$$

where,

ψ is pressure head

p is fluid pressure

γ is the specific weight

ρ is the density of the fluid

and, g is acceleration due to gravity

In this system, the opening of the control valve depends on the control action provides by the controller. Initially, the valve is in close position, when there is no control signal. The pressure header in the loop, which is being constantly pressurized by a discharge pump, is continuously monitored by pressure transmitter and pressure gauge. The desired pressure is obtained by applying suitable control action to the valve. The set-up has a provision of controlling the loop manually with conventional PID controllers. However, to make the system automatic and tune it on-line, we interface the system with the controller (designed in LABVIEW environment). The PCI6236 DAQ card is used for receiving and transmitting the data. We choose the input and output of the DAQ card, *i.e.*, 4 to 20 mA and 0 to 10 V DC respectively.



Figure 6.7: Hardware set-up of real time pressure loop.



Figure 6.8: Pneumatic control valve with positioner in real time pressure loop.



Figure 6.9: Real time pressure loop connected with PC for operation.

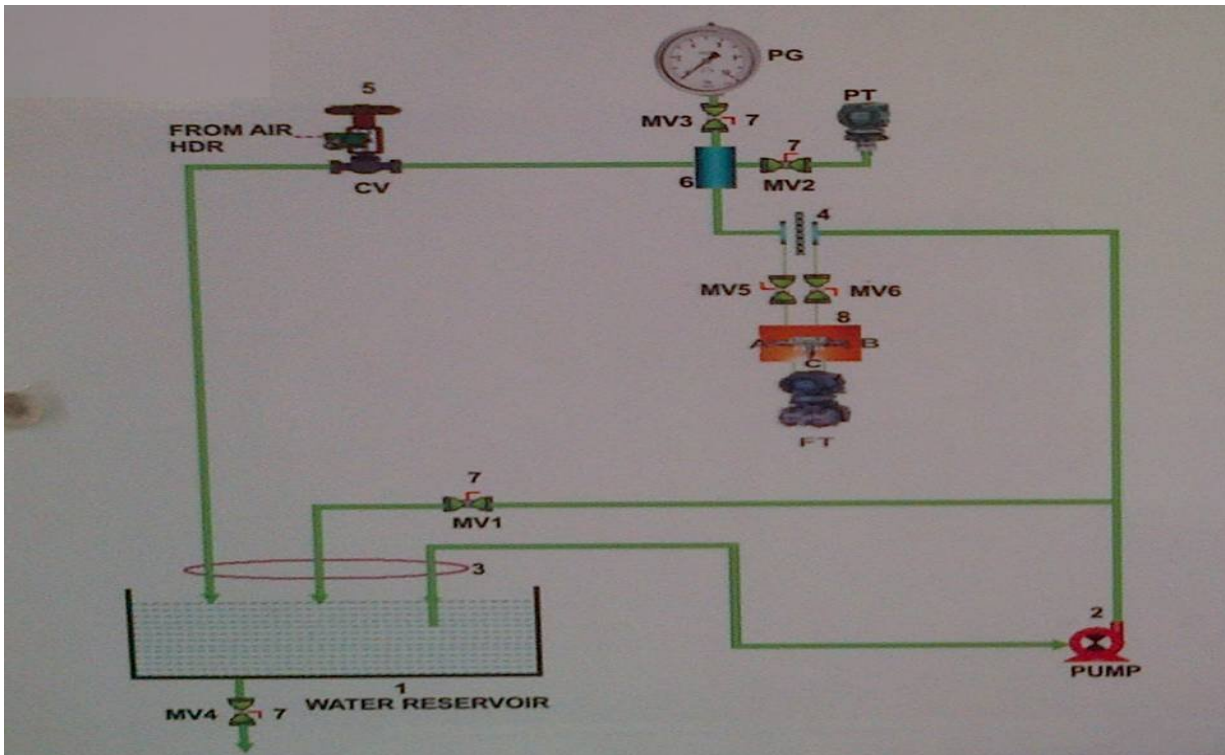


Figure 6.10: Schematic diagram of pressure loop.

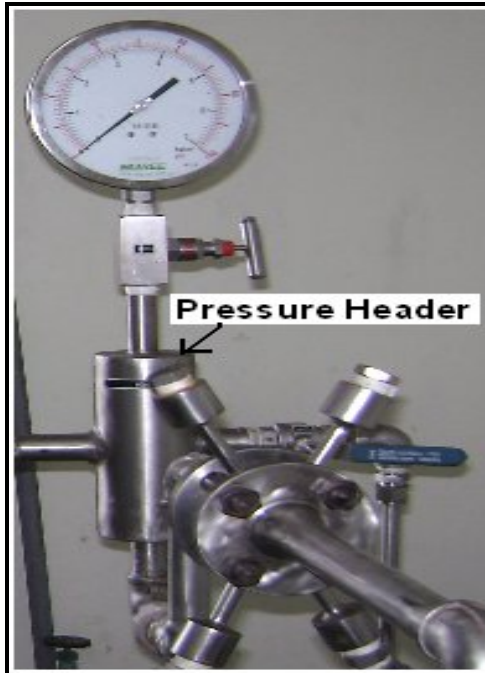


Figure 6.11: Pressure header in pressure loop.

Pressure vs. control valve opening characteristic is found inverse in nature as plotted in Fig. 6.12 and the corresponding values are shown in Table 6.3. Current (*amp*) vs. pressure (*psi*) calibration curve of the system is represented graphically in Fig. 6.13 based on the data presented in Table 6.4, which is linear in nature.

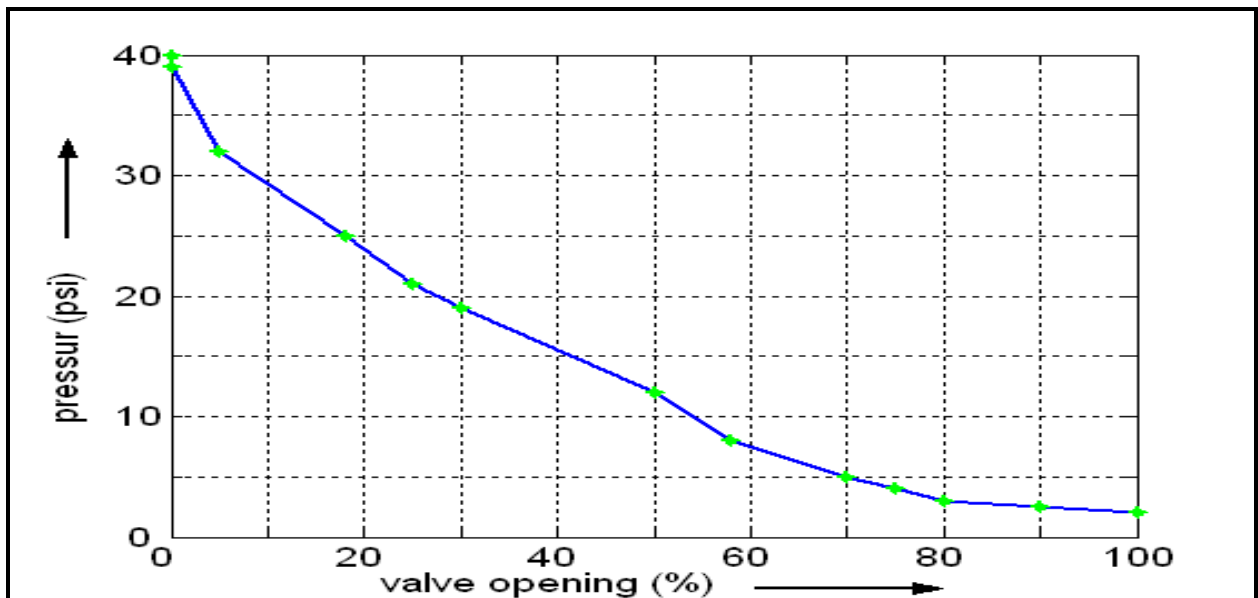


Figure 6.12: Control valve characteristics.

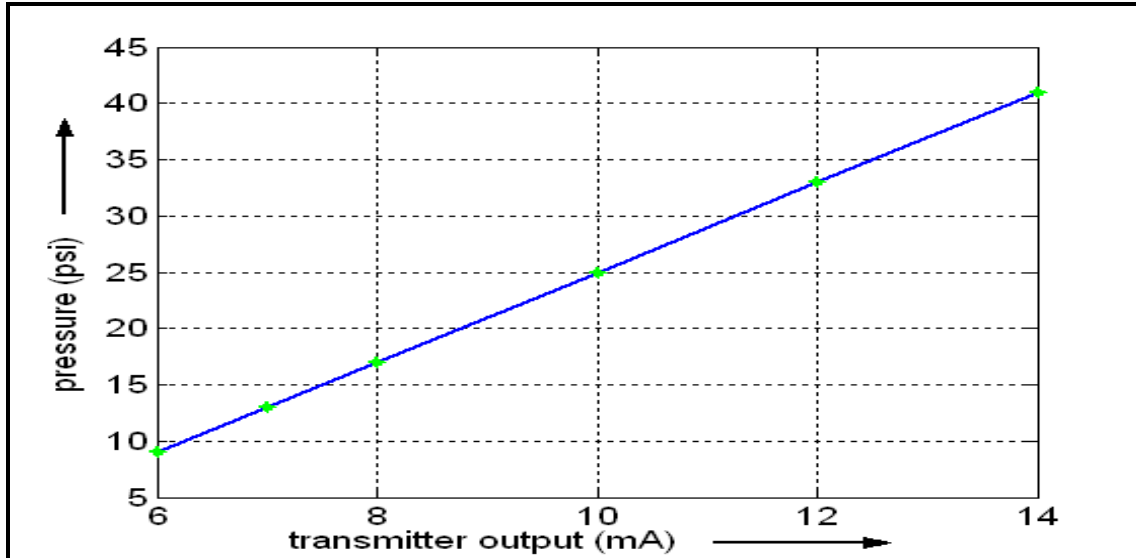


Figure 6.13: Pressure vs. current calibration curve.

Table 6.3: Valve opening and corresponding pressure

% Valve Opening	Pressure (psi)
0	40
0	39
5	32
18	25
25	21
30	19
50	12
58	8
70	5
75	4
80	3
90	2.5
100	2

Table 6.4: Pressure transmitter and corresponding pressure gauge output

Transmitter output (mA)	Pressure (psi)
6	9
7	13
8	17
10	25
12	33
14	41

In this set-up, to obtain the desired pressure, it is required to give suitable control action to the valve. Initially, we test the loop with FPIC (with 49 rules) and STFPIC (with 49 control rules and 49 gain rules) in LABVIEW environment. Input and output of the DAQ card are 4 to 20 mA and 0 to 10 $V DC$ respectively. Then the proposed SOM-STFPIC₁ (with 25 identified gain rules) is tested on the pressure control system with a constant set-point of 25 psi (10 mA). The SOM-STFPIC₁ provides almost identical performance as original STFPIC, but it outperforms the conventional Fuzzy PI controller as shown in Fig. 6.14, Fig. 6.15 and Table 6.5. Real-time experiment on the system further illustrates the advantages of the proposed rule-extraction scheme. From Table 6.5, we observe that the different performance parameters such as IAE , $ITAE$, and ISE are reduced significantly when controlled by STFPIC or SOM-STFPIC₁ compared to FPIC.

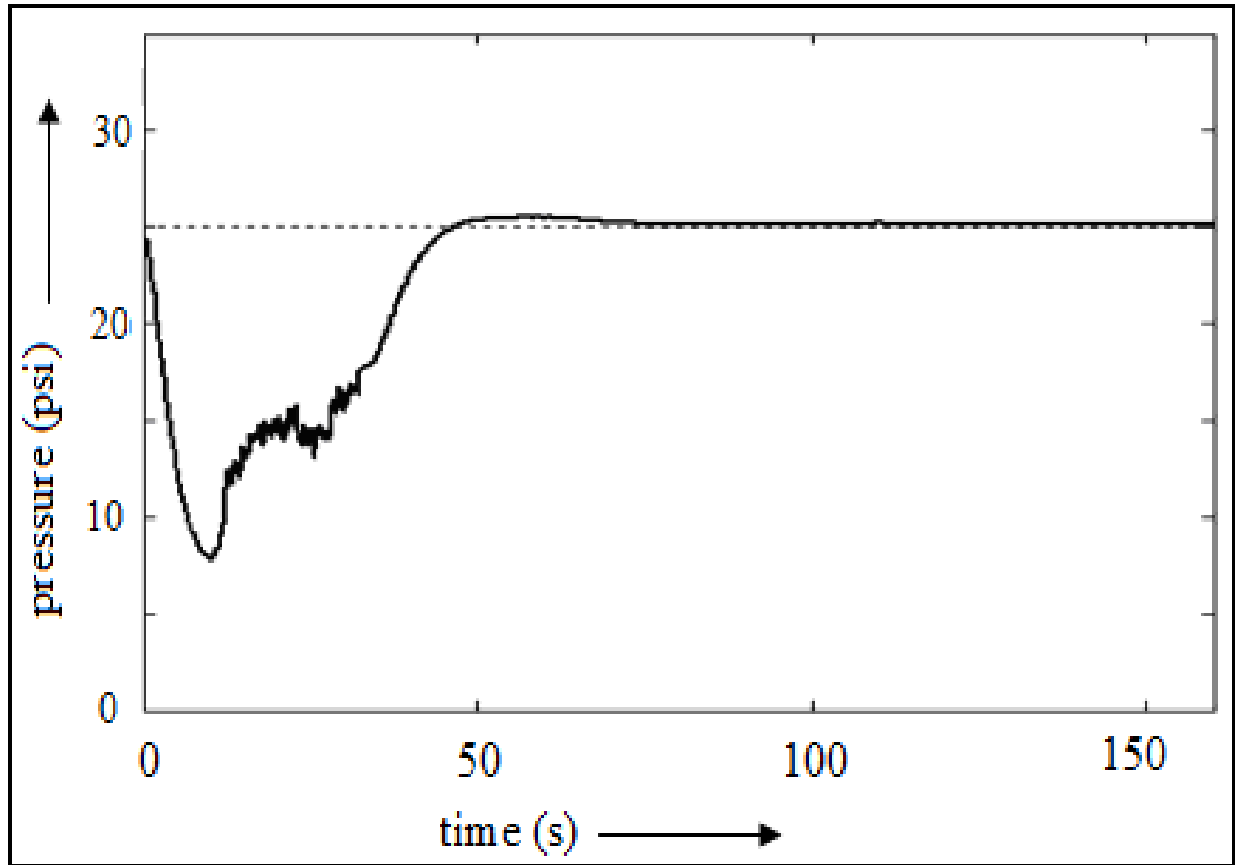


Figure 6.14: Process response for a set-point of 25 psi (10 mA) with STFPIC.

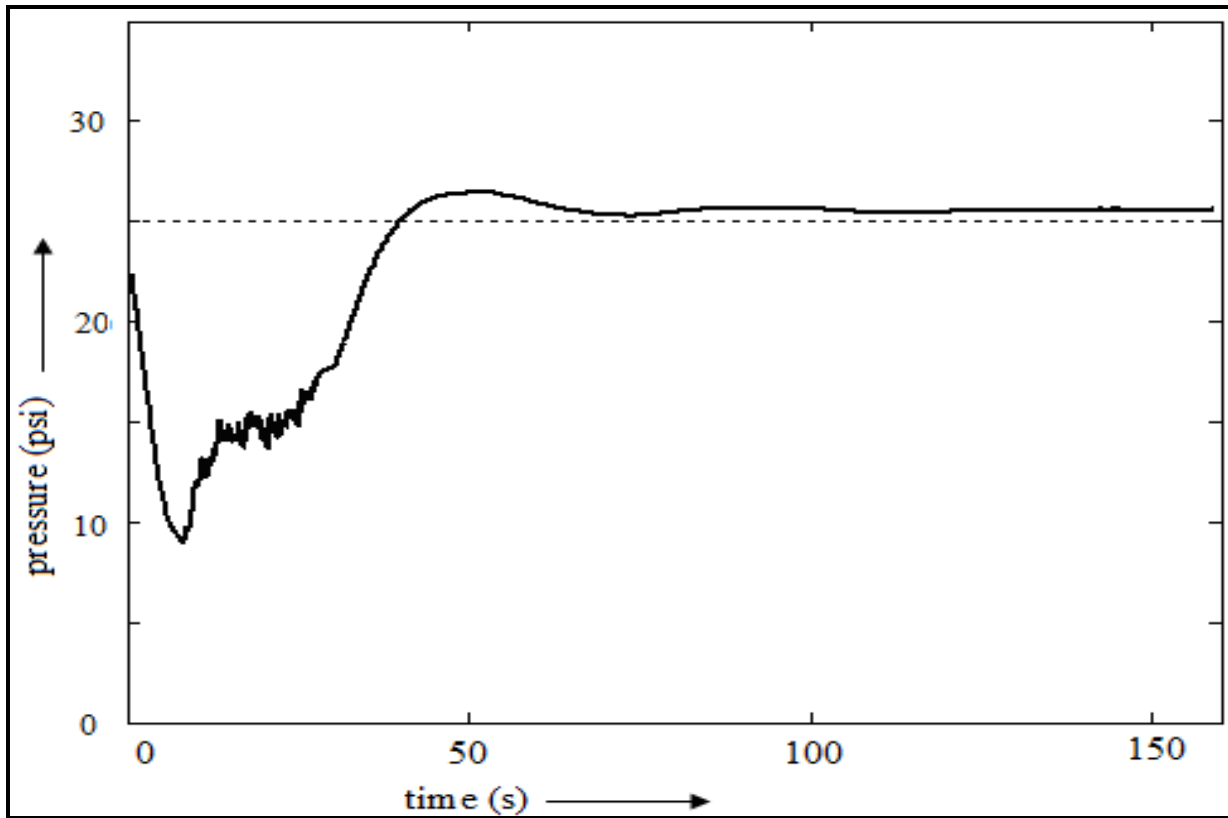


Figure 6.15: Process response for a set-point of 25 *psi* (10 *mA*) with SOM-STFPIC₁.

Table 6.5: Performance analysis of the process

Controller Type	IAE	ITAE	ISE
FPIC	65.1666	2639.100	0.2920
STFPIC	32.2104	715.446	0.1383
SOM-STFPIC ₁	32.3511	915.940	0.1138

The study of SOM-STFPIC₁ with sudden load change is depicted in Fig. 6.16. The study reveals that it can fix the system in its desired pressure easily even at load change. The pressure evolution for a set-point change from 25 to 33 *psi* and again from 33 to 25 *psi* using SOM-STFPIC₁ is presented in Fig. 6.17.

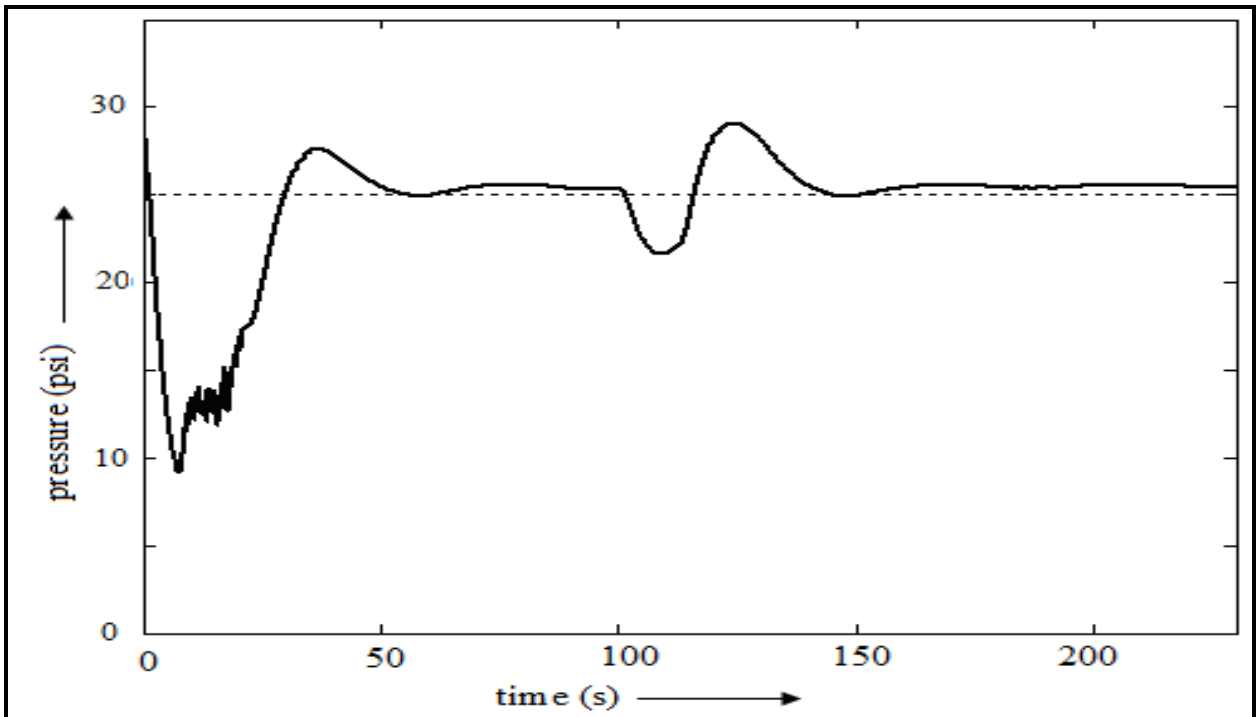


Figure 6.16: Pressure evolution using SOM-STFPIC₁ after load change at 100s.

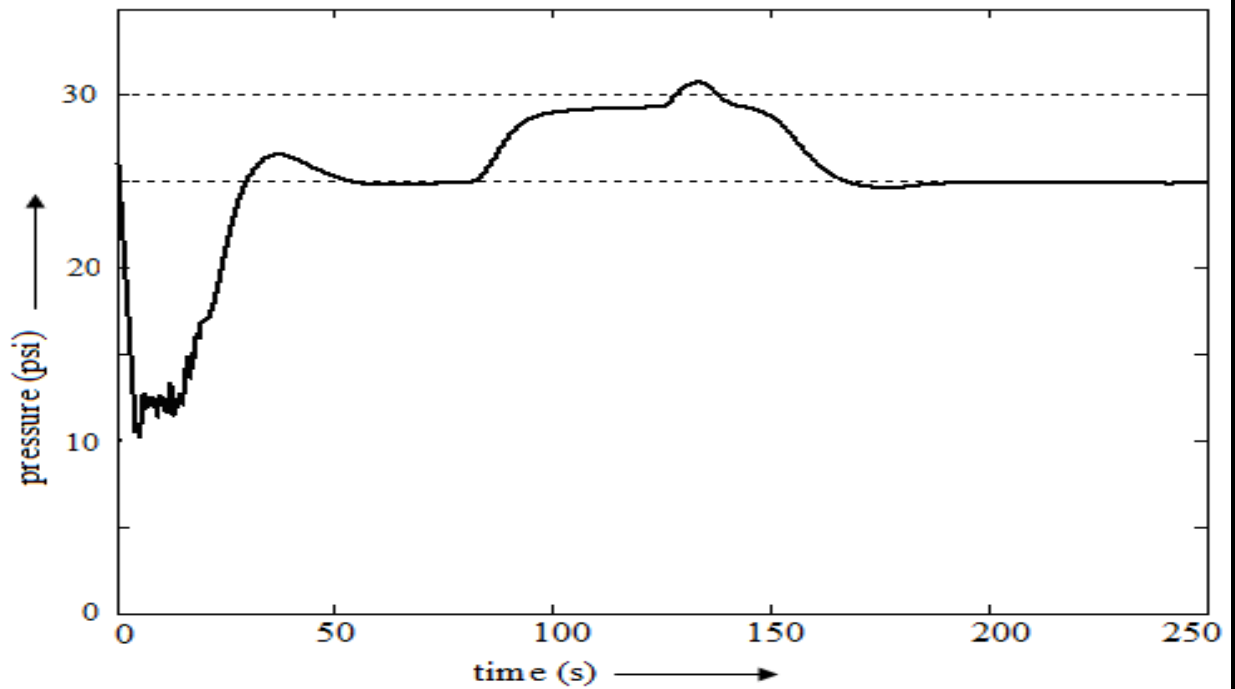


Figure 6.17: Pressure evolution after set-point changes from 25 to 33 *psi* and 33 to 25 *psi* for SOM-STFPIC₁.

6.3 Conclusion

The technique of rule extraction, which is integrated with system identification to model any unknown system, is discussed in the previous chapter. The proposed method can extract required number of rules to model an unknown system from its input-output data. In this chapter, the gain part of the fuzzy controllers are modeled using the proposed scheme and is tested in two practical real processes. Even with a significant reduction of rule-base, the developed controller exhibits effective and improved performance in real time systems compared to its conventional fuzzy counterpart. The proposed twin control scheme for overhead crane reduces the computational complexity and is easy to understand.

CHAPTER 7

Conclusion

7.1 Thesis contributions

In this chapter overall conclusion is made by highlighting the contributions. The thesis has investigated on few key problems associated with high order, nonlinear and complex systems.

The following are the salient contributions of the thesis:

- ✚ Developed efficient self-tuning scheme for fuzzy controller for application in different important processes like HVAC system and Inverted Pendulum. In this proposed scheme as discussed in *chapter-2*, we implemented an operator's strategy while running a plant. In this scheme, one can design the fuzzy *if-then* rules for fuzzy controller and as well as the fuzzy rule-base for gain updating factor according to their knowledge.
- ✚ In *chapter-2* of the thesis, a fuzzy self-tuning scheme has been studied, in which a constant SF multiplier has been considered irrespective of the process. To parameterize the output gain of STFPIC, relay feedback tuning approach has been proposed in *chapter-3*. Instead of fixed gain, this technique helped us to update the output SF of the self-tuning fuzzy controller on-line based on the process trend as well as the dynamics of the system. This modification of tuning method worked effectively in different linear and nonlinear systems with varying dead-time. Even a considerable reduction of rule-base did not deteriorate the performance compared to its conventional fuzzy and non-fuzzy controllers. The scheme has also been successfully implemented for the speed control of a DC motor.

- ✚ A new non-fuzzy auto-tuning method has been proposed and its application to real time process control has been investigated in *chapter-4*. Gain updating fuzzy rules used to tune the controllers adds some processing time, in view of that in *chapter-4* a non-fuzzy tuning scheme has been implemented. The most important feature of the proposed scheme was that it depends neither on the process being controlled nor on the controller used. Even with significant reduction of rule-base, adaptive fuzzy controller exhibited effective and improved performance compared to its conventional fuzzy and non-fuzzy controllers for wide variety of second order integrating, nonlinear and non-minimum phase system with varying dead-time. Other contribution of this chapter was the twin control scheme for overhead crane that reduced the computational complexity to a great extent.
- ✚ In *chapter-2*, 49 control rules and 49 gain rules were used to realize a system. But the question was- did we really need that much of rules? In *chapter-5*, we tried to find the answer of this query using SOM-based rule extraction scheme. Identification of ill-defined and uncertain system is a very difficult task. In such cases, it is required to extract input-output relationship based on the information obtained from the system, so that one can describe the input-output behavior of a given system by a set of fuzzy rules. In this context, in *chapter-5*, we proposed a technique of rule extraction which was integrated with system identification to model any unknown system. The proposed method could extract required number of rules for an unknown system from its input-output data. The proposed scheme has been tested successfully in controller design, even with significant reduction of rule-base; controllers exhibited improved performance in different simulated systems. Comparative performance study ensured that the proposed rule extraction technique could be successfully implemented to model any complex process. In the proposed rule extraction scheme we observed many similar type of fuzzy sets or MFs identified in the rule-base. To reduce the number of MFs in *chapter-5*, we have also suggested a similarity measure based technique to refine the extracted fuzzy model.
- ✚ In *chapter-6*, the performance of rule extraction scheme has been studied in different real time systems like overhead crane control and water pressure control.

7.2. Future scope

It has been established that self-tuning and adaptive fuzzy PI and PD controller can be useful for complex systems, however further work is require to understand the extent of its usefulness. Some proposals in this regard are presented next.

- The utility of the proposed self-tuning and adaptive fuzzy scheme can be demonstrated in other complex processes and with others types of fuzzy controllers (*e.g.*, Type-2 Fuzzy Controller).
- We know that the output scaling factor or gain of the fuzzy controller is a very important parameter. We implemented the concept of dynamic gain in *chapter- 3* in this regard. In place of relay feedback tuning other adaptive or tuning schemes may be checked for further fine tuning of output gain.
- A new adaptive scheme for fuzzy controller is proposed in this thesis, but still there is scope for further research on development of more effective adaptive fuzzy schemes.
- Investigation can also be made to check the stability of the nonlinear systems discussed.
- The proposed system identification or rule extraction scheme is based on unsupervised SOM clustering technique. Other clustering techniques can be tried for this purpose.
- Apart from the possibilities mentioned above, the work carried out in the present thesis can be extended in many directions. However we like to emphasis on rule reduction scheme. In this regard, we have highlighted similarity measure scheme in *chapter- 5*.

BIBLIOGRAPHY

- [1] F. D. S. Cardoso, J. F. Martins and V. F. Pires, "A comparative study of a PI, Neural Network and Fuzzy Genetic approach controllers for an AC drive", *5th International Workshop on Advanced Motion Control*, pp.375-380, 1998.
- [2] Y. Zhongming and W. Bin, "A review on induction motor online fault diagnosis", *Proceedings of 3rd International Power Electronics and Motion Control Conference*, vol.3, pp.1353-1358, 2000.
- [3] R. Echavarria, S. Horta and M. Oliver, "A three phase motor drive using IGBTs and constant V/F speed control with slip regulation", *Proceedings of 4th IEEE International Power Electronics Congress*, pp.87-91, 1995.
- [4] A. Munoz-Garcia, T. A. Lipo and D. W. Novotny, "A new induction motor V/F control method capable of high-performance regulation at low speeds", *IEEE Transactions on Industry Applications*, vol.34(4), pp.813-821, 1998.
- [5] M. G. Simoes, B. K. Bose and R. J. Spiegel, "Fuzzy logic based intelligent control of a variable speed cage machine wind generation system", *IEEE Transactions on Power Electronics*, vol.12, pp.87-95, 1997.
- [6] A. K. Pal and J. Chakraborty, "Adaptive fuzzy control of inverted pendulum with fuzzy-based set-point weighting scheme", *Proceedings of IEEE international Conference of Emerging Applications of Information Technology*, pp.46-51, 2014.
- [7] R. R. Yager and L. A. Zadeh, "An introduction to fuzzy logic application in intelligent systems", *The Springer International Series in Engineering and Computer Science*, 1992.
- [8] H. J. Zimmermann, "Fuzzy sets theory and its applications", *Dordrecht: Kluwer, Academic Publishers*, 1993.
- [9] J. H. Kim, K. C. Kim and E. K. P. Chong, "Fuzzy pre-compensated PID controllers", *IEEE Transactions on Control System Technology*, vol.2(4), 1994.
- [10] C. H. Joon, C. K. Bo and W. B. Hyeun, "Fuzzy-PID hybrid control: automatic rule generation using genetic algorithm", *Fuzzy Sets and Systems*, vol.92(3), pp.305-316, 1997.
- [11] W. Z. Qiao and M. Mizumoto, "PID type fuzzy controller and parameters adaptive method", *Fuzzy Sets and Systems*, vol.78(1), pp.23-35, 1996.
- [12] M. Sugeno, "Industrial applications of fuzzy control", *Elsevier Science Inc.*, 1985.
- [13] M. Sugeno and K. Tanaka, "Successive identification of a fuzzy model and its application to prediction of a complex system", *Fuzzy Sets and Systems*, vol.42, pp.315-334, 1991.
- [14] M. Sugeno and T. Yasukawa, "A fuzzy logic based approach to qualitative modeling", *IEEE Transactions on Fuzzy System*, vol.1(1), pp.7-31, 1993.
- [15] R. M. Tong, "The evaluation of fuzzy models derived from experimental data", *Fuzzy Sets and Systems*, vol.4(1), pp.1-12, 1980.
- [16] R. Palm, "Sliding mode fuzzy control", *Proceedings of Fuzz IEEE*, pp.519-526, 1992.

- [17] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control", *IEEE Transactions on Systems, Man, Cybernetics*, vol.15(1), pp.116-132, 1985.
- [18] W. Pedrycz, "An identification algorithm in fuzzy relational systems", *Fuzzy Sets and Systems*, vol.13, pp.153-167, 1984.
- [19] W. Pedrycz and J. V. D. Oliveira, "Optimization of fuzzy models", *IEEE Transactions on System, Man, Cybernetics*, vol.26(4), pp.627-634, 1996.
- [20] R. Alcalá, J. Casillas, O. Cordon, A. Gonzalez and F. Herrera, "A genetic rule weighting and selection process for fuzzy control of heating, ventilation and air conditioning systems", *Engineering application of Artificial Intelligence*, vol.28, pp.279-296, 2005.
- [21] Q. Xiong, W. J. Cai and M. He, "A practical decentralized PID auto-tuning method for TITO systems under closed-loop control", *International Journal of Innovative Computing, Information and Control*, vol.2(2), pp.305-322, 2006.
- [22] R. K. Mudi and N. R. Pal, "A robust self-tuning scheme for PI and PD type fuzzy controllers", *IEEE Transactions on Fuzzy Systems*, vol.7(1), pp.2-16, 1999.
- [23] A. K. Pal and R. K. Mudi, "Self-tuning fuzzy PI controller and its application to HVAC system", *International Journal of Computational Cognition*, vol.6(1), pp.25-30, 2008.
- [24] Z. Gao, T. A. Trautzsch and J. G. Dawson, "A stable self-tuning fuzzy logic control system for industrial temperature regulation", *IEEE Transactions on Industry Applications*, vol.38(2), pp.414-424, 2002.
- [25] C. H. Chou and H. C. Lu, "A heuristic self-tuning fuzzy controller", *Fuzzy Sets and Systems*, vol.61(3), pp.249-264, 1994.
- [26] J. H. Chiang, "Support vector learning mechanism for fuzzy rule-based modeling: A New Approach", *IEEE Transactions on Fuzzy Systems*, vol.12(1), pp.1-12, 2004.
- [27] A. K. Pal and S. K. Bag, "Characteristics of user defined neuro-fuzzy controller for nonlinear processes", *Journal of Systems Science and Engineering*, vol.15(2), pp.53-60, 2007.
- [28] Y. Lin and G. A. Cunningham III, "A new approach to Fuzzy-Neural system modeling", *IEEE Transactions on Fuzzy Systems*, vol.3(2), pp.190-198, 1995.
- [29] S. Mitra and Y. Hayashi, "Neuro-Fuzzy rule generation: survey in soft computing framework", *IEEE Transactions on Neural networks*, vol.11(3), pp.748-768, 2000.
- [30] K. Ogata, "Modern Control Engineering", *Englewood Cliffs, NJ: Prentice-Hall*, 1970.
- [31] F. G. Shinskey, "Process control systems-application, design, and tuning", *New York: McGraw-Hill*, 1998.
- [32] C. Bissel, "Control engineering", *New York: Chapman and Hall*, 1994.
- [33] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers", *Transactions of the ASME*, vol.64, pp.759-68, 1942.
- [34] J. C. Basilio and S. R. Matos, "Design of PI and PID controllers with transient performance specification", *IEEE Transactions on Education*, vol.45(4), pp.364-70, 2002.

- [35] R. K. Mudi, C. Dey and T. T. Lee, "An improved auto-tuning scheme for PI controllers", *ISA Transactions*, vol.47, pp.45-52, 2008.
- [36] L. A. Zadeh, "Fuzzy sets", *Information and Control*, vol.8, pp.338-353, 1965.
- [37] J. S. R. Jang, C. T. sun and E. Mizutani, "Neuro-Fuzzy and Soft Computing", *Pearson Education*, 2004.
- [38] James Vernon, "Fuzzy logic system", *Control systems principles, UK*, pp.1-12, 2006.
- [39] S. N. Sivanandam and S. N. Deepa, "Fuzzy set theory", *Principles of soft computing, Wiley-India edition*, 2010.
- [40] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller", *International Journal of Man-Machine Studies*, vol.7(1), pp.1-13, 1975.
- [41] I. Iancu, "A Mamdani type fuzzy logic controller", *Published by Intech.*, chapter(16), pp. 325-350, 2012.
- [42] D. Dirankov, H. Hellendorn and M. Reinfrank, "An introduction to fuzzy control", *New York: Springer-Verlag*, 1993.
- [43] T. J. Ross, "Fuzzy Logic with engineering applications", *McGraw-Hill, Inc.*, 1995.
- [44] C. J. Harris, C. G. Moore and M. Brown, "Intelligent control - aspects of fuzzy logic and neural nets", *Singapore : World Scientific*, 1993.
- [45] F. O. Karry and C. D. Silva, "Soft Computing and intelligent systems design", *Pearson*, 2009.
- [46] S. Haykin, "Neural Networks", *Pearson Education (Singapore), Indian Branch*, 6th reprint, 2004.
- [47] J. C. Bezdek, R. Ehrlich and W. Full, "FCM: The fuzzy C-Means clustering algorithm", *Computers and Geosciences*, vol.10(2/3), pp.191-203, 1984.
- [48] R. R. Yager and D. P. Filev, "Approximate clustering via the mountain method", *IEEE Transactions on Systems, Man, Cybernetics*, vol.24(8), pp.1279-1284, 1994.
- [49] S. Chiu, "Method and software for extracting fuzzy classification rules by subtractive clustering", *Proceedings of North American Fuzzy Information Processing*, pp.461-465, 1996.
- [50] T. Kohonen, "Self-Organizing Maps", *Springer Series in Information Sciences, Springer, Berlin, Heidelberg, New York*, vol.30(3), 1995, 1997, 2001.
- [51] M. Yoshida, Y. Tsutsumi and T. Ishida, "Gain tuning method for design of fuzzy control systems", *Proceedings of International Conference on Fuzzy Logic Neural Networks, Fukuoka, Japan*, pp.405-408, 1990.
- [52] T. Iwasaki and A. Morita, "Auto-tuning controller with fuzzy identification", *Proceedings of International Conference on Fuzzy Logic Neural Networks, Fukuoka, Japan*, pp.401-404, 1990.
- [53] S. Hayashi, "Auto-tuning fuzzy PI controller", *Proceedings of International Fuzzy System Association, Brussels, Belgium*, pp.41-44, 1991.

- [54] H. Nomura, I. Hayashi and N. Wakami, "A self-tuning method of fuzzy control by decent method", *Proceedings of International Fuzzy System Association*, Brussels, Belgium, pp.155-158, 1991.
- [55] W. C. Daugherty, B. Rathakrishnan and J. Yen., "Performance evaluation of a self-tuning fuzzy controller", *Proceedings of IEEE International Conference on Fuzzy Systems*, pp.389-397, 1992.
- [56] Y. Maeda, J. Murakami and T. Takagi, "FRASH – A fuzzy real-time auto-tuning shell with expressive function of fuzzy algorithms", *Singapore International Conference on Intelligent Control and Instrumentation*, pp.1363-1368, 1992.
- [57] L. Zheng, "A practical guide to tune of proportional and integral (PI) like fuzzy controllers", *Proceedings of Fuzz IEEE, San Diego, CA*, pp.633-641, 1992.
- [58] M. Maeda and S. Murakami, "A self-tuning fuzzy controller", *Fuzzy Sets and Systems*, vol.51, pp.29-40, 1992.
- [59] S. Z. He, S. Tan, F. L. Xu and P. Z. Wang, "Fuzzy self-tuning of PID controller", *Fuzzy Sets and Systems*, vol.56, pp.37-46, 1993.
- [60] H. K. Tonshoff and A. Walter, "Self-tuning fuzzy-controller for process control in internal grinding", *Fuzzy Sets and Systems*, vol.63, pp.359-373, 1994.
- [61] J. Lee, "On methods for improving performance of PI-type fuzzy logic controllers", *IEEE Transactions on Fuzzy Systems*, vol.1, pp.298-301, 1993.
- [62] R. K. Mudi and N. R. Pal, "A note on fuzzy PI-type controllers with resetting action", *Fuzzy Sets and Systems*, vol.121, pp.149-159, 2001.
- [63] H. C. Lui, M. K. Gu, T. H. Goh and P. Z. Wang, "A self-tuning adaptive resolution (STAR) fuzzy control algorithm", *IEEE World Congress on Computational Intelligence*, vol.3, pp.1508-1513, 1994.
- [64] K. B. Ramkumar and M. Chidambaram, "Fuzzy self-tuning PI controller for bioreactors", *Bioprocess Engineering*, vol.12(5) pp.263-267, 1995.
- [65] C. H. Jung, C. S. Ham and Kuhn-II Lee, "A real-time self-tuning fuzzy controller through scaling factor adjustment for the steam generator of NPP", *Fuzzy Sets and Systems*, vol.74(1), pp.53-60, 1995.
- [66] E. Chiricozzi, F. Parasiliti, M. Tursini and D. Q. Zhang, "Fuzzy self-tuning PI control of PM synchronous motor drives", *Proceedings of IEEE international conference on power electronics and drive systems, Singapore*, pp.749-754, 1995.
- [67] K. Shimojima, T. Fukuda and Y. Uchikawa, "Self-tuning fuzzy modeling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm", *Fuzzy Sets and Systems*, vol.71(3), pp.295-309, 1995.
- [68] R. Palm, "Scaling of fuzzy controller using the cross-correlation", *IEEE Transactions on Fuzzy Systems*, vol.3, pp.116-123, 1995.
- [69] H. X. Li and H. B. Gatland, "Conventional fuzzy control and its enhancement", *IEEE Transactions on Systems, Man, Cybernetics*, vol.26(5), pp.791-797, 1996.

- [70] H. Miyata and M. Ohkita, "Self-Tuning of fuzzy reasoning by the steepest descent method and its application to a parallel parking", *IEICE Transactions on Information and Systems*, vol.E79-D(5), pp.561-569, 1996.
- [71] D. H. Wang and T. Y. Chai, "A robust adaptive control algorithm via neural network model", *Acta Automatica Sinica*, vol.22(4), pp.447-451, 1996.
- [72] A. Routray, P. K. Dash and S. K. Panda, "A fuzzy self-tuning PI controller for HVDC links", *IEEE Transactions on power Electronics*, vol.11(5), pp.669-679, 1996.
- [73] J. Y. Chen and Y. H. Lin, "A self-tuning fuzzy controller design", *Proceedings of IEEE international Conference on Neural Networks*, vol.3, pp.1358-1362, 1995.
- [74] Y. G. Leu, T. T. Lee and W. Y. Wang, "On-line tuning of fuzzy-neural network for adaptive control of nonlinear dynamical systems", *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics*, vol.27(6), pp.1034-1043, 1997.
- [75] S. Takagi, T. Oki, T. Yamamoto and M. Kaneda, "A skill-based PID controller using artificial neural networks", *IEEE International Conference on Neural Networks*, vol.5, pp.4454-4459, 1997.
- [76] H. Ying, "Constructing nonlinear variable gain controllers via the Takagi-Sugeno fuzzy control", *IEEE Transactions on Fuzzy Systems*, vol.6(2), pp.226-234, 1998.
- [77] H. Y. Chung, B. C. Chen and J. J. Lin, "A PI-type fuzzy controller with self-tuning scaling factors", *Fuzzy Sets and Systems*, vol.93, pp.23-28, 1998.
- [78] Z. W. Woo, H. Y. Chung, J. J. Lin, "A PID type fuzzy controller with self-tuning scaling factors", *Fuzzy Sets and Systems*, vol.115, pp.321-326, 2000.
- [79] W. D. Chang, R. Hwang and J. G. Chsieh, "A self-tuning PID control for a class of nonlinear system based on Lyapunov approach", *Journal of Process Control*, vol.12, pp.233-242, 2002.
- [80] E. Yesil, M. Guzelkaya and I. Eksin, "Self tuning fuzzy PID type load and frequency controller", *Energy Conversion and Management*, vol.45(3), pp.377-390, 2004.
- [81] S. J. Huang and H. Y. Chen., "Adaptive sliding controller with self-tuning fuzzy compensation for vehicle suspension control", *Mechatronics*, vol.16(10), pp.607-622, 2006.
- [82] J. MacQueen, "Some methods for classification and analysis of multivariable observations", *Proceedings of the 5th Berkeley Symposium on Mathematical Statics and Probability*, pp.281-297, 1967.
- [83] I. S. Dhillon, Y. Q. Guan and J. Kogan, "Iterative clustering of high dimensional text data augmented by local search", *Proceedings of IEEE International Conference on Data Mining*, pp.131-138, 2002.
- [84] J. C. Dunn, "A fuzzy relative of the iso-data process and its use in detecting compact well-separated clusters", *Journal of Cybernetics*, vol.3(3), pp.32-57, 1973.
- [85] F. Hoppner, F. Klawonn, R. Kruse and T. Runkler, "Fuzzy cluster analysis: Methods for classification, data analysis and image recognition", *John Wiley and Sons*, 1999.
- [86] Y. Yoshinari, W. Pedrycz, and K. Hirota, "Construction of fuzzy models through clustering techniques", *Fuzzy Sets and Systems*, vol.54(2), pp.157-166, 1993.

- [87] R. R. Yager and D. P. Filev, "Generation of fuzzy rules by mountain clustering", *Journal of Intelligent Fuzzy Systems*, vol.2, pp.209-219, 1994.
- [88] S. L. Chiu, "Extracting fuzzy rules for pattern classification by cluster estimation", *Proceedings of International Fuzzy System Association*, pp.1-4, 1995.
- [89] S. L. Chiu, "Fuzzy model identification based on cluster estimation", *Journal of Intelligent Fuzzy System*, vol.2, pp.267-278, 1994.
- [90] R. Babuska and U. Kaynak, "Application of compatible cluster merging to fuzzy modeling of multivariable systems", *Proceedings of European Congress on Fuzzy and Intelligent Technologies*, pp.565-569, 1995.
- [91] U. Kaynak, R. Babuska and M. Setnes, "Methods for simplifications of fuzzy models", *Proceedings of NATO Advanced Science Institutes Series*, pp.1-10, 1996.
- [92] S. K. Sin and J. P. de Figueiredo, "Fuzzy system design through fuzzy clustering and optimal pre-defuzzification", *Proceedings of IEEE FUZZ*, pp.190-195, 1993.
- [93] X. L. Xie and G. A. Beni, "Validity measure for fuzzy clustering", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.3, pp.841-846, 1991.
- [94] R. Krishnapuram and J. M. Keller, "A possibilistic approach to clustering", *IEEE Transactions on Fuzzy Systems*, vol.1, pp.98-110, 1993.
- [95] Y. Nakamori and M. Royke, "Identification of fuzzy prediction models through hyperellipsoidal clustering", *IEEE Transactions on System, Man, Cybernetics*, vol.24, pp.1153-1173, Aug.1994.
- [96] D. E. Gustafson and W. C. Kesel, "Fuzzy clustering with a fuzzy covariance matrix", *Proceedings of IEEE FUZZ*, pp.761-766, 1997.
- [97] R. Krishnapuram and C. P. Freg, "Fitting an unknown number of lines and planes to image data through compatible cluster merging", *Pattern Recognition*, vol.25(4), pp.385-400, 1993.
- [98] R. Babuska, M. Setnes, U. Kaynak and R. H. Van Nauta Lemke, "Simplification of fuzzy rule bases," *Proceedings of European Congress on Fuzzy and Intelligent Technologies*, pp.1115-1119, 1996.
- [99] T. A. Runkler and R. H. Palm, "Identification of nonlinear systems using regular fuzzy c-elliptotype clustering", *Proceedings of IEEE FUZZ*, pp.1026-1030, 1996.
- [100] M. Delgado, A. F. Gomez-Skarmeta and F. Martin, "A fuzzy clustering based rapid-prototyping for fuzzy modeling", *IEEE Transactions on Fuzzy Systems*, vol.5, pp.223-233, 1997.
- [101] N. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy c-means model", *IEEE Transactions on Fuzzy Systems*, vol.3, pp.370-379, 1995.
- [102] I. Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.11, pp.773-781, 1989.
- [103] C. C. Wong and C. C. Chen, "A Hybrid clustering and gradient decent approach for fuzzy modeling", *IEEE Transactions on Systems, Man, Cybernetics, Part B, Cybernetics*, vol.29(6), pp.686-693, 1999.

- [104] K. Pal, R. K. Mudi and N. R. Pal, “A new scheme for fuzzy rule-based system identification and its application to self-tuning fuzzy controllers”, *IEEE Transactions on Systems, Man, Cybernetics, Part B*, vol.32(4), pp.470-482, 2002.
- [105] N. R. Pal, R. K. Mudi, K. Pal and D. Patranabis, “Rule extraction through exploratory data analysis for self-tuning fuzzy controller”, *International Journal of Fuzzy systems*, vol.6(2), pp.71-80, 2004
- [106] N. R. Pal and S. Saha, “Simultaneous structure identification and fuzzy rule generation for Takagi–Sugeno models”, *IEEE Transactions on Systems, Man, Cybernetics, Part B, Cybernetics*, vol.38, pp.1626-1638, 2008.
- [107] T. C. Havens, J. C. Bezdek, C. Leckie, L. O. Hall and M. Palaniswami, “Fuzzy C-means algorithms for very large data”, *IEEE Transactions on Fuzzy Systems*, vol.20(6), pp.1130–1146, 2012.
- [108] E. G. Mansoori, “FRBC: A fuzzy rule-based clustering algorithm”, *IEEE Transactions on Fuzzy Systems*, vol.19(5), pp.960–971, 2011.
- [109] L. O. Hall and D. B. Goldgof, “Convergence of the single-pass and online fuzzy c-means algorithms,” *IEEE Transactions on Fuzzy Systems*, vol.19(4), pp.792–794, 2011.
- [110] R. Krishnapuram, A. Joshi, O. Nasraoui and L. Yi, “Low-complexity fuzzy relational clustering algorithms for web mining”, *IEEE Transactions on Fuzzy Systems*, vol.9(4), pp.595-607, 2001.
- [111] R. N. Dave and S. Sen, “Robust fuzzy clustering of relational data”, *IEEE Transactions on Fuzzy Systems*, vol.10(6), pp.713-727, 2002.
- [112] H. Frigui, C. Hwang, and F. C. Rhee, “Clustering and aggregation of relational data with applications to image database categorization”, *Pattern Recognition*, vol.40(11), pp.3053-3068, 2007.
- [113] H. Frigui and C. Hwang, “Fuzzy clustering and aggregation of relational data with instance-level constraints”, *IEEE Transactions on Fuzzy Systems*, vol.16(6), pp.1565-1581, 2008.
- [114] H. C. Huang, Y. Y. Chuang, and C. S. Chen, “Multiple kernel fuzzy clustering”, *IEEE Transactions on Fuzzy Systems*, vol.20(1), pp.120-134, 2012.
- [115] X. W. Yang, G. Q. Zhang, J. Lu, and J. Ma, “A kernel fuzzy C-means clustering-based fuzzy support vectormachine algorithm for classification problems with outliers or noises”, *IEEE Transactions on Fuzzy Systems*, vol.19(1), pp.105-115, 2011.
- [116] J. H. Chiang and P. Y. Hao, “Support vector learning mechanism for fuzzy rule-based modeling: A new approach”, *IEEE Transactions on Fuzzy Systems*, vol.12(1), pp.1-12, 2004.
- [117] S. S. Cheng, H. C. Fu and H. M. Wang, “Model-based clustering by probabilistic self-organizing maps”, *IEEE Transactions on Neural Networks*, vol.20(5), pp.805-826, 2009.
- [118] A. Celikyilmaz and I. B. Turksen, “Enhanced fuzzy system models with improved fuzzy clustering algorithm”, *IEEE Transactions on Fuzzy Systems*, vol.16(3), pp.779-794, 2008.

- [119] T. Kohonen, S. Kaski, K. Lagus, J. Salojarvi and J. Honkela, "Self organization of a massive document collection", *IEEE Transactions on Neural Networks*, vol.11(3), pp.574-585, 2000.
- [120] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing Map", *IEEE Transactions on Neural Networks*, vol.11(3), pp.586-600, 2000.
- [121] C. C. Yang and N. K. Bose, "Generating fuzzy membership function with self-organizing feature map", *Pattern Recognition Letters, Elsevier*, vol.27, pp.356-365, 2006.
- [122] N. Hashimoto, T. Hatanaka, K. Uosaki and A. Kitamura, "Nonlinear system identification by SOM with GP based local modeling", *Annual Structural Integrity Conference and Exhibition*, pp.2360-2363, 2007.
- [123] G. A. Barreto and A. F. R. Araujo, "Identification and control of dynamical systems using the Self-Organizing Map", *IEEE Transactions on Neural Networks*, vol.15(5), pp.1244-1259, 2004.
- [124] G. Tambouratzis, "Assessing the effectiveness of feature groups in author recognition tasks with the SOM model", *IEEE Transactions on Systems, Man, Cybernetics, Part C, Applications and Reviews*, vol.36 (2), pp.249-259, 2006.
- [125] K. Taşdemir, P. Milenov, and B. Tapsall, "Topology-based hierarchical clustering of Self-Organizing Maps", *IEEE Transactions on Neural Networks*, vol.22(3), pp.474-485, 2011.
- [126] A. K. Pal and I. Naskar, "Design of self-tuning fuzzy PI controller in LABVIEW for control of real time process", *International journal of Electronics and Computer Science Engineering*, vol.2(2), pp.538-545, 2013.
- [127] J. Lee, "On methods for improving performance of PI-type fuzzy logic controllers", *IEEE Transactions on Fuzzy Systems*, vol.1, pp.298-301, 1993.
- [128] H. A. Malki, H. Li, and G. Chen, "New design and stability analysis of fuzzy proportional-derivative control systems", *IEEE Transactions on Fuzzy Systems*, vol.2, pp.245-254, 1994.
- [129] A. K. Pal and R. K. Mudi, "Development of neuro-fuzzy controller for application to HVAC system, inverted pendulum and other processes", *International Journal of Computational Cognition*, vol.6(2), pp.1-6, 2008.
- [130] P. B. Deshpande and R. H. Ash, "Elements of computer process control with advanced control applications", *Englewood Cliffs, NJ: Prentice-Hall*, 1983.
- [131] K. J. Åström, C. C. Hang, P. Persson, and W. K. Ho, "Toward intelligent PID control", *Automatica*, vol.28(1), pp.1-9, 1992.
- [132] Swenson and S. Don, "HVAC: heating, ventilating, and air conditioning", *Homewood, Illinois: American Technical Publishers*, 1995.
- [133] Rock and Zhu, "Designer's guide to ceiling-based air diffusion", *ASHRAE, Inc., New York, USA*, 2002.
- [134] Dianat and I. I. Nazari, "Characteristic of unintentional carbon monoxide poisoning in Northwest Iran-Tabriz", *International Journal of Injury Control and Promotion*, 2011.

- [135] Ashrae, “Ventilation and infiltration chapter”, *Fundamentals volume of the ASHRAE Handbook, Atlanta, Georgia*, 2005.
- [136] Q. Bi, W. J. Cai, Q. G. Wang, C. C. Hang, L. E. Lock, Y. Sun, K. D. Liu, Y. Zhang and B. Zou, “Advanced controller auto-tuning and its application in hvac systems”, *Control Engineering Practice*, vol.8(6), pp.633–644, 2000.
- [137] Z. R. Radakovic, V. M. Milosevic and S. B. Radakovic, “Application of temperature fuzzy controller in an indirect resistance furnace”, *Applied Energy*, vol.73, pp.167-182, 2002.
- [138] H. R. Benerji, Fuzzy Logic Controllers, in: R. R. Yager, L. A. Zadeh (Eds.), “An introduction to fuzzy logic application in intelligent systems”, *Kluwer, Boston, MA*, 1992.
- [139] Q. G. Wang, C. C. Hang, Y. Zhang and Q. Bi, “Multivariable controller auto-tuning with its application in HVAC systems”, *Proceedings of the American Control Conference, California*, 1999.
- [140] W. Jian and C. Wenjian, “Development of an adaptive neuro-fuzzy method for supply air pressure control in HVAC system”, *IEEE International Conference on System, Man, Cybernetics*, 2000.
- [141] Y. W. Huang and P. C. Tung, “Fuzzy PD system in adaptive control systems having input saturation”, *Journal of Control and Intelligent Systems*, vol.35(3), pp.217-222, 2007.
- [142] A. K. Pal and R. K. Mudi, “An adaptive fuzzy controller for overhead crane”, *Proceedings of IEEE International Conference on Advanced Communication Control and Computing Technologies, Ramanathapuram*, pp.300-304, 2012.
- [143] Feedback Instruments Ltd., “Manual on digital pendulum control experiments”, *Manual no. 33-936S Ed01 122006*.
- [144] A. K. Pal and J. Chakraborty, “Design a fuzzy logic controller with a non-fuzzy tuning scheme for swing up and stabilization of inverted pendulum”, *Advances in Intelligent Systems and Computing, Publisher: Springer*, vol.308, pp.221-230, 2015.
- [145] C. Dey, R. K. Mudi and T. T. Lee, “Dynamic set point weighted PID controller”, *Journal of Control and Intelligent Systems*, vol.37(4), pp.212-219, 2009.
- [146] T. Hägglund and K. J. Åström, “Revisiting the Ziegler-Nichols step response method for PID control”, *Journal of Process Control*, vol.14(6), pp.635-650, 2004.
- [147] M. N. Uddin, T. S. Radwan and M. A. Rahman, “Fuzzy logic based position control of a PMSM servo drive”, *Journal of Control and Intelligent Systems*, vol.35(4), pp.293-299, 2007.
- [148] A. K. Pal and R. K. Mudi, “Development of a self-tuning fuzzy controller through relay feedback approach”, in *Book: Communications in Computer and Information Science, Publisher: Springer-Verlag Berlin Heidelberg*, vol.250, pp.424-426, 2011.
- [149] K. J. Åström and T. Hägglund, “Automatic tuning of simple regulators with specifications on phase and amplitude margins”, *Automatica*, vol.20, pp.645-651, 1984.
- [150] H. P. Haung, J. C. Jeng and K. Y. Luo, “Auto-tune system using single run relay feedback test and model based controller design”, *Journal of process control*, vol.15, 2005.

- [151] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers", *Transactions of ASME*, vol.64, pp.759-68, 1942.
- [152] T. H. Lee, Q. G. Wang, and K. K. Tan, "A modified relay based technique for improved critical point estimation in process control", *IEEE Transactions on Control Systems Technology*, vol.3(3), pp.330-337, 1995.
- [153] S. W. Sung, J. H. Park, and I. Lee, "Modified relay feedback method", *Industrial Engineering and Chemistry Research*, vol.34(11), pp.4133-4135, 1995.
- [154] S. Shen, H. Yu, and C. C. Yu, "Use of the saturation relay feedback for auto-tune identification", *Chemical Engineering Science*, vol.51(8), pp.1187-1198, 1996.
- [155] R. K. Mudi and N. R. Pal, "A self-tuning fuzzy PI controllers", *Fuzzy Sets and Systems*, vol.115, pp.327-338, 2000.
- [156] A. K. Pal and R. K. Mudi, "Speed control of DC motor using relay feedback tuned PI, fuzzy PI and self-tuned fuzzy PI controller", *Control Theory and Informatics*, vol.2(1), pp.24-32, 2012.
- [157] A. Ahmad, "Active sway suppression techniques of a granty crane system", *European Journal of Scientific Research*, vol.27(3), pp.322-333, 2009.
- [158] A. K. Pal, R. K. Mudi, and R. R. De Maity, "A Non-fuzzy self-tuning scheme of PD-type FLC for overhead crane control", *Advances in Intelligent Systems and Computing*, Publisher: Springer-Verlag Berlin Heidelberg, vol.199, pp.35-42, 2013.
- [159] Ritu Rani De(Maity), R. K. Mudi and A. K. Pal, "A PD-type self-tuning FLC for second-order systems with dead-time", *Proceedings of IEEE International Conference on Advanced Communication Control and Computing Technologies, Ramanathapuram*, pp.409-413, 2012.
- [160] K. S. Hong and Q. H. Ngo, "Port automation: modeling and control of container cranes", *International Conference on Instrumentation, Control and Automation*, pp.19-26, 2009.
- [161] A. Rahman, A. Nayfeh and Z. Masoud, "Dynamics and control of cranes- a review", *Journal of vibration and control*, vol.9, pp.863-908, 2003.
- [162] J. J. Hamalainen, A. Marttinen, L. Baharova and J. Virkkuen, "Optimal path planning for a trolley crane: fast and smooth transfer of load", *IEE Proceedings of Control Theory and Applications*, vol.142(1), pp.51-57, 1995.
- [163] C. Li and C. Y. Lee, "Fuzzy motion control of an auto-warehousing crane system", *IEEE Transactions on Industrial Electronics*, vol.48(5), pp.983-994, 2001.
- [164] G. Bartolini, A. Pisano and E. Usai, "Second order sliding mode control of container cranes", *Automatica*, vol.38, pp.1783-1790, 2002.
- [165] M. S. Park, D. Chwa and S. K. Hong, "Antisway tracking control of overhead cranes with system uncertainty and actuator nonlinearity using an adaptive fuzzy sliding mode control", *IEEE Transactions on Industrial Electronics*, vol.55(11), 2008.
- [166] H. Lee, Y. Liang, and D. Segura, "A new approach for the antiswing control of overhead cranes with high-speed load hoisting", *International Journal of Control*, vol.76(15), pp.1493-1499, 2003.

- [167] J. H. Yang and K. S. Yang, "Adaptive coupling control for overhead crane systems", *Mechatronics*, vol.17(2/3), pp.143-152, 2007.
- [168] K. L. Sorensen, W. Singhose and S. Dickerson, "A controller enabling precise positioning and sway reduction in bridge and grany cranes", *Control Engineering Practice*, vol.15, pp.825-837, 2007.
- [169] Q. H. Ngo, K. S. Hong and I. H. Jung, "Adaptive control of an axially moving system", *Journal of Mechanical Science and Technology*, vol.23, pp.3071-3078, 2009.
- [170] Y. C. Liang and K. K. Koh., "Concise anti-swing approach for fuzzy crane control", *IEE Eletronics Letters*, vol.3(2), pp.167-168, 1997.
- [171] D. Liu, J. Yi, D. Zhao and W. Wang, "Adaptive sliding mode fuzzy control for a two dimensional overhead crane", *Mechatronics*, vol.15, pp.505-522, 2005.
- [172] M. A. Karkoub and M. Zribi, "Modelling and energy based nonlinear control of crane lifters", *IEE Proceedings of Control Theory and Applications*, vol.149(3), pp.209-216, 2002.
- [173] A. Piazzzi and A. Visioli, "Optimal dynamic inversion-based control of an overhead crane", *IEE Proceedings of Control Theory and Applications*, vol.149(5), pp.405-411, 2002.
- [174] C. Chang, S. Hsu and K. Chiang, "A practical fuzzy controllers scheme of overhead crane", *Journal of Control Theory and Applications*, vol.3, pp.266-270, 2005.
- [175] D. Antic, Z. Jovanovic, S. Peric, S. Nikolic, M. Milojkovic and M. Milosevic., "Anti-swing fuzzy controller applied in a 3D crane system", *Engineering, Technology and Applied science research*, vol.2(2), pp.196-200, 2012.
- [176] P. E. Wellstead, "Introduction to physical system modellling", *Academic Press Ltd.*, 1979.
- [177] A. K. Pal and R. K. Mudi, "An adaptive PD-type FLC and its real time implementation to overhead crane control", *International Journal of Emerging Technologies in Computational and Applied Sciences*, vol.6(2), pp.178-183, 2013.
- [178] M. Eftekhari and S. D. Katebi, "Extracting compact fuzzy rules for nonlinear system modeling using subtractive clustering, GA and unscented filter", *Journal of Applied Mathematical Modelling*, vol.32, pp.2634-2651, 2008.
- [179] N. R. Pal, K. Pal, J. C. Bezdek, and T. A. Runkler, "Some issues in system identification using clustering", *IEEE International Joint Conference on Neural Networks, Piscataway*, pp. 2524-2529, 1997.
- [180] M. Eftekhari, S. D. Katebi, M. Karami and A. H. Jahanmiri, "Eliciting transparent fuzzy model using differential evolution", *Journal of Applied Soft Computing*, vol.8, pp.466-476, 2008.
- [181] Y. C. Chen, N. R. Pal and I. F. Chung, "An integrated mechanism for feature selection and fuzzy rule extraction for classification", *IEEE Transactions on Fuzzy Systems*, vol.20(4), pp.683-698, 2012.
- [182] R. R.Yager, "Perception-based granular probabilities in risk modeling and decision making", *IEEE Transactions on Fuzzy Systems*, vol.14(2), pp.329-339, 2006.

- [183] D. R. Wu and J. M. Mendel, "Linguistic summarization using if-then rules and interval type-2 fuzzy sets," *IEEE Transactions on Fuzzy Systems*, vol.19(1), pp.136-151, 2011.
- [184] N. Kanagaraj, P. Sivashanmugam and S. Paramasivam, "A fuzzy logic based supervisory hierarchical control scheme for real time pressure control", *International Journal of Automation and Computing*, vol.6(1), pp.88-96, 2009.
- [185] T. Kohonen, "Self-Organization and associative memory", 3rd edition *Spinger-Verlag, Berlin*, 1989.
- [186] J. C. Bezdek, "Pattern recognition with fuzzy objective function algorithm", *Plenum Press, New York*, 1981.
- [187] P. C. Chang and T. W. Liao, "Combining SOM and fuzzy rule base for flow time prediction in semiconductor manufacturing factory", *Journal of Applied Soft Computing*, vol.6(2), pp.198-206, 2006.
- [188] D. Brugger, M. Bogdan and W. Rosenstiel, "Automatic cluster detection in Kohonen's SOM", *IEEE Transactions on Neural Networks*, vol.19(3), pp.442-459, 2008.
- [189] Y. Fukuyama and M. Sugeno, "A new method of choosing the number of clusters for the fuzzy c-means method", *Proceedings of 5th Fuzzy Systems Symposium*, pp.247-250, 1989.
- [190] P. H. Huang and Y. S. Chang, "Fuzzy rule based qualitative modeling", *Proceedings of IEEE Fuzzy Systems*, vol.2, pp.1261-1265, 1996.
- [191] C. K. Chak, G. Feng and J. Ma, "An adaptive fuzzy neural network for MIMO system model approximation in high dimensional space", *IEEE Transactions on System, Man, Cybernetics, Part B, Cybernetics*, vol.28, pp.436-446, 1998.
- [192] K. B. Cho and B. H. Wang, "Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction", *Fuzzy Sets and Systems*, vol.83(3), pp.325-339, 1996.
- [193] E. Kim, M. Park, S. Ji and M. Park, "A new approach to fuzzy modeling", *IEEE Transactions on Fuzzy Systems*, vol.5(3), pp.328-337, 1997.
- [194] K. W. Wong, T. D. Gedeon, C. C. Fung and P. M. Wong, "Fuzzy rule extraction using self-organizing neural network and association rules", *Proceedings of IEEE Region 10 International Conference on electrical and electronic Technology*, pp.403-408, 2001.
- [195] J. E. Moreno, O. Castillo, J. R. Castro, L. G. Martinez and P. Melin, "Data mining for extraction of fuzzy if-then rules using Mamdani and Takagi-Sugeno-Kang FIS", *Engineering Letters*, vol.15(1), pp.82-88, 2007.
- [196] X. Liu, X. Feng and W. Pedrycz, "Extraction of fuzzy rules from decision trees: An axiomatic fuzzy sets (AFS) approach", *Journal of Data and Knowledge Engineering, Elsevier*, vol.84, pp.1-25, 2013.
- [197] C. Gao, Q. Ge and L. Jian, "Rule extraction from fuzzy-based blast furnace SVM multiclassifier for decision making", *IEEE Transaction on fuzzy systems*, vol.22(3), pp.586-596, 2014.

- [198] A. K. Pal, R. K. Mudi, and C. Dey, "Rule extraction through Self-Organizing Map for a self-tuning fuzzy logic controller", *Advanced Materials Research (MEMS, NANO and Smart Systems)*, vols.403-408, pp.4957-4964, 2012.
- [199] L. T. Koczy and K. Hirota, "Size reduction by interpolation in fuzzy rule bases", *IEEE Transactions on Systems, Man, Cybernetics, Part B: Cybernetics*, vol.27(1), 1997.
- [200] P. Baranyi, L. T. Kóczy, and T. D. Gedeon, "A generalized concept for fuzzy rule interpolation," *IEEE Transactions on Fuzzy Systems*, vol.12, pp.820-837, 2004.
- [201] S. Guillaume, "Designing fuzzy inference systems from data: an interpretability-oriented review", *IEEE Transactions on Fuzzy Systems*, vol.9(3), pp.426-443, 2001.
- [202] Y. Jin, "Fuzzy modeling of high-dimensional systems: complexity reduction and interpretability improvement", *IEEE Transactions on Fuzzy Systems*, vol.8(2), pp.212-221, 2000.
- [203] J. Yen and L. Wang, "Simplifying fuzzy rule-based models using orthogonal transformation methods", *IEEE Transactions on Systems, Man, Cybernetics, Part B*, vol.29, pp.13-24, 1999.
- [204] M. Setnes and R. Babuska, "Rule base reduction: some comments on the use of orthogonal transforms", *IEEE Transactions on Systems, Man, Cybernetics, Part C: Applications and Reviews*, vol.31(20), 2001.
- [205] M. Setnes, R. Babuska and H. B. Verbruggen, "Complexity reduction in fuzzy modeling", *Mathematics and Computers in Simulation*, vol.46, pp.507-516, 1998.
- [206] M. Setnes, R. Babuska, U. Kaymak and H. R. V. N. Lemke, "Similarity measures in fuzzy rule base simplification", *IEEE Transactions on Systems, Man, Cybernetics, Part B: Cybernetics*, vol.28(3), pp.376-386, 1998.
- [207] R. Ketata, H. Bellaaj, M. Chtourou, and M. B. Amer, "Adjustment of membership functions, generation and reduction of fuzzy rule-base from numerical data", *Malaysian Journal of Computer Science*, vol.20(2), pp.147-169, 2007.
- [208] D. Chakraborty and N. R. Pal, "A neuro-fuzzy scheme for simultaneous feature selection and fuzzy rule-based classification", *IEEE Transactions on Neural Networks*, vol.15(1), pp.110-123, 2004.