Efficient sequential fuzzy-neural algorithms for aircraft fault-tolerant control

Rong, Haijun

2007

Rong, H. (2007). Efficient sequential fuzzy-neural algorithms for aircraft fault-tolerant control. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/3507>

<https://doi.org/10.32657/10356/3507>

Nanyang Technological University

Downloaded on 26 Apr 2023 19:22:27 SGT

Efficient Sequential Fuzzy-Neural Algorithms for Aircraft Fault-Tolerant Control

Rong Haijun

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University

in fulfillment of the requirements for the degree of

Doctor of Philosophy

2007

Statement of Originality

I hereby certify that the content of this thesis is the result of work done by me and has not been submitted for a higher degree to any other University or Institution.

.....

Date

.....

Rong Haijun

Acknowledgments

I would like to express my deepest gratitude to my supervisor Prof. N. Sundararajan for his intelligent guidance and patient nurture over the past years. I have learned so much from his ways of critical thinking and his analytic insights into the problems helped me greatly in the accomplishment of this thesis. I am proud to have a such a great mentor on my way towards research and he has made my stay at Nanyang Technological University a truly valuable experience.

I want to thank Associate Prof. P. Saratchandran and Assistant Prof. Guang-Bin Huang for their numerous helpful comments and enlightening discussions throughout my research. I have been receiving their help in my research work from all aspects. Their time and assistance were a valuable contribution to this thesis. I would like to thank Dr Abhay for his helpful discussions and useful comments on my research.

Thanks are also dedicated to my colleagues and the technicians at Computer Control Lab who are always ready to offer their help and encouragements.

Special thanks are given to my family for their fundamental support and encouragement during my study in Singapore.

Finally, I acknowledge the School of Electrical and Electronics Engineering, Nanyang Technological University for providing me the financial support and this precious opportunity of study.

Summary

This thesis presents the development of new fuzzy-neural learning algorithms for the fault-tolerant control of aircraft, specifically, in the area of autoland control under actuator failures and severe winds.

In the classic approaches to design a fuzzy inference system, the fuzzy rules are determined by a domain expert *a priori* and then this is maintained unchanged during the learning. These fixed fuzzy rules may not be appropriate and hence poor performance may be observed in realtime applications where the environment or model often meets unpredicted disturbances or damages. In comparison to the conventional methods, fuzzy inference systems based on neural networks, called fuzzy-neural systems, have begun to exhibit great potential for adapting to the changes by utilizing the learning ability and adaptive capability of neural networks. Thus, a fuzzy inference system can be built using the standard structure of neural networks. Nevertheless, the determination of the number of fuzzy rules and the adjustment of the parameters in the if-then fuzzy rules are still open issues.

In this thesis we develop two efficient sequential, fuzzy-neural schemes where the first scheme can determine the number of fuzzy rules during learning and modify the parameters in fuzzy rules simultaneously while the second scheme can update the parameters for the fuzzy rules at an extremely high speed. Their applications and superior performance to real-life problem in the aircraft autoland control under actuator failures and severe winds are presented.

Recently, a novel Growing and Pruning method for Radial Basis Function network

(GAP-RBF) has been developed by Huang *et al.* [42]. GAP-RBF uses the concept of significance of hidden neurons to realize parsimonious networks and achieve fast computation by adjusting only the parameters of the nearest neuron every time instead of all the neurons without losing the approximation performance. On the basis of the functional equivalence between a fuzzy inference system and a radial basis function network, a Sequential Adaptive Fuzzy Inference System (SAFIS) is developed to realize a compact fuzzy system with lesser number of rules by using the GAP-RBF neural network to extract fuzzy rules during learning. Similar to GAP-RBF, SAFIS uses the concept of *influence* of a fuzzy rule for adding and removing rules during learning. The *influence* of a fuzzy rule is defined as its contribution to the system output in a statistical sense when the input data is uniformly distributed. When there are no addition of fuzzy rules, only the parameters of the “closest” (in a Euclidean sense) rule are updated using an Extended Kalman Filter (EKF) scheme. The performance of SAFIS is compared with several existing algorithms based on two nonlinear system identification benchmark problems and a chaotic time series prediction problem. Results indicate that SAFIS produces similar or better accuracies with lesser number of rules compared to other algorithms.

In addition, an On-line, Sequential, Fuzzy Extreme Learning Machine (OS-Fuzzy-ELM) has been developed based on the Extreme Learning Machine (ELM) proposed by Huang *et al.* [47]. OS-Fuzzy-ELM handles the fuzzy inference systems (including both Takagi-Sugeno-Kang (TSK) and Mamdani type of fuzzy models) with any bounded nonconstant piecewise continuous membership function and any type T-norm fuzzy logic operation. Further, the learning in OS-Fuzzy-ELM can be done in a one-by-one or chunk-by-chunk (a block of data) mode with fixed or varying chunk sizes. In OS-Fuzzy-ELM, the parameters of the fuzzy membership functions need not be adjusted during training and one can randomly assign the values to them and then analytically determine the consequent parameters. The performance of OS-Fuzzy-ELM has been evaluated with other well-known

learning algorithms using benchmark problems drawn from the areas of function approximation and classification. The study results show that the proposed OS-Fuzzy-ELM produces similar or better approximation and classification accuracies with an extremely fast learning speed.

The main contribution of this thesis is to develop two adaptive, fault-tolerant, fuzzy control strategies based on the above two algorithms for a high performance fighter automatic landing problem under the failures of stuck control surfaces and severe winds. The autolanding problem is executed in four phases of flight segments consisting of a wing-level flight, a coordinated turn, glide slope descent and finally the flare maneuver. In the control strategy the fuzzy controller augments an existing conventional controller called Baseline Trajectory Following Controller (BTFC) by means of its online learning. BTFC has been designed using classical control methods under normal operating conditions with winds but was found not capable of handling failures. For this study, the following fault scenarios have been considered. i) Single fault of either aileron or elevator stuck at certain deflections and ii) Double fault cases where one aileron and one elevator at the same or opposite direction are stuck at different deflections. Besides a severe wind model is incorporated in the fault scenarios. The work is separated into two parts. In the first part, the proposed SAFIS and OS-Fuzzy-ELM algorithms are respectively utilized as the fuzzy controller together with BTFC controller for the aircraft autolanding problem and their performance is compared with a neural aided BTFC controller and BTFC controller. The simulation results show that the proposed SAFIS aided BTFC and OS-Fuzzy-ELM aided BTFC control schemes have a clear improvement for the fault tolerant envelope for both single and double faults. In the second part, the proposed SAFIS and OS-Fuzzy-ELM are compared based on the aircraft autolanding problem and the simulation results illustrate that OS-Fuzzy-ELM achieves similar or slightly larger stuck-up deflections of the actuators for both single and double faults.

To summarize, the work in this thesis focuses on the development of two new sequential fuzzy-neural schemes and their applications in the area of aircraft autoland control under actuator failures and severe winds.

Table of Contents

Acknowledgments	i
Summary	ii
Table of Contents	vi
List of Figures	xi
List of Tables	xviii
Nomenclature for the Aircraft Model	xxii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	5
1.3 Major Contributions of the Thesis	6
1.4 Organization of the Thesis	7
2 A Brief Review of Fuzzy-Neural Systems	9
2.1 Fuzzy Inference Systems	9
2.1.1 Fuzzy Sets	11

TABLE OF CONTENTS

vii

2.1.2	Fuzzy Rules	14
2.1.3	Fuzzifier	15
2.1.4	Fuzzy Inference Engine	16
2.1.5	Defuzzifier	17
2.1.6	Universal Approximation of Fuzzy Inference Systems	19
2.2	Neural Networks	20
2.2.1	SLFNs with Sigmoidal Activation Functions	22
2.2.2	Radial Basis Function Networks	23
2.2.3	Learning Algorithms	25
2.3	Functional Equivalence between fuzzy inference systems and radial basis function networks	28
2.4	Fuzzy-Neural Systems	29
3	Sequential Adaptive Fuzzy Inference System (SAFIS)	34
3.1	Architecture of SAFIS	35
3.1.1	“Influence” of a Fuzzy Rule	38
3.2	SAFIS Algorithm	41
3.2.1	Adding a Fuzzy Rule	41
3.2.2	Allocation of Antecedent and Consequent Parameters	42
3.2.3	Parameter Adjustment	43
3.2.4	Removing a Fuzzy Rule	44
3.2.5	Selection of Predefined Parameters	46
3.3	Performance Evaluation of SAFIS	48
3.3.1	Nonlinear Dynamic System Identification	49

TABLE OF CONTENTS

viii

3.3.2	Mackey-Glass Time Series Prediction	54
4	Aircraft Autolanding Fault-Tolerant Control Problem	59
4.1	Aircraft Mathematical Model	66
4.1.1	Aerodynamic Coefficients	70
4.2	Aircraft Landing Task and Wind Model	74
4.3	Failure Types	77
4.4	Baseline Trajectory Following Controller	83
5	SAFIS Based Aircraft Fault-Tolerant Autolanding Control	96
5.1	Selection of SAFIS Parameters	99
5.2	Single Surface Failure	103
5.3	Two Surface Failures	111
5.3.1	Left Elevator and Left Aileron Stuck at Different Deflections	112
5.3.2	Left Elevator and Right Aileron Stuck at Different Deflections	118
6	Online Sequential Fuzzy Extreme Learning Machine (OS-Fuzzy-ELM)	121
6.1	Extreme Learning Machine (ELM)	122
6.2	Structure of Fuzzy Inference Systems	125
6.2.1	SLFN Equivalence of Fuzzy Inference Systems	128
6.3	Online Sequential Fuzzy Extreme Learning Machine	136
6.4	Parameter Determination of OS-Fuzzy-ELM	143
6.5	Performance Evaluation of OS-Fuzzy-ELM	149
6.5.1	Performance Evaluation of OS-Fuzzy-ELM: One-by-One Case	151

TABLE OF CONTENTS

ix

6.5.2	Performance Evaluation of OS-Fuzzy-ELM: Chunk-by-Chunk Mode	159
6.5.3	Performance Comparison of OS-Fuzzy-ELM: Product vs Min T-Norm	163
6.5.4	Performance Comparison of OS-Fuzzy-ELM: TSK vs Mamdani Model	167
7	Aircraft Autolanding Fault-Tolerant Control with OS-Fuzzy-ELM Algorithm	174
7.1	Single Surface Failure	176
7.2	Two Surface Failures	186
7.2.1	Left Elevator and Left Aileron Stuck at Different Deflections	187
7.2.2	Left Elevator and Right Aileron Stuck at Different Deflections	194
8	Comparison of OS-Fuzzy-ELM and SAFIS for Aircraft Autolanding Fault-Tolerant Control Problem	197
8.1	Single Surface Failure	198
8.2	Two Surface Failures	204
8.2.1	Left Elevator and Left Aileron Stuck at Different Deflections	205
8.2.2	Left Elevator and Right Aileron Stuck at Different Deflections	210
9	Conclusions and Future Work	213
9.1	Conclusions	213
9.2	Recommendations for Future Work	218
	Appendix A	220
A.1	Graphs for the Membership Functions	220

TABLE OF CONTENTS

x

Appendix B	223
B.1 Simulation Block of the High Performance Fighter Aircraft	223
Appendix C	225
C.1 Data of the Lateral and Longitudinal Aircraft Dynamics for BTFC Controller Design	225
Appendix D	228
D.1 Results for Parameter Determination	228
D.2 Results for Chunk-by-Chunk Mode	231
D.3 Results for Different T-norm Operators	233
D.4 Results for Different Fuzzy Models	237
Author's Publications	241
Bibliography	242

List of Figures

2.1	Structure of SLFN with additive activation function	22
2.2	Structure of RBF network	24
3.1	Structure of SAFIS	36
3.2	(a) Rule update process between different algorithms for nonlinear identification problem 1 during the whole observation (b) Rule update process between different algorithms for nonlinear identification problem 1 between 0 and 1000 observation	51
3.3	Membership functions of input variable $y(t)$ for nonlinear identification problem 2	53
3.4	Membership functions of input variable $u(t)$ for nonlinear identification problem 2	53
3.5	Rule update process between different algorithms for nonlinear identification problem 2	54
3.6	Rule update process between different algorithms for Mackey-Glass time series prediction	55
3.7	Feedback-error-learning control strategy	58
4.1	Definition of aircraft dynamics in the body-fixed axes	67
4.2	The landing trajectory	75

LIST OF FIGURES

xii

4.3	Wind model during landing	76
4.4	Feasible range for left elevator stuck failure	79
4.5	Feasible range for left aileron stuck failure	80
4.6	Feasible range for left elevator and left aileron stuck failures	81
4.7	Feasible range for left elevator and right aileron stuck failures	82
4.8	The baseline control strategy for autoland	83
4.9	Waypoints and trajectory geometry	84
4.10	Longitudinal Baseline Trajectory Following Controller (BTFC) . . .	88
4.11	Lateral Baseline Trajectory Following Controller (BTFC)	89
4.12	BTFC controller performance for the altitude (h) and velocity (V_T) under winds and no failures	90
4.13	BTFC controller performance for the sideslip angle (β) and lateral position (Y) under winds and no failures	91
4.14	BTFC controller performance for the altitude (h) and velocity (V_T) under winds and left elevator failure stuck at -10deg	93
4.15	BTFC controller performance for the sideslip angle (β) and lateral position (Y) under winds and left elevator failure stuck at -10deg .	94
5.1	SAFIS aided BTFC control strategy for autoland	97
5.2	SAFIS implementation block for the longitudinal direction	98
5.3	SAFIS implementation block for the lateral direction	99
5.4	Best and average solutions in the GA population during a typical run	102
5.5	Comparison of SAFIS aided BTFC, EMRAN aided BTFC and BTFC for the altitude (h) and velocity (V_T) under the left elevator stuck at -10deg	103

LIST OF FIGURES

xiii

5.6	Comparison of SAFIS aided BTFC, EMRAN aided BTFC and BTFC for the sideslip angle (β) and lateral position (Y) under the left elevator stuck at -10deg	104
5.7	Left and right elevator control signals under the left elevator stuck at -10deg	106
5.8	Left and right aileron control signals under the left elevator stuck at -10deg	107
5.9	Failure tolerance under left elevator stuck conditions	108
5.10	Touchdown points for SAFIS aided BTFC under left elevator stuck conditions	109
5.11	Failure tolerance under left aileron stuck conditions	110
5.12	Touchdown points for SAFIS aided BTFC under left aileron stuck conditions	111
5.13	Comparison of SAFIS aided BTFC, EMRAN aided BTFC and BTFC for the altitude (h) and velocity (V_T) under the left elevator stuck at -10deg and the left aileron stuck at 10deg	112
5.14	Comparison of SAFIS aided BTFC, EMRAN aided BTFC and BTFC for the sideslip angle (β) and lateral position (Y) under the left elevator stuck at -10deg and the left aileron stuck at 10deg	113
5.15	Left and right elevator control signals under the left elevator stuck at -10deg and the left aileron stuck at 10deg	115
5.16	Left and right aileron control signals under the left elevator stuck at -10deg and the left aileron stuck at 10deg	116
5.17	Failure tolerance under left elevator and left aileron stuck conditions	117
5.18	Failure tolerance under left elevator and right aileron stuck conditions	118
6.1	Structure of Fuzzy Inference Systems	126

LIST OF FIGURES

xiv

6.2	Equivalent SLFN Structure of Fuzzy Inference Systems	129
6.3	The tendency of training and testing accuracies for Gaussmf and TSK fuzzy model according to the varying number of rules	145
6.4	The tendency of training and testing accuracies for Trimf and TSK fuzzy model according to the varying number of rules	145
6.5	The tendency of training and testing accuracies for Cauchymf and TSK fuzzy model according to the varying number of rules	146
6.6	The tendency of training and testing accuracies for Gaussmf and Mamdani fuzzy model according to the varying number of rules . . .	147
6.7	The tendency of training and testing accuracies for Trimf and Mamdani fuzzy model according to the varying number of rules	148
6.8	The tendency of training and testing accuracies for Cauchymf and Mamdani fuzzy model according to the varying number of rules . . .	148
6.9	Training time comparison for nonlinear system identification problem based on the same membership functions between different algorithms.	154
6.10	Training time comparison for California housing prediction based on the same membership functions between different algorithms. . .	157
7.1	OS-Fuzzy-ELM aided BTFC control strategy for autolandings	174
7.2	OS-Fuzzy-ELM implementation block for the longitudinal direction . . .	175
7.3	OS-Fuzzy-ELM implementation block for the lateral direction . . .	176
7.4	Comparison of OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC for the altitude (h) and velocity (V_T) under the left elevator stuck at -10deg	178

LIST OF FIGURES

xv

7.5	Comparison of OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC for the sideslip angle (β) and lateral position (Y) under the left elevator stuck at -10deg	179
7.6	Left and right elevator control signals under the left elevator stuck at -10deg	181
7.7	Left and right aileron control signals under the left elevator stuck at -10deg	182
7.8	Failure tolerance under left elevator stuck conditions	183
7.9	Touchdown points for OS-Fuzzy-ELM aided BTFC under left elevator stuck conditions	184
7.10	Failure tolerance under left elevator stuck conditions	185
7.11	Touchdown points for OS-Fuzzy-ELM aided BTFC under left elevator stuck conditions	186
7.12	Comparison of OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC for the altitude (h) and velocity (V_T) under the left elevator stuck at -10deg and the left aileron stuck at 10deg	188
7.13	Comparison of OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC for the sideslip angle (β) and lateral position (Y) under the left elevator stuck at -10deg and the left aileron stuck at 10deg .	189
7.14	Left and right elevator control signals under the left elevator stuck at -10deg and the left aileron stuck at 10deg	190
7.15	Left and right aileron control signals under the left elevator stuck at -10deg and the left aileron stuck at 10deg	192
7.16	Failure tolerance under left elevator and left aileron stuck conditions	193
7.17	Failure tolerance under left elevator and right aileron stuck conditions	194

LIST OF FIGURES

xvi

8.1	Comparison of OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC for the altitude (h) and velocity (V_T) under the left elevator stuck at -10deg	198
8.2	Comparison of OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC for the sideslip angle (β) and lateral position (Y) under the left elevator stuck at -10deg	199
8.3	Left and right elevator control signals under the left elevator stuck at -10deg	201
8.4	Left and right aileron control signals under the left elevator stuck at -10deg	202
8.5	Failure tolerance under left elevator stuck conditions	203
8.6	Failure tolerance under left elevator stuck conditions	204
8.7	Comparison of OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC for the altitude (h) and velocity (V_T) under the left elevator stuck at -10deg and the left aileron stuck at 10deg	205
8.8	Comparison of OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC for the sideslip angle (β) and lateral position (Y) under the left elevator stuck at -10deg and the left aileron stuck at 10deg	206
8.9	Left and right elevator control signals under the left elevator stuck at -10deg and the left aileron stuck at 10deg	208
8.10	Left and right aileron control signals under the left elevator stuck at -10deg and the left aileron stuck at 10deg	209
8.11	Failure tolerance under left elevator and left aileron stuck conditions	210
8.12	Failure tolerance under left elevator and right aileron stuck conditions	211

LIST OF FIGURES

xvii

A.1 Graph Examples of Eight Membership Functions: a) Trimf; b) Trapmf; c) Gaussmf; d) Gauss2mf; e) Cauchymf; f) Pimf; g) Dsigmf; h) Psigmf.	222
B.1 The high performance fighter aircraft simulation block	224

List of Tables

3.1	Effect of parameter e_g on system performance (number of rules and the testing RMS error) under different $\epsilon_{\max}$ values and $\kappa = 1.0$, $\gamma=0.997$	47
3.2	Effect of parameter e_g on system performance (number of rules and the testing RMS error) under different κ values and $\epsilon_{\max} = 10.0$, $\gamma=0.997$	48
3.3	Effect of parameter e_g on system performance (number of rules and the testing RMS error) under different γ values and $\epsilon_{\max} = 10.0$, $\kappa = 1.0$	48
3.4	Results of nonlinear identification problem 1	50
3.5	Results of nonlinear identification problem 2	52
3.6	Results of Mackey-Glass time series prediction	55
4.1	Specification of control surfaces	68
4.2	Mass and Geometry Data in aircraft model	69
4.3	Pillbox conditions at touchdown	83
4.4	Equilibrium sates	86
4.5	Equilibrium control inputs	86
5.1	The parameter bounds of SAFIS used in the GA optimization . . .	101

5.2	The parameter values of SAFIS after GA optimization	102
5.3	Performance comparison between SAFIS aided BTFC and EMRAN aided BTFC control schemes under the left elevator stuck at -10deg	105
5.4	Performance comparison between SAFIS aided BTFC and EMRAN aided BTFC control schemes under the left elevator stuck at -10deg and the left aileron stuck at 10deg	114
6.1	Effect of parameter $\tilde{N}$ on system performance (training and testing accuracies) for TSK fuzzy model	144
6.2	Effect of parameter $\tilde{N}$ on system performance (training and testing accuracies) for Mamdani fuzzy model	147
6.3	Specification of Benchmark Datasets	149
6.4	Performance comparison for nonlinear system identification	153
6.5	Performance comparison for California housing prediction between different algorithms	156
6.6	Performance comparison between different algorithms for DNA clas- sification	158
6.7	Performance comparison implemented in different chunk sizes for nonlinear system identification	160
6.8	Performance comparison implemented in different chunk sizes for California housing prediction	161
6.9	Performance comparison implemented in different chunk sizes for DNA classification	162
6.10	Performance comparison between Product and Min T-norm opera- tors for nonlinear system identification	164
6.11	Performance comparison between Product and Min T-norm opera- tors for California housing prediction	165

LIST OF TABLES

xx

6.12	Performance comparison between Product and Min T-norm operators for DNA classification	166
6.13	Parameter specification of TSK and Mamdani fuzzy models for each problem	167
6.14	Performance comparison between TSK and Mamdani fuzzy models for nonlinear system identification	168
6.15	Performance comparison between TSK and Mamdani fuzzy models for California housing prediction	169
6.16	Performance comparison between TSK and Mamdani fuzzy models for DNA classification	171
7.1	Performance comparison between OS-Fuzzy-ELM and EMRAN . .	177
7.2	Performance comparison between OS-Fuzzy-ELM and EMRAN . .	187
8.1	Performance comparison between OS-Fuzzy-ELM and SAFIS	200
8.2	Performance comparison between OS-Fuzzy-ELM and SAFIS	207
D.1	Effect of parameter $\tilde{N}$ on system performance (training and testing accuracies) for TSK model	228
D.2	Effect of parameter $\tilde{N}$ on system performance (training and testing accuracies) for Mamdani model	230
D.3	Performance comparison implemented in different chunk sizes for nonlinear system identification	231
D.4	Performance comparison implemented in different chunk sizes for California housing prediction	232
D.5	Performance comparison implemented in different chunk sizes for DNA classification	233

D.6	Performance comparison between Product and Min T-norm operators for nonlinear system identification	233
D.7	Performance comparison between Product and Min T-norm operators for California housing prediction	235
D.8	Performance comparison between Product and Min T-norm operators for DNA classification	236
D.9	Performance comparison between TSK and Mamdani fuzzy models for nonlinear system identification	237
D.10	Performance comparison between TSK and Mamdani fuzzy models for California housing prediction	238
D.11	Performance comparison between TSK and Mamdani fuzzy models for DNA classification	239

Nomenclature for the Aircraft Model

X, Y, Z	aircraft body axes
X_{cg}	center-of-gravity location, m
X_{cgr}	reference center-of-gravity location for aerodynamic data, m
α	angle of attack, deg
β	angel of sideslip, deg
u, v, w	component of aircraft velocity along X, Y, Z body axes, m/s
$\dot{u}, \dot{v}, \dot{w}$	aircraft acceleration along X, Y, Z body axes, m/s ²
p	aircraft roll rate about X body axis, rad/s
q	aircraft pitch rate about Y body axis, rad/s
r	aircraft yaw rate about Z body axis, rad/s
$\dot{p}$	aircraft roll acceleration about X body axis, rad/s ²
$\dot{q}$	aircraft pitch acceleration about Y body axis, rad/s ²
$\dot{r}$	aircraft yaw acceleration about Z body axis, rad/s ²
ϕ, θ, ψ	aircraft roll, pitch, yaw angle about X, Y, Z body axes respectively, rad
x, y, z	inertial location of the aircraft, m
I_x, I_y, I_z	moments of inertia about X, Y, Z body axes respectively, kgm ²
I_{xz}, I_{yz}, I_{xy}	cross moment of inertia, kgm ²
$C_{x,t}, C_{y,t}, C_{z,t}$	total aerodynamic force coefficients in X, Y, Z axes respectively
$C_{l,t}, C_{m,t}, C_{n,t}$	total aerodynamic moment coefficients in X, Y, Z axes respectively
$\bar{q}$	free-stream dynamic pressure, N/m ²

$\bar{c}$	wing mean aerodynamic chord, m
S	area of the wings, m ²
b	wing span, m
g	acceleration due to gravity, m/s ²
h	altitude, m
m	aircraft mass, N
χ	flight path angle, rad
T	engine thrust, N
H_e	engine angular momentum, kgm ² /s
V_T	total airspeed, m/s
a_x, a_y, a_z	forward, normal and sideways acceleration respectively, g
δ_h	deflection angle of elevator, deg
δ_a	deflection angle of aileron, deg
δ_r	deflection angle of rudder respectively, deg
δ_{sb}	deflection angle of speed brake, deg
δ_{lef}	deflection angle of leading edge flap, deg
δ_t	throttle position in the non-dimensional range [0, 1]

Chapter 1

Introduction

1.1 Motivation

Recently, Fault Tolerant Control (FTC) has achieved more attention in the dynamic system area due to its ability to accommodate system component failures automatically and then maintain overall system stability and acceptable performance requirements in the event of such failures. FTC approaches are generally classified into two categories, viz., passive FTC and active FTC [100]. In the passive FTC approaches the robust control techniques are utilized to make the closed-loop systems insensitive to uncertainties and restricted faults. Consequently the approach does not lead to reconfigure the control systems by using online fault information and has a limited fault tolerant capability. In active FTC approaches, the control systems are reconfigured according to the faults in the systems by using a pre-determined control law or combining a new control scheme online to maintain the stability and acceptable performance of the overall system. As a result, the active FTC approaches have higher fault-tolerant capability compared to the passive FTC approaches. In this thesis, active FTC approaches are considered for the study.

Although active FTC approaches utilize the information of the unanticipated faults

to reconfigure the controllers and maintain the system performance, they have many different structures. According to [100], the active FTC methods can be classified into the following categories: those that are 1) based on off-line control laws, 2) based on on-line control laws, 3) based on Fault Detection and Identification (FDI) scheme to tolerate the unanticipated faults and 4) dependent on the use of a baseline controller. To design the off-line control laws, a good understanding on the fault information is necessary. This is not suitable for real-time fault-tolerant control when unpredictable faults may occur. FDI often suffers from missed-alarm and false-alarm rates where the occurrence of a fault can not be checked correctly. In this case the controlled system may become unstable or its performance may become deteriorated. The FTC control system structure considered in this thesis incorporates online control laws, a baseline controller and reconfiguration of the controllers to accommodate the faults.

Many areas of industry are furnished with fault-tolerant control systems to improve their reliability, maintainability and survivability by redesigning the controllers in the presence of faults. Fault-tolerant control systems for high-performance aircrafts called fault-tolerant flight control are critical to maintain the aircraft survivability and to increase the probability of mission success in the presence of sensor/actuator failures or control surfaces impairments since these failures may cause catastrophic flight accidents and lead to the loss of the aircraft. With modern aircrafts becoming sophisticated, the failure modes become complex and are unanticipated. Although experienced pilots can often cope with certain simple and anticipated failures like the high probability failures, certain complicated and unanticipated events are beyond their capabilities of handling. Consequently there is the need for aircraft fault-tolerant control scheme to reconfigure the aircraft control laws to mitigate the consequences of severe failures in the aircraft. In general, the aircraft fault-tolerant control schemes aim to lead the aircraft back to the equilibrium state, and even execute the scheduled operations in case of sensor/actuator failures or control surfaces impairments.

Among all the aircraft flight phases, autoland under failures is a very challenging fault-tolerant flight control problem since the aircraft flies at a considerable low altitude and low speed during landing and this is more prone to flight accidents due to some uncertain factors and failures. Modern aircrafts are equipped with automatic landing system for safe landing. The automatic landing system is coupled with the Instrument Landing Systems (ILS) or Global Positioning Systems (GPS) to guide the aircraft into the proper altitude, position, and approach angle during the landing phase. In general, autonomous landing by manned and unmanned aircraft includes two tasks, that is, to compute in realtime the desired optimal/near-optimal trajectories for landing and to track the reference trajectories in the presence of external disturbances. This requires that the controllers designed for nominal conditions are able to cope quickly with the failure conditions. However, the nominal automatic landing controllers work only within a specified operational safety envelope. When the conditions are beyond this envelope, such as wind disturbances or actuator failures, the nominal controllers often cannot be used. Thus only conventional controllers alone are no longer sufficient to provide enough control since they are unable to cope with these failures. In this case, new control approaches based on model-free intelligence methods have to be explored that adapt to the sudden changes in the environment as well as actuator failures.

Fuzzy inference systems are commonly used as model-free tools to model dynamic systems because of their powerful ability to approximate any continuous nonlinear function. It has been shown theoretically that a fuzzy inference system can approximate any function to any specified accuracy provided that sufficient rules are available. Specifically a fuzzy inference system consists of a series of linguistic rules in the format of “if-then” rules to approximate any unknown function mapping. This enables the approximation capabilities of human reasoning to be applied in knowledge-based systems.

The classic approach for fuzzy inference systems is to design the fuzzy rules by a domain expert *a priori* and then maintain them unchanged during learning. It is

well known that the environment or model often meets unpredicted disturbances or damages in realtime applications. This requires the fuzzy inference systems to be adaptive to track the transformation. An appropriate way to achieve this is to develop fuzzy inference systems based on neural networks, that is fuzzy-neural systems because neural networks have the learning ability. Fuzzy-neural systems are built by utilizing the standard structure of neural networks to approximate a fuzzy inference process. However, two issues need to be considered for designing an appropriate fuzzy-neural system; that is a structure identification and parameter adjustment. Structure identification is related with the determination of number of fuzzy rules while parameter adjustment is to adjust the parameters existing in the antecedent and consequent parts of the fuzzy rules.

Many methods have been proposed to realize the above two tasks for designing the fuzzy-neural systems, such as fuzzy c-mean clustering [9, 119], least-squares [51], back-propagation [22, 55] and so on. All these methods usually are classified as per two learning modes, viz., batch and sequential modes. In the batch mode, the total training data is available before learning and further many iterative training epoches are required to obtain good performance while in the sequential mode, the training data comes one-by-one and further only one learning epoch is used and the training stops after all the training data is processed. To suit real-time aircraft autolanding fault-tolerant control application, the learning algorithm for fuzzy-neural systems has to be in a sequential mode where the faults can be identified and controlled online without knowing them beforehand. This is impossible to achieve for batch learning algorithms.

In this study, we first propose two sequential fuzzy-neural schemes and then explore their applications in the aircraft autolanding problem under severe winds and actuator failures.

1.2 Objectives

The primary objective of this research is to develop appropriate fuzzy-neural schemes for aircraft fault-tolerant control, in particular for autoland fault-tolerant control under sudden environmental changes and actuator failures. More specifically, the objectives of the work in this thesis can be summarized as:

- To develop a sequential algorithm for designing a fuzzy-neural system. In this algorithm, the determination of the fuzzy rules and adjustment of the premise and consequent parameters in fuzzy rules can be achieved simultaneously. In the determination of the fuzzy rules, the fuzzy rules can be allocated and removed automatically without predefining them. When no new fuzzy rules are added the parameters existing in fuzzy rules are updated.
- To develop a *fast* sequential algorithm for training the fuzzy-neural system. In this algorithm, the consequent parameters of the fuzzy rules can be updated at a high speed by randomly assigning the values for the antecedent parameters existing in the fuzzy membership functions. In this case, the fuzzy-neural system becomes a linear system and then the consequent parameters can be analytically determined in a single step.
- To explore the above fuzzy-neural schemes for aircraft autoland under actuator failures and severe winds. Simulation studies are carried out based on a high performance fighter with five primary control surfaces consisting of two elevators, two ailerons and one rudder. The actuator failure scenarios comprise of single fault of either aileron or elevator stuck at certain deflections and double fault cases where one aileron and one elevator stuck at the same or opposite direction at different deflections. Moreover a typical wind model incorporating both the turbulence and a severe microburst has been used for the landing problem. During the landing the aircraft is required to satisfy certain strict touchdown conditions.

1.3 Major Contributions of the Thesis

Major contributions of the thesis are:

- A Sequential Adaptive Fuzzy Inference System called SAFIS is developed based on the functional equivalence between a fuzzy inference system and a Radial Basis Function (RBF) network. Specifically SAFIS uses the Growing and Pruning RBF (GAP-RBF) neural network proposed by Huang *et al.* [42]. In SAFIS, the concept of “Influence” of a fuzzy rule is introduced and using this the fuzzy rules are added or removed based on the input data received so far. If the input data does not warrant adding of fuzzy rules, then only the parameters of the “closest” (in a Euclidean sense) rule are updated using an Extended Kalman Filter (EKF) scheme.
- Another sequential fuzzy neural algorithm called Online Sequential Fuzzy Extreme Learning Machine (OS-Fuzzy-ELM) is proposed based on the Extreme Learning Machine (ELM) [41, 46]. The algorithm can be utilized for both the TSK and Mamdani type fuzzy models with any bounded nonconstant piecewise continuous membership function and for any type T-norm fuzzy logic operation. In OS-Fuzzy-ELM, the parameters of the fuzzy membership functions need not be adjusted during training and one can randomly assign the values to them and then analytically determine the consequent parameters. Also, OS-Fuzzy-ELM can learn the data sequentially in a one-by-one or chunk-by-chunk mode.
- Two adaptive fuzzy control strategies which respectively utilize the proposed SAFIS and OS-Fuzzy-ELM algorithms to aid a Baseline Trajectory Following Controller (BTFC) are developed for the aircraft automatic landing under the failures of stuck control surfaces and severe winds. BTFC has been designed using classical control methods under normal operating conditions with winds. The following fault scenarios are considered: i) Single fault of either aileron or elevator stuck at certain deflections and ii) Double fault

cases where one aileron and one elevator at the same or opposite direction are stuck at different deflections. Besides a severe wind model is considered in the failure scenarios and tight touchdown condition dispersions have been specified. Simulation results illustrate the proposed SAFIS aided BTFC and OS-Fuzzy-ELM aided BTFC clearly improve the fault tolerant envelopes achieved earlier by a neural aided BTFC and BTFC controllers.

- The performance of the proposed OS-Fuzzy-ELM and SAFIS algorithms are compared for the same aircraft autoland problem and the simulation results show that OS-Fuzzy-ELM achieves slightly better fault tolerant envelopes for all the failure scenarios considered.

1.4 Organization of the Thesis

The thesis is organized as follows:

Chapter 2 provides a brief review on fuzzy-neural systems. It covers both fuzzy inference systems and neural networks. Besides the relationship and integration between them are given in this chapter, specifically with reference to the work in this thesis.

In Chapter 3 the architecture of the proposed SAFIS algorithm is first presented and then the “Influence” of a fuzzy rule is introduced. Based on this, the SAFIS algorithm is given in detail by describing the adding and pruning criteria and parameter update method. The performance evaluation of the proposed SAFIS algorithm is carried out based on several benchmark problems including two nonlinear system identification problems and the Mackey-Glass time series prediction problem.

Chapter 4 mainly presents an overview of the aircraft autoland fault-tolerant control problem. A nonlinear six degree-of-freedom mathematical model for a high performance fighter is first introduced and then the landing tasks executed by the

aircraft is presented. The wind models and actuator failure types used for simulation study are given. Besides, the control structure for solving the aircraft fault-tolerant control during the autolanding is described in the chapter together with a conventional controller called Baseline Trajectory Following Controller (BTFC). The application of the proposed SAFIS algorithm in aircraft autolanding fault-tolerant control problem is explored in Chapter 5. Its performance is evaluated based on the failure types as described in Chapter 4 and compared with those achieved from the conventional BTFC and a neural aided BTFC controller.

In Chapter 6, the ELM algorithm is first reviewed and then the details of the proposed OS-Fuzzy-ELM algorithm are described. The performance evaluation of the proposed OS-Fuzzy-ELM is presented by using three benchmark problems drawn from function approximation and classification fields. Furthermore the performance of the proposed OS-Fuzzy-ELM algorithm is verified by using chunk-by-chunk learning modes, different T-norm operators and fuzzy models.

In Chapter 7 the OS-Fuzzy-ELM algorithm is used for the aircraft autolanding fault-tolerant control problem. It is also compared with the BTFC and neural aided BTFC controllers. The comparison of OS-Fuzzy-ELM and SAFIS algorithms by using the aircraft autolanding problem is described in Chapter 8.

Finally, conclusions from this study and future work are summarized in Chapter 9.

Chapter 2

A Brief Review of Fuzzy-Neural Systems

This chapter presents a brief overview of fuzzy-neural systems which are built by incorporating fuzzy inference systems using the structure of neural networks. Thus before reviewing the fuzzy-neural systems, the basic concepts about fuzzy inference systems and the neural networks relevant to the work in this thesis are described. Although they have different theoretical bases, they also have the common property, viz., they can determine the nonlinear mapping based on numerical data. Besides they are functionally equivalent under some minor conditions.

2.1 Fuzzy Inference Systems

Fuzzy inference systems are developed based on the fuzzy logic and fuzzy set theory introduced by Zadeh in 1965 [137] and have been widely used in many disciplines such as engineering, economics and other areas [27, 32, 50, 61, 67, 82, 83, 94–96, 101, 113, 118, 121, 126]. A fuzzy inference system using fuzzy if-then rules can model the qualitative aspects of human knowledge and reasoning processes for dealing with ill-defined and uncertain systems without employing precise quantitative analy-

sis. Takagi and Sugeno [121] first explored the fuzzy identification systematically by using a fuzzy model which described fuzzy rules by local linear input-output functions. This kind of fuzzy model has been employed in numerous practical applications like control [101, 118], prediction and inference [61]. An adaptive fuzzy controller was proposed by Mamdani [82, 83] by using the proportional and derivative error signals and the control actions were produced based on the plant performance. The proposed controller was successfully applied in controlling a steam engine of a model industrial plant. Fuzzy systems can be utilized as fuzzy controllers for autonomous mobile robots which have complex control architectures. Saffiotti *et al.* [113] presented a fuzzy controller for an autonomous mobile robot to pursue strategic goals such as a reactive behavior to avoid obstacles on the way and a goal-oriented behavior to reach a given location. In addition the fuzzy systems have been employed as the fuzzy controllers to solve the aircraft fault-tolerant problem during landing phase for achieving the safe landing under disturbances [27, 94, 95]. Many applications in the pattern classification and prediction field have been explored successfully by the researchers [32, 50, 67, 96, 126] using fuzzy systems. Gopal *et al.* [32] utilized fuzzy logic for classification of Partial Discharge (PD) patterns for the diagnosis of High Voltage insulation system. Wei and Mendel [126] employed fuzzy inference systems to construct a classifier for non-ideal environments where precise probabilistic methods are difficult or impossible to use. In [96] fuzzy inference systems were employed to predict the Gross Domestic Product (GDP) development by designing a prediction model. Konjic *et al.* [67] utilized fuzzy inference systems to predict load curves at low voltage substations used by different types consumers such as residents, industry and so on. In [50] fuzzy systems were employed to predict the global solar radiation data. A higher accuracy was achieved by these compared with the conventional methods. From the above overview, it can be seen that the fuzzy inference systems are very useful to solve many practical problems which involve a high level of uncertainty, complexity, or nonlinearity and are difficult to solve by using conventional modelling methods. In general a fuzzy inference system consists of four principal

components, viz., a fuzzifier, a fuzzy rule base, a fuzzy inference engine, and a defuzzifier. Next we will give a brief description for each of them which will help the algorithms described in later chapters.

2.1.1 Fuzzy Sets

Fuzzy set theory is an extension of classical set theory assessed in binary terms, that is an element either belongs or does not belong to the set. In the fuzzy set, the membership of the elements in relation to the set is gradually assessed with the aid of a membership function $\mu \rightarrow [0, 1]$. In general, any bounded nonconstant continuous function can be chosen as a candidate for the membership function [16, 134]. The following list gives the type of membership functions which are most commonly used [53].

(1) Triangular Membership Function

The triangular membership function (Trimf) includes two parameters (c, a) and is given by,

$$\text{Trimf}(x; c, a) = \begin{cases} 0 & x \leq c - a \\ \frac{x-c+a}{a} & c - a < x \leq c \\ \frac{c+a-x}{a} & c < x \leq c + a \\ 0 & c + a < x \end{cases} \quad (2.1)$$

(2) Trapezoid Membership Function

Trapezoid membership function (Trapmf) includes four parameters $\{c_1, a_1, c_2, a_2\}$

and is given as,

$$\text{Trapmf}(x; c_1, a_1, c_2, a_2) = \begin{cases} 0 & x \leq c_1 - a_1 \\ \frac{x - c_1 + a_1}{a_1} & c_1 - a_1 < x \leq c_1 \\ 1 & c_1 < x \leq c_2 \\ \frac{c_2 + a_2 - x}{a_2} & c_2 < x \leq c_2 + a_2 \\ 0 & c_2 + a_2 < x \end{cases} \quad (2.2)$$

(3) Gaussian Membership Function

Gaussian membership function (Gaussmf) includes two parameters $\{c, a\}$ and is given by,

$$\text{Gaussmf}(x; c, a) = \exp \left(- \left(\frac{x - c}{a} \right)^2 \right) \quad (2.3)$$

(4) Two-sided Gaussian Membership Function

Two-sided Gaussian membership function (Gauss2mf) includes four parameters $\{c_1, a_1, c_2, a_2\}$ and is given by,

$$\text{Gauss2mf}(x; c_1, a_1, c_2, a_2) = \begin{cases} \exp \left(- \left(\frac{x - c_1}{a_1} \right)^2 \right) & x \leq c_1 \\ 1 & c_1 < x < c_2 \\ \exp \left(- \left(\frac{x - c_2}{a_2} \right)^2 \right) & x \geq c_2 \end{cases} \quad (2.4)$$

(5) Cauchy Membership Function

The Cauchy membership function (Cauchymf) includes two parameters $\{c, a\}$ and is given by,

$$\text{Cauchymf}(x; c, a) = \frac{1}{1 + \left(\frac{x - c}{a} \right)^2} \quad (2.5)$$

(6) π -shaped Membership Function

π -shaped membership function is the product of S membership function and

Z membership function.

π -shaped membership function (Pimf) includes two parameters $\{c, a\}$ and is given by,

$$\text{Pimf}(x; c, a) = \begin{cases} S(x; c - a, c) & x \leq c \\ Z(x; c, c + a) & x > c \end{cases} \quad (2.6)$$

where $S(x; c - a, c)$ is the S membership function and given by,

$$S(x; c - a, c) = \begin{cases} 0 & x \leq c - a \\ 2 \left(\frac{x - c + a}{a} \right)^2 & c - a < x \leq \frac{2c - a}{2} \\ 1 - 2 \left(\frac{c - x}{a} \right)^2 & \frac{2c - a}{2} < x \leq c \\ 1 & c < x \end{cases} \quad (2.7)$$

$Z(x; c, c + a)$ is the Z membership function and given by,

$$Z(x; c, c + a) = \begin{cases} 1 & x \leq c \\ 1 - 2 \left(\frac{x - c}{a} \right)^2 & c < x \leq \frac{2c + a}{2} \\ 2 \left(\frac{c + a - x}{a} \right)^2 & \frac{2c + a}{2} < x \leq c + a \\ 0 & c + a < x \end{cases} \quad (2.8)$$

(7) Difference between two Sigmoidally-shaped Membership Functions

The sigmoid function includes two parameters $\{c, a\}$ and is given by,

$$f(x; c, a) = \frac{1}{1 + \exp(ax + c)} \quad (2.9)$$

The membership function of the difference between two sigmoidal functions (Dsigmf) includes four parameters $\{c_1, a_1, c_2, a_2\}$ and is given by,

$$\text{Dsigmf}(x; c_1, a_1, c_2, a_2) = f(x; c_1, a_1) - f(x; c_2, a_2) \quad (2.10)$$

(8) Product of two Sigmoidally-shaped Membership Functions

The membership function of the product of two sigmoidal functions (Psigmf) includes four parameters $\{c_1, a_1, c_2, a_2\}$ and is given as,

$$\text{Psigmf}(x; c_1, a_1, c_2, a_2) = f(x; c_1, a_1) * f(x; c_2, a_2) \quad (2.11)$$

To visualize each membership function described above, the graphs from these membership functions are illustrated in Appendix A. Among these membership functions, the triangular and Gaussian membership functions are commonly used by many researchers [6, 7, 55, 62, 63, 70, 124, 131].

2.1.2 Fuzzy Rules

The fuzzy rule base comprises of a series of fuzzy rules in the format of “if-then” form and consistent with the human languages. The fuzzy rules are generally classified into two types. One type is that the antecedent (if) part and the consequent (then) part are both described by the fuzzy sets. The second type is that only the antecedent part is described by fuzzy sets whereas the consequent part is described by real values. The most commonly used Mamdani fuzzy model uses the first type of fuzzy rules while the Takagi-Sugeno-Kang (TSK) fuzzy model utilizes the second type of fuzzy rules.

(1) Mamdani Fuzzy Model

The Mamdani fuzzy model is given by the following rules [82],

$$\text{Rule } i : \text{if } (x_1 \text{ is } A_{1i}) \text{ AND } (x_2 \text{ is } A_{2i}) \cdots \text{ AND } (x_n \text{ is } A_{ni}), \text{ then} \\ (y_1 \text{ is } B_{1i}) \cdots (y_m \text{ is } B_{mi})$$

where $A_{ji} (j = 1, 2, \dots, n; i = 1, 2, \dots, \tilde{N})$ and $B_{ki} (k = 1, 2, \dots, m; i = 1, 2, \dots, \tilde{N})$ are the fuzzy sets of the j th input variable x_j and the k th output variable y_k in rule i , n is the dimension of the input vector $\mathbf{x} (\mathbf{x} = [x_1, \dots, x_n]^T)$, m is the dimension of the output vector $\mathbf{y} (\mathbf{y} = [y_1, \dots, y_m]^T)$, and $\tilde{N}$ is the number of fuzzy rules.

(2) TSK Fuzzy Model

The TSK fuzzy model is given by the following rules [121],

Rule i : if $(x_1 \text{ is } A_{1i}) \text{ AND } (x_2 \text{ is } A_{2i}) \text{ AND } \cdots \text{ AND } (x_n \text{ is } A_{ni})$,
then $(y_1 \text{ is } \omega_{1n}) \cdots (y_m \text{ is } \omega_{mi})$

where $\omega_{ki}(k = 1, 2, \dots, m; i = 1, 2, \dots, \tilde{N})$ is the crisp value and it may be any function of the input variables but the commonly used is a constant value or a linear combination of the input variables. In case of a linear function, it is given by $\omega_{ki} = q_{ki0} + q_{ki1}x_1 + \cdots + q_{kin}x_n$.

Note that different from the above fuzzy rules, the classification rules are proposed in [108, 130] to solve the pattern recognition problem. Although the classification rules is constructed by using a series of IF-THEN rules, the IF part states a condition over the data and the THEN part includes a class label, which is different from fuzzy rules where the IF part is the fuzzy sets and the THEN part is either fuzzy sets or real values. Besides the learning scheme of classification rules is different from that of fuzzy rules. The emphasis of this thesis is focused on the fuzzy rules and not classification rules, thus the learning scheme details for classification rules will not be described here. The following gives the details about the learning scheme of fuzzy rules.

2.1.3 Fuzzifier

The fuzzifier aims to perform a mapping from a crisp input $\mathbf{x}'$ into a fuzzy set A' . One of the most commonly used fuzzifier methods is the singleton fuzzifier, that is $\mu_{A'}(\mathbf{x}') = 1$ for $\mathbf{x}' = \mathbf{x}$ and $\mu_{A'}(\mathbf{x}') = 0$ for $\mathbf{x}' \neq \mathbf{x}$. All the studies in this thesis are based on the singleton fuzzifier.

2.1.4 Fuzzy Inference Engine

In fuzzy rule i , a fuzzy implication is employed to define a fuzzy set as given below,

$$\Psi_i : A_{1i} \otimes A_{2i} \otimes \cdots \otimes A_{ni} \longrightarrow B_{1i} + \cdots + B_{mi} \quad (2.12)$$

where ‘ $\otimes$ ’ is the T-norm operator, ‘ $+$ ’ represents the union of the independent variables.

T-norm includes many types, such as minimum, algebraic product and so on. When the degree to which the given j th input variable x_j and the k th output variable y_k satisfy the quantifier A_{ji} and B_{ki} in rule i are specified by their membership function $\mu_{A_{ji}}(x_j)$ and $\mu_{B_{ki}}(y_k)$, the minimum T-norm operation is given by,

$$\begin{aligned} R_i &= \mu_{A_{1i}}(x_1) \otimes \mu_{A_{2i}}(x_2) \otimes \cdots \otimes \mu_{A_{ni}}(x_n) \\ &= \min \{ \mu_{A_{1i}}(x_1), \mu_{A_{2i}}(x_2), \cdots, \mu_{A_{ni}}(x_n) \} \end{aligned} \quad (2.13)$$

For the algebraic product it is given as,

$$R_i = \mu_{A_{1i}}(x_1) * \mu_{A_{2i}}(x_2) * \cdots * \mu_{A_{ni}}(x_n) = \prod_{j=1}^n \mu_{A_{ji}}(x_j) \quad (2.14)$$

Equation 2.12 with any membership function as described above is expressed as,

$$\mu_{\Psi_i}(\mathbf{x}, \mathbf{y}) = \mu_{A_{1i}}(x_1) \otimes \mu_{A_{2i}}(x_2) \otimes \cdots \otimes \mu_{A_{ni}}(x_n) \otimes (\mu_{B_{1i}}(y_1) + \mu_{B_{2i}}(y_2) \cdots + \mu_{B_{mi}}(y_m)) \quad (2.15)$$

The fuzzy inference engine aims to determine a mapping from the fuzzy sets in the input space to the fuzzy sets in the output space based on the sup-star composition. Let A' be an arbitrary fuzzy set in the input space, then the fuzzy set B in the

output space is given by,

$$\begin{aligned}
 \mu_B(\mathbf{y}) &= \mu_{A' \circ \Psi_i}(\mathbf{y}) \\
 &= \sup_{\mathbf{x}'} [\mu_{A'}(\mathbf{x}') \otimes \mu_{\Psi_i}(\mathbf{x}', \mathbf{y})] \\
 &= \sup_{\mathbf{x}'} [\mu_{A'}(\mathbf{x}') \otimes \mu_{A_{1i}}(x'_1) \otimes \mu_{A_{2i}}(x'_2) \otimes \mu_{A_{ni}}(x'_n) \otimes (\mu_{B_{1i}}(y_1) \\
 &\quad + \mu_{B_{2i}}(y_2) \cdots + \mu_{B_{mi}}(y_m))]
 \end{aligned} \tag{2.16}$$

where $\circ$ denotes the sup-star composition where star represents the T-norm operation.

2.1.5 Defuzzifier

The defuzzifier performs a mapping from the fuzzy sets in the output space to crisp points in the output space. Many schemes including center average, mean of maximum, maximum criterion, etc [53, 83] have been proposed to realize the defuzzifier. Since the center average defuzzifier is employed in this thesis, only its equation is given below,

$$\mathbf{y} = \frac{\sum_{i=1}^{\tilde{N}} \bar{\mathbf{y}}_i \mu_B(\bar{\mathbf{y}}_i)}{\sum_{i=1}^{\tilde{N}} \mu_B(\bar{\mathbf{y}}_i)} \tag{2.17}$$

where $\bar{\mathbf{y}}_i = [\bar{y}_{1i}, \dots, \bar{y}_{mi}]$ and $\bar{y}_{ki}$ is the point at which B_{ki} achieves its maximum value, that is $\mu_B(\beta_{ki}) = 1$.

For the TSK fuzzy model, its consequence is the crisp values and thus the defuzzifier operation is ignored. Based on fuzzy models introduced above, the fuzzy inference systems can be classified into two types, viz., Mamdani fuzzy inference systems and TSK fuzzy inference systems.

For Mamdani fuzzy systems, by using a center average defuzzifier and singleton

fuzzifier, the system output $\mathbf{y}$ for given input $\mathbf{x}$ is given by [124],

$$\mathbf{y} = \frac{\sum_{i=1}^{\tilde{N}} \beta_i \mu_{A' \circ \Psi_i}(\beta_i)}{\sum_{i=1}^{\tilde{N}} \mu_{A' \circ \Psi_i}(\beta_i)} \quad (2.18)$$

where

$$\begin{aligned} \mu_{A' \circ \Psi_i}(\beta_i) = \sup_{\mathbf{x}'} [\mu_{A'}(\mathbf{x}') \otimes \mu_{A_{1i}}(x'_1) \otimes \mu_{A_{2i}}(x'_2) \otimes \mu_{A_{ni}}(x'_n) \otimes (\mu_{B_{1i}}(\beta_{1i}) \\ + \mu_{B_{2i}}(\beta_{2i}) \cdots + \mu_{B_{mi}}(\beta_{mi}))] \end{aligned} \quad (2.19)$$

Due to the singleton fuzzifier, $\mu_{A'}(\mathbf{x}') = 1$ for $\mathbf{x}' = \mathbf{x}$ and because of center average defuzzifier, $\mu_{B_{1i}}(\beta_{1i}) = \mu_{B_{2i}}(\beta_{2i}) = \cdots = \mu_{B_{mi}}(\beta_{mi}) = 1$

Thus equation 2.18 becomes as,

$$\mathbf{y} = \frac{\sum_{i=1}^{\tilde{N}} \beta_i R_i}{\sum_{i=1}^{\tilde{N}} R_i} \quad (2.20)$$

where $\beta_i = [\beta_{1i}, \cdots, \beta_{mi}]$ and β_{ki} is the point at which B_{ki} achieves its maximum value, that is $\mu_B(\beta_{ki}) = 1$.

For the TSK fuzzy systems, since the consequent parts are the crisp values, the defuzzifier is removed. The system crisp output is achieved by the weighted average sum of each rule's output and given by

$$\mathbf{y} = \frac{\sum_{i=1}^{\tilde{N}} \omega_i R_i}{\sum_{i=1}^{\tilde{N}} R_i} \quad (2.21)$$

where R_i is the weight and computed based on equation 2.13. $\omega_i = [\omega_{1i}, \omega_{2i}, \cdots, \omega_{ki}]$ ($k = 1, 2, \cdots, m; i = 1, 2, \cdots, \tilde{N}$) is the crisp output of rule i and its elements may

be any function of the input variables. In case of linear function, the k th element ω_{ki} equals to $\omega_{ki} = q_{ki0} + q_{ki1}x_1 + \cdots + q_{kin}x_n$.

2.1.6 Universal Approximation of Fuzzy Inference Systems

As introduced above, fuzzy inference systems have been utilized widely in a number of practical applications, especially in the areas of control and modelling. The fuzzy inference systems aim to approximate a desired control or decision up to a given level of accuracy. Since the desired control or decision and the fuzzy systems can be denoted as functions on some appropriate input and output spaces, from mathematics point of view, fuzzy inference systems aim to find a mapping from the input space to the output space. The theories that have been developed by researchers [16,134,138,139] have demonstrated that a fuzzy inference system based on Mamdani and TSK fuzzy models can approximate any continuous functions under certain conditions.

Zeng and Singh [138, 139] has proved the Mamdani fuzzy system with Pseudo Trapezoid-Shaped (PTS) membership functions, the algebraic product operator and centroid defuzzifier can approximate any target continuous function on a compact domain to any degree of accuracy. PTS membership function is defined as follows [139]: A membership function g of a fuzzy set is called a Pseudo Trapezoid-Shaped (PTS) membership function if g is a continuous function in U given by

$$g(x; a, b, c, d, h) = \begin{cases} I(x), & x \in [a, b) \\ h, & x \in [b, c] \\ D(x), & x \in (c, d] \\ 0, & x \in U - [a, d] \end{cases} \quad (2.22)$$

where $[a, b] \subset U \subset R$, $a \leq b \leq c \leq d$, $I(x) \geq 0$ is a monotonically increasing function in $[a, b)$ and $D(x) \geq 0$ is a monotonically decreasing function in $(c, d]$.

Furthermore, it is shown in [16] that the Mamdani fuzzy system with any bounded nonconstant continuous membership function and any types of T-norm fuzzy logic operation can approximate any target continuous function on a compact domain to any degree of accuracy.

Ying [134] has proved that the fuzzy systems based on TSK fuzzy models with any bounded nonconstant continuous membership functions and any types of T-norm operators can uniformly approximate any multivariate continuous function on a compact domain to any degree of accuracy.

After briefly describing the concepts of fuzzy inference systems, the basic concepts about neural networks are briefly introduced in the next section.

2.2 Neural Networks

A Neural Network (NN) [36] is made of a number of basic computational units called neurons, each of which generates a linear or nonlinear transformation of the data by the activation functions and is connected to each other by synaptic weights. NNs are able to learn, generalize and adapt to unknown uncertainties. Thus they have been successfully applied in many fields such as pattern classification, identification and control of highly nonlinear dynamic systems etc. [10, 17, 20, 49, 64, 65, 72, 73, 86, 90–92, 105, 112, 114, 122, 127]. Khan *et al.* [64] first explored the application of NNs in diagnostic classification of cancer using gene expression data. Specifically the NN was used to classify the Small, Round Blue-Cell Tumors (SRBCT) into 4 diagnostic categories based on gene expression signatures. By using the NN all the samples are correctly classified, even for the samples that often present diagnostic difficulties in clinical practice. Narendra and Parthasarathy [92] employed NNs for identifying several models which represented the nonlinear dynamic systems. Chen *et al.* [20] has explored the application of NNs in the nonlinear dynamic system identification. These papers illustrated that NNs were capable of successfully identifying nonlinear dynamic systems. NNs have

been found to be particularly useful for controlling highly uncertain, nonlinear and complex systems and thus many neural-network-based control structures have been proposed for controlling nonlinear dynamic systems. As described in [2] the neural-network-based control schemes proposed in the literature can be classified into two categories; one class [90–92] is to use the neural networks to relieve the complex diagnostics, tedious modelling, unwieldy tuning or excessive computational effort in conventional control schemes and the other class [72, 114, 122] is to use neural networks as the controllers and update neural networks by using available input signals obtained either from the real plant or a model simulation. NNs have been successfully applied in building the predictive models for predicting MPEG-coded video source traffic [10, 17]. The predictive models built using NNs were operating in real-time and captured the inherent non-stationarities and nonlinearities associated with the MPEG-coded real-time video streams.

Based on the above review, one can find that NNs are very useful to identify and control nonlinear systems due to their learning and adaptive ability. However NNs may use different structures to achieve the modelling task. According to different connection structure among neurons and the flow of the input information NNs can be classified broadly into two classes, viz., feedforward neural networks [14, 87] and Recurrent Neural Networks (RNNs) [26, 110]. Although RNNs are appropriate to represent dynamic systems, the feedback neurons are difficult to be trained [80]. In this study, we only concentrate on the feedforward neural networks. Among the feedforward neural networks, Single Layer Feedforward Networks (SLFNs) are the most commonly used ones because of their simple structure. The basic structure of SLFNs comprises of an input layer, a hidden layer and an output layer. In a SLFN, the activation functions existing in the hidden layer may be of different forms but the most commonly used ones are the Sigmoid and radial basis functions. A SLFN with the hidden layer node using radial basis functions is called as Radial Basis Function (RBF) network. The work in this thesis mainly concentrates on the SLFN with Sigmoid active functions and the RBF networks and a brief description of them is given below.

2.2.1 SLFNs with Sigmoidal Activation Functions

Figure 2.1 illustrates the SLFN structure with $\tilde{N}$ hidden nodes using Sigmoidal activation functions. Assuming the network with n input variables $\mathbf{x} = [x_1, x_2, \dots, x_n]$

Figure 2.1: Structure of SLFN with additive activation function

and m output variables $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_m]$, the output of the SLFN can be represented by

$$\hat{\mathbf{y}} = \sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{a}_i \cdot \mathbf{x} + b_i), \quad \mathbf{x} \in \mathbf{R}^n, \mathbf{a}_i \in \mathbf{R}^n \quad (2.23)$$

where β_i is the weight connecting the i th hidden node to the output node; $\mathbf{a}_i$ is the weight vector connecting the input layer to the i th hidden node and b_i is the bias of the i th hidden node. $\mathbf{a}_i \cdot \mathbf{x}$ denotes the inner product of vectors $\mathbf{a}_i$ and $\mathbf{x}$

in $\mathbf{R}^n$; $g(x) : R \rightarrow R$ is the Sigmoidal activation function of the hidden node and given by,

$$g(x) = \frac{1}{1 + \exp(-x)} \quad (2.24)$$

2.2.2 Radial Basis Function Networks

The RBF networks are originally motivated by the locally tuned response in biological neurons and their activation functions of the hidden neurons are based on the RBFs determined by the distances between the input vector and prototype vectors. RBF was firstly used in neural networks by Broomhead *et al.* [14] where a two-layer network structure was set up and RBF was employed as computation units in the hidden layer. For RBF network, typical selections of RBFs are [14,107]:

(1) Thin-plate-spline function

$$g(r) = r^2 \log(r) \quad (2.25)$$

(2) Multiquadric function

$$g(r) = (r^2 + b^2)^{1/2} \quad (2.26)$$

(3) Inverse multiquadric function

$$g(r) = \frac{1}{(r^2 + b^2)^{1/2}} \quad (2.27)$$

(4) Gaussian function

$$g(r) = \exp(-r^2/b^2) \quad (2.28)$$

where r represents the Euclidean distance between a center $\mathbf{a}$ and the input data point $\mathbf{x}$, i.e. $r = \|\mathbf{x} - \mathbf{a}\|$, b is a real value parameter and denotes the width of RBFs.

Among these types of RBFs, Gaussian function is the most widely used in many applications because it is capable of making an accurate global mapping and refining local features without much alteration on the already learned mapping [88].

Figure 2.2 illustrates the structure of RBF networks with the Gaussian basis func-

Figure 2.2: Structure of RBF network

tions. From the figure it can be found that different from SLFNs with Sigmoid active functions in the RBF networks the input layer directly feeds the input variables into the hidden layer without linking weights. Then the hidden layer utilizes a set of Gaussian basis functions for the input vectors to realize a nonlinear transformation when the input variables are mapped into the hidden unit space. Same

with SLFNs with Sigmoidal activation functions the units in the output layer are the linear combination of hidden unit outputs. Assuming the RBF network with n input variables $\mathbf{x} = [x_1, x_2, \dots, x_n]$, $\tilde{N}$ hidden neurons and m output variables $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_m]$, the outputs are given by

$$\hat{\mathbf{y}} = \sum_{i=1}^{\tilde{N}} \beta_i g_i(\mathbf{x}) \quad (2.29)$$

where β_i is the weight connecting the i th hidden neuron to the output neurons and $g_i(\mathbf{x})$ is the response of the i th hidden neuron for an input vector $\mathbf{x}$:

$$g_i(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{a}_i\|^2}{b_i^2}\right) \quad (2.30)$$

where $\mathbf{a}_i$ and b_i are the center and width of the i th hidden neuron.

2.2.3 Learning Algorithms

Many algorithms like back-propagation algorithm [36], Extreme Learning Machine (ELM) [44–46, 48], Resource Allocating Network (RAN) [104] etc. have been proposed to train the two kinds of neural networks. However, based on different learning modes, the algorithms used for training the above two types of neural networks can be broadly divided into two classes namely batch learning schemes and sequential learning schemes. In batch learning, it is assumed that the complete training data is available before the training commences. The training usually involves cycling the data over a number of epochs. In sequential learning, the data arrives one by one and after the learning of each data it is discarded and the notion of epoch does not exist. In practical applications, new training data arrives sequentially and to handle this using batch learning, one has to retrain the network all over again, resulting in large training time. Hence, in these cases, sequential learning algorithms are generally preferred over batch learning algorithms as they do not require retraining whenever a new data is received. Thus, if one strictly

applies the above features of the sequential algorithms, many of the existing algorithms are not sequential. One major bottleneck seems to be that they need the entire training data ready for training before the training procedure starts and thus they are not really sequential. This point is highlighted in a brief review of the existing algorithms given below.

One of the most popular methods for training the SLFNs is the back-propagation algorithm which is based on the gradient of the output error [36]. In the algorithm the input data is repeatedly presented to the networks and then the output of the network is compared to the target output for obtaining the output error. The error is back-propagated to adjust the linking weights for SLFNs with Sigmoid hidden nodes or the parameters existing in the hidden nodes and linking weights between hidden nodes and output nodes for SLFNs with RBF nodes such that the output of the neural networks is close to the target output for further decreasing the output error. If a stopping criterion such as a predefined maximum training epoch, a threshold for the change of the weights and so on is satisfied, the training process will stop. Many improved algorithms on the back-propagation algorithm like Quasi-Newton and Levenberg-Marquardt algorithms [19, 103, 129] have been employed to improve the convergence speed by using the second order information of the cost function. However the gradient descent based learning methods easily converge to local minima if the cost function is not convex with respect to its parameters. Furthermore such learning algorithms may require many iterative learning steps to obtain better learning performance and this increases the computation effort.

Recently a new fast learning algorithm referred to as ELM has been developed for the SLFNs [44–46, 48]. Different from the gradient descent based learning methods, in ELM the input weights and bias for the SLFNs with Sigmoid nodes and the parameters in the hidden nodes for the SLFNs with RBF nodes are randomly assigned and based on this a SLFN is considered as the linear system and the output weights are analytically determined using the generalized inverse operation

of the hidden layer output matrix. ELM has been successfully applied in some applications [35, 47, 76, 132, 133, 140]. It has been shown that ELM can provide better generalization performance at extremely high learning speed [45–47, 76, 140]. The universal approximation capability of ELM has been rigorously proved in Huang, *et al.* [41] using an incremental method (named Incremental Extreme Learning Machine (I-ELM)). However the algorithm is a batch learning algorithm because it requires the whole training data available before training.

Some sequential learning algorithms have been proposed for the SLFNs with RBF nodes and they included RAN and its extensions. In these algorithms the number of RBF hidden nodes need not be predefined. Platt [104] developed the RAN algorithm for RBF network in which the hidden neurons were sequentially recruited based on the novelty of the new data and the parameters were updated by means of the Least Mean Square (LMS) algorithm. Another improved algorithm of RAN called RANEKF was introduced by Kadirkamanathan and Niranjan [60] utilized the Extended Kalman Filter (EKF) instead of LMS algorithm to estimate the network parameters. This algorithm achieved more compact network size and better accuracy than RAN. However RAN and RANEKF can not remove any insignificant hidden neurons during learning.

Lu *et al.* [136] proposed a pruning strategy based on the consistent contribution of each hidden neuron to the overall network output to remove those insignificant neurons. The resulting network called Minimal RAN (MRAN) was shown to improve the RAN and RANEKF and the simulation results from several applications in the field of function approximation and pattern classification [135, 136] showed that MRAN achieved more compact network size and better approximation accuracy than RAN and RANEKF. However in MRAN some parameters have to be chosen *a priori* by trial and error such as the data window used to compute the contribution of hidden neurons in the growing and pruning criterion of the hidden neurons. This increases the algorithm complexity.

A recent new sequential learning algorithm called Growing and Pruning RBF net-

work (GAP-RBF) was proposed by Huang *et al.* [42] to grow and prune the hidden neurons of RBF networks based on a neurons's significance. Significance of a neuron was defined as its contribution to the network output averaged over all the input data received so far and was computed from the statistical point by considering that the input data followed the uniform distribution. This avoided the selection of data window to calculate the contribution of the hidden neurons existing in MRAN.

Actually a RBF network can functionally be equivalent with a fuzzy inference system under certain conditions therefore a RBF network can be employed to determine the fuzzy rules automatically during learning by linking the RBF hidden neurons with the fuzzy rules. This will be introduced next.

2.3 Functional Equivalence between fuzzy inference systems and radial basis function networks

From the above description for the fuzzy inference systems and the RBF networks, one can find that they have different origins, viz., fuzzy inference systems are from the cognitive science and RBF networks are from the physiology area. However common characteristics not only in the operation on data but also in the learning process to achieve desired mappings are shared by them [52]. Jang and Sun [52] have proved that a standard RBF neural network and a fuzzy inference system are functionally equivalent under some minor conditions. They are given as:

- (1) The number of basis function units is equal to the number of fuzzy if-then rules.
- (2) The consequent part of each fuzzy rule is a constant.
- (3) The membership functions of the premise variable within each fuzzy rule are

Gaussian functions with the same variance.

- (4) The T-norm operator used to compute the firing strength of each rule is of multiplication type.
- (5) Both RBF network and the fuzzy inference system under consideration use the same method (i.e., normalized or nonnormalized calculation) to derive the overall outputs.

Based on these conditions, the membership value of linguistic variables A_{ji} is given by the following Gaussian basis function,

$$\mu_{A_{ji}}(x_j) = \exp \left(- \left(\frac{x_j - a_{ji}}{b_i} \right)^2 \right) \quad (2.31)$$

By means of the multiplication T-norm, the firing strength of the i th rule for the antecedent part is given by,

$$R_i = \prod_{j=1}^n \mu_{A_{ji}}(x_j) = \exp \left(- \left(\frac{\mathbf{x} - \mathbf{a}_i}{b_i} \right)^2 \right) \quad (2.32)$$

Thus, the basis function units in the RBF neural networks are related with the if-then fuzzy rules and able to be utilized to achieve the firing strength of the antecedent part. Recently, fuzzy-neural systems have appeared using the benefits of both the fuzzy systems and neural networks. A brief review of them is given in the next section.

2.4 Fuzzy-Neural Systems

Although fuzzy inference systems utilize the fuzzy set theory to approximate human reasoning capabilities from the mathematical point, the finally achieved performance mainly depends on the fuzzy rules. If the fuzzy rules are not appropriate and deviate from the requirement of the system itself, this may result in poor

performance. Besides, although the rules are correct, it is hard to determine the appropriate parameters for the fuzzy rules. Inappropriate parameters also may result in poor performance.

To solve these problems, many researchers have built fuzzy-neural systems by incorporating the fuzzy inference process in the structure of neural networks and then the learning ability of neural networks were used to adjust the fuzzy rules. Except for some special fuzzy-neural systems which made use of fuzzy neurons and fuzzy weights [34, 102], most of the recent fuzzy-neural systems [5, 7, 22, 51, 55, 63, 70, 131] have been built based on the standard feedforward network with local fields to approximate the fuzzy inference systems with local properties. In these fuzzy-neural systems, the neurons with local fields correspond to the fuzzy rules and the proposed algorithms for designing the fuzzy-neural systems have considered two issues, that is, a structure identification and the parameter adjustment. Structure identification is related with the determination of number of fuzzy rules while parameter adjustment is to adjust the parameters existing in the antecedent and consequent parts of fuzzy rules.

Jang [51] has developed an Adaptive-Network-based Fuzzy Inference System (ANFIS) where a hybrid learning method was utilized to identify the system parameters. The parameters in the membership functions were updated by a gradient descent method and the parameters in the consequent parts were adjusted by means of a least-square method. The number of fuzzy rules was determined according to a grid-type partition which resulted in the exponential increase of the number of fuzzy rules as the input variables increased. Chiu [21] solved this problem by selecting some significant input variables from all the input variables as the input of the fuzzy systems. However these algorithms require cycling the whole training data over a number of learning cycles (epochs). Thus, they are batch learning algorithms. Besides, in these algorithms the number of fuzzy rules are determined beforehand and cannot be varied according to learning process.

Many researchers [22, 55, 70, 131] have employed the functional equivalence between

a RBF neural network and a fuzzy inference system to achieve the determination of the number of fuzzy rules and parameter adjustment simultaneously during learning. These schemes utilize the learning capabilities of the RBF for changing the rules as well as adjusting the parameters since the hidden neurons of the RBF networks are related to the fuzzy rules.

A hierarchically self-organizing approach proposed by Cho and Wang [22] automatically generated fuzzy rules without predefining the number of fuzzy rules based on the error and distance criterion of fuzzy basis functions. The parameters in the fuzzy rules were modified by the gradient descent algorithm. However, the algorithm requires cycling the whole training data over a number of learning cycles (epochs) and hence it is not a truly sequential learning scheme.

Juang and Lin [55] have proposed a Self-constructing Neural Fuzzy Inference Network (SONFIN) in which the fuzzy rules were extracted online from the training data together with the parameter update for all existing fuzzy rules using the gradient descent method. For adding a new fuzzy rule, SONFIN utilized the distance criterion between the new input data and the center of the Gaussian membership function in the existing fuzzy rules. Although this algorithm is sequential in nature, it does not remove the fuzzy rules once created even though that rule is not effective. This may result in a structure where the number of rules may be large.

In most of the real applications, not all fuzzy rules contribute significantly to the system performance during the entire time period. A fuzzy rule may be active initially, but may later contribute little to the system output. For this reason, the insignificant fuzzy rules have to be removed during learning to realize a compact fuzzy system structure. Using the ideas of adding and pruning hidden neurons to form a minimal RBF network in [136], a hierarchical on-line self-organizing learning algorithm for Dynamic Fuzzy Neural Networks (DFNN) has been proposed in [131]. Another on-line Self-Organizing Fuzzy Neural Network (SOFNN) proposed by Leng *et al.* [70] also included a pruning method. The pruning method utilized the Optimal Brain Surgeon (OBS) approach to determine the importance of each

rule. In the two algorithms, the least-square method was utilized to update the parameters for all the existing fuzzy rules. However, in these two algorithms the pruning criteria need all the past data received so far. Hence, they are not strictly sequential and further requires increased memory for storing all the past data.

A Dynamic Evolving Neural-Fuzzy Inference System (DENFIS) was proposed by Kasabov and Song [63] where the fuzzy rules were created depending on the position of the input vector in the input space and the output was dynamically calculated based on m -most active fuzzy rules which have been created during the past learning process. Angelov and Filev [7] proposed an evolving Takagi-Sugeno model (eTS) that recursively updated TS model structure based on the *potential* of the input data (defined based on its distances to all other data points received so far). In this algorithm, a new rule was added when the potential of the new data was higher than the potential of the existing rules or a new rule was modified when the potential of the new data was higher than the potential of the existing rules and the new data was close to an old rule. These two algorithms are truly sequential learning algorithms. However, the algorithms cannot simplify the rule base during learning by ignoring the rules which may become irrelevant with the future data samples when the data sample sequentially arrives. A simplified version of the eTS learning algorithm that simplified the rule base, called the `simpl_eTS` was proposed by Angelov and Filev [5]. The algorithm utilized the concept of the scatter which was similar to the notion of potential but computationally more efficient. The algorithm could simplify the rule base to make the rules representative based on the population of each rule determined by the number of the data samples that belonged to a particular cluster. If the population of a rule was less than 1% of the total data at the moment of appearance of a rule, the rule was ignored from the rule base by setting its firing strength to zero. Besides, these algorithms employed the least-square method to modify the parameters of the existing fuzzy rules.

Compared with the above learning algorithms, the sequential fuzzy-neural schemes which are discussed in this thesis, have the following distinguishing features:

- (1) The training observations are sequentially (one-by-one or chunk-by-chunk with varying or fixed chunk length) presented to the learning algorithm.
- (2) At any time, only the newly arrived single or chunk of observations (instead of the entire past data) are seen and learned.
- (3) A single or a chunk of training observations is discarded as soon as the learning procedure for that particular (single or chunk of) observation(s) is completed.
- (4) The learning algorithm has no prior knowledge as to how many training observations will be presented.

Specifically two fuzzy-neural schemes satisfying the above features are proposed in this thesis. The first scheme can learn the training data one-by-one mode while the second scheme can learn the training data not only one-by-one but also chunk-by-chunk (with fixed or varying length) mode.

In this chapter a brief review of the fuzzy-neural systems is given together with the concept description of the fuzzy inference systems and neural networks. In the next chapter we present the first sequential fuzzy-neural scheme developed in this thesis and its performance is evaluated by using several benchmark problems including two nonlinear system identification problems and the Mackey-Glass time series prediction problem.

Chapter 3

Sequential Adaptive Fuzzy Inference System (SAFIS)

In this chapter, a Sequential Adaptive Fuzzy Inference System (SAFIS) is developed to realize a compact fuzzy system with lesser number of rules by using the concepts from GAP-RBF neural network [42]. Different from the previous work [22, 70, 131] where all the past data received have to be stored, the proposed SAFIS algorithm only makes use of the current data during learning. The SAFIS algorithm has two aspects: determination of the fuzzy rules and the adjustment of the premise and consequent parameters in the fuzzy rules. SAFIS uses the concept of influence of a fuzzy rule to add and remove rules during learning. Similar to the significance of a neuron in GAP-RBF, the influence of a fuzzy rule is also a mark of its contribution to the system output in a statistical sense when the input data is uniformly distributed. The learning procedure of SAFIS algorithm is similar to that of GAP-RBF algorithm, that is addition and removal of a fuzzy rule is equivalent to the addition and removal of a neuron according to the function equivalence described in Section 2. The difference between them is that the growing and pruning criteria in SAFIS are based on the influence of a fuzzy rule while the GAP-RBF makes use of the significance of a neuron to add and remove the hidden neurons. However conceptually they are utilized as the mark to add

and delete fuzzy rules and neurons.

3.1 Architecture of SAFIS

Generally, a wide class of MIMO nonlinear dynamic systems can be represented by the nonlinear discrete model with an input-output description form:

$$\mathbf{y}(t) = \mathbf{f}[\mathbf{y}(t-1), \mathbf{y}(t-2), \dots, \mathbf{y}(t-k+1); \mathbf{u}(t), \mathbf{u}(t-1), \dots, \mathbf{u}(t-p+1)] \quad (3.1)$$

where $\mathbf{y}$ is a vector containing m system outputs, $\mathbf{u}$ is a vector for r system inputs; $\mathbf{f}$ is a nonlinear vector function, representing m hypersurfaces of the system, and k and p are the maximum lags of the output and input respectively.

Selecting $[\mathbf{y}(t-1), \dots, \mathbf{y}(t-k+1); \mathbf{u}(t), \mathbf{u}(t-1), \dots, \mathbf{u}(t-p+1)]$, $\mathbf{y}(t)$ as the fuzzy system's input-output $\mathbf{x}_t, \mathbf{y}_t$ at time t , the above equation becomes

$$\mathbf{y}_t = \mathbf{f}(\mathbf{x}_t) \quad (3.2)$$

The aim of the new SAFIS algorithm is to approximate $\mathbf{f}$ such that

$$\hat{\mathbf{y}}_t = \hat{\mathbf{f}}(\mathbf{x}_t) \quad (3.3)$$

where $\hat{\mathbf{y}}_t$ is the output of SAFIS. This means that the objective is to minimize the error between the system output and the output of SAFIS, $\|\mathbf{y}_t - \hat{\mathbf{y}}_t\|$. Before describing the details of the algorithm, the structure of SAFIS network is first described below.

The structure of SAFIS illustrated by Figure 3.1 consists of five layers to realize the following fuzzy rule model:

Rule i : if $(x_1 \text{ is } A_{1i}) \text{ AND } \dots \text{ AND } (x_n \text{ is } A_{ni})$, then $(\hat{y}_1 \text{ is } \omega_{i1}) \dots (\hat{y}_m \text{ is } \omega_{im})$

where $\omega_{ik} (k = 1, 2, \dots, m; i = 1, 2, \dots, \tilde{N})$ is a constant consequent parameter in

rule i , $A_{ji}(j = 1, 2, \dots, n)$ is fuzzy set of the j th input variable x_j in rule i , n is the dimension of the input vector $\mathbf{x}(\mathbf{x} = [x_1, \dots, x_n]^T)$, $\tilde{N}$ is the number of fuzzy rules, m is the dimension of the output vector $\hat{\mathbf{y}}(\hat{\mathbf{y}} = [\hat{y}_1, \dots, \hat{y}_m]^T)$. In SAFIS, the number of fuzzy rules $\tilde{N}$ varies. Initially, there is no fuzzy rule and then during learning fuzzy rules are added and removed.

Figure 3.1: Structure of SAFIS

Layer 1: In layer 1, each node represents an input variable and directly transmits the input signal to layer 2.

Layer 2: In this layer each node represents the membership value of each input variable. SAFIS utilizes the function equivalence between a RBF network and a FIS and thus its antecedent part (if part) in fuzzy rules is achieved by Gaussian

functions of the RBF network. The membership value $\mu_{A_{ji}}(x_i)$ of the j th input variable x_j in the i th rule is given by

$$\mu_{A_{ji}}(x_j) = \exp\left(-\frac{(x_j - c_{ji})^2}{a_i^2}\right), i = 1, 2, \dots, \tilde{N} \quad (3.4)$$

where $\tilde{N}$ is the number of the Gaussian functions, c_{ji} is the center of the i th Gaussian function for the j th input variable, a_i is the width of the i th Gaussian function. In SAFIS, the width of all the input variables in the i th Gaussian function is the same.

Layer 3: Each node in the layer represents the if part of if-then rules obtained by the algebraic product T-norm operation and the total number of such rules is $\tilde{N}$. The firing strength (if part) of the i th rule is given by

$$R_i(\mathbf{x}) = \prod_{j=1}^n \mu_{A_{ji}}(x_j) = \exp\left(-\sum_{j=1}^n \frac{(x_j - c_{ji})^2}{a_i^2}\right) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{a_i^2}\right) \quad (3.5)$$

Layer 4: The nodes in the layer are named as normalized nodes whose number is equal to the number of the nodes in third layer. The i th normalized node is given by

$$G_i(\mathbf{x}) = \frac{R_i(\mathbf{x})}{\sum_{i=1}^{\tilde{N}} R_i(\mathbf{x})} \quad (3.6)$$

Layer 5: Each node in this layer corresponds to an output variable, which is given by the weighted sum of the output of each normalized rule. The system output is calculated by

$$\hat{\mathbf{y}} = \sum_{i=1}^{\tilde{N}} \omega_i G_i(\mathbf{x}) = \frac{\sum_{i=1}^{\tilde{N}} \omega_i R_i(\mathbf{x})}{\sum_{i=1}^{\tilde{N}} R_i(\mathbf{x})} \quad (3.7)$$

where $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_m]^T$, $\boldsymbol{\omega}_i = [\omega_{i1}, \omega_{i2}, \dots, \omega_{im}]^T$.

Similar to the significance concept of a neuron in GAP-RBF [42], the SAFIS algorithm uses the concept of “influence” of a rule to realize the growing and pruning of fuzzy rules. It is described below.

3.1.1 “Influence” of a Fuzzy Rule

As per equation (3.7), the contribution of the i th rule to the overall output for an input observation $\mathbf{x}_l$ is given by:

$$E(i, l) = |\boldsymbol{\omega}_i| \frac{R_i(\mathbf{x}_l)}{\sum_{i=1}^{\tilde{N}} R_i(\mathbf{x}_l)} \quad (3.8)$$

Then the contribution of the i th rule to the overall output based on all input data N received so far is obtained by:

$$E(i) = |\boldsymbol{\omega}_i| \frac{\sum_{l=1}^N R_i(\mathbf{x}_l)}{\sum_{i=1}^{\tilde{N}} \sum_{l=1}^N R_i(\mathbf{x}_l)} \quad (3.9)$$

Dividing both the numerator and denominator by N in equation (3.9), the equation becomes

$$E(i) = |\boldsymbol{\omega}_i| \frac{\sum_{l=1}^N R_i(\mathbf{x}_l)/N}{\sum_{i=1}^{\tilde{N}} \sum_{l=1}^N R_i(\mathbf{x}_l)/N} \quad (3.10)$$

Using the significance concept of GAP-RBF, the *influence* of the i th fuzzy rule is defined as its statistical contribution to the overall output of SAFIS. When $N \rightarrow \infty$

the influence of the i th rule is given by

$$E_{\text{inf}}(i) = \lim_{N \rightarrow \infty} E(i) = \lim_{N \rightarrow \infty} |\omega_i| \frac{\sum_{l=1}^N R_i(\mathbf{x}_l)/N}{\sum_{i=1}^{\tilde{N}} \sum_{l=1}^N R_i(\mathbf{x}_l)/N} \quad (3.11)$$

Calculation of $E_{\text{inf}}(i)$ using the above equation requires the knowledge of $(\mathbf{x}_l, \mathbf{y}_l)$, $l = 1, \dots, N$. In the truly sequential learning scheme, this is not possible. An alternate way of calculating $E_{\text{inf}}(i)$ is by using the distribution of the inputs and follows the same approach as introduced in [43]. In order to compute $E_{\text{inf}}(i)$ one has to compute first E_i defined by

$$E_i = \lim_{N \rightarrow \infty} \frac{\sum_{l=1}^N R_i(\mathbf{x}_l)}{N} \quad (3.12)$$

Assume that the observations $(\mathbf{x}_l, \mathbf{y}_l)$, $l = 1, \dots$, are drawn from a sampling range X with a sampling density function $p(\mathbf{x})$. Consider a situation where N observations have been learned by the sequential learning scheme. Let the sampling range X be divided into M small spaces $\Delta_k, k = 1, \dots, M$. The size of Δ_k is represented by $S(\Delta_k)$. Since the sampling density function is $p(\mathbf{x})$ there are around $N \cdot p(\mathbf{x}_k) \cdot S(\Delta_k)$ samples in each Δ_k , where $\mathbf{x}_k$ is any point chosen in Δ_k . When the number of input observations N is large and Δ_k is small. From equation (3.12), we have

$$\begin{aligned} E_i &\approx \lim_{M \rightarrow \infty} \frac{\sum_{k=1}^M R_i(\mathbf{x}_k) \cdot N p(\mathbf{x}_k) \cdot S(\Delta_k)}{N} \\ &= \lim_{M \rightarrow \infty} \sum_{k=1}^M R_i(\mathbf{x}_k) \cdot p(\mathbf{x}_k) \cdot S(\Delta_k) \\ &= \int_X R_i(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \\ &= \int_X \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{a_i^2}\right) p(\mathbf{x}) d\mathbf{x} \end{aligned} \quad (3.13)$$

Equation (3.13) shows that the joint density function of the input variable $p(\mathbf{x})$ need to be known for calculating E_i . However in real implementation it is hard to obtain it. Thus we give an assumption that the distribution of the n attributes $(x_1, \dots, x_j, \dots, x_n)^T$ of observations $\mathbf{x}$ are independent from each other. According to the assumption, a simplified model for Equation (3.13) can be obtained where the joint density function $p(\mathbf{x})$ of $\mathbf{x}$ can be written as: $p(\mathbf{x}) = \prod_{j=1}^n p_j(x)$. Here $p_j(x)$ is the density function of the j th attribute x_j of observations. In this case, equation (3.13) is simplified as:

$$E_i = \prod_{j=1}^n \left(\int_{b_j}^{d_j} \exp\left(-\frac{\|x - c_{ji}\|^2}{a_i^2}\right) p_j(x) dx \right) \quad (3.14)$$

where (b_j, d_j) is the interval of the j th attribute x_j of observations, $j = 1, \dots, n$.

When the input samples follow uniform distribution, the sampling density function $p(\mathbf{x})$ is given by $p(\mathbf{x}) = \frac{1}{S(X)}$, where $S(X)$ is the size of the range X given by $S(X) = \int_{\mathbf{x}} 1 d\mathbf{x}$. Substituting for $p(\mathbf{x})$ in equation (3.13) we get

$$E_i = \int_X \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{a_i^2}\right) \frac{1}{S(X)} d\mathbf{x} \quad (3.15)$$

Note that in general the width a_i of a rule i is much less than the size of range X , the above equation can be approximated as

$$\begin{aligned} E_i &\approx \frac{1}{S(X)} \left(2 \int_0^{+\infty} \exp\left(-\frac{x^2}{a_i^2}\right) dx \right)^n \\ &= \frac{\pi^{n/2} a_i^n}{S(X)} \\ &= \frac{(1.8a_i)^n}{S(X)} \end{aligned} \quad (3.16)$$

Thus, based on equation (3.16) the influence of the i th rule is given by:

$$E_{\text{inf}}(i) = |\omega_i| \frac{(1.8a_i)^n}{\sum_{i=1}^{\tilde{N}} (1.8a_i)^n} \quad (3.17)$$

Remark 1: The significance of a neuron proposed in GAP-RBF [42] is defined based on the average contribution of an individual neuron to the output of the RBF network. Under this definition, one may need to estimate the input distribution range $S(X)$. However, the influence of a rule introduced here is different from the significance of a neuron proposed in GAP-RBF [42]. In fact the influence of a neuron is defined as the relevant significance of the neuron compared to summation of significance of all the existing RBF neurons. Seen from equation (3.17), with the introduction of influence one need not estimate the input distribution range $S(X)$ and the implementation has been simplified.

Influence of a rule is utilized for the addition and deletion of a fuzzy rule in SAFIS algorithm as indicated below.

3.2 SAFIS Algorithm

The learning algorithm of SAFIS can automatically add and remove fuzzy rules using ideas similar to GAP-RBF for hidden neurons [42]. A description of dynamically adding and removing the fuzzy rules along with the details of parameter adjustment when there are no addition of rules is given below.

3.2.1 Adding a Fuzzy Rule

SAFIS begins with no fuzzy rules. When the first input $\mathbf{x}_1, \mathbf{y}_1$ is received, it is translated into the first rule whose parameters are given as, $\mathbf{c}_1 = \mathbf{x}_1, \boldsymbol{\omega}_1 = \mathbf{y}_1, a_1 = \kappa \|\mathbf{x}_1\|$. Then as new inputs $\mathbf{x}_t, \mathbf{y}_t$ ($t(>1)$ is the time index) are received sequentially during learning, growing of fuzzy rules is based on the following two criteria which

are the distance criterion and the influence of the newly added fuzzy rule $\tilde{N} + 1$:

$$\begin{cases} \|\mathbf{x}_t - \mathbf{c}_{nr}\| > \epsilon_t \\ E_{inf}(\tilde{N} + 1) = |\mathbf{e}_t| \frac{(1.8\kappa\|\mathbf{x}_t - \mathbf{c}_{nr}\|)^n}{\sum_{i=1}^{\tilde{N}+1} (1.8a_i)^n} > e_g \end{cases} \quad (3.18)$$

where ϵ_t , e_g are thresholds to be selected appropriately, $\mathbf{x}_t$ is the latest input data, $\mathbf{c}_{nr}$ is the center of the fuzzy rule nearest to $\mathbf{x}_t$. e_g is the growing threshold and is chosen according to the desired accuracy of SAFIS. $\mathbf{e}_t = \mathbf{y}_t - \hat{\mathbf{y}}_t$, $\mathbf{y}_t$ is the true value, $\hat{\mathbf{y}}_t$ is the approximated value, κ is an overlap factor that determines the overlap of fuzzy rules in the input space, ϵ_t is the distance threshold which decays exponentially and is given by

$$\epsilon_t = \max \{ \epsilon_{\max} \times \gamma^t, \epsilon_{\min} \} \quad (3.19)$$

where $\epsilon_{\max}$, $\epsilon_{\min}$ are the largest and smallest length of interest, γ is the decay constant. This equation shows that initially it is the largest length of interest in the input space which allows fewer fuzzy rules to coarsely learn the system and then it decreases exponentially to the smallest length of interest in the input space which allows more fuzzy rules to finely learn the system.

3.2.2 Allocation of Antecedent and Consequent Parameters

When the new fuzzy rule $\tilde{N} + 1$ is added, its corresponding antecedent and consequent parameters are allocated as follows:

$$\begin{cases} \omega_{\tilde{N}+1} &= \mathbf{e}_t \\ \mathbf{c}_{\tilde{N}+1} &= \mathbf{x}_t \\ a_{\tilde{N}+1} &= \kappa\|\mathbf{x}_t - \mathbf{c}_{nr}\| \end{cases} \quad (3.20)$$

3.2.3 Parameter Adjustment

In parameter modification, SAFIS utilizes a winner rule strategy similar to the work done by Huang *et al.* [42]. The key idea of the winner rule strategy is that only the parameters related to the selected winner rule are updated by the extended Kalman Filter (EKF) algorithm in every step. The ‘winner rule’ is defined as the rule that is closest (in the Euclidean distance sense) to the current input data as in [42]. The parameter vector existing in all the fuzzy rules is given by

$$\begin{aligned}\boldsymbol{\theta}_t &= \begin{bmatrix} \boldsymbol{\theta}_1 & \cdots & \boldsymbol{\theta}_{\text{nr}} & \cdots & \boldsymbol{\theta}_{\tilde{N}} \end{bmatrix}^T \\ &= \begin{bmatrix} \boldsymbol{\omega}_1, \mathbf{c}_1, a_1, & \cdots, & \boldsymbol{\omega}_{\text{nr}}, \mathbf{c}_{\text{nr}}, a_{\text{nr}}, & \cdots, & \boldsymbol{\omega}_{\tilde{N}}, \mathbf{c}_{\tilde{N}}, a_{\tilde{N}} \end{bmatrix}^T\end{aligned}\quad (3.21)$$

where $\boldsymbol{\theta}_{\text{nr}} = [\boldsymbol{\omega}_{\text{nr}}, \mathbf{c}_{\text{nr}}, a_{\text{nr}}]$ is the parameter vector of the nearest fuzzy rule and its gradient is derived as follows:

$$\begin{aligned}\dot{\boldsymbol{\omega}}_{\text{nr}} &= \frac{\partial \hat{\mathbf{y}}_t}{\partial \boldsymbol{\omega}_{\text{nr}}} = \frac{\partial \hat{\mathbf{y}}_t}{\partial R_{\text{nr}}} \frac{\partial R_{\text{nr}}}{\partial \boldsymbol{\omega}_{\text{nr}}} = \frac{R_{\text{nr}}}{\sum_{i=1}^{\tilde{N}} R_i} \\ \dot{\mathbf{c}}_{\text{nr}} &= \frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{c}_{\text{nr}}} = \frac{\partial \hat{\mathbf{y}}_t}{\partial R_{\text{nr}}} \frac{\partial R_{\text{nr}}}{\partial \mathbf{c}_{\text{nr}}} = \frac{\boldsymbol{\omega}_{\text{nr}} - \hat{\mathbf{y}}_t}{\sum_{i=1}^{\tilde{N}} R_i} \frac{\partial R_{\text{nr}}}{\partial \mathbf{c}_{\text{nr}}} \\ \dot{a}_{\text{nr}} &= \frac{\partial \hat{\mathbf{y}}_t}{\partial a_{\text{nr}}} = \frac{\partial \hat{\mathbf{y}}_t}{\partial R_{\text{nr}}} \frac{\partial R_{\text{nr}}}{\partial a_{\text{nr}}} = \frac{\boldsymbol{\omega}_{\text{nr}} - \hat{\mathbf{y}}_t}{\sum_{i=1}^{\tilde{N}} R_i} \frac{\partial R_{\text{nr}}}{\partial a_{\text{nr}}}\end{aligned}\quad (3.22)$$

$$\begin{aligned}\frac{\partial R_{\text{nr}}}{\partial \mathbf{c}_{\text{nr}}} &= 2R_{\text{nr}} \frac{\mathbf{x}_t - \mathbf{c}_{\text{nr}}}{a_{\text{nr}}^2} \\ \frac{\partial R_{\text{nr}}}{\partial a_{\text{nr}}} &= 2R_{\text{nr}} \frac{\|\mathbf{x}_t - \mathbf{c}_{\text{nr}}\|^2}{a_{\text{nr}}^3}\end{aligned}$$

After obtaining the gradient vector of the nearest fuzzy rule, that is $\mathbf{B}_{nr} = [\dot{\omega}_{nr}, \dot{c}_{nr}, \dot{a}_{nr}]^T$, EKF is used to update its parameters as follows:

$$\begin{aligned} \mathbf{K}_t &= \mathbf{P}_{t-1} \mathbf{B}_t [\mathbf{R}_t + \mathbf{B}_t^T \mathbf{P}_{t-1} \mathbf{B}_t]^{-1} \\ \boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} + \mathbf{K}_t \mathbf{e}_t \\ \mathbf{P}_t &= [\mathbf{I} - \mathbf{K}_t \mathbf{B}_t^T] \mathbf{P}_{t-1} + Q_0 \mathbf{I} \end{aligned} \quad (3.23)$$

When a new rule is added, the dimension of $\mathbf{P}_t$ increases to

$$\begin{pmatrix} \mathbf{P}_{t-1} & 0 \\ 0 & p_0 \mathbf{I}_{Z_1 \times Z_1} \end{pmatrix} \quad (3.24)$$

where Z_1 is the dimension of the parameters introduced by the newly added rule, p_0 is an initial value of the uncertainty assigned to the newly allocated rule. In the chapter, p_0 is set to 1.0 for all the examples.

3.2.4 Removing a Fuzzy Rule

If the influence of rule i is less than a certain pruning threshold e_p , the i th rule's influence to the output is negligible and should be removed. The pruning threshold e_p is chosen a priori. Given the pruning threshold e_p , rule i will be removed if

$$E_{\text{inf}}(i) = |\omega_i| \frac{(1.8a_i)^n}{\sum_{i=1}^{\tilde{N}} (1.8a_i)^n} < e_p \quad (3.25)$$

Considering the Gaussian function $R(x) = \exp(-\frac{x^2}{\sigma^2})$, its first and second derivatives will approach zero much faster when x moves away from zero. Thus, in EKF the gradient vector of the parameters for all the rules except the nearest rule will approach zero more quickly than that of the nearest rule

$\left(\frac{R_{nr}}{\sum_{i=1}^{\tilde{N}} R_i}, \frac{2(\omega_{nr} - \hat{y}_t) R_{nr}}{\sum_{i=1}^{\tilde{N}} R_i} \frac{\mathbf{x}_t - \mathbf{c}_{nr}}{a_{nr}^2}, \frac{2(\omega_{nr} - \hat{y}_t) R_{nr}}{\sum_{i=1}^{\tilde{N}} R_i} \frac{\|\mathbf{x}_t - \mathbf{c}_{nr}\|^2}{a_{nr}^3} \right)$. In this case, one may only need to adjust parameters of the nearest rule without adjusting the parameters of all

rules when a new observation enters and a new rule needs not be added. At the same time, all rules need not be checked for possible pruning. If a new observation arrives and the growing criterion (3.17) is satisfied, a new rule will be added. The existing rules will maintain their influence because their parameters remain unchanged after learning the new observation. Simultaneously the newly added rule is also influencing and therefore it is not necessary to check for pruning after a new rule is added. If the growing criterion (3.17) is not satisfied after a new observation arrives, a new rule will not be added and only the parameters of the nearest rule will be modified. As such only the nearest rule needs to be checked for pruning.

The SAFIS algorithm is summarized below.

Proposed SAFIS Algorithm: Given the growing and pruning thresholds e_g, e_p , for each observation $(\mathbf{x}_t, \mathbf{y}_t)$, where $\mathbf{x}_t \in R^n$, $\mathbf{y}_t \in R^m$ and $t = 1, 2, \dots$, do

compute the overall system output:

$$\hat{\mathbf{y}}_t = \frac{\sum_{i=1}^{\tilde{N}} \omega_i R_i(\mathbf{x}_t)}{\sum_{i=1}^{\tilde{N}} R_i(\mathbf{x}_t)} \quad (3.26)$$

$$R_i(\mathbf{x}_t) = \exp\left(-\frac{1}{a_i^2} \|\mathbf{x}_t - \mathbf{c}_i\|^2\right)$$

where $\tilde{N}$ is the number of fuzzy rules.

calculate the parameters required in the growth criterion:

$$\epsilon_t = \max\{\epsilon_{\max} \gamma^t, \epsilon_{\min}\}, \quad (0 < \gamma < 1) \quad (3.27)$$

$$\mathbf{e}_t = \mathbf{y}_t - \hat{\mathbf{y}}_t$$

apply the criterion for adding rules:

$$\text{If } \|\mathbf{x}_t - \mathbf{c}_{\text{nr}}\| > \epsilon_t \text{ and } E_{\text{inf}}(\tilde{N} + 1) = |\mathbf{e}_t| \frac{(1.8\kappa \|\mathbf{x}_t - \mathbf{c}_{\text{nr}}\|)^n}{\sum_{i=1}^{\tilde{N}+1} (1.8a_i)^n} > e_g$$

allocate a new rule $\tilde{N} + 1$ with

$$\begin{aligned}\omega_{\tilde{N}+1} &= \mathbf{e}_t \\ \mathbf{c}_{\tilde{N}+1} &= \mathbf{x}_t \\ a_{\tilde{N}+1} &= \kappa \|\mathbf{x}_t - \mathbf{c}_{\text{nr}}\|\end{aligned}\tag{3.28}$$

Else

adjust the system parameters ω_{nr} , $\mathbf{c}_{\text{nr}}$, a_{nr} for the nearest rule only by using the EKF method.

check the criterion for pruning the rule:

$$\text{If } E_{\text{inf}}(\text{nr}) = |\omega_{\text{nr}}| \frac{(1.8a_{\text{nr}})^n}{\sum_{i=1}^{\tilde{N}} (1.8a_i)^n} < e_p$$

remove the nr-th rule

reduce the dimensionality of EKF

Endif

Endif

3.2.5 Selection of Predefined Parameters

In SAFIS, some parameters need to be selected in advance. They include the distance thresholds $(\epsilon_{\text{max}}, \epsilon_{\text{min}}, \gamma)$, the overlap factor (κ) for determining the width of the newly added rule, the growing threshold (e_g) for a new rule and the pruning threshold (e_p) for removing an insignificant rule. Based on the observation from many experiments, a general selection procedure for the predefined parameters is given as follows: ϵ_{max} is set to around the upper bound of input variables; ϵ_{min} is set to around 10% of ϵ_{max} ; γ is suggested to be selected from the range $[0.9, 0.999]$ according to the problem considered; e_p is set to around 10% of e_g . The overlap factor (κ) is utilized to initialize the width of the newly added rule and chosen according to different problems. κ is suggested to be chosen in the range $[1.0, 2.0]$. The growing threshold e_g is chosen according to the system performance. The smaller e_g the better the system performance, but the resulting system structure is more complex.

An example is given to illustrate the effects of the parameters $(e_g, \kappa, \epsilon_{\max}, \gamma)$ on the system structure and performance. Consider the following two-dimensional *sinc* function:

$$z = \text{sinc}(x, y) = \frac{\sin(x)\sin(y)}{xy} \quad (3.29)$$

In the simulation, 2500 training data pairs (x, y) are drawn from the input range $[-10, 10] \times [-10, 10]$. At the same time, 100 testing data pairs (x, y) are drawn from the same input range.

The general rule for choosing the parameters $(\epsilon_{\min}, e_p)$ is obeyed. $\epsilon_{\min}$ is set to 10% of $\epsilon_{\max}$; e_p is set to the 10% of e_g . The parameters $e_g, \epsilon_{\max}, \kappa$ and γ are observed in the range $[0.001, 0.05], [1.0, 10.0], [1.0, 2.0]$ and $[0.9, 0.999]$ respectively to illustrate their effect on the resulting system structure and testing accuracy. Table 3.1, Table 3.2 and Table 3.3 give the effect of parameter e_g on system performance in terms of number of rules and the testing RMS error under different $\epsilon_{\max}, \kappa$ and γ values. From the these tables, it is easy to find that with the increase of e_g the number of rules is decreased and also system performance (testing RMS error) become worse with the same $\epsilon_{\max}, \kappa$ and γ value. The three tables further illustrate that changing the three parameters $(\epsilon_{\max}, \kappa, \gamma)$ by a small amount does not result in the significant variations in the finally achieved system complexity and testing accuracy. However these parameters are problem-dependent and need to be determined according to the problem considered. Besides the above guidelines for setting the parameters, one may find the optimal parameters using search techniques like GA for some complex problems.

Table 3.1: Effect of parameter e_g on system performance (number of rules and the testing RMS error) under different $\epsilon_{\max}$ values and $\kappa = 1.0, \gamma = 0.997$

$\epsilon_{\max} \backslash e_g$	0.001	0.005	0.01	0.05
1.0	(61, 0.0198)	(17, 0.0385)	(11, 0.0535)	(2, 0.0912)
5.0	(45, 0.0233)	(17, 0.0385)	(11, 0.0535)	(2, 0.0912)
10.0	(41, 0.0249)	(14, 0.0386)	(9, 0.0461)	(2, 0.0912)

Table 3.2: Effect of parameter e_g on system performance (number of rules and the testing RMS error) under different κ values and $\epsilon_{\max} = 10.0$, $\gamma=0.997$

$\kappa \backslash e_g$	0.001	0.005	0.01	0.05
1.0	(41, 0.0249)	(14, 0.0386)	(9, 0.0461)	(2, 0.0912)
1.5	(50, 0.0350)	(18, 0.0586)	(15, 0.0598)	(3, 0.1382)
2.0	(52, 0.0557)	(25, 0.0902)	(15, 0.1384)	(3, 0.1391)

Table 3.3: Effect of parameter e_g on system performance (number of rules and the testing RMS error) under different γ values and $\epsilon_{\max} = 10.0$, $\kappa = 1.0$

$\gamma \backslash e_g$	0.001	0.005	0.01	0.05
0.9	(58, 0.0167)	(17, 0.0385)	(9, 0.0535)	(2, 0.0912)
0.93	(58, 0.0167)	(17, 0.0385)	(9, 0.0535)	(2, 0.0912)
0.96	(41, 0.0229)	(16, 0.0394)	(9, 0.0538)	(2, 0.0912)
0.99	(45, 0.0201)	(16, 0.0373)	(9, 0.0524)	(2, 0.0912)
0.992	(38, 0.0229)	(16, 0.0373)	(10, 0.0600)	(2, 0.0912)
0.995	(40, 0.0247)	(19, 0.0327)	(13, 0.0340)	(2, 0.0912)
0.997	(41, 0.0249)	(14, 0.0386)	(9, 0.0461)	(2, 0.0912)
0.999	(30, 0.0357)	(17, 0.0534)	(10, 0.0636)	(3, 0.0931)

3.3 Performance Evaluation of SAFIS

Three benchmark problems including two nonlinear system identification problems and one chaotic time series (Mackey-Glass) prediction problem are utilized to verify the performance of SAFIS. For the first system identification problem, performance of SAFIS is compared with other well known sequential algorithms such as MRAN [136], RANEKF [60], eTS [7], Simpl_eTS [5] and Hybrid Algorithm (HA) [125]. For the second benchmark system identification problem performance of SAFIS is compared with MRAN [136], RANEKF [60], eTS [7], Simpl_eTS [5] and SONFIN [55]. For the chaotic time series prediction problem, the comparison is done with MRAN [136], RANEKF [60], eTS [6] and Simpl_eTS [5]. In all the studies, the parameters (r, Ω) for eTS and Simpl_eTS where r is the distance and Ω is the LSE parameter [5,7] are tuned to obtain the best performance. Performance comparison is done in terms of accuracy and the complexity (the number of rules) of the fuzzy system. For these problems, the SAFIS algorithm goes through the training data sequentially in a single pass and builds up the fuzzy inference system by adding and

removing the rules along with their parameters. Then its performance is evaluated on the unseen test data.

3.3.1 Nonlinear Dynamic System Identification

Narendra and Parthasarathy [92] have suggested two special forms of the nonlinear system model given in equations (3.30) and (3.31).

Model I:

$$y(t) = f[y(t-1), y(t-2), \dots, y(t-k)] + \sum_{i=1}^p \beta_i u(t-i) \quad (3.30)$$

Model II:

$$y(t) = f[y(t-1), y(t-2), \dots, y(t-k)] + g[u(t-1), u(t-2), \dots, u(t-p)] \quad (3.31)$$

These two models of nonlinear systems have been used here for performance comparison.

Selecting $[y(t-1), y(t-2), \dots, y(t-k), u(t-1), u(t-2), \dots, u(t-p)]$ and $y(t)$ as the input-output of SAFIS, the identified model is given by this equation

$$\hat{y}(t) = \hat{f}(y(t-1), y(t-2), \dots, y(t-k), u(t-1), u(t-2), \dots, u(t-p)) \quad (3.32)$$

where $\hat{f}$ is the SAFIS approximation and $\hat{y}(t)$ is the output of the SAFIS.

3.3.1.1 Identification Problem 1

The first nonlinear dynamic system to be identified represents model I and is described by Wang and Yen [125]

$$y(t) = \frac{y(t-1)y(t-2)(y(t-1)-0.5)}{1+y^2(t-1)+y^2(t-2)} + u(t-1) \quad (3.33)$$

The equilibrium state of the unforced system given by equation (3.33) is $(0, 0)$. As in [125], the input $u(t)$ is uniformly selected in the range $[-1.5, 1.5]$ and the test input $u(t)$ is given by $u(t) = \sin(2\pi t/25)$. 5000 and 200 observation data are produced for the purpose of training and testing. The different parameter values for SAFIS are chosen as: $\gamma = 0.997$, $\epsilon_{\max} = 1.0$, $\epsilon_{\min} = 0.1$, $\kappa = 1.0$, $e_g = 0.05$, $e_p = 0.005$.

Table 3.4: Results of nonlinear identification problem 1

Methods	No. of Rules	Training RMSE	Testing RMSE
SAFIS	17	0.0539	0.0221
MRAN	22	0.0371	0.0271
RANEKF	35	0.0273	0.0297
Simpl_eTS ($r = 2.0$, $\Omega = 10^6$)	22	0.0528	0.0225
eTS ($r = 1.8$, $\Omega = 10^6$)	49	0.0292	0.0212
HA [125]	28	0.0182	0.0244

The average performance comparison of SAFIS with MRAN, RANEKF, eTS, Simpl_eTS and HA is shown in Table 3.4 based on 50 experimental trials. From the table it can be seen that SAFIS obtains similar testing accuracy compared to MRAN, RANEKF, eTS, Simpl_eTS and HA. However, SAFIS achieves this accuracy with smallest number of rules. It is worth noting that HA is based on GA iterative learning and is not sequential. The evolution of the fuzzy rules for SAFIS, MRAN, RANEKF, eTS and Simpl_eTS for a typical run is shown by Figure 3.2(a). It can be seen from the figure that SAFIS produces the least number of rules. Besides, Figure 3.2(b) gives a clear illustration for the rule evolution tendency between 0 and 1000 observation and shows that SAFIS can automatically add and delete a rule during learning.

(a)

(b)

Figure 3.2: (a) Rule update process between different algorithms for nonlinear identification problem 1 during the whole observation (b) Rule update process between different algorithms for nonlinear identification problem 1 between 0 and 1000 observation

3.3.1.2 Identification Problem 2

The second nonlinear dynamic system to be identified represents model II and is described by Juang and Lin [55]

$$y(t) = \frac{y(t-1)}{1 + y^2(t-1)} + u^3(t-1) \quad (3.34)$$

In accordance with [55], the input signal $u(t)$ is given by $\sin(2\pi t/100)$. 50000 and 200 observation data are produced for the purpose of training and testing. The SAFIS parameter values chosen are : $\gamma = 0.997, \epsilon_{\max} = 2.0, \epsilon_{\min} = 0.2, \kappa = 2.0, e_g = 0.03, e_p = 0.003$. The input variables $y(t), u(t)$ respectively follow the uniform sample distribution in the range $[-1.5, 1.5]$ and $[-1.0, 1.0]$.

Table 3.5: Results of nonlinear identification problem 2

Methods	No. of Rules	Testing RMSE
SAFIS	8	0.0116
MRAN	10	0.0129
RANEKF	11	0.0184
Simpl_eTS ($r = 0.075, \Omega = 10^6$)	18	0.0122
eTS ($r = 1.0, \Omega = 10^6$)	19	0.0082
SONFIN [55]	10	0.0130

Table 3.5 shows the performance comparison of SAFIS with MRAN, RANEKF, eTS, Simpl_eTS and SONFIN [55]. It can be seen from the table that SAFIS achieves similar accuracy with a lesser number of rules. Figure 3.3 and Figure 3.4 show the final membership functions of input variables $y(t), u(t)$ achieved by SAFIS. From the two figures, one can clearly see that the input variable membership functions are distributed in their own entire range. Besides the testing accuracy of SAFIS is slightly better than those of MRAN, RANEKF, Simpl_eTS and SONFIN, which verifies that the learning performance of SAFIS is not lost by only modifying the nearest fuzzy rule instead of all fuzzy rules during the learning. The evolution of the fuzzy rules for SAFIS, MRAN, RANEKF, eTS and Simpl_eTS is shown by Figure 3.5. It can be seen from the figure that SAFIS is able to add and delete rules during learning and produces the least number of rules.

Figure 3.3: Membership functions of input variable $y(t)$ for nonlinear identification problem 2

Figure 3.4: Membership functions of input variable $u(t)$ for nonlinear identification problem 2

Figure 3.5: Rule update process between different algorithms for nonlinear identification problem 2

3.3.2 Mackey-Glass Time Series Prediction

The chaotic Mackey-Glass time series is generated from the following differential equation [6]:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \quad (3.35)$$

where $\tau = 17$ and $x(0) = 1.2$. For the purpose of training and testing, 6000 samples are produced by means of the fourth-order Runge-Kutta method with the step size 0.1. The prediction task is to predict the value $x(t+85)$ from the input vector $[x(t-18) \ x(t-12) \ x(t-6) \ x(t)]$ for any value of the time t . As in [6], the observations between $t = 201$ and $t = 3200$ and the observations between $t = 5001$ and $t = 5500$ are extracted from the series and used as training and testing data. For this problem, the parameters for SAFIS are selected as follows: $\gamma = 0.98$, $\epsilon_{\max} = 1.6$, $\epsilon_{\min} = 0.16$, $\kappa = 1.68$, $e_g = 0.0005$, $e_p = 0.00005$. The data follow a uniform sample distribution in the range $[0.4, 1.4]$.

Table 3.6 shows the prediction accuracies and the number of rules obtained by SAFIS, eTS and Simpl_eTS. For comparison purposes, the prediction accuracy is based on the Non-Dimensional Error Index (NDEI) defined as the RMSE divided by the standard deviation of the true output values. As observed from Table 3.6, all the algorithms produce similar accuracies, however SAFIS obtains the smallest number of fuzzy rules. The evolution of the fuzzy rules for SAFIS, MRAN, RANEKF, eTS and Simpl_eTS is shown in Figure 3.6.

Table 3.6: Results of Mackey-Glass time series prediction

Methods	No. of Rules	Testing NDEI
SAFIS	6	0.376
MRAN	14	0.375
RANEKF	18	0.378
Simpl_eTS($r = 0.25$, $\Omega = 750$) [5]	11	0.394
eTS($r = 0.25$, $\Omega = 750$) [6]	9	0.380

Figure 3.6: Rule update process between different algorithms for Mackey-Glass time series prediction

In this chapter, a sequential fuzzy inference system called SAFIS is presented to

automatically construct a FIS using the training data during the learning process. Specifically SAFIS algorithm implements the structure identification and parameter adjustment for a FIS using the ideas from GAP-RBF algorithm. SAFIS algorithm utilizes the influence of a fuzzy rule to add and remove the fuzzy rules during learning. At the same time the SAFIS algorithm utilizes the EKF to update the parameters of the nearest rule instead of all the rules without losing the approximation performance. Its performance has been evaluated by some benchmark problems including two nonlinear system identification problems and the Mackey-Glass time series prediction problem. The simulation results from these benchmark problems show that compared with other algorithms, SAFIS produces similar or better testing accuracies with lesser number of rules.

In this chapter, the proposed SAFIS is mainly used to construct a suitable identification model subjected to the same input as the real plant by producing an estimated $\hat{\mathbf{y}}$ to approximate the target output $\mathbf{y}$. As analysed in [92], the above systems are Bounded Input and Bounded Output (BIBO) stable since the input is bounded and thus resulting in a bounded output. Furthermore we can use the proposed SAFIS algorithm as a controller to control nonlinear dynamic systems. Many neural/fuzzy-neural based on control schemes have been developed and range from learning the inverse dynamics of the system without any guarantee of stability to the direct adaptive control that guarantees stability of the overall systems. Without describing all the control strategies in detail, we will here concentrate on the most commonly used control strategies, that is, indirect adaptive control and direct adaptive control schemes. The adaptive inverse control [54, 128], stable direct adaptive control [24, 71, 114], and fixed stabilizing controller [31, 74, 120] (i.e., feedback-error-learning controller) all belong to these two categories.

In indirect adaptive control [54, 92, 128], generally two neural networks or fuzzy-neural systems are used: one is for identifying the forward/inverse dynamics of the systems, and the other is updated online using the information from the identified model to implement a suitable control law by connecting in cascade with the

system to be controlled. In direct adaptive control [24, 31, 71, 74, 114, 120] the neural networks or fuzzy-neural systems have no explicit attempt to determine a model of the process dynamics, and their parameters are directly adjusted to reduce some norm of the output error. The indirect control strategy depends strongly on the identification procedure, and generally there is no strict mathematical proof to guarantee the stability of the tracking error. The stable direct control strategy [24, 71, 114] can guarantee the stability of the tracking error using Lyapunov stability theory in the design of learning rule for neural networks and fuzzy-neural systems. However the stable direct control strategy generally only focuses on the tracking performance while ignoring the ability of the neural network or fuzzy-neural system to capture the underlying nonlinear function. Kawato [31] proposed an online control strategy to investigate the possibility of using the neural network for online function approximation and control of a nonlinear dynamic system. In the control scheme with the name of “feedback-error-learning”, a RBF network is used to learn the inverse dynamics of the system, which is shown in Figure 3.7. In the feedback-error-learning strategy, the total control effort, u_t , is composed of the output of the neural controller, u_{nn} , and the output of the conventional feedback controller, u_c . If the total control output u_t does not reflect the desired control effort, the parameters of the neural controller are adjusted according to certain tuning rules by using the output of the conventional controller. The feedback-error-learning has the advantage of learning the true inverse dynamics without requiring training the network initially off-line and allowing the controller to track the system dynamics which may change along with time by utilizing the online data. This proposed control scheme has been successfully utilized to control the movement of an industrial robotic manipulator. Later the scheme is further utilized by Li *et al.* [74, 120] for the aircraft flight control application by developing different conventional baseline controllers and neural network structure. The conventional controller in the inner-loop plays an important role in this strategy. It is not only used to stabilize the overall system, but also provides error signals to train the network on-line.

Figure 3.7: Feedback-error-learning control strategy

In the next chapter we adapt the feedback-error-learning control scheme by using the proposed SAFIS algorithm as the fuzzy controller instead of the neural controller together with a conventional controller named Baseline Trajectory Following Controller (BTFC) for aircraft autoland fault-tolerant control problem and show how the SAFIS algorithm is used as a controller to improve the fault-tolerance envelope for the problem.

Chapter 4

Aircraft Autolanding

Fault-Tolerant Control Problem

Traditional flight control systems mainly used hardware redundancy to handle component failures. In this case, the increase of the components will cause a substantial weight penalty and further require additional resources for deployment. Analytic redundancy instead of hardware redundancy has opened up the field of the aircraft fault-tolerant control to regain the aircraft control under failures such as actuator stuck conditions, control surface damages, sensor failures or a combination of them. Aircraft fault-tolerant control aims to achieve this: when the aircraft suffers from the failures, the controller still can lead the aircraft back to the equilibrium state, and even execute the scheduled flight in the presence of the failures.

The methods developed for the aircraft fault-tolerant control are usually classified as the scheme dependent on failure detection and identification (FDI) and the second scheme which is independent of FDI. In the approaches based on FDI [18, 84, 85, 99, 106], FDI units first detect and identify the possible faults among a set of pre-built faults and then a reconfigurable mechanism is utilized to report the fault information and reconfigure the controllers for handling the faults. This approach is effective when the fault conditions are understood well and their number is

small. However when failure conditions increase, it becomes difficult and time-consuming to detect and identify all the possible failures. In aircraft fault-tolerant control, aircrafts have multiple actuators and thus multiple actuator failures and an infinite variety of surface damages are possible. It is difficult to predict all the fault information and build the complete failure models. Consequently the approaches based on FDI cannot handle unpredictable failures which may occur in the real-time flight phases. In addition FDI units easily suffer from the missing-alarm and false-alarm rates due to modelling errors and disturbances because it is hard to build an exact agreement between the system and its model especially for aircraft dynamics with highly nonlinear properties.

The approaches that are independent of FDI [3, 13, 89, 97] reconfigure the flight controller by identifying the dynamic behavior of the aircraft in real-time, and designing a controller automatically. Hence these approaches can accommodate a larger class of failures including some unanticipated failures since they implement real-time control reconfiguration.

Many conventional methods that are independent of FDI have been developed for aircraft fault-tolerant control and they include model-following, adaptive control and so on. Morse *et al.* [89] proposed a Model Following Reconfigurable Flight Control (MFRFC) scheme together with PID to reconfigure the controllers for an experimental aircraft model AFTI/F-16. Three faults were considered in their study: 1) a right horizontal tail surface fault, 2) a double flaperon and rudder fault and 3) a double horizontal tail surface fault. The simulation results showed that the proposed MFRFC system could maintain system performance even in the presence of quadruple faults.

The adaptive control approaches carry out the reconfigurable flight control by using their online learning capabilities to redesign the control laws and then accommodate the faults. Ahmed-Zaid *et al.* [3] made use of the adaptive control theory to accommodate the failures for the F-16 fighter aircraft. In the absence of failures, the nominal controller was designed based on gain scheduling to achieve a good

degree of robustness. In the event of failures, the nominal control system was augmented with a Hybrid Adaptive Linear Quadratic Control (HALQC) scheme. The HALQC held the online capability of learning to accommodate the altered aircraft dynamics caused by the surface failure and was designed by using the reduced order linearized lateral dynamics of the F-16. Its performance was tested for the following three faults: 1) an inaccurate value of dynamic pressure $\bar{q}$, 2) the total loss of the horizontal stabilizer control input and 3) the total loss of the rudder control. The simulation results illustrated that the augmented control system was able to accommodate these failures and maintained good performance.

Pachter *et al.* [97] used the model predictive control with one-step-ahead actuator rate constraint enforcement to maximize aircraft tracking performance after control surface failure. When the aircraft sustained the control surface damage, the deflection and deflection rate of the control surface increased that resulted in actuator rate saturation. The proposed scheme was evaluated by the linearized second-order pitch plane dynamics of a derivative F-16 fighter and the simulation results illustrated that good tracking performance was obtained even after a severe failure.

Bodson and Groszkiewicz [13] employed multivariable adaptive control techniques to develop three adaptive algorithms based on indirect adaptive control, the direct output error and the direct input error. All the methods were designed using the reduced-order linear aircraft models and aimed to redesign the flight control laws when the aircraft was subjected to actuator failures or surface damages. Simulation study was carried out using a twin-engine jet aircraft and the results demonstrated that these adaptive algorithms were able to achieve specified system requirements under failures.

In the conventional methods, the fault-tolerant controller is designed using the linearized mathematical model of aircraft based on a set of pre-selected equilibrium states or trim points within the flight envelope. Actually aircraft dynamics is a highly uncertain and time-varying nonlinear system. The linearized model can-

not embody more complete information contained in the nonlinear model and is valid only for small perturbation from its equilibrium or trim point. Thus specific performance requirement used for designing the linear controller cannot directly be applied to the nonlinear system because performance can degrade severely if the aircraft moves away from the design trim point. To overcome some of the limitations and drawbacks of linear fault-tolerant controller design, the nonlinear fault-tolerant flight control techniques have to be developed. However it is difficult to build the exact nonlinear mathematical models for designing the nonlinear controller because of their complexity and unpredictable faults. In this case, the model-free intelligent methods become attractive tools for developing nonlinear fault-tolerant flight controllers by using their learning ability to build the nonlinear mapping from the actual data.

Fuzzy-neural systems can accommodate the failures by utilizing the learning and adaptive capabilities of neural networks for reconfiguring the controllers and thus they are powerful tools to solve aircraft fault-tolerant control problem. Diao and Passino [25] proposed two intelligent fault-tolerant fuzzy-neural controllers based on stable indirect and direct adaptive methods. The fuzzy-neural controllers were based on the Takagi-Sugeno fuzzy models and were trained by means of the Levenberg-Marquardt method to learn the unknown dynamics caused by faults and then accommodate the faults. The proposed control schemes were evaluated through the component level simulation of the General Electric XTE46 turbine engine and by extensive analysis of system zero dynamics and asymptotic tracking abilities for the two control schemes. The simulation results showed that both fuzzy-neural controllers achieved asymptotic tracking performance.

Kwong *et al.* [68] introduced a direct adaptive fuzzy control scheme to reconfigure controllers for a F-16 aircraft. In the scheme the Fuzzy Model Reference Learning Controller (FMRLC) was utilized as the fuzzy inverse model and modified by the reference model signals so that the closed-loop system behaved like the reference model in case of failure. The simulation study was implemented under the aileron

failures and the results demonstrated that in the proposed control scheme all the responses achieved in presence of aileron failures eventually matched those of the unimpaired aircraft with the nominal control laws.

Aircraft has many flight phases but autoland is a very challenging task among all aircraft flight phases since aircraft flies at a considerable low altitude and low speed during landing and any failure here will result in more accidents. Some uncertain factors like wind turbulence and wind shear become critical for aircraft landing because of low altitude and low speed [77]. Besides, when undergoing actuator failures, aircraft still is required to track a desired trajectory during landing phase and land in a specified touchdown region satisfying certain requirements, such as acceptable ranges of speed, pitch angle and altitude rate and so on. Many conventional methods including robust and optimal control methods [4, 11, 29, 37, 66, 77, 79, 115, 117, 123] have been utilized for the design of automatic landing control systems that are tolerant to the environment changes or actuator failures.

Ghahia and Alouani [29] presented an automatic landing controller design based on Linear Quadratic Regulator/Loop Transfer Recovery (LQG/LTR) for a typical commercial airplane. The proposed controller was evaluated using the longitudinal motion of the linearized aircraft system and the simulation results showed that good system tracking of the nominal trajectory was achieved either in the absence or in the presence of wind shear. Similarly Alonge [4] employed the LQG/LTR control techniques to design a landing control system for an Unmanned Aerial Vehicle (UAV). The simulation study was carried out for landing with gust, rear or front wind and the results confirmed the robustness characteristics of the designed control system. Based on LQR controller a precision longitudinal control of an UAV for glidepath tracking was proposed by Kim *et al.* [66]. The performance of the strategy was validated with regard to atmospheric disturbance and simulation results showed that the proposed strategy maintained the touch down points (TDP) under wind turbulence.

Subrahmanyam [117] made use of the finite horizon H_∞ techniques for the F/A-

18A Automatic Carrier Landing System by synthesizing the output feedback controller. The longitudinal equations of motion were considered for verifying the proposed method and the results indicated that a constant flight path angle during carrier landing was maintained under the worst vertical gust conditions and the obtained response of vertical rate command was satisfactory. A mixed H_2/H_∞ control technique was developed by Shue and Agarwal [115] for autolanding system of a commercial airplane (Boeing 747-200). The H_2 method was used to select an optimal trajectory for landing and the H_∞ technique was employed to minimize the effect of the disturbance. To illustrate the potential of the proposed method it was validated based on the linear model of the aircraft in longitudinal motion. The simulation results showed that the glide slope capture motion and flare maneuver of the aircraft were accomplished quite well and the amplitudes of all maneuvers were within Federal Aviation Administration requirements.

Although all the methods above guarantee good performance for aircraft autolanding under wind disturbances, the actuator faults have not been considered in these studies. Liao *et al.* [77] proposed the reliable automatic landing control system based on the solvability of linear matrix inequalities and polytopic fault models against actuator stuck faults and wind disturbances during aircraft landing. The approach was verified on a six-degree-of-freedom nonlinear aircraft model and simulation results showed that the proposed reliable autolanding controller was able to effectively handle tracking accuracy and robustness against wind disturbances and reliability against actuator stuck faults during the whole landing process.

From the above overview, it can be seen that the conventional methods are useful for aircraft landing when aircraft suffers from wind disturbances and actuator failures. However the conventional control methods are only able to guarantee the safe landing under a small range of disturbance conditions and failures. Therefore it is desirable to develop an intelligent autolanding controller for expanding the operational envelope to include more safe responses under a wider range of conditions and failures. In this case fuzzy-neural systems based controller design has

been developed by many researchers for an aircraft landing problem [56–59, 81]. They use their learning capabilities to increase the flight controller’s adaptation to different environments.

Juang and Chio [57, 58] proposed a fuzzy-neural controller combined with a linearized inverse aircraft model to improve the performance of conventional automatic landing systems. A multilayered fuzzy neural network used as the controller provided the control signals during the aircraft landing phase and was trained based on the Backpropagation Through Time (BPTT) method. The error signals used to back-propagate through the controller was provided by the linearized inverse aircraft model. A commercial aeroplane was used for simulation study and the simulation results showed that the fuzzy controller achieved good landing performance in severe wind turbulence. In [56, 59] the authors investigated the fuzzy-neural controller trained by Particle Swarm Optimization (PSO) method for the aircraft automatic landing under different wind disturbances. The simulation results illustrated that the fuzzy-neural controller with PSO method achieved better performance and the convergence rate was better than the fuzzy-neural controller with the BPTT method.

Malaek *et al.* [81] developed a fuzzy-neural control scheme for landing phase of a jet transport aircraft in the presence of different wind patterns. In this scheme, the outer loop utilized the fuzzy-neural controller based on Adaptive Network-based Fuzzy Inference System and the inner loop utilized the PID conventional controller. At the same time other three controllers named PID, Neuro based on Generalized Regression Neural Networks (GRNN) and hybrid Neuro-PID (its inner loop is PID and outer loop is GRNN) controllers were developed. Two different wind patterns named Strong wind (weaker than the JFK Downburst) and Very Strong wind (stronger than the JFK Downburst) were considered for the evaluation of their performance which classified at two levels named Level I (desired) and Level II (acceptable). Simulation results illustrated that all controllers satisfied the performance of level II in the presence of Strong wind but only ANFIS-PID controller

satisfied the performance of level I. In case of Very Strong wind, the simulation results showed that only ANFIS-PID controller satisfied the Level I performance and only Neuro-PID controller satisfied the Level 2 performance. Consequently among the four controllers the fuzzy-neural controller achieved the best performance and extended the flight envelope of the aircraft.

Although these fuzzy-neural control schemes achieve desired performance under different wind patterns, it is worth noting that in all the above work on landing, failure of the actuators is not considered though severe winds have been taken into account. Actuator failure during landing is a severe condition resulting in instability. The aircraft autoland problem studied in this thesis considers not only severe wind disturbances but also the actuator failures during landing.

The aircraft autoland problem studied in this thesis consists of four phases of flight segments for a high performance fighter executing a wing-level flight, a coordinated turn, glide slope descent and finally the flare maneuver. The aircraft model has five primary control surfaces which consist of two elevators, two ailerons and one rudder. The trajectory segments corresponding to these four phases are flown with severe winds as specified. The touchdown conditions that are to be satisfied are stringent and are called, for convenience, as the “touchdown pillbox”. In addition during the landing, the aircraft suffers from actuator faults which correspond to elevator and aileron surfaces stuck at different deflections either alone or in combination. Before describing the proposed control scheme, a brief description of the aircraft mathematical model is given below.

4.1 Aircraft Mathematical Model

The aircraft model considered in this study is based on a high performance fighter which is similar to the F-16 aircraft. The detailed derivation for the nonlinear equations of the aircraft motion is given in [93, 116, 141]. Here they are briefly presented.

Generally the aircraft motion equations are derived by assuming the aircraft is a rigid body and then using the body-fixed axis system. Figure 4.1 shows the aircraft model based on the body-fixed axis used in the simulation study.

Figure 4.1: Definition of aircraft dynamics in the body-fixed axes

In the body-fixed axis system, the origin lies in the center of gravity of aircraft and x axis, y axis and z axis point forward through the nose, through the right starboard wing and downwards along the origin. Some variables are shown in the figure to describe the flight motions and their directions are positive. These variables include velocities (u, v and w) where u is the longitudinal velocity, v is the lateral velocity and w is the normal velocity; the angular rates (p, q and r) which are roll, pitch and yaw angular velocities and angles (α, β) where α is the

angle-of-attack and β is the side-slip angle. The figure further shows the control surfaces whose deflections are commanded to control flight motions and the true control variables. The control surfaces include two elevators, two ailerons and one rudder. The pitch motion is controlled by the two elevators which are on either side of the axis of symmetry and commanded with identical signals. The roll motion is controlled by two ailerons which are on either wing and commanded in differential mode. The rudder is used to control the yaw motion. Their saturation limits and sign of convention are summarized in Table 4.1 where δ_e is the deflection of elevator, δ_a is the deflection of aileron and δ_r is the deflection of rudder.

Table 4.1: Specification of control surfaces

Control surfaces	Control effectiveness	Sign of convention	
		Positive	Effect
Elevator (δ_e)	$\pm 25^\circ$	Trailing edge down	Negative pitching moment
Aileron (δ_a)	$\pm 20^\circ$	Right wing trailing edge down	Negative rolling moment
Rudder (δ_r)	$\pm 30^\circ$	Trailing edge left	Negative yawing moment

The dynamics of the actuators is modelled as first order transfer function with a time constant of 50msec and is given by the following equation,

$$\frac{y(s)}{u(s)} = \frac{20}{s + 20} \quad (4.1)$$

Failure of a particular control surface consists of the control surface getting stuck at a specific deflection. If the instantaneous deflection of the surface is not the same as the failed position, the surface achieves its stuck position with a first order lag given by equation 4.1. The aircraft is powered by an after-burning turbofan jet engine and the thrust generated by the engine is denoted by T . The mass and geometry parameter values of the aircraft are summarized in Table 4.2.

Based on the assumption of the rigid body and the body-fixed axis system, the aircraft dynamics is represented by 12 scalar first order differential equations, two

Table 4.2: Mass and Geometry Data in aircraft model

Symbol	Unit	Value
Weight	N	91,188
I_x	kgm ²	12,875
I_y	kgm ²	75,674
I_z	kgm ²	85,552
I_{xz}	kgm ²	1,331
I_{xy}	kgm ²	0
I_{yz}	kgm ²	0
H_e	kgm ²	216.9
b	m	9.144
S	m ²	27.87
$\bar{c}$	m	3.45
x_{cg}	m	$0.3\bar{c}$
x_{cgr}	m	$0.35\bar{c}$
δ_{sb}	deg	60
δ_{lef}	deg	25

each for the six degrees of freedom. The state variables comprise of the components of the velocity vector (u, v and w), the angular rate vector (p, q and r), the vector of Euler angles (ϕ, θ and ψ) where ϕ is the roll angle, θ is the pitch angle and ψ is the yaw angle and the position vector (x, y and h) along x, y and z axis.

Force equations

$$\begin{aligned}
 \dot{u} &= rv - qw - g \sin \theta + \frac{\bar{q}}{m} SC_{x,t} + \frac{T}{m} \\
 \dot{v} &= pw - ru + g \cos \theta \sin \phi + \frac{\bar{q}}{m} SC_{y,t} \\
 \dot{w} &= qu - pv + g \cos \theta \cos \phi + \frac{\bar{q}}{m} SC_{z,t}
 \end{aligned} \tag{4.2}$$

Moment equations

$$\begin{aligned}
 \dot{p} &= \frac{I_x - I_z}{I_x} qr + \frac{I_{xz}}{I_x} (\dot{r} + pq) + \frac{\bar{q} Sb}{I_x} C_{l,t} + \frac{H_e q}{I_x} \\
 \dot{q} &= \frac{I_z - I_x}{I_y} pr + \frac{I_{xz}}{I_y} (r^2 - p^2) + \frac{\bar{q} S \bar{c}}{I_y} C_{m,t} - \frac{H_e r}{I_y} \\
 \dot{r} &= \frac{I_x - I_y}{I_z} pq + \frac{I_{xz}}{I_z} (\dot{p} - qr) + \frac{\bar{q} Sb}{I_z} C_{n,t} + \frac{H_e q}{I_z}
 \end{aligned} \tag{4.3}$$

Kinematics equations

$$\begin{aligned}
\dot{\phi} &= p + \tan \theta (q \sin \phi + r \cos \phi) \\
\dot{\theta} &= q \cos \phi - r \sin \phi \\
\dot{\psi} &= \frac{q \sin \phi + r \cos \phi}{\cos \theta}
\end{aligned} \tag{4.4}$$

Navigation equations

$$\begin{aligned}
\dot{x} &= u \cos \psi \cos \theta + v (\cos \psi \sin \theta \sin \phi - \sin \psi \cos \phi) \\
&\quad + w (\cos \psi \sin \theta \cos \phi + \sin \psi \sin \phi) \\
\dot{y} &= u \sin \psi \cos \theta + v (\sin \psi \sin \theta \sin \phi + \sin \psi \cos \phi) \\
&\quad + w (\sin \psi \sin \theta \cos \phi - \cos \psi \sin \phi) \\
\dot{h} &= u \sin \theta - v \sin \phi \cos \theta - w \cos \phi \cos \theta
\end{aligned} \tag{4.5}$$

Auxiliary equations

$$\begin{aligned}
V_T &= \sqrt{u^2 + v^2 + w^2}; \quad \alpha = \tan^{-1}\left(\frac{w}{u}\right); \quad \beta = \sin^{-1}\left(\frac{v}{V_T}\right); \quad \chi = \tan^{-1}\left(\frac{\dot{y}}{\dot{x}}\right) \\
a_x &= \frac{\bar{q} S C_{x,t} + T}{mg}; \quad a_y = \frac{\bar{q} S C_{y,t}}{mg}; \quad a_z = \frac{-\bar{q} S C_{z,t}}{mg}
\end{aligned} \tag{4.6}$$

4.1.1 Aerodynamic Coefficients

The total aerodynamic coefficients $C_{x,t}, C_{y,t}, C_{z,t}, C_{l,t}, C_{m,t}, C_{n,t}$ used in the simulation are derived from low-speed static and dynamic wind tunnel tests as described in [93]. The complete aerodynamic data acquired from the wind tunnel tests covers a wide range of angle of attack $-30^\circ \leq \alpha \leq 90^\circ$ and the sideslip angle $-30^\circ \leq \beta \leq 30^\circ$. The various aerodynamic contributions to a particular force or moment are added to obtain the total coefficient equations.

For the X-axis force coefficient $C_{x,t}$,

$$C_{x,t} = C_x(\alpha, \beta, \delta_e) + \Delta C_{x,lef} \left(1 - \frac{\delta_{lef}}{25}\right) + \frac{q\bar{c}}{2V_T} \left[C_{x,q}(\alpha) + \Delta C_{x,q,lef}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right) \right]$$

$$(4.7)$$

where

$$\Delta C_{x,lef} = C_{x,lef}(\alpha, \beta) - C_x(\alpha, \beta, \delta_e = 0^\circ) \quad (4.8)$$

For the Y-axis force coefficient $C_{y,t}$,

$$\begin{aligned} C_{y,t} = & C_y(\alpha, \beta) + \Delta C_{y,lef} \left(1 - \frac{\delta_{lef}}{25}\right) + \left[\Delta C_{y,\delta_a} + \Delta C_{y,\delta_a,lef} \left(1 - \frac{\delta_{lef}}{25}\right) \right] \frac{\delta_a}{20} \\ & + \Delta C_{y,\delta_r} \left(\frac{\delta_r}{30}\right) + \frac{rb}{2V_T} \left[C_{y,r}(\alpha) + \Delta C_{y,r,lef}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right) \right] \\ & + \frac{pb}{2V_T} \left[c_{y,p}(\alpha) + \Delta C_{p,lef}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right) \right] \end{aligned} \quad (4.9)$$

where

$$\begin{aligned} \Delta C_{y,lef} &= C_{y,lef}(\alpha, \beta) - C_y(\alpha, \beta) \\ \Delta C_{y,\delta_a} &= C_{y,\delta_a}(\alpha, \beta) - C_y(\alpha, \beta) \\ \Delta C_{y,\delta_a,lef} &= C_{y,\delta_a,lef}(\alpha, \beta) - C_{y,lef}(\alpha, \beta) - \Delta C_{y,\delta_a} \\ \Delta C_{y,\delta_r} &= C_{y,\delta_r}(\alpha, \beta) - C_y(\alpha, \beta) \end{aligned} \quad (4.10)$$

For the Z-axis force coefficient $C_{z,t}$,

$$C_{z,t} = C_z(\alpha, \beta, \delta_e) + \Delta C_{z,lef} \left(1 - \frac{\delta_{lef}}{25}\right) + \frac{q\bar{c}}{2V_T} \left[C_{z,q}(\alpha) + \Delta C_{z,q,lef}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right) \right] \quad (4.11)$$

where

$$\Delta C_{z,lef} = C_{z,lef}(\alpha, \beta) - C_z(\alpha, \beta, \delta_e = 0^\circ) \quad (4.12)$$

For the X axis moment coefficient $C_{l,t}$,

$$\begin{aligned} C_{l,t} = & C_l(\alpha, \beta, \delta_e) + \Delta C_{l,lef}(1 - \frac{\delta_{lef}}{25}) + [\Delta C_{l,\delta_a} + \Delta C_{l,\delta_a,lef}(1 - \frac{\delta_{lef}}{25})] \frac{\delta_a}{20} \\ & + \Delta C_{l,\delta_r} \frac{\delta_r}{30} + \frac{rb}{2V_T} \left[C_{l,r}(\alpha) + \Delta C_{l,r,lef}(\alpha)(1 - \frac{\delta_{lef}}{25}) \right] \\ & + \frac{pb}{2V_T} \left[C_{l,p}(\alpha) + \Delta C_{l,p,lef}(\alpha)(1 - \frac{\delta_{lef}}{25}) \right] + \Delta C_{l,\beta}(\alpha)\beta \end{aligned} \quad (4.13)$$

where

$$\begin{aligned} \Delta C_{l,lef} &= C_{l,lef}(\alpha, \beta) - C_l(\alpha, \beta, \delta_e = 0^\circ) \\ \Delta C_{l,\delta_a} &= C_{l,\delta_a}(\alpha, \beta) - C_l(\alpha, \beta, \delta_e = 0^\circ) \\ \Delta C_{l,\delta_a,lef} &= C_{l,\delta_a,lef}(\alpha, \beta) - C_{l,lef}(\alpha, \beta) - \Delta C_{l,\delta_a} \\ \Delta C_{l,\delta_r} &= C_{l,\delta_r}(\alpha, \beta) - C_l(\alpha, \beta, \delta_e = 0^\circ) \end{aligned} \quad (4.14)$$

For the Y-axis moment coefficient $C_{m,t}$,

$$\begin{aligned} C_{m,t} = & C_m(\alpha, \beta, \delta_e) + C_{z,t} [x_{cgr} - x_{cg}] + \Delta C_{m,lef}(1 - \frac{\delta_{lef}}{25}) \\ & + \frac{q\bar{c}}{2V_T} [C_{m,q}(\alpha) + \Delta C_{m,q,lef}(\alpha)] (1 - \frac{\delta_{lef}}{25}) + \Delta C_m(\alpha) + \Delta C_{m,ds}(\alpha, \delta_e) \end{aligned} \quad (4.15)$$

where

$$\Delta C_{m,lef} = C_{m,lef}(\alpha, \beta) - C_m(\alpha, \beta, \delta_e = 0^\circ) \quad (4.16)$$

For the Z axis moment coefficient $C_{n,t}$,

$$\begin{aligned} C_{n,t} = & C_n(\alpha, \beta, \delta_e) + \Delta C_{n,lef}(1 - \frac{\delta_{lef}}{25}) + [\Delta C_{n,\delta_a} + \Delta C_{l,\delta_a,lef}(1 - \frac{\delta_{lef}}{25})] \frac{\delta_a}{20} \\ & + \Delta C_{n,\delta_r} \frac{\delta_r}{30} + \frac{rb}{2V_T} \{ [C_{n,r}(\alpha) + \Delta C_{n,r,lef}(\alpha)(1 - \frac{\delta_{lef}}{25})] \\ & + \frac{pb}{2V_T} [C_{n,p}(\alpha) + \Delta C_{n,p,lef}(\alpha)(1 - \frac{\delta_{lef}}{25})] p \} + \Delta C_{n,\beta}(\alpha)\beta - C_{y,t} [x_{cgr} - x_{cg}] \frac{\bar{c}}{b} \end{aligned} \quad (4.17)$$

where

$$\begin{aligned}
 \Delta C_{n,lef} &= C_{n,lef}(\alpha, \beta) - C_n(\alpha, \beta, \delta_e = 0^\circ) \\
 \Delta C_{n,\delta_a} &= C_{n,\delta_a}(\alpha, \beta) - C_n(\alpha, \beta, \delta_e = 0^\circ) \\
 \Delta C_{n,\delta_a,lef} &= C_{n,\delta_a,lef}(\alpha, \beta) - C_{n,lef}(\alpha, \beta) - \Delta C_{n,\delta_a} \\
 \Delta C_{n,\delta_r} &= C_{n,\delta_r}(\alpha, \beta) - C_n(\alpha, \beta, \delta_e = 0^\circ)
 \end{aligned} \tag{4.18}$$

The simulation block of the high performance aircraft model constructed using Simulink blocks in the Matlab environment is given in Appendix B.

The elevators present in the aircraft model are split into the left and right elevators and are differentially used to handle failures. This leads to the estimation of the aerodynamic parameters for different angle of attack and side slip angle. The differential use of the elevators mainly affects the rolling moment. One should calculate the rolling moment from different values of the left and right deflections. Hence in this work the Computation Fluid Dynamic (CFD) approach has been used to calculate the aerodynamic coefficients for different deflections of the left and right elevators at varying angle of attack and side slip angle [98]. The table look-up procedure is used to calculate the aerodynamic parameters at intermediate deflections. Thus in these equations for the aerodynamic coefficients, the deflections of the elevators are replaced by the deflections of the left and right elevators. For example in equation 4.17 the first term $C_n(\alpha, \beta, \delta_e)$ is replaced with $C_n(\alpha, \beta, \delta_e^l, \delta_e^r)$ where δ_e^l and δ_e^r are the deflections of the left and right elevators. In a similar way, the CFD approach is utilized to split the ailerons into the left and right ailerons and the aerodynamic parameters need to be estimated by using the deflections of the left and right ailerons.

4.2 Aircraft Landing Task and Wind Model

The landing task executed by the aircraft described above is shown in Figure 4.2. It consists of the following six distinct phases:

- Segment 1: Level flight at 600m, heading -90° (from East to West). Velocity is maintained at 83m/s for this and all the other segments except the flare (segment 6).
- Segment 2: A coordinated right turn with bank angle 40° at 600m to align with the runway 0° (heading north).
- Segment 3: A level flight at 600m heading 0° (towards north).
- Segment 4: Descent on glide-slope of -6° to altitude of 300m.
- Segment 5: Descent on glide-slope of -3° to altitude of 12m.
- Segment 6: Flare and touchdown. Velocity is reduced from 83m/s to 79m/s.

During landing along this trajectory, the aircraft is subjected to wind disturbances. The major wind disturbances consist of wind shear and turbulence which are commonly encountered during landing. Wind shear is defined as a change in wind speed and direction within a short distance. It can be either in horizontal or vertical direction. Vertical wind shear is known as microburst or downburst and this is most critical during landing. Turbulence is modeled using a Dryden distribution and is assumed to exist in the horizontal direction. A typical wind model which incorporates both the turbulence and a severe microburst has been used for the whole landing trajectory.

Mathematically, the wind model along the x-body axis is represented by

$$\begin{aligned}\mu_g &= \mu_{g1} + \mu_{gc} \\ \dot{\mu}_{g1} &= 0.2|\mu_{gc}|\sqrt{2a_u}N_1 - a_\mu\mu_{g1}\end{aligned}\tag{4.19}$$

Figure 4.2: The landing trajectory

where μ_{g1} is turbulence component and μ_{gc} is the mean wind.

$$\mu_{gc} = \begin{cases} -\mu_0(1 + \ln(h/510)/\ln(51)) & h \geq 3 \\ 0 & h < 3 \end{cases} \quad (4.20)$$

where μ_0 is 6m/sec and N_1 is the Gaussian random noises with mean zero and variance 100, and

$$a_\mu = \begin{cases} U_0/(100\sqrt[3]{h}) & h > 70 \\ U_0/600 & h \leq 70 \end{cases} \quad (4.21)$$

where U_0 is the trimmed aircraft speed, which is 72m/sec. To produce a wind shear in z-body axis (microburst) at a given height, the direction of the wind is suddenly changed (from up to down) as

$$\omega_g = -\omega_0(1 + \ln(h/510)/\ln(51)) \quad (4.22)$$

where ω_0 is 12m/sec above the height of h_{shear} and -12 m/sec under h_{shear} , h_{shear} is 91m. The wind shear in y-body axis is a step change of magnitude 10m/s and -20m/s at altitudes 470m and 190m respectively. The equation is given by:

$$v_g = \begin{cases} -10 & h < 190 \\ 10 & 190 \leq h < 470 \\ 0 & h \geq 470 \end{cases} \quad (4.23)$$

Figure 4.3 illustrates the wind model described by the above equations along x, y and z-body axis.

Figure 4.3: Wind model during landing

Not only the winds but also the actuator failures are considered in the aircraft autoland fault-tolerant control problem. Next the actuator failure types are introduced.

4.3 Failure Types

The aircraft model consists of five primary control surfaces including two elevators, two ailerons and one rudder. The two elevators are deflected to produce the pitch motion. Under normal condition they are commanded with identical signals and on either side of the axis of symmetry. If one of them fails and is stuck in a hard position, the other healthy surface is utilized to control the aircraft. The two ailerons are deflected to generate the roll motion and they are on either wing and commanded in differential mode. If one of them fails, the other one is utilized to control the aircraft. The yaw motion is controlled by the rudder.

The failure scenarios considered in this thesis are given as follows: i) Single fault of either aileron or elevator stuck at certain deflections and ii) Double fault cases where one aileron and one elevator stuck at the same or opposite direction at different deflections. Furthermore, in the case of single fault, only single left elevator or single left aileron is considered because similar results from the right control surfaces can be obtained (because of the symmetry properties). Similarly for the case of double faults, results for only left elevator and left aileron failures along with left elevator and right aileron failures are given. The winds as described above are also considered in the failure scenarios.

The failures described above may occur at any point during the landing phase where the aircraft flies a total of six segments before touchdown. However the most crucial flight segment is the level turn followed by the descent phase where maximum demands are placed on control system if the failure occurs before the level turn. Thus in this thesis the failures are injected at 10sec for the elevator and 8sec for the aileron since these points are very close to the initiation of turn and

any potential instabilities would be exacerbated due to the turning maneuver.

The stuck positions of the failed control surface(s) can take any values within the permissible range of deflections but not all possible stuck positions are feasible. This is because the remaining healthy surfaces cannot trim out the residual moments caused by some surface failures. The feasible regions in which a single and two failures can be tolerated are determined based on the intersection of the following trim regions.

- Region of level flight trim ($p = q = r = \gamma = 0$, 6dof accelerations=0).
- Region of level descent trim ($p = q = r = 0$, $\gamma = -6\text{deg}$, 6dof accelerations=0).
- Region of level turning trim ($\phi = 40\text{deg}$, 6dof accelerations=0).

Figure 4.4 to Figure 4.7 give the trim computation for the above three trim conditions for all the failure scenarios. From these figures it can be found that since the level turn is the most demanding maneuver, its trim regions lie in the intersections of all the trim regions from the three conditions. Thus the feasible regions are determined by the level turn set.

Figure 4.4: Feasible range for left elevator stuck failure

Figure 4.5: Feasible range for left aileron stuck failure

Figure 4.6: Feasible range for left elevator and left aileron stuck failures

Figure 4.7: Feasible range for left elevator and right aileron stuck failures

For a successful autolanding, the aircraft need to follow the desired trajectories closely and satisfy the strict touchdown pillbox conditions defined in Table 4.3.

Table 4.3: Pillbox conditions at touchdown	
X-distance	$-100\text{m} \leq x \leq 300\text{m}$
Y-distance	$ y \leq 5\text{m}$
Total Velocity	$V_T \geq 60\text{m/s}$
Sink Rate	$\dot{h} \leq 1.0\text{m/s}$
Bank Angle	$ \phi \leq 10\text{deg}$

To achieve the objective, a baseline control scheme is utilized and for this a conventional controller named Baseline Trajectory Following Controller (BTFC) is designed using the classic loop design method. Figure 4.8 gives the control architecture for BTFC.

Figure 4.8: The baseline control strategy for autolanding

A block called tracking command generator is incorporated in the scheme since the landing task is autonomous. The block produces the references commands, $\mathbf{r}$, which are used as the input to the BTFC controller. The output of the BTFC controller, $\mathbf{u}_c$, is used as the input signals to drive the actuators. At the same time the wind model described above is considered for designing the BTFC controller. Next the details for designing the BTFC controller is given.

4.4 Baseline Trajectory Following Controller

BTFC is designed based on classical loop shaping SISO design techniques. In the trajectory following control design task a reference command generator (labelled ' $\mathbf{r}$ '

in Figure 4.8) is required to produce the reference signals whose outputs are acted upon by the classical feedback controller. The reference signals consist of altitude, velocity, cross distance from desired track and the angular error of the aircraft velocity vector from the desired track vector. Since the segments of trajectory are either straight lines or arcs of circles, the cross distance is the length of the perpendicular in case of line segments and is the difference between the distance to the center of the circular arc and the radius of the turn in case of the circular arcs. In the similar way, the angular error of the velocity vector is calculated based on the components of the aircraft velocity in the X-Y plane and the direction of the desired trajectory nearest to the aircraft. Figure 4.9 shows the geometry of the trajectory in both cases and their computations are given by equation 4.24 and equation 4.25.

Figure 4.9: Waypoints and trajectory geometry

- Straight Line Segment:

$$\begin{aligned}
q &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\
\delta &= \text{sgn}[(x - x_1)(y_2 - y_1) - (x_2 - x_1)(y - y_1)] \cdot \\
&\quad \text{abs} [((y_2 - y_1)x - (x_2 - x_1)y + (x_2 - x_1)y_1 - (y_2 - y_1)x_1)/q] \\
l &= \sqrt{(x - x_1)^2 + (y - y_1)^2} \\
s &= \sqrt{l^2 - \delta^2} \\
\psi_{ref} &= \tan^{-1} [(y_2 - y_1)/(x_2 - x_1)]
\end{aligned} \tag{4.24}$$

- Turn Segment

$$\begin{aligned}
q &= r\Delta\theta \\
l &= \sqrt{(x - x_r)^2 + (y - y_r)^2} \\
\psi_{ref} &= \tan^{-1} [(x - x_r)/(y - y_r)] \\
s &= r(\psi_t - \theta_0)
\end{aligned} \tag{4.25}$$

When these quantities are achieved, the altitude and velocity reference signals are calculated by using linear interpolation between the beginning and end point values of each segment. For the straight and turn segments they are computed as follows:

$$\begin{aligned}
w_r &= s/q \\
w_l &= 1 - w_r \\
h_{ref} &= h_1 w_l + h_2 w_r \\
V_{ref} &= V_1 w_l + V_2 w_r
\end{aligned} \tag{4.26}$$

During the flare phase, the desired altitude is computed by the following exponential equation:

$$h_{ref} = h_2 + (h_1 - h_2) \cdot e^{-2.5s/q} \tag{4.27}$$

The linearized model about the equilibrium point can represent the local dynamics of an aircraft well and is amenable to analysis and control system design. In the design of BTFC controller, a linear aircraft model generated based on the level flight segment 1 is utilized for the longitudinal and lateral-directional axes separately. The condition is sufficient for the entire flight envelope since in the whole trajectory the angle of attack, sideslip or March Number have no significant changes. The equilibrium states and the corresponding control inputs are given by Table 4.4 and Table 4.5 based on the trim calculation of the level flight segment 1.

Table 4.4: Equilibrium states

State Variables	Values
u (m/s)	81.31
w (m/s)	16.08
q (rad/s)	0
θ (rad)	11.19
h (m)	600
v (m/s)	0
p (rad/s)	0
r (rad/s)	0
ϕ (rad)	0
ψ (rad)	0
y (m)	0

Table 4.5: Equilibrium control inputs

Control Surfaces	Deflections
δ_e (deg)	-0.64
δ_a (deg)	0
δ_r (deg)	0
δ_t (deg)	0.22

Then the linearized aircraft model obtained by using the trim states and inputs is given by the following equation,

$$\begin{aligned}\Delta\dot{x} &= A\Delta x + B\Delta u \\ \Delta\dot{y} &= C\Delta x + D\Delta u\end{aligned}\tag{4.28}$$

where $\Delta x = [\Delta u, \Delta w, q, \Delta \theta, \Delta h, v, p, r, \phi, \psi, y]^T$; $\Delta u = [\Delta \delta_e, \delta_a, \delta_r, \Delta \delta_t]^T$; $\Delta y = [\gamma, q, \Delta \theta, \Delta h, \Delta V_T, p, r, \phi, a_y, \Delta \chi]^T$. The symbol Δ represents the deviation from the trim value for a particular variable which has a non-zero value at trim. Furthermore the linear model shown by equation 4.28 can be decoupled into the longitudinal and lateral dynamics for the BTFC controller design. In the lateral dynamics, the state variables, control input and output variables are as follows: $\Delta x = [v, p, r, \phi, \psi, y]^T$; $\Delta u = [\delta_a, \delta_r]^T$; $\Delta y = [p, r, \phi, a_y, \Delta \chi]^T$. The corresponding A, B, C and D matrices for the lateral dynamics are given in Appendix C.

In the longitudinal dynamics the state, control input and output variables are given as: $\Delta x = [\Delta u, \Delta w, q, \Delta \theta, \Delta h]^T$; $\Delta u = [\Delta \delta_e, \Delta \delta_t]^T$; $\Delta y = [\gamma, q, \Delta \theta, \Delta h, \Delta V_T]^T$. The corresponding A, B, C and D matrices for the longitudinal dynamics are given in Appendix C.

The BTFC controller processes the four reference signals generated by the tracking controller in the following way [98]:

- Altitude control is achieved by generating the desired attitude command. The flight path angle error is also used for feedback in order to provide damping to the altitude signal. The innermost loop is the pitch rate loop (Figure 4.10).
- Airspeed control is achieved by manipulating the throttle (Figure 4.10).
- Tracking of trajectories in the X-Y plane is achieved using heading angle as inner loop. This loop in turn commands the bank angle loop with the roll rate loop as the innermost (Figure 4.11).
- Sideslip is minimized by commanding the rudder using estimated sideslip rate given by $(r - p * \tan \alpha)$ and lateral acceleration as feedback signals (Figure 4.11).

Figure 4.10 and Figure 4.11 present the designed BTFC controller for longitudinal and lateral-directional axes. In the two figures all the loops are designed with a

minimum gain margin of 6dB and phase margin of at least 45 deg. The design details about the determination of the parameters for all the loops are given in [98].

Figure 4.10: Longitudinal Baseline Trajectory Following Controller (BTFC)

Figure 4.11: Lateral Baseline Trajectory Following Controller (BTFC)

When the aircraft flies with winds and no failures, BTFC controller can land the aircraft successfully and meet the touchdown specifications. The simulation results of landing trajectory achieved by BTFC controller under winds are shown in Figure 4.12 and Figure 4.13. Figure 4.12 shows the altitude h and velocity V_T and Figure 4.13 shows the sideslip angle (β) and lateral position (Y).

Figure 4.12: BTFC controller performance for the altitude (h) and velocity (V_T) under winds and no failures

Figure 4.13: BTFC controller performance for the sideslip angle (β) and lateral position (Y) under winds and no failures

Besides the reference signals are shown in the two figures so one can compare them with the simulation results. The numbers at the top of the two plots represent the landing phases where 1 represents the first level flight phase, 2 shows the right turn and so on. The simulation stops when the landing gear touches the ground. From the two figures it can be seen that in case of winds the BTFC controller can follow the desired trajectory and the touchdown points as listed in Table 4.3. Figure 4.12 shows that there is an initial velocity jump at $t = 0$ because the model is trimmed according to the straight level flight and also there is a gust at $t = 0$ as shown in Figure 4.3 resulting in an instantaneous increase in forward velocity. However the

perturbation quickly disappears. In segment 2 the velocity shows a dip because of the initial loss of the lift at the beginning of the turn. The deviations of velocity at the beginning of segment 4 and segment 5 occur due to the change in glide slope. The velocity deviations at about 120sec and 130sec are due to the wind shear in the y and z axis whose magnitude changes from -12m/s to 23m/s and from -12m/s to 11m/s respectively. From Figure 4.13 it can be observed that the lateral deviations of the ground track occurring in the turn phase in segment 2 and segment 3 are eventually being corrected by the subsequent flight phases. In addition Figure 4.13 illustrates that there is a sharp deviation of sideslip angle β in segment 4 and segment 5 because of the side gust in the y and z axis whose magnitude changes from 10m/s to -20m/s and from 10m/s to -10m/s separately.

Although BTFC controller can land the aircraft successfully and meet the touchdown pillbox under winds, the BTFC controller cannot handle certain actuator failures. In case of the above failure scenarios, the simulation studies of BTFC controller are conducted at one to two degree intervals of control surface deflections to obtain the region in which a single and two surface failures can be tolerated. After the simulation studies it is found that BTFC controller can not tolerate the elevator failure when the touchdown pillbox conditions are required to be met. Figure 4.14 and Figure 4.15 show the simulation results achieved by BTFC controller for the single elevator failure stuck at -10deg.

Figure 4.14: BTFC controller performance for the altitude (h) and velocity (V_T) under winds and left elevator failure stuck at -10°

Figure 4.15: BTFC controller performance for the sideslip angle (β) and lateral position (Y) under winds and left elevator failure stuck at -10deg

From these two figures it can be observed that BTFC controller has no tolerance to this failure because the landing task can not be completed successfully and the altitude drops at about 50sec. For other failure cases like single aileron failure and combined failures the BTFC has a very limited fault tolerance capability to tolerate them.

In this chapter the aircraft mathematical model, landing task and wind model used in the autoland problem have been described. The chapter also gives the

actuator failure types considered in this study where the feasible failure regions are determined based on the trim conditions from the level flight, level descent and level turn. To achieve the safe landing under winds and actuator failures, the baseline control scheme based on BTFC controller is utilized for the autoland problem. Thus the details about the design of BTFC controller are given in the chapter. BTFC controller performs well under winds but has a limited fault-tolerant capability to handle the actuator failures. To improve the performance of BTFC controller, an adaptive fuzzy controller will be used to augment the BTFC controller through its online learning.

In the next chapter the proposed SAFIS algorithm is incorporated in the control architecture described in Chapter 3 to improve the fault-tolerant capability of the BTFC controller for executing the landing task under actuator failures and winds.

Chapter 5

SAFIS Based Aircraft Fault-Tolerant Autolanding Control

In this chapter, SAFIS algorithm developed in Chapter 3 is utilized as a fuzzy controller for solving the high performance fighter aircraft landing problem under failures described in Chapter 4. Different from the previous work [56, 57, 81], the fuzzy controller based on SAFIS algorithm automatically determines the number of fuzzy rules in the learning process and furthermore handles not only severe winds but also actuator failures.

The SAFIS algorithm acts as an aid to the existing BTFC controller for solving the aircraft autolanding fault-tolerant control problem based on the feedback-error-learning scheme described in Chapter 3 when it is used as a fuzzy controller. The whole control scheme, viz., SAFIS aided BTFC controller architecture is shown in Figure 5.1.

Figure 5.1: SAFIS aided BTFC control strategy for autolanding

The reference signals produced by the tracking command generator and the aircraft outputs are utilized as the inputs for the SAFIS fuzzy controller. The sum of the outputs from the SAFIS fuzzy controller, u_f , and BTFC controller, u_c , is utilized as the total input signals, u_t , to drive the actuators. The BTFC controller not only is utilized to stabilize the overall system but also provide the error learning signals for training the SAFIS fuzzy controller online. In this control architecture the SAFIS fuzzy controller acts as an aid to the BTFC controller through an online learning scheme and thus when the actuator faults are present the SAFIS fuzzy controller utilizes its online learning ability to handle faults and reconfigure the control signals.

The details of how this scheme works is described below. Consider the aircraft dynamics given by the following equation

$$\dot{x} = f(x, u) \quad (5.1)$$

where f is assumed to be smooth and have bounded first derivatives in the neighborhood of the trajectory. The elevator, aileron and rudder control signals can be generated by inverting the nonlinear dynamic equations for an aircraft [15]. As a

result, the desired control input is obtained by the implicit function theorem,

$$u = f^{-1}(\dot{x}, x) \quad (5.2)$$

The function representing the inverse aircraft dynamics is varying over a period of time during landing. SAFIS utilizes its online learning ability to approximate the varying function and generate immediate corrective action on the basis of the states and their derivatives. The total landing motion is separated into the lateral and longitudinal direction motion using the assumption that the aerodynamic control inputs have no significant coupling. Thus in the simulations, two SAFIS fuzzy controllers given by Figure 5.2 and Figure 5.3 separately are designed for the longitudinal and lateral direction motion.

Figure 5.2: SAFIS implementation block for the longitudinal direction

Figure 5.3: SAFIS implementation block for the lateral direction

In the two figures, the input variables are the states and their derivatives separately existing in the longitudinal and lateral direction motion equations and the output variables are the control inputs for the longitudinal and lateral direction motion.

In the following sections, the performance of SAFIS aided BTFC control strategy is evaluated on the fault scenarios described in Chapter 3. Results are also presented for BTFC and a neural aided BTFC [98] as a comparison. The results show that SAFIS aided BTFC improves the fault tolerant capabilities compared to the other two schemes.

5.1 Selection of SAFIS Parameters

In SAFIS, some parameters need to be pre-selected before learning. They are, the distance thresholds ($\epsilon_{\max}$, $\epsilon_{\min}$, γ), the overlap factor for determining the width of the newly added rule (κ), the growing threshold for adding a new rule (e_g), the

pruning threshold for removing an insignificant rule (e_p) and the Kalman Filter parameters (p_0, q, r). When SAFIS is employed to solve the aircraft autoland fault-tolerant control problem, its parameters have to be tuned. However it is hard and impossible to change the parameters for SAFIS each time the failure conditions change. It is well known that Genetic Algorithm (GA) is an effective method to find the global optimal solution. Thus we have used a GA optimization scheme [38] to select the optimal parameters for SAFIS suitable for all the failure cases. The GA algorithm is briefly summarized below.

GA Optimization Algorithm:

Begin

Set the population size, number of generations and bounds for GA variables

Set generation counter to 1

Initialize the starting population $P(g)$ within the given bounds

Evaluate their fitness

While (Maximum generation not reached) do

Increment generation counter $g = g + 1$

Store member with best fitness among previous generation $M^* = \text{best}(P(g))$

Select members from the previous generation to form current generation $P_e(g)$

While (Number of crossovers not exhausted) do

Randomly pick two parents from current generation $P_e(g)$

Apply crossover operator and generate two children

Evaluate their fitness

Replace the parents in the current generation $P_e(g)$ with the child

End

While (Number of mutations not exhausted) do

Randomly pick a parent from current generation $P_e(g)$

Apply mutation operator and generate a child

Evaluate its fitness

Replace the parent in the current generation $P_e(g)$ with the child

End

Swap previous generation with current generation $P(g) = P_e(g)$

Find the worst in the generation $P(g)$ and replace it with the previous best M^*

End

In the GA optimization process, a real-valued representation chromosome is utilized to represent the SAFIS parameters. Their bounds are set in the range illustrated in Table 5.1. The corresponding crossover and mutation operators for real-valued representation are multi-type. The mutation approaches comprise of uniform mutation, non-uniform mutation with the shape parameter of 3, multi-non-uniform mutation with the shape parameter of 3, and boundary mutation. The crossover approaches consist of simple crossover, arithmetic crossover, and heuristic crossover with the try parameter of 3. The selection approach is based on the normalized geometric ranking selection with probability 0.08 of selecting the best individual. The population size is 20 and the generation number is 20.

Table 5.1: The parameter bounds of SAFIS used in the GA optimization

Parameters	$\epsilon_{\max}$	$\epsilon_{\min}$	γ	κ	e_g	e_p	p_0	q	r
Lower Bound	0.1	0.01	0.9	1.0	0.0001	0.00001	0.5	0.0001	0.5
Upper Bound	1.0	0.1	0.999	2.0	0.001	0.0001	1000	1.0	1000

The cost function is constructed based on eleven simulations with widely different two surface failures (i.e., different values of the failed position of control surfaces) taken from the feasible fault-tolerant envelope. The minimum lateral deviation is required at the beginning of the runway when the aircraft is executing the landing task. The lateral deviation from this point is computed for each of the eleven simulations and their norm is used in the cost function for each evaluation. The GA algorithm works by maximizing the cost function, whereas we need to minimize the distance from runway threshold. Thus the GA is given -ve of the norm of the distance for optimization. 20 function evaluations are implemented by the GA optimization process described above and thus the total number of complete trajectory simulations attempted from one generation to the next is $11 \times 20 = 220$. The GA is successful in reducing the cost function by about 30% from the trial and error values chosen initially. Figure 5.4 shows a typical plot of the reduction in cost function.

The finally achieved SAFIS parameter values by using GA optimization method are listed in Table 5.2 which is given as,

Figure 5.4: Best and average solutions in the GA population during a typical run

Table 5.2: The parameter values of SAFIS after GA optimization

Parameters	$\epsilon_{\max}$	$\epsilon_{\min}$	γ	κ	e_g	e_p	p_0	q	r
Values	0.5081	0.0161	0.9456	1.5103	0.0007	0.000012	551.18	0.6530	795.07

5.2 Single Surface Failure

In this section, a single failure of elevator or aileron is studied along with severe winds. First, we present the results for elevator failure. Here, left elevator is stuck at -10° at the beginning of the turn. Figure 5.5 and Figure 5.6 show the altitude (h), the velocity (V_T), sideslip angle (β) and lateral position (Y) during the landing phase for the SAFIS aided BTFC, EMRAN aided BTFC and BTFC.

Figure 5.5: Comparison of SAFIS aided BTFC, EMRAN aided BTFC and BTFC for the altitude (h) and velocity (V_T) under the left elevator stuck at -10°

Figure 5.6: Comparison of SAFIS aided BTFC, EMRAN aided BTFC and BTFC for the sideslip angle (β) and lateral position (Y) under the left elevator stuck at -10deg

The numbers at the top of the two figures represent the different segments of the trajectory. As discussed in Chapter 4, BTFC alone is unable to cope with this failure and achieve a safe landing as the altitude drops around 50sec. However from the two figures it can be seen that SAFIS aided BTFC and EMRAN aided BTFC controllers are not only able to land the aircraft but also satisfy the touchdown performance requirements. Furthermore, it can be seen from the two figures that SAFIS aided BTFC and EMRAN aided BTFC control schemes are able to follow the reference trajectory closely. As described in Chapter 4, a large sideslip seen in

Figure 5.6 around 110s is due to the abrupt step inputs in the side gust profile but these excursions are quickly damped out by the controllers. Table 5.3 shows the RMS trajectory errors along with the number of rules/neurons for SAFIS aided BTFC and EMRAN aided BTFC schemes. From the table, it can be seen that the trajectory error and the number of rules for SAFIS are smaller than those of EMRAN.

Table 5.3: Performance comparison between SAFIS aided BTFC and EMRAN aided BTFC control schemes under the left elevator stuck at -10deg

Method	SAFIS aided BTFC	EMRAN aided BTFC
Trajectory error (RMSE)	8.8573	8.9206
Velocity error (RMSE)	3.4243	3.8146
# of rules/neurons for δ_e	6	28
# of rules/neurons for δ_a	14	15

To analyze the control scheme performance in more detail, Figure 5.7 shows the deflection of the left elevator ($\delta_{e-left-t}$) and right elevator ($\delta_{e-right-t}$) along with the BTFC ($\delta_{e-right-c}$) and SAFIS/EMRAN components ($\delta_{e-right-f/n}$) of the control signals. The rule and neuron history based on SAFIS and EMRAN for the elevator δ_e is also shown.

Figure 5.7: Left and right elevator control signals under the left elevator stuck at -10deg

In the figure, the solid line represents the results for SAFIS aided BTFC and the dotted line represents the results from EMRAN aided BTFC. From the figure it is to be noted that the control signals for SAFIS aided BTFC and EMRAN aided BTFC are similar. However, the control signal for SAFIS aided BTFC is less oscillatory than that of EMRAN aided BTFC. Also, SAFIS only needs 6 rules for learning while EMRAN needs 28 neurons for learning.

The left and right aileron signals are given in Figure 5.8 and they are similar.

Figure 5.8: Left and right aileron control signals under the left elevator stuck at -10deg

The fault tolerant capabilities of SAFIS aided BTFC, EMRAN aided BTFC and BTFC for a single elevator surface failure are given in Figure 5.9.

Figure 5.9: Failure tolerance under left elevator stuck conditions

Each point in the figure indicates a successful landing meeting the touchdown pillbox requirements. From the figure, it can be seen that SAFIS aided BTFC is able to meet the pillbox requirements for a wider range of deflections (-12 to 18 degrees) compared to EMRAN aided BTFC (-12 to 12 degrees). No points corresponding to BTFC are shown in the figure because it can not meet the pillbox requirements for the entire range of failures from -18deg to +25deg. Furthermore, SAFIS aided BTFC is able to meet the pillbox requirements during continuous elevator deflections without “gaps” but EMRAN aided BTFC is unable to meet the pillbox requirements at -8, 0 and 8 degrees of elevator deflections. In [98], a detailed analysis has been given at the -8, 0 and 8 degree cases and we can observe that the three cases violate only one condition of the pillbox, namely

the y-deviation at touchdown. Further, the amount of deviation from the pillbox condition is not large, being less than 1.5m.

The pillbox in the x-y plane is shown in Figure 5.10 along with the touchdown points for SAFIS. It can be seen from Figure 5.10 that all these touchdown points lie inside the pillbox.

Figure 5.10: Touchdown points for SAFIS aided BTFC under left elevator stuck conditions

Similar results for the case of single aileron failure are shown in Figure 5.11.

Figure 5.11: Failure tolerance under left aileron stuck conditions

From this figure, one can note that the SAFIS aided BTFC has a wider aileron failure tolerance range (-14 to 14 degrees) compared to BTFC (-7 to 4 degrees) and a slightly wider failure tolerance range than EMRAN aided BTFC (-7 to 20 degrees) with the exception of failure at 2 degree (for EMRAN). As to the range from 15 degree to 20 degree where EMRAN can tolerant the failures, SAFIS still is able to land the aircraft successfully but is unable to satisfy the touchdown pillbox requirements by only violating the y-distance criterion.

Figure 5.12 gives the touchdown points for SAFIS together with the pillbox in the x-y plane and we can observe that these touchdown points are inside the pillbox.

Figure 5.12: Touchdown points for SAFIS aided BTFC under left aileron stuck conditions

5.3 Two Surface Failures

In this section, two failure cases are considered. Specifically, for the first case the left elevator and left aileron are stuck at different deflections. For the second case the left elevator and right aileron are considered.

5.3.1 Left Elevator and Left Aileron Stuck at Different Deflections

We first present a typical trajectory result when the left elevator is stuck at -10° and the left aileron is stuck at $+10^\circ$. Figure 5.13 and Figure 5.14 show the altitude (h), the velocity (V_T), sideslip angle (β) and lateral position (Y) during the landing phase for the SAFIS aided BTFC, EMRAN aided BTFC and BTFC along with the reference trajectory.

Figure 5.13: Comparison of SAFIS aided BTFC, EMRAN aided BTFC and BTFC for the altitude (h) and velocity (V_T) under the left elevator stuck at -10° and the left aileron stuck at 10°

Figure 5.14: Comparison of SAFIS aided BTFC, EMRAN aided BTFC and BTFC for the sideslip angle (β) and lateral position (Y) under the left elevator stuck at -10deg and the left aileron stuck at 10deg

From the two figures, it can be seen that BTFC alone is unable to cope with this failure and achieve a safe landing as the altitude drops around 30sec. However, SAFIS aided BTFC and EMRAN aided BTFC controllers are not only able to land the aircraft but also satisfy the touchdown performance requirements and also follow the reference trajectory closely.

Table 5.4 shows the RMS trajectory errors along with the number of rules/neurons for SAFIS aided BTFC and EMRAN aided BTFC schemes. It can be seen from the table that the trajectory error and the number of rules for SAFIS are smaller than those of EMRAN.

Table 5.4: Performance comparison between SAFIS aided BTFC and EMRAN aided BTFC control schemes under the left elevator stuck at -10deg and the left aileron stuck at 10deg

Method	SAFIS aided BTFC	EMRAN aided BTFC
Trajectory error (RMSE)	8.8978	8.9127
Velocity error (RMSE)	3.2761	3.8097
# of rules/neurons for δ_e	8	32
# of rules/neurons for δ_a	22	25

Figure 5.15 shows the deflection of the left elevator ($\delta_{e-left-t}$) and right elevator ($\delta_{e-right-t}$).

Figure 5.15: Left and right elevator control signals under the left elevator stuck at -10deg and the left aileron stuck at 10deg

In the figure, the solid line represents the results from SAFIS aided BTFC and the dotted line represents the results from EMRAN aided BTFC. The figure also shows the control signals for the BTFC component ($\delta_{e-right-c}$) along with the SAFIS/EMRAN components ($\delta_{e-right-f/n}$). The rule and neuron update process based on SAFIS and EMRAN for the elevator δ_e is also shown. It is seen that EMRAN scheme needs around 30 neurons for learning whereas SAFIS requires only around 10 rules indicating that SAFIS can do the job with a compact network.

Similar results for the left and right ailerons are given in Figure 5.16.

Figure 5.16: Left and right aileron control signals under the left elevator stuck at -10deg and the left aileron stuck at 10deg

The fault tolerant capabilities of SAFIS aided BTFC, EMRAN aided BTFC and BTFC for the whole range of deflections are given in Figure 5.17. The edge of the fault tolerance envelopes achieved by SAFIS aided BTFC, EMRAN aided BTFC and BTFC is given by the solid line, dashed line and dashdot line. In their respective fault tolerant regions each point which indicates a successful landing meeting the touchdown pillbox requirements is represented by o, + and $\times$.

Figure 5.17: Failure tolerance under left elevator and left aileron stuck conditions

From the figure, it can be seen that SAFIS aided BTFC is able to meet the pillbox requirements for a wider range of deflections compared to EMRAN aided BTFC as well as BTFC. It can be seen from the figure that the fault tolerance region achieved by SAFIS aided BTFC covers those of EMRAN aided BTFC and BTFC. But EMRAN aided BTFC controller does not cover the entire BTFC controller region due to its inability to meet the tight pillbox requirements.

It can also be observed from the figure that SAFIS aided BTFC has no “gaps” in the whole region whereas the regions covered by EMRAN aided BTFC and BTFC have “gaps” indicating their inability to meet the tight pillbox requirements at those stuck deflections.

5.3.2 Left Elevator and Right Aileron Stuck at Different Deflections

The fault tolerant envelope achieved by SAFIS aided BTFC, EMRAN aided BTFC and BTFC under both left elevator and right aileron failures is given in Figure 5.18. Their fault tolerance envelope edge is given by the solid line, dashed line and dashdot line respectively. In their respective fault tolerant regions each point which indicates a successful landing meeting the touchdown pillbox requirements is represented by o, + and $\times$.

Figure 5.18: Failure tolerance under left elevator and right aileron stuck conditions

From the figure, it can also be seen that SAFIS aided BTFC is able to meet the pillbox requirements for a wider range of deflections compared to EMRAN aided BTFC as well as BTFC. It can be seen from the figure that the fault tolerance

region achieved by SAFIS aided BTFC covers those of EMRAN aided BTFC and BTFC. But EMRAN aided BTFC controller does not cover the entire BTFC controller region due to its inability to meet the tight pillbox requirements. It can be further observed from the figure that SAFIS aided BTFC has no “gaps” in the whole region whereas the regions covered by EMRAN aided BTFC and BTFC have “gaps” indicating their inability to meet the tight pillbox requirements at those stuck deflections.

We have also looked at the pitch angle at touch down for all these cases as it is critical to ensure that the aircraft tail does not touch the ground when it lands. For the normal and all the failure cases, the pitch angle at touch down is in the range of 9-10 degrees. A typical value of 15 degree is safe for most aircraft (depending on the take-off AOA plus some margin). Since we are specifically looking for touch down speeds higher than 60m/s, the pitch angle criteria is satisfied.

To summarize in this chapter, a control strategy based on SAFIS aided BTFC has been presented for a high performance fighter aircraft during the landing phase when aircraft is subject to actuator stuck faults and severe winds. Based on the study, the following conclusions can be observed.

- SAFIS algorithm is utilized as the fuzzy controller to aid the existing conventional BTFC controller. SAFIS not only can learn online but also automatically determine the number of fuzzy rules during learning.
- For single surface failure cases, the BTFC controller is unable to meet the touchdown dispersions under elevator failures and is able to meet the requirements under aileron stuck failures only for small deflections (-8 and 4 degrees for aileron). The neural aided BTFC controller (EMRAN) enlarges the fault tolerant envelope (-12 to 12 degrees for elevator and -8 to 20 degrees for aileron) for the same case of single surface failure. However, in the fault tolerant envelope achieved by EMRAN aided BTFC there are some “gaps”

where the touchdown pillbox is not met under those stuck degrees (2 degree for aileron and -8, 0 and 8 degrees for elevator). The proposed fuzzy aided BTFC controller (SAFIS) achieves not only a continuous fault tolerant range but also a larger fault tolerant envelope (-12 to 18 degrees for elevator and -14 to 14 degrees for aileron) for single elevator and aileron failures compared with BTFC controller and EMRAN aided BTFC controller.

- For two surface failure cases like the combined elevator and aileron failures, the fault tolerant envelope achieved by SAFIS aided BTFC controller is enlarged and furthermore there are no “gaps” in the whole region compared with BTFC controller and EMRAN aided BTFC controller whose regions have “gaps” indicating their inability to meet the tight pillbox requirements at those stuck deflections (see Figure 5.17 and Figure 5.18).
- Compared with EMRAN aided BTFC, SAFIS aided BTFC has a smooth rule generation process and requires lesser number of fuzzy rules during learning (see Figure 5.7, Figure 5.8, Figure 5.15 and Figure 5.16).

One disadvantage of the SAFIS algorithm is that it requires many control parameters to be determined by trial and error before learning. In addition, SAFIS only can be applied for a specified fuzzy membership function, that is Gaussian membership function. In the next chapter, a new fast training algorithm which includes a smaller number of control parameters and which can be applied for any fuzzy membership function satisfying bounded nonconstant piecewise continuous functions is introduced to train the fuzzy-neural systems.

Chapter 6

Online Sequential Fuzzy Extreme Learning Machine (OS-Fuzzy-ELM)

In this chapter, a fast fuzzy-neural algorithm called Online Sequential Fuzzy Extreme Learning Machine (OS-Fuzzy-ELM) is developed based on ELM [44–46, 48]. Jang and Sun [52] proved the functional equivalence between a Gaussian radial basis function neural network and a fuzzy inference system with Gaussian membership functions under some mild conditions. This chapter further proves the functional equivalence between a generalized single-hidden layer feedforward network (SLFN) and the fuzzy inference system with any bounded nonconstant piecewise continuous membership function, thus making the fuzzy inference system with the Gaussian membership functions as a special case of SLFNs. Based on such functional equivalence, ELM can be extended to the fuzzy inference systems and the resulting fuzzy learning algorithm is referred to as Fuzzy Extreme Learning Machine (Fuzzy-ELM) that can handle any bounded nonconstant piecewise continuous membership function in a unified framework. However, Fuzzy-ELM is a batch learning algorithm and thus it needs the complete training data available before the training commences. This is not suitable for some practical applications

where new training data may arrive sequentially. Hence, a sequential learning version for Fuzzy-ELM called Online Sequential Fuzzy-ELM (OS-Fuzzy-ELM) is developed in this chapter.

OS-Fuzzy-ELM has been developed for both the TSK [121] and Mamdani type fuzzy models [82]. Different from other well-known learning algorithms such as ANFIS [51], DENFIS [63], SAFIS [111], eTS [6, 7] and SimpleTS [5] in OS-Fuzzy-ELM the parameters of the membership functions are randomly generated and based on this the consequent parameters are analytically determined. Moreover OS-Fuzzy-ELM can learn the training data not only one-by-one but also chunk-by-chunk (with fixed or varying length) modes and discard the data for which the training has already been done.

Since OS-Fuzzy-ELM is developed based on ELM, a brief review for the ELM [44–46, 48] is first given in the next section.

6.1 Extreme Learning Machine (ELM)

ELM is a recently developed algorithm by Huang, *et al.* [44, 46, 48] for Single-hidden Layer Feedforward Networks (SLFNs). In ELM, one may randomly choose the input weights and the hidden neurons' biases and based on this SLFNs become a linear system and then the output weights of SLFNs are determined through a generalized inverse operation of the hidden layer output matrices [46, 48]. Here input weights are the weights linking input neurons and hidden neurons and output weights are the weights linking hidden neurons and output neurons. Many activation functions existing in hidden layer nodes can be used in ELM and they include Sigmoid function and Radial Basis Functions (RBFs). When the active functions are the RBFs, the center and width parameters of RBFs are randomly selected and then the generalized inverse operation of the hidden layer output matrices is used to estimate the output weights.

Universal approximation capability of ELM has been analyzed in Huang, *et al.* [41]

in an incremental method and it has been shown that a SLFN with randomly generated Sigmoid or RBF nodes for a widespread of activation functions can universally approximate any continuous target function on any compact subspace of the Euclidean space $\mathbf{R}^n$. The details of ELM can be referred in [41,46]. Here it is briefly described for later development of the OS-Fuzzy-ELM.

The output of a SLFN with $\tilde{N}$ hidden nodes (Sigmoid or RBF nodes) as described in Chapter 2 can be represented by

$$f_{\tilde{N}}(\mathbf{x}) = \sum_{i=1}^{\tilde{N}} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}), \quad \mathbf{x} \in \mathbf{R}^n, \mathbf{a}_i \in \mathbf{R}^n \quad (6.1)$$

where $\mathbf{a}_i$ and b_i are the learning parameters of hidden nodes and β_i is the weight connecting the i th hidden node to the output node. $G(\mathbf{a}_i, b_i, \mathbf{x})$ is the output of the i th hidden node with respect to the input $\mathbf{x}$. For Sigmoid hidden node with the Sigmoidal activation function $g(x) : R \rightarrow R$, $G(\mathbf{a}_i, b_i, \mathbf{x})$ is given by

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i), \quad b_i \in R \quad (6.2)$$

where $\mathbf{a}_i$ is the weight vector connecting the input layer to the i th hidden node and b_i is the bias of the i th hidden node. $\mathbf{a}_i \cdot \mathbf{x}$ denotes the inner product of vectors $\mathbf{a}_i$ and $\mathbf{x}$ in $\mathbf{R}^n$.

For RBF hidden node with Gaussian activation function $g(x) : R \rightarrow R$, $G(\mathbf{a}_i, b_i, \mathbf{x})$ is given by

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(b_i \|\mathbf{x} - \mathbf{a}_i\|), \quad b_i \in R^+ \quad (6.3)$$

where $\mathbf{a}_i$ and b_i are the center and impact factor of i th RBF node. R^+ indicates the set of all positive real values.

For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbf{R}^n \times \mathbf{R}^m$, if a SLFN with $\tilde{N}$ hidden nodes can approximate these N samples, it then implies that there exist β_i , $\mathbf{a}_i$ and

b_i such that

$$f_{\tilde{N}}(\mathbf{x}_j) = \sum_{i=1}^{\tilde{N}} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j, \quad j = 1, \dots, N. \quad (6.4)$$

Equation (6.4) can be written compactly as:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \quad (6.5)$$

where

$$\mathbf{H}(\mathbf{a}_1, \dots, \mathbf{a}_{\tilde{N}}, b_1, \dots, b_{\tilde{N}}, \mathbf{x}_1, \dots, \mathbf{x}_N) = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}_N) \end{bmatrix}_{N \times \tilde{N}} \quad (6.6)$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \quad (6.7)$$

$\mathbf{H}$ is called the hidden layer output matrix of the network [39, 40]; the i th column of $\mathbf{H}$ is the i th hidden node's output vector with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ and the j th row of $\mathbf{H}$ is the output vector of the hidden layer with respect to input $\mathbf{x}_j$.

In ELM one can randomly generate hidden nodes with random parameters $(\mathbf{a}_i, b_i)$ for SLFNs and thus equation (6.5) becomes a linear system and the output weights $\boldsymbol{\beta}$ are estimated as:

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^\dagger \mathbf{T} \quad (6.8)$$

where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse [109] of the hidden layer output matrix $\mathbf{H}$ and equal to $(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T$. Calculation of the output weights is done in

a single step here. There is no need for any lengthy training procedure where the network parameters are adjusted interactively with appropriately chosen control parameters (learning rate and learning epochs, etc). The *three-step* ELM algorithm [45, 46] can be summarized as follows:

ELM Algorithm: Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \dots, N\}$, activation function g , and hidden node number $\tilde{N}$

step 1 Randomly assign hidden node parameters $(\mathbf{a}_i, b_i)$, $i = 1, \dots, \tilde{N}$.

step 2 Calculate the hidden layer output matrix $\mathbf{H}$.

step 3 Calculate the output weight β : $\beta = \mathbf{H}^\dagger \mathbf{T}$.

The universal approximation capability of ELM in Huang, *et al.* [41] is also valid for fuzzy inference systems. In the following sections we show how it can be extended to a fuzzy inference system.

6.2 Structure of Fuzzy Inference Systems

The fuzzy models considered here are TSK fuzzy model [121] and Mamdani fuzzy model [83]. The structure of fuzzy inference systems illustrated by Figure 6.1 consists of five layers that realize the above two fuzzy models.

Layer 1: In Layer 1, each node represents an input variable and directly transmits the input signal to Layer 2.

Layer 2: In this layer each node represents the membership value of each input variable. The membership value $\mu_{A_{ji}}(x_j)$ of the j th input variable x_j in the i th rule can be achieved by any bounded nonconstant piecewise continuous membership function g :

$$\mu_{A_{ji}}(x_j; c_{ji}, a_i) = g(x_j; c_{ji}, a_i) \quad (6.9)$$

Figure 6.1: Structure of Fuzzy Inference Systems

where c_{ji} and a_i are the parameters existing in the membership function g corresponding to the j th input variable x_j and the i th rule.

Layer 3: Each node in this layer represents the if part of if-then rules obtained by fuzzy logic AND operation, which can be any type of T-norm such as the product composition. The firing strength (if part) of the i th rule is given by

$$R_i(\mathbf{x}; \mathbf{c}_i, a_i) = \mu_{A_{1i}}(x_1; c_{1i}, a_i) \otimes \mu_{A_{2i}}(x_2; c_{2i}, a_i) \otimes \cdots \otimes \mu_{A_{ni}}(x_n; c_{ni}, a_i) \quad (6.10)$$

where symbol $\otimes$ represents any type of T-norm operation. If the triangular membership function with the algebraic product operation is employed it will be simply

as

$$R_i(\mathbf{x}; \mathbf{c}_i, a_i) = \prod_{j=1}^n \mu_{A_{ji}}(x_j; c_{ji}, a_i) = \prod_{j=1}^n \left(1 - \frac{|x_j - c_{ji}|}{a_i}\right) \quad (6.11)$$

Layer 4: The nodes in this layer are named as normalized nodes whose number is equal to the number of the nodes in the third layer. The i th normalized node is equal to the following equation:

$$G(\mathbf{x}; \mathbf{c}_i, a_i) = \frac{R_i(\mathbf{x}; \mathbf{c}_i, a_i)}{\sum_{i=1}^{\tilde{N}} R_i(\mathbf{x}; \mathbf{c}_i, a_i)} \quad (6.12)$$

Similar to Zeng and Singh [139] G can be called Fuzzy Basis Function (FBF). Different from Zeng and Singh [139] where only product operation is used, any T-norm fuzzy logic operation including the product operation can be used in the fuzzy basis function G defined here.

Layer 5: Each node in this layer corresponds to an output variable.

For Mamdani fuzzy model, by using center average defuzzifier [124] the system output $\hat{\mathbf{y}}$ for given input $\mathbf{x}$ is calculated by,

$$\hat{\mathbf{y}} = \frac{\sum_{i=1}^{\tilde{N}} \beta_i R_i(\mathbf{x}; \mathbf{c}_i, a_i)}{\sum_{i=1}^{\tilde{N}} R_i(\mathbf{x}; \mathbf{c}_i, a_i)} = \sum_{i=1}^{\tilde{N}} \beta_i G(\mathbf{x}; \mathbf{c}_i, a_i) \quad (6.13)$$

where $\beta_i = [\beta_{1i}, \dots, \beta_{mi}]$ and $\beta_{ki} (k = 1, 2, \dots, m)$ is the point at which B_{ki} achieves its maximum value, that is $\mu_B(\beta_{ki}) = 1$.

For TSK fuzzy model, its system output is achieved by the weighted sum of the output of each normalized rule. As such the system output $\hat{\mathbf{y}}$ for given input $\mathbf{x}$ is

calculated by

$$\hat{\mathbf{y}} = \frac{\sum_{i=1}^{\tilde{N}} \beta_i R_i(\mathbf{x}; \mathbf{c}_i, a_i)}{\sum_{i=1}^{\tilde{N}} R_i(\mathbf{x}; \mathbf{c}_i, a_i)} = \sum_{i=1}^{\tilde{N}} \beta_i G(\mathbf{x}; \mathbf{c}_i, a_i) \quad (6.14)$$

where the weight $\beta_i = [\beta_{1i}, \dots, \beta_{mi}]$ is the consequent crisp outputs of each rule and here it is a linear function on input variables. For β_{ki} , it equals to $\beta_{ki} = q_{ki0} + q_{ki1}x_1 + \dots + q_{kin}x_n$.

Remark 1: In equations (6.13) and (6.14), the antecedent (if) part of fuzzy rules (if-then rules) for the two fuzzy models, $R_i(\cdot)$, is same in the form and $R_i(\cdot) / \sum_{i=1}^{\tilde{N}} R_i(\cdot)$ represents the normalized firing strength of fuzzy rules while the consequent (then) part $\beta_i (= [\beta_{1i}, \dots, \beta_{mi}]^T)$ is same in the form but represents different meaning. For the Mamdani fuzzy model, the $\beta_{ki} (k = 1, \dots, m)$ contains the linguistic information since it is related with the linguistic variable B_{ki} while the β_{ki} in the TSK fuzzy model is only the crisp value and has no linguistic information. Note that when the Mamdani fuzzy model applies the center average defuzzifier, the obtained model output equation (6.13) is functionally equivalent to the output of the TSK model where the consequent parts are constant.

6.2.1 SLFN Equivalence of Fuzzy Inference Systems

The five layer fuzzy inference system in Figure 6.1 can be shown to be equivalent to a SLFN given in Figure 6.2. The equations (6.13) and (6.14) represent an equivalent SLFN given by Figure 6.2 where $G(\cdot)$ represents the activation function of the hidden node and β represents the output weight vector. The activation functions for the hidden nodes in the SLFN are based on the membership functions of the fuzzy inference system. As an example, when the membership function g is triangular and the fuzzy logic operation is the product operation, the output of the i -hidden node of the equivalent SLFN with input $\mathbf{x} = [x_1, \dots, x_n]^T$ for the

Figure 6.2: Equivalent SLFN Structure of Fuzzy Inference Systems

fuzzy inference system (Figure 6.1) is:

$$G(\mathbf{x}; \mathbf{c}_i, a_i) = \frac{R_i(\mathbf{x}; \mathbf{c}_i, a_i)}{\sum_{i=1}^{\tilde{N}} R_i(\mathbf{x}; \mathbf{c}_i, a_i)} = \frac{\prod_{j=1}^n \left(1 - \frac{|x_j - c_{ji}|}{a_i}\right)}{\sum_{i=1}^{\tilde{N}} \prod_{j=1}^n \left(1 - \frac{|x_j - c_{ji}|}{a_i}\right)} \quad (6.15)$$

For N arbitrary distinct training samples $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in \mathbf{R}^n$ and $\mathbf{t}_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T \in \mathbf{R}^m$, according to the equivalent SLFN structure of the fuzzy inference system illustrated by Figure 6.2, the mathematical model of the fuzzy inference system with $\tilde{N}$ fuzzy rules and the corresponding parameters

β_i , $\mathbf{c}_i$ and a_i as shown in Figure 6.1 is given as,

$$f_{\tilde{N}}(\mathbf{x}_j) = \sum_{i=1}^{\tilde{N}} \beta_i G(\mathbf{x}_j; \mathbf{c}_i, a_i) = \mathbf{t}_j, \quad j = 1, \dots, N. \quad (6.16)$$

Specifically for Mamdani fuzzy model the consequent parameters are given by the following matrix form according to equation (6.13),

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_{11} & \cdots & \beta_{m1} \\ \vdots & \cdots & \vdots \\ \beta_{1\tilde{N}} & \cdots & \beta_{m\tilde{N}} \end{bmatrix}_{\tilde{N} \times m} \quad (6.17)$$

As such the equation (6.16) for the Mamdani model can be written in the following compact form,

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \quad (6.18)$$

where $\mathbf{H}$ is the normalization node output matrix shown in Figure 6.1 and given as,

$$\begin{aligned} & \mathbf{H}(\mathbf{c}_1, \dots, \mathbf{c}_{\tilde{N}}, a_1, \dots, a_{\tilde{N}}; \mathbf{x}_1, \dots, \mathbf{x}_N) \\ &= \begin{bmatrix} G(\mathbf{x}_1; \mathbf{c}_1, a_1) & \cdots & G(\mathbf{x}_1; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_N; \mathbf{c}_1, a_1) & \cdots & G(\mathbf{x}_N; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}} \end{aligned} \quad (6.19)$$

For TSK fuzzy model, the consequent parameters are the linear equation of the input variables and given by

$$\beta_i = \mathbf{x}_e^T \mathbf{q}_i \quad (6.20)$$

where $\mathbf{x}_e^T$ is the extended input vector by appending the input vector $\mathbf{x}$ with 1, that is $[1, \mathbf{x}^T]^T$; $\mathbf{q}_i$ is the parameters existing in the TSK model for the i th fuzzy

rule and given by

$$\mathbf{q}_i = \begin{bmatrix} q_{1i0} & \cdots & q_{mi0} \\ \vdots & \cdots & \vdots \\ q_{1in} & \cdots & q_{min} \end{bmatrix}_{(n+1) \times m} \quad (6.21)$$

Thus the output equation (6.16) for the TSK model becomes as,

$$f_{\tilde{N}}(\mathbf{x}_j) = \sum_{i=1}^{\tilde{N}} \mathbf{x}_{je}^T \mathbf{q}_i G(\mathbf{x}_j; \mathbf{c}_i, a_i) = \mathbf{t}_j, \quad j = 1, \dots, N. \quad (6.22)$$

The equation further is written in the following compact form,

$$\mathbf{H}\mathbf{q} = \mathbf{T} \quad (6.23)$$

where $\mathbf{q}$ is the parameter matrix for TSK model and $\mathbf{H}$ is the input matrix that is weighted by normalized firing strength of fuzzy rules and they are given as,

$$\mathbf{q} = \begin{bmatrix} q_{110} & \cdots & q_{m10} \\ \vdots & \vdots & \vdots \\ q_{11n} & \cdots & q_{m1n} \\ q_{120} & \cdots & q_{m20} \\ \vdots & \cdots & \vdots \\ q_{12n} & \cdots & q_{m2n} \\ \vdots & \cdots & \vdots \\ q_{1\tilde{N}0} & \cdots & q_{m\tilde{N}0} \\ \vdots & \cdots & \vdots \\ q_{1\tilde{N}n} & \cdots & q_{m\tilde{N}n} \end{bmatrix}_{\tilde{N}(n+1) \times m} \quad (6.24)$$

sequence $\{(\mathbf{c}_i, a_i)\}_{i=1}^{\tilde{N}}$, $\lim_{\tilde{N} \rightarrow \infty} \|f(\mathbf{x}) - f_{\tilde{N}}(\mathbf{x})\| = 0$ holds with probability one.

Proof. Without loss of generality assume that the equivalent SLFN has only one linear output node. The extension of all the analysis conducted here to multilinear output node cases is straightforward. Let $\Delta = \|e_{\tilde{N}-1}\|^2 - \|e_{\tilde{N}}\|^2$, where $e_{\tilde{N}-1} = f(\mathbf{x}) - f_{\tilde{N}-1}(\mathbf{x})$, then we have

$$\begin{aligned} \Delta &= \|e_{\tilde{N}-1}\|^2 - \|f(\mathbf{x}) - (f_{\tilde{N}-1}(\mathbf{x}) + \beta_{\tilde{N}}G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}))\|^2 \\ &= \|e_{\tilde{N}-1}\|^2 - \|e_{\tilde{N}-1} - \beta_{\tilde{N}}G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x})\|^2 \\ &= 2\beta_{\tilde{N}}\langle e_{\tilde{N}-1}, G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}) \rangle - \beta_{\tilde{N}}^2\|G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x})\|^2 \\ &= \|G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x})\|^2 \left(\frac{\langle e_{\tilde{N}-1}, G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}) \rangle^2}{\|G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x})\|^4} - \left(\beta_{\tilde{N}} - \frac{\langle e_{\tilde{N}-1}, G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}) \rangle}{\|G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x})\|^2} \right)^2 \right) \end{aligned} \quad (6.26)$$

When $\beta_{\tilde{N}} = \langle e_{\tilde{N}-1}, G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}) \rangle / \|G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x})\|^2$, Δ achieves its maximum, that is $\Delta_{\max} = \langle e_{\tilde{N}-1}, G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}) \rangle^2 / \|G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x})\|^2 \geq 0$. Equivalently when $\beta_{\tilde{N}} = \langle e_{\tilde{N}-1}, G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}) \rangle / \|G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x})\|^2$, $\|e_{\tilde{N}}\| = \|f(\mathbf{x}) - (f_{\tilde{N}-1}(\mathbf{x}) + \beta_{\tilde{N}}G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}))\|$ is minimized. Further when $\beta_{\tilde{N}} = \langle e_{\tilde{N}-1}, G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}) \rangle / \|G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x})\|^2$, we have $e_{\tilde{N}} \perp G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x})$ derived as,

$$\begin{aligned} \langle e_{\tilde{N}}, G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}) \rangle &= \langle e_{\tilde{N}-1} - \beta_{\tilde{N}}G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}), G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}) \rangle \\ &= \langle e_{\tilde{N}-1}, G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}) \rangle - \beta_{\tilde{N}}\langle G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}), G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x}) \rangle \\ &= 0 \end{aligned} \quad (6.27)$$

As such the sequence $\{\|e_{\tilde{N}}\|\}$ converges and is bound below by zero. Actually when $\tilde{N} \rightarrow +\infty$, $\|e_{\tilde{N}}\| \rightarrow 0$, that is $\lim_{\tilde{N} \rightarrow +\infty} \|e_{\tilde{N}}\| = 0$. We can prove this by contradiction method. The sequence $\{\|e_{\tilde{N}}\|\}$ converges and thus there exists $r \geq 0$ such as $\lim_{\tilde{N} \rightarrow \infty} \|e_{\tilde{N}}\| = r$. If $r > 0$, the sequence $\{\|e_{\tilde{N}}\|\}$ is decreasing and bound below by r , that is $\|e_{\tilde{N}}\| \geq r$ for all $\tilde{N} > 0$. Thus there exists $\bar{N}_1$ for $\forall \delta > 0$ so that

$r \leq \|e_{\tilde{N}}\| < r + \delta$ when $\tilde{N} > \tilde{N}_1$. This implies that an infinite number of $e_{\tilde{N}} (\forall \tilde{N} > \tilde{N}_1)$ is covered by a compact set and based on this there exists a subsequence $\{e_{\tilde{N}_k}\}$ which converges to a limit denoted by $\|e^*\| = \lim_{k \rightarrow +\infty} \|e_{\tilde{N}_k}\| = r$. Since $e_{\tilde{N}_k} \in L^2(X)$ and $L^2(X)$ is complete, we have $e^* \in L^2(X)$. Furthermore there should exist $G(\mathbf{a}^*, b^*, \mathbf{x})$ such that $G(\mathbf{a}^*, b^*, \mathbf{x})$ is not orthogonal to e^* . Otherwise e^* is orthogonal to $\text{span}\{G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x})\}$ which is contradictory to the fact $\text{span}\{G(\mathbf{a}_{\tilde{N}}, b_{\tilde{N}}, \mathbf{x})\}$ is dense in $L^2(X)$. The limit e^* of a subsequence of $\{e_{\tilde{N}_k}\}$ should satisfy $\|e^*\| = r = 0$. If $\|e^*\| = r \neq 0$, there exists a $G(\mathbf{a}^*, b^*, \mathbf{x})$ which is not orthogonal to e^* , then there exists $e_{\tilde{N}_k+p+1}$, where $0 \leq p \leq L(\tilde{N}_k)$ and $L(\tilde{N}_k)$ is a function of k such that $\|e_{\tilde{N}_k+p+1}\| < \lim_{k \rightarrow +\infty} \|e_{\tilde{N}_k}\| = r$, which is contradictory to the fact that $\|e_{\tilde{N}_k+p+1}\| \geq r$. Thus r must be zero. $\square$

In fact, zero approximation error is not required in real applications by adding infinite nodes to the network. Instead, the training error obtained using the network is only expected to be smaller than a given nonzero target error $\epsilon > 0$. Based on this, we have the following.

Theorem 6.2.2. *Given any small positive value $\epsilon > 0$ and any bounded nonconstant piecewise continuous membership function g , if $\text{span}\{G(\mathbf{x}; \mathbf{c}, a) : (\mathbf{c}, a) \in \mathbf{R}^n \times R\}$ is dense in L^p for every $p \in [1, \infty)$, then there exists a finite $\tilde{N} (< \infty)$ such that for any continuous target function f and any randomly generated sequence $\{(\mathbf{c}_i, a_i)\}_{i=1}^{\tilde{N}}$, $\|f(\mathbf{x}) - f_{\tilde{N}}(\mathbf{x})\| < \epsilon$ holds with probability one.*

Proof. The validity of the theorem is straightforward based on theorem 6.2.1. $\square$

Finally according to the universal approximation theorem introduced by Ying [134] for the TSK fuzzy model and the universal approximation theorems introduced by Zeng and Singh [139] and Castro and Delgado [16] for the Mamdani fuzzy model (described in Chapter 2), we then have the following universal approximation theorem for the TSK and Mamdani fuzzy systems:

Theorem 6.2.3. *Given any small positive value $\epsilon > 0$ and any bounded nonconstant piecewise continuous membership function g and any types of T -norm fuzzy*

logic operation, if $\text{span}\{G(\mathbf{x}; \mathbf{c}, a) : (\mathbf{c}, a) \in \mathbf{R}^n \times R\}$ is dense in L^p for every $p \in [1, \infty)$, then there exists a finite $\tilde{N} (< \infty)$ such that for any continuous target function f and any randomly generated sequence $\{(\mathbf{c}_i, a_i)\}_{i=1}^{\tilde{N}}$, $\|f(\mathbf{x}) - f_{\tilde{N}}(\mathbf{x})\| < \epsilon$ holds with probability one.

Remark 2: Theorem 6.2.3 shows that the parameters of the membership functions g can be randomly generated unlike the conventional implementation of the fuzzy inference systems where they need to be tuned.

Based on the functional equivalence between SLFNs and the fuzzy inference systems shown above, the ELM for SLFNs can be linearly extended to the fuzzy inference systems and the resulted algorithm is called Fuzzy-ELM. The *three-step* Fuzzy-ELM algorithm similar to ELM can be summarized as follows:

Fuzzy-ELM Algorithm: Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \dots, N\}$, membership function g , and fuzzy rule number $\tilde{N}$,

step 1 Randomly assign membership function parameters $(\mathbf{c}_i, a_i)$, $i = 1, \dots, \tilde{N}$.

step 2 Calculate the matrix $\mathbf{H}$ (cf. equation (6.19) and equation (6.25)) for Mamdani and TSK fuzzy models.

step 3 Calculate the parameter matrix $\boldsymbol{\theta}$: $\boldsymbol{\theta} = \mathbf{H}^\dagger \mathbf{T}$.

It should be noted that Fuzzy-ELM requires the availability of complete training data and thus is a batch learning method. But in many real applications the training data arrive chunk-by-chunk or one-by-one (a special case of chunk) and hence the Fuzzy-ELM algorithm has to be modified for this case to make it sequential. The description of sequential version of the Fuzzy-ELM called as Online Sequential Fuzzy-ELM (OS-Fuzzy-ELM) is described below. Fuzzy-ELM will be a special case of OS-Fuzzy-ELM when all the training observations are present in one learning iteration.

6.3 Online Sequential Fuzzy Extreme Learning Machine

Recently an Online Sequential ELM (OS-ELM) algorithm has been developed by Liang, *et al.* [75] to learn the data one-by-one or chunk-by-chunk (a block of data) with fixed or varying chunk size. Using the procedures similar to the derivation of OS-ELM [75], OS-Fuzzy-ELM is presented below.

Rewrite the parameter matrix estimate of $\boldsymbol{\theta}$ as,

$$\boldsymbol{\theta} = \mathbf{H}^\dagger \mathbf{T} \quad (6.28)$$

where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse of the matrix $\mathbf{H}$ and given by,

$$\mathbf{H}^\dagger = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \quad (6.29)$$

The equation (6.28) is referred to as the least-squares solution to $\mathbf{H}\boldsymbol{\theta} = \mathbf{T}$. Sequential implementation of the least-squares solution of equation (6.28) results in the Online Sequential Fuzzy ELM (OS-Fuzzy-ELM).

When a chunk of initial training set $\aleph_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$, $N_0 \geq \tilde{N}$ is given, the batch Fuzzy-ELM algorithm is used to estimate the parameter matrix $\boldsymbol{\theta}$ by considering the problem of minimizing $\|\mathbf{H}_0 \boldsymbol{\theta} - \mathbf{T}_0\|$.

For Mamdani fuzzy model,

$$\begin{aligned} & \mathbf{H}_0(\mathbf{c}_1, \dots, \mathbf{c}_{\tilde{N}}, a_1, \dots, a_{\tilde{N}}; \mathbf{x}_1, \dots, \mathbf{x}_{N_0}) \\ &= \begin{bmatrix} G(\mathbf{x}_1; \mathbf{c}_1, a_1) & \cdots & G(\mathbf{x}_1; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_{N_0}; \mathbf{c}_1, a_1) & \cdots & G(\mathbf{x}_{N_0}; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) \end{bmatrix}_{N_0 \times \tilde{N}} \end{aligned} \quad (6.30)$$

For TSK fuzzy model,

$$\begin{aligned}
 & \mathbf{H}_1(\mathbf{c}_1, \dots, \mathbf{c}_{\tilde{N}}, a_1, \dots, a_{\tilde{N}}; \mathbf{x}_{N_0+1}, \dots, \mathbf{x}_{N_1}) \\
 &= \begin{bmatrix} G(\mathbf{x}_{N_0+1}; \mathbf{c}_1, a_1) & \cdots & G(\mathbf{x}_{N_1}; \mathbf{c}_1, a_1) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_{N_0+1}; \mathbf{c}_1, a_1)x_{(N_0+1)n} & \cdots & G(\mathbf{x}_{N_1}; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}})x_{N_1n} \\ G(\mathbf{x}_{N_0+1}; \mathbf{c}_2, a_2) & \cdots & G(\mathbf{x}_{N_1}; \mathbf{c}_2, a_2) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_{N_0+1}; \mathbf{c}_2, a_2)x_{(N_0+1)n} & \cdots & G(\mathbf{x}_{N_1}; \mathbf{c}_2, a_2)x_{N_1n} \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_{N_0+1}; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) & \cdots & G(\mathbf{x}_{N_1}; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_{N_0+1}; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}})x_{(N_0+1)n} & \cdots & G(\mathbf{x}_{N_1}; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}})x_{N_1n} \end{bmatrix}^T \quad (6.34) \\
 & \quad \quad \quad \tilde{N}(n+1) \times N_1
 \end{aligned}$$

Based on both chunks of training data sets $\aleph_0$ and $\aleph_1$, the output weight $\boldsymbol{\theta}$ becomes as,

$$\boldsymbol{\theta}^{(1)} = \mathbf{K}_1^{-1} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \quad (6.35)$$

where

$$\mathbf{K}_1 = \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \quad (6.36)$$

Furthermore $\mathbf{K}_1$ can be written as

$$\begin{aligned}
 \mathbf{K}_1 &= [\mathbf{H}_0^T \quad \mathbf{H}_1^T] \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \\
 &= \mathbf{K}_0 + \mathbf{H}_1^T \mathbf{H}_1
 \end{aligned} \quad (6.37)$$

and

$$\begin{aligned}
\begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} &= \mathbf{H}_0^T \mathbf{T}_0 + \mathbf{H}_1^T \mathbf{T}_1 \\
&= \mathbf{K}_0 \mathbf{K}_0^{-1} \mathbf{H}_0^T \mathbf{T}_0 + \mathbf{H}_1^T \mathbf{T}_1 \\
&= \mathbf{K}_0 \boldsymbol{\theta}^{(0)} + \mathbf{H}_1^T \mathbf{T}_1 \\
&= (\mathbf{K}_1 - \mathbf{H}_1^T \mathbf{H}_1) \boldsymbol{\theta}^{(0)} + \mathbf{H}_1^T \mathbf{T}_1 \\
&= \mathbf{K}_1 \boldsymbol{\theta}^{(0)} - \mathbf{H}_1^T \mathbf{H}_1 \boldsymbol{\theta}^{(0)} + \mathbf{H}_1^T \mathbf{T}_1
\end{aligned} \tag{6.38}$$

Combining the equation (6.38), $\boldsymbol{\theta}^{(1)}$ is given by

$$\begin{aligned}
\boldsymbol{\theta}^{(1)} &= \mathbf{K}_1^{-1} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} = \mathbf{K}_1^{-1} (\mathbf{H}_0^T \mathbf{T}_0 + \mathbf{H}_1^T \mathbf{T}_1) \\
&= \mathbf{K}_1^{-1} (\mathbf{K}_1 \boldsymbol{\theta}^{(0)} - \mathbf{H}_1^T \mathbf{H}_1 \boldsymbol{\theta}^{(0)} + \mathbf{H}_1^T \mathbf{T}_1) \\
&= \boldsymbol{\theta}^{(0)} + \mathbf{K}_1^{-1} \mathbf{H}_1^T (\mathbf{T}_1 - \mathbf{H}_1 \boldsymbol{\theta}^{(0)})
\end{aligned} \tag{6.39}$$

where

$$\mathbf{K}_1 = \mathbf{K}_0 + \mathbf{H}_1^T \mathbf{H}_1 \tag{6.40}$$

As new data arrives we can further generalize the equation (6.39) to obtain a recursive algorithm for updating the least-squares solution of parameter matrix $\boldsymbol{\theta}$, which is similar to the recursive least squares algorithm [23]. Given $(k+1)$ -th chunk of data set $\mathfrak{N}_{k+1} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{\sum_{j=0}^{k+1} N_j / (\sum_{j=0}^k N_j + 1)}$, where $k \geq 0$ and N_{k+1} denotes the number of observations in the $(k+1)$ -th chunk, we have

$$\begin{aligned}
\mathbf{K}_{k+1} &= \mathbf{K}_k + \mathbf{H}_{k+1}^T \mathbf{H}_{k+1} \\
\boldsymbol{\theta}^{(k+1)} &= \boldsymbol{\theta}^{(k)} + \mathbf{K}_{k+1}^{-1} \mathbf{H}_{k+1}^T (\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\theta}^{(k)})
\end{aligned} \tag{6.41}$$

For Mamdani fuzzy model,

$$\begin{aligned} & \mathbf{H}_{k+1}(\mathbf{c}_1, \dots, \mathbf{c}_{\tilde{N}}, a_1, \dots, a_{\tilde{N}}; \mathbf{x}_{\Delta_1}, \dots, \mathbf{x}_{\Delta_2}) \\ &= \begin{bmatrix} G(\mathbf{x}_{\Delta_1}; \mathbf{c}_1, a_1) & \cdots & G(\mathbf{x}_{\Delta_1}; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_{\Delta_2}; \mathbf{c}_1, a_1) & \cdots & G(\mathbf{x}_{\Delta_2}; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) \end{bmatrix}_{N_{k+1} \times \tilde{N}} \end{aligned} \quad (6.42)$$

For TSK fuzzy model,

$$\begin{aligned} & \mathbf{H}_{k+1}(\mathbf{c}_1, \dots, \mathbf{c}_{\tilde{N}}, a_1, \dots, a_{\tilde{N}}; \mathbf{x}_{\Delta_1}, \dots, \mathbf{x}_{\Delta_2}) \\ &= \begin{bmatrix} G(\mathbf{x}_{\Delta_1}; \mathbf{c}_1, a_1) & \cdots & G(\mathbf{x}_{\Delta_2}; \mathbf{c}_1, a_1) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_{\Delta_1}; \mathbf{c}_1, a_1)x_{\Delta_1 n} & \cdots & G(\mathbf{x}_{\Delta_2}; \mathbf{c}_1, a_1)x_{\Delta_2 n} \\ G(\mathbf{x}_{\Delta_1}; \mathbf{c}_2, a_2) & \cdots & G(\mathbf{x}_{\Delta_2}; \mathbf{c}_2, a_2) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_{\Delta_1}; \mathbf{c}_2, a_2)x_{\Delta_1 n} & \cdots & G(\mathbf{x}_{\Delta_2}; \mathbf{c}_2, a_2)x_{\Delta_2 n} \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_{\Delta_1}; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) & \cdots & G(\mathbf{x}_{\Delta_2}; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_{\Delta_1}; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}})x_{\Delta_1 n} & \cdots & G(\mathbf{x}_{\Delta_2}; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}})x_{\Delta_2 n} \end{bmatrix}^T \end{aligned} \quad (6.43)$$

$\tilde{N}(n+1) \times N_{k+1}$

where Δ_1 and Δ_2 denote $(\sum_{j=0}^k N_j) + 1$ and $\sum_{j=0}^{k+1} N_j$.

$$\mathbf{T}_1 = \begin{bmatrix} \mathbf{t}_{(\sum_{j=0}^k N_j)+1}^T \\ \vdots \\ \mathbf{t}_{\sum_{j=0}^{k+1} N_j}^T \end{bmatrix}_{N_{k+1} \times m} \quad (6.44)$$

In equation 6.41, $\mathbf{K}_{k+1}^{-1}$ is used to compute $\boldsymbol{\theta}^{(k+1)}$ from $\boldsymbol{\theta}^{(k)}$ instead of $\mathbf{K}_{k+1}$. The

update formula for $\mathbf{K}_{k+1}^{-1}$ is achieved using the Sherman-Morrison formula [30]

$$\begin{aligned}\mathbf{K}_{k+1}^{-1} &= (\mathbf{K}_k + \mathbf{H}_{k+1}^T \mathbf{H}_{k+1})^{-1} \\ &= \mathbf{K}_k^{-1} - \mathbf{K}_k^{-1} \mathbf{H}_{k+1}^T (\mathbf{I} + \mathbf{H}_{k+1} \mathbf{K}_k^{-1} \mathbf{H}_{k+1}^T)^{-1} \mathbf{H}_{k+1} \mathbf{K}_k^{-1}\end{aligned}\quad (6.45)$$

By making $\mathbf{P}_{k+1} = \mathbf{K}_{k+1}^{-1}$, we have the following equations for updating $\boldsymbol{\theta}^{(k+1)}$,

$$\begin{aligned}\mathbf{P}_{k+1} &= \mathbf{P}_k - \mathbf{P}_k \mathbf{H}_{k+1}^T (\mathbf{I} + \mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T)^{-1} \mathbf{H}_{k+1} \mathbf{P}_k \\ \boldsymbol{\theta}^{(k+1)} &= \boldsymbol{\theta}^{(k)} + \mathbf{P}_{k+1} \mathbf{H}_{k+1}^T (\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\theta}^{(k)})\end{aligned}\quad (6.46)$$

The equation 6.46 constitutes the recursive formula for $\boldsymbol{\theta}^{(k+1)}$.

Remark 3: Equation (6.46) illustrates that the sequential implementation of the least-squares solution to equation (6.28) is similar to recursive least-squares algorithm in Chong and Zak [23] and thus its convergence results can be applied here.

The **Online Sequential Fuzzy Extreme Learning Machine** (OS-Fuzzy-ELM) is summarized as follows:

Proposed OS-Fuzzy-ELM Algorithm: Given certain membership function g and rule number $\tilde{N}$ for a specific application, the data $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \dots\}$ arrives sequentially.

step 1 Initialization Phase: Initialize the learning using a small chunk of initial training data $\aleph_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$ from the given training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \dots\}$, $N_0 \geq \tilde{N}$.

- (1) Assign random centers $\mathbf{c}_i$ and width a_i for membership function g , $i = 1, \dots, \tilde{N}$.
- (2) Calculate the initial matrix $\mathbf{H}_0$ (cf. equation (6.30) and equation (6.31)) for Mamdani and TSK models.
- (3) Estimate the initial parameter matrix $\boldsymbol{\theta}^{(0)} = \mathbf{P}_0 \mathbf{H}_0^T \mathbf{T}_0$, where $\mathbf{P}_0 = (\mathbf{H}_0^T \mathbf{H}_0)^{-1}$ and $\mathbf{T}_0 = [\mathbf{t}_1, \dots, \mathbf{t}_{N_0}]^T$.
- (4) Set $k = 0$.

step 2 Sequential Learning Phase:

Present the $(k+1)$ -th chunk of new observations: $\aleph_{k+1} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=(\sum_{j=0}^k N_j)+1}^{\sum_{j=0}^{k+1} N_j}$, where N_{k+1} denotes the number of observations in the $(k+1)$ -th chunk, do

- (1) Calculate the partial matrix $\mathbf{H}_{k+1}$ for the $(k+1)$ -th chunk of data $\aleph_{k+1}$ (cf. equation (6.42) and equation (6.43)) for the Mamdani and TSK models.

Set $\mathbf{T}_{k+1} = [\mathbf{t}_{(\sum_{j=0}^k N_j)+1}, \dots, \mathbf{t}_{\sum_{j=0}^{k+1} N_j}]^T$.

- (2) Calculate the parameter matrix $\boldsymbol{\theta}^{(k+1)}$:

$$\begin{aligned} \mathbf{P}_{k+1} &= \mathbf{P}_k - \mathbf{P}_k \mathbf{H}_{k+1}^T (\mathbf{I} + \mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T)^{-1} \mathbf{H}_{k+1} \mathbf{P}_k \\ \boldsymbol{\theta}^{(k+1)} &= \boldsymbol{\theta}^{(k)} + \mathbf{P}_{k+1} \mathbf{H}_{k+1}^T (\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\theta}^{(k)}) \end{aligned} \quad (6.47)$$

- (3) Set $k = k + 1$. Go to *step 2*.

The OS-Fuzzy-ELM consists of two main phases. Initialization phase is to train the fuzzy inference systems using the Fuzzy-ELM method with some batch of training data in the initialization stage and these initialization training data will be discarded as soon as initialization phase is completed. For this the required training data is very small, which may be equal to the number of rules. For example, if there are 10 rules, we may need 10 training samples to initialize the learning. After initialization phase, the OS-Fuzzy-ELM will learn the training

data one-by-one ($N_{k+1} \equiv 1$) or chunk-by-chunk (with fixed or varying size) and then all the training data will be discarded once the learning procedure on these data is completed. If $N_0 = N$, the OS-Fuzzy-ELM becomes Fuzzy-ELM and thus the Fuzzy-ELM is a special case of OS-Fuzzy-ELM when all the training data is present in one iteration.

In OS-Fuzzy-ELM, the control parameter to be selected is the number of fuzzy rules $\tilde{N}$. In the next section, the determination of the parameter $\tilde{N}$ for the proposed OS-Fuzzy-ELM is analyzed in detail using an example. As theorem 6.2.3 describes, OS-Fuzzy-ELM can be used for any bounded nonconstant piecewise continuous membership function. Thus in the following sections the results for OS-Fuzzy-ELM are based on the kind of membership functions listed in Chapter 2. However to give a brief illustration of the main points, only the results from the three membership functions, viz., Gaussmf, Trimf and Cauchymf are given below while the results from other membership functions including Psigmf, Dsigmf, Gauss2mf, Trapmf and Pimf are given in Appendix D.

6.4 Parameter Determination of OS-Fuzzy-ELM

Generally better training and testing performance can be achieved by increasing the number of the parameter $\tilde{N}$. However this does not mean that arbitrarily large number of parameter $\tilde{N}$ is required for good system performance for training and testing accuracies and this will be illustrated by the following example. Here a nonlinear static function is utilized to illustrate the effect of the parameter $\tilde{N}$ on the system performance in terms of training and testing accuracies when the parameter $\tilde{N}$ changes. The nonlinear static function is given by,

$$f = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2 \quad (6.48)$$

3375 training data and 125 testing data are drawn from the input space $[1, 6] \times [1, 6] \times [1, 6]$ and $[1.5, 5.5] \times [1.5, 5.5] \times [1.5, 5.5]$ separately. A uniformly distributed

noise in the range of $[-0.2, 0.2]$ is added to all the training samples while testing data remain noise-free. The parameter $\tilde{N}$ changes in the range $[5, 50]$ for the TSK model with an interval increase of 5 and in the range $[20, 110]$ for Mamdani model with an interval increase of 10.

Table 6.1 gives the effect of parameter $\tilde{N}$ on system performance in terms of the training RMS error and the testing RMS error under different values and TSK fuzzy model. To give a clear illustration, Figure 6.3, Figure 6.4 and Figure 6.5 show the relation between the performance of OS-Fuzzy-ELM and its number of rules. As observed in these figures, the performance of OS-Fuzzy-ELM has a clear improvement initially but further becomes stable with the fuzzy rules increasing.

Table 6.1: Effect of parameter $\tilde{N}$ on system performance (training and testing accuracies) for TSK fuzzy model

# rules $\tilde{N}$	Gaussmf		Trimf		Cauchymf	
	Training (RMSE)	Testing (RMSE)	Training (RMSE)	Testing (RMSE)	Training (RMSE)	Testing (RMSE)
5	0.6482	0.4769	0.6861	0.5289	0.5696	0.4543
10	0.3272	0.2881	0.3982	0.3044	0.2795	0.2531
15	0.1863	0.1346	0.2001	0.1456	0.1828	0.1370
20	0.1672	0.0901	0.1724	0.1031	0.1439	0.0753
25	0.1459	0.0647	0.1333	0.0638	0.1266	0.0455
30	0.1368	0.0505	0.1229	0.0495	0.1200	0.0356
35	0.1218	0.0426	0.1187	0.0414	0.1177	0.0306
40	0.1150	0.0344	0.1160	0.0353	0.1160	0.0272
45	0.1139	0.0307	0.1124	0.0311	0.1158	0.0262
50	0.1127	0.0275	0.1119	0.0289	0.1152	0.0257

Figure 6.3: The tendency of training and testing accuracies for Gaussmf and TSK fuzzy model according to the varying number of rules

Figure 6.4: The tendency of training and testing accuracies for Trimf and TSK fuzzy model according to the varying number of rules

Figure 6.5: The tendency of training and testing accuracies for Cauchymf and TSK fuzzy model according to the varying number of rules

Table 6.2 demonstrates the effect of parameter $\tilde{N}$ on system performance in terms of the training RMS error and the testing RMS error under different values and Mamdani fuzzy model. Figure 6.6, Figure 6.7 and Figure 6.8 give a clear illustration about the relation between the performance of OS-Fuzzy-ELM and its number of rules. From these figures one still can observe that the performance of OS-Fuzzy-ELM has a clear improvement initially but further becomes stable with the fuzzy rules increasing.

Table 6.2: Effect of parameter $\tilde{N}$ on system performance (training and testing accuracies) for Mamdani fuzzy model

# rules $\tilde{N}$	Gaussmf		Trimf		Cauchymf	
	Training (RMSE)	Testing (RMSE)	Training (RMSE)	Testing (RMSE)	Training (RMSE)	Testing (RMSE)
20	0.6444	0.4639	0.4589	0.4063	0.5799	0.4500
30	0.3295	0.2881	0.3644	0.3393	0.3752	0.3180
40	0.2672	0.2353	0.2515	0.2162	0.2664	0.2273
50	0.2135	0.1813	0.2145	0.1817	0.2169	0.1757
60	0.1758	0.1134	0.1657	0.1137	0.1762	0.1241
70	0.1646	0.0902	0.1516	0.0881	0.1559	0.0890
80	0.1487	0.0721	0.1456	0.0767	0.1420	0.0707
90	0.1402	0.0591	0.1344	0.0603	0.1357	0.0588
100	0.1364	0.0507	0.1232	0.0511	0.1308	0.0498
110	0.1284	0.0483	0.1150	0.0495	0.1276	0.0437

Figure 6.6: The tendency of training and testing accuracies for Gaussmf and Mamdani fuzzy model according to the varying number of rules

Figure 6.7: The tendency of training and testing accuracies for Trimf and Mamdani fuzzy model according to the varying number of rules

Figure 6.8: The tendency of training and testing accuracies for Cauchymf and Mamdani fuzzy model according to the varying number of rules

Thus in OS-Fuzzy-ELM the number of fuzzy rules $\tilde{N}$ may be selected by trial and error till the performance of OS-Fuzzy-ELM will appear no large variations with the further increase of the parameter $\tilde{N}$. All the results here are obtained based on the product T-norm and one-by-one sequential mode. However this is also true to other implementation modes and T-norms because similar results can be achieved under these conditions. This will be analysed in the next section.

6.5 Performance Evaluation of OS-Fuzzy-ELM

In this section, the performance of the proposed OS-Fuzzy-ELM is evaluated in detail using benchmark problems. Performance evaluation of OS-Fuzzy-ELM has been carried out on the benchmark problems described in Table 6.3 which includes a nonlinear system identification problem, a real world regression application, i.e., California Housing regression problem [12] and a real world classification application, i.e., DNA classification problem [12].

Table 6.3: Specification of Benchmark Datasets

Dataset	# Input Attributes	# Classes	# Training Data	# Testing Data
Nonlinear System Identification	3	-	5,000	200
California Housing	8	-	10,320	10,320
DNA	180	3	2,000	1,186

The performance of OS-Fuzzy-ELM is first compared with other popular sequential learning algorithms such as ANFIS [51], DENFIS [63], SAFIS [111], eTS [6] and Simpl.eTS [5] in a one-by-one learning mode for identification and regression problems. For classification problem, the comparison of OS-Fuzzy-ELM is carried out with the algorithms FNN [69] and FSVMNN [78]. Performance evaluation of OS-Fuzzy-ELM is then carried out in a chunk-by-chunk learning mode, different T-norm operations and different fuzzy models. Also, the comparison has been done based on different membership functions used by these algorithms.

All the simulations have been conducted in MATLAB 6.5 environment running in an ordinary PC with 1.7 GHZ CPU.

OS-Fuzzy-ELM, ANFIS, DENFIS, SAFIS, eTS and Simpl_eTS are applied for TSK fuzzy model. The consequent parameters of SAFIS are constant while the consequent parameters of OS-Fuzzy-ELM, ANFIS, DENFIS, eTS and Simpl_eTS are the linear equation on the input variables. In addition, OS-Fuzzy-ELM and ANFIS are applied for any bounded nonconstant piecewise continuous membership function, however SAFIS and eTS are applied for Gaussian form membership function and Simpl_eTS is applied for Cauchy form membership function. DENFIS can in theory be utilized for any bounded nonconstant piecewise membership function but only the triangular membership function is applied for the comparison as described in [63]. OS-Fuzzy-ELM, DENFIS, SAFIS, eTS and Simpl_eTS go through the training data sequentially in a single pass, however for ANFIS the training is done over a number of passes.

In OS-Fuzzy-ELM, the control parameter to be selected is the number of fuzzy rules $\tilde{N}$. In SAFIS [111], the parameters to be decided include the distance thresholds ($\epsilon_{\max}$, $\epsilon_{\min}$, γ), the overlap factor (κ) for determining the width of the newly added rule, the growing threshold (e_g) for a new rule and the pruning threshold (e_p) for removing an insignificant rule. The control parameters used in eTS [6] and Simpl_eTS [5] are the radius (r) and the parameter of RLS algorithm (Ω). For ANFIS [51], the control parameters are the learning rate, initial step size, forgetting factor and the number of membership functions. The control parameters of the DENFIS algorithm [63] are the distance threshold and the number of rules in a dynamic FIS. The parameters for each algorithm are determined depending on each problem but they are tuned to obtain the best performance in terms of accuracy.

6.5.1 Performance Evaluation of OS-Fuzzy-ELM: One-by-One Case

We first evaluate and compare the performance of the proposed OS-Fuzzy-ELM with other one-by-one learning mode algorithms: SAFIS, ANFIS, DENFIS, eTS and SimpleTS. For each problem, the results are averaged over 50 trials and compared in terms of averaged training accuracy, testing accuracy, training time and the required number of fuzzy rules.

6.5.1.1 Function Approximation Problems

In this section, we consider a nonlinear system identification problem as well as a real world regression problem.

6.5.1.1.1 Nonlinear system identification

The nonlinear dynamic system to be identified is described by [92]

$$y(n) = \frac{y(n-1)y(n-2)(y(n-1) + 2.5)}{1 + y^2(n-1) + y^2(n-2)} + u(n-1) \quad (6.49)$$

The equilibrium state of the unforced system given by equation (6.49) is $(0, 0)$. The training input $u(n)$ is uniformly selected in the range $[-2, 2]$ and the testing input $u(n)$ is given by $u(n) = \sin(2\pi n/25)$. 5000 and 200 observation data are produced for the purpose of training and testing. A uniformly distributed noise in the range of $[-0.2, 0.2]$ is added to all the training samples while testing data remain noise-free.

Selecting $[y(n-1), y(n-2), u(n-1)]$ and $y(n)$ as the input-output of OS-Fuzzy-ELM, the identified model is given by this equation

$$\hat{y}(n) = \hat{f}(y(n-1), y(n-2), u(n-1)) \quad (6.50)$$

where $\hat{f}$ is the OS-Fuzzy-ELM approximation and $\hat{y}(n)$ is the output of the OS-Fuzzy-ELM.

Table 6.4 gives the performance comparison for OS-Fuzzy-ELM, ANFIS, SAFIS, eTS, SimpleTS and DENFIS in terms of training accuracy, testing accuracy, number of fuzzy rules and training time. For the comparison purposes, OS-Fuzzy-ELM uses the corresponding membership functions used by the other algorithms. For this study, ANFIS's training uses 50 epochs for each trial. From the table, it can be seen that OS-Fuzzy-ELM obtains similar or better testing accuracy than other algorithms and further it can be found that OS-Fuzzy-ELM requires a smaller number of fuzzy rules than eTS, SimpleTS and DENFIS and less training time compared to all the other algorithms. To highlight the training time comparison, we also show the same in Figure 6.9. From this figure, it can be clearly seen that OS-Fuzzy-ELM takes the lowest training time and ANFIS takes the highest.

Table 6.4: Performance comparison for nonlinear system identification

Algorithms	Gaussmf			Trimf			Cauchymf		
	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Testing (RMSE)
OS-Fuzzy-ELM	0.1279	0.0490	30	16.261	0.1321	0.0530	30	15.772	0.0402
ANFIS	0.1311	0.0588	27	149.98	0.1332	0.0588	27	175.08	0.0479
SAFIS	0.1493	0.0533	30	22.473	N/A	N/A	N/A	N/A	N/A
eTS	0.1620	0.0638	31	29.250	N/A	N/A	N/A	N/A	N/A
Simpl.eTS	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.1169
DENFIS	N/A	N/A	N/A	N/A	0.2246	0.1204	276	82.052	N/A

Figure 6.9: Training time comparison for nonlinear system identification problem based on the same membership functions between different algorithms.

6.5.1.1.2 Regression Problem - California Housing Prediction

The main objective of the California Housing problem is to predict the median California housing price based on the information collected using all the block groups in California from the 1990 Census and the dataset is obtained from the StatLib repository [12].

California Housing database consist of 20640 observations and each observation consists of eight continuous inputs (median income, housing median age, total rooms, total bedrooms, population, households, latitude, and longitude) and one continuous output (median house value). For this problem, 10320 training data and 10320 testing data are randomly generated. The input and output attributes are normalized in the range $[0, 1]$.

The training and testing accuracies for OS-Fuzzy-ELM, ANFIS, SAFIS, eTS, SimpleTS and DENFIS are compared in Table 6.5. Here, OS-Fuzzy-ELM uses the corresponding membership functions used by the other algorithms. As in the previous case, ANFIS's training uses 50 epochs for each trial. It can be clearly seen from the table that OS-Fuzzy-ELM obtains similar or better testing accuracy than other algorithms. The table also gives the comparison in terms of the number of fuzzy rules and the training time. The training time comparison between different algorithms based on the same membership function is shown in Figure 6.10. Here also, OS-Fuzzy-ELM requires lesser training time compared to all the other algorithms.

Table 6.5: Performance comparison for California housing prediction between different algorithms

Algorithms	Gaussmf				Trimf				Cauchymf			
	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)
OS-Fuzzy-ELM	0.1311	0.1344	5	3.7616	0.1317	0.1363	5	3.7441	0.1302	0.1320	5	3.7515
ANFIS	0.1391	0.1395	8	186.14	0.1417	0.1426	8	183.87	0.1378	0.1380	8	188.57
SAFIS	0.1346	0.1355	8	23.627	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
eTS	0.1334	0.1358	8	16.850	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Simpl.eTS	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.1616	0.1616	10	34.942
DENFIS	N/A	N/A	N/A	N/A	0.1674	0.1675	23	41.164	N/A	N/A	N/A	N/A

Figure 6.10: Training time comparison for California housing prediction based on the same membership functions between different algorithms.

6.5.1.2 Classification Problem - DNA

For classification studies, the DNA [12] benchmark problem is used here. The database “Primate splice-junction gene sequences (DNA) with associated imperfect domain theory” is known as the DNA problem. Splice junctions are points on a DNA sequence at which ‘superfluous’ DNA is removed during the process of protein creation in higher organisms. The aim of the DNA problem is, given a sequence of DNA, to recognize the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out). This consists of three subtasks: recognizing exon/intron boundaries (referred to as EI sites), intron/exon boundaries (IE sites), and neither (n sites). A given sequence of DNA consists of 60 elements (called “nucleotides” or “base-pairs”). The symbolic variables representing nucleotides are replaced by 3 binary indicator variables, thus resulting in 180 binary attributes. The 2000 training and 1186 testing sets are fixed according to [12], but order of training set is randomly shuffled for each trial.

Table 6.6 gives the classification accuracy comparison between OS-Fuzzy-ELM, FSVMNN and FNN. For this problem, use of ANFIS, SAFIS, eTS, SimpleTS and DENFIS results in a large number of rules and hence a large training time or system overflow. Thus, they are not reported here. From the table it can be seen that OS-Fuzzy-ELM produces similar training and testing accuracy but with lesser number of fuzzy rules compared with FSVMNN. Compared with FNN, OS-Fuzzy-ELM obtains better classification accuracy.

Table 6.6: Performance comparison between different algorithms for DNA classification

Algorithms	Training Accuracy (%)	Testing Accuracy (%)	# Rules
OS-Fuzzy-ELM	96.61	94.28	5
FSVMNN [78]	93.60	94.20	334
FNN [69]	N/A	89.75	N/A

6.5.2 Performance Evaluation of OS-Fuzzy-ELM: Chunk-by-Chunk Mode

The performance of OS-Fuzzy-ELM in the chunk-by-chunk mode is given in Table 6.7 for nonlinear system identification, Table 6.8 for California Housing prediction problem and Table 6.9 for the DNA classification problem. These tables also give the results for fixed and varying (randomly between 10 and 30) chunk sizes. For the purpose of comparison, results of chunk size 1 (1-by-1 learning mode) and also the results for chunk size of the entire training set (the batch mode) have also been presented. From these tables, one can note that the training and testing accuracies obtained by implementing in different chunk sizes are similar for the same membership functions. Also, there is no large variations of training and testing accuracies according to different membership functions. The tables also present the training time and number of fuzzy rules for different chunk sizes. From the tables, it can be seen that the 1-by-1 learning mode takes the largest time and the batch learning mode takes the least time. The training time taken by the chunk of 20-by-20 and [10,30] falls in between. This is consistent with the fact that normally sequential learning algorithms take more training time than batch algorithms.

Table 6.7: Performance comparison implemented in different chunk sizes for nonlinear system identification

Chunk Sizes	Gaussmf				Trimf				Cauchymf			
	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)
Batch	0.1278	0.0491	30	1.1434	0.1321	0.0530	30	1.1116	0.1217	0.0402	30	1.1103
1-by-1	0.1279	0.0490	30	16.261	0.1321	0.0530	30	15.772	0.1217	0.0402	30	14.592
20-by-20	0.1281	0.0491	30	1.3114	0.1271	0.0525	30	1.3195	0.1217	0.0401	30	1.2004
[10, 30]	0.1269	0.0492	30	1.3344	0.1333	0.0536	30	1.2972	0.1226	0.0406	30	1.2822

Table 6.8: Performance comparison implemented in different chunk sizes for California housing prediction

Chunk Sizes	Gaussmf				Trimf				Cauchymf			
	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)
Batch	0.1311	0.1340	5	0.4410	0.1321	0.1357	5	0.4263	0.1301	0.1319	5	0.4253
1-by-1	0.1311	0.1344	5	3.7616	0.1317	0.1363	5	3.7441	0.1302	0.1320	5	3.7515
20-by-20	0.1312	0.1344	5	0.5390	0.1321	0.1365	5	0.4919	0.1303	0.1321	5	0.5247
[10, 30]	0.1314	0.1340	5	0.6092	0.1313	0.1365	5	0.5897	0.1310	0.1329	5	0.5863

Table 6.9: Performance comparison implemented in different chunk sizes for DNA classification

Chunk Sizes	Gaussmf				Trimf				Cauchymf			
	Training (%)	Testing (%)	# Rules	Training Times (Secs)	Training (%)	Testing (%)	# Rules	Training Times (Secs)	Training (%)	Testing (%)	# Rules	Training Times (Secs)
Batch	96.72	94.33	5	24.419	96.61	94.29	5	23.062	96.72	94.34	5	20.600
1-by-1	96.61	94.28	5	2164.8	96.56	94.28	5	2177.8	96.51	94.21	5	2177.6
20-by-20	96.65	94.14	5	128.515	96.33	94.15	5	127.85	96.40	94.23	5	127.66
[10, 30]	96.66	94.16	5	126.487	96.43	94.12	5	123.92	94.63	94.14	5	121.63

6.5.3 Performance Comparison of OS-Fuzzy-ELM: Product vs Min T-Norm

Theorem 6.2.3 illustrates that OS-Fuzzy-ELM can be used for any types of T-norm operators, thus another commonly used T-norm operator named ‘Min’ is utilized to verify this. Table 6.10, Table 6.11 and Table 6.12 present the results using ‘Min’ T-norm for nonlinear system identification, California housing prediction and DNA classification problems together with the results using ‘Product’ T-norm. From these tables it can be found that OS-Fuzzy-ELM is effective for the ‘Min’ T-norm for all the implementation modes and all the membership functions and further it can be observed that the results under these conditions are similar. Besides, by comparing the ‘Product’ T-norm and ‘Min’ T-norm, one can note from these tables that the results from them are similar.

Table 6.10: Performance comparison between Product and Min T-norm operators for nonlinear system identification

Chunk Sizes	T-norm Types	Gaussmf				Trimf				Cauchymf			
		Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)
Batch	Product	0.1278	0.0491	30	1.1434	0.1321	0.0530	30	1.1116	0.1217	0.0402	30	1.1103
	Min	0.1440	0.0649	30	1.6375	0.1487	0.0640	30	1.6097	0.1425	0.0688	30	1.6109
1-by-1	Product	0.1279	0.0490	30	16.261	0.1321	0.0530	30	15.772	0.1217	0.0402	30	14.592
	Min	0.1406	0.0644	30	16.378	0.1496	0.0649	30	16.152	0.1423	0.0690	30	15.070
20-by-20	Product	0.1281	0.0491	30	1.3114	0.1271	0.0525	30	1.3195	0.1217	0.0401	30	1.2004
	Min	0.1416	0.0624	30	1.7278	0.1500	0.0629	30	1.6980	0.1414	0.0682	30	1.6927
[10, 30]	Product	0.1269	0.0492	30	1.3344	0.1333	0.0536	30	1.2972	0.1226	0.0406	30	1.2822
	Min	0.1420	0.0603	30	1.8269	0.1497	0.0661	30	1.7827	0.1425	0.0665	30	1.7922

Table 6.11: Performance comparison between Product and Min T-norm operators for California housing prediction

Chunk Sizes	T-norm Types	Gaussmf				Trimf				Cauchymf			
		Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)
Batch	Product	0.1311	0.1340	5	0.4410	0.1321	0.1357	5	0.4263	0.1301	0.1319	5	0.4253
	Min	0.1330	0.1365	5	0.6116	0.1337	0.1370	5	0.5987	0.1327	0.1348	5	0.5959
1-by-1	Product	0.1311	0.1344	5	3.7616	0.1317	0.1363	5	3.7441	0.1302	0.1320	5	3.7515
	Min	0.1330	0.1364	5	3.8657	0.1337	0.1370	5	3.7894	0.1327	0.1327	5	3.8125
20-by-20	Product	0.1312	0.1344	5	0.5390	0.1321	0.1365	5	0.4919	0.1303	0.1321	5	0.5247
	Min	0.1330	0.1363	5	0.6706	0.1337	0.1370	5	0.6585	0.1327	0.1347	5	0.6510
[10, 30]	Product	0.1314	0.1340	5	0.6092	0.1313	0.1365	5	0.5897	0.1310	0.1329	5	0.5863
	Min	0.1334	0.1352	5	0.8050	0.1331	0.1386	5	0.7798	0.1336	0.1380	5	0.8046

Table 6.12: Performance comparison between Product and Min T-norm operators for DNA classification

Chunk Sizes	T-norm Types	Gaussmf				Trimf				Cauchymf			
		Training (%)	Testing (%)	# Rules	Training Times (Secs)	Training (%)	Testing (%)	# Rules	Training Times (Secs)	Training (%)	Testing (%)	# Rules	Training Times (Secs)
Batch	Product	96.72	94.33	5	24.419	96.61	94.29	5	23.062	96.72	94.34	5	20.600
	Min	97.35	92.49	5	24.712	97.05	92.68	5	23.578	97.60	92.52	5	21.606
1-by-1	Product	96.61	94.28	5	2164.8	96.56	94.28	5	2177.8	96.51	94.21	5	2177.6
	Min	97.52	92.59	5	2192.2	97.15	92.41	5	2179.3	97.00	92.08	5	2191.7
20-by-20	Product	96.65	94.14	5	128.515	96.33	94.15	5	127.85	96.40	94.23	5	127.66
	Min	97.96	92.91	5	129.21	97.50	92.40	5	128.02	97.05	92.45	5	128.76
[10, 30]	Product	96.66	94.16	5	126.487	96.43	94.12	5	123.92	94.63	94.14	5	121.63
	Min	97.01	92.99	5	126.69	97.97	92.41	5	124.68	97.01	92.13	5	122.00

6.5.4 Performance Comparison of OS-Fuzzy-ELM: TSK vs Mamdani Model

Although the results presented in the previous sections are all based on TSK model, we have also evaluated OS-Fuzzy-ELM for Mamdani model for the three examples on all learning modes. Table 6.13 summaries the number of rules and update parameters for both TSK and Mamadani models for these problems.

Table 6.13: Parameter specification of TSK and Mamdani fuzzy models for each problem

Problems	Fuzzy models	# Input Dimension	# Output Dimension	# Rules	# Parameters
Nonlinear System Identification	TSK	3	1	30	120
	Mamdani			110	110
California Housing Prediction	TSK	8	1	5	45
	Mamdani			35	35
DNA Classification	TSK	180	3	5	2715
	Mamdani			200	600

Table 6.14 and Table 6.15 present the training and testing accuracies for the non-linear identification and California housing prediction problems using TSK and Mamdani fuzzy models using all the membership functions and the ‘Product’ T-norm. From the tables it can be seen that Mamdani produces similar training accuracies as TSK.

The training time and number of rules taken by Mamdani and TSK are given in the tables. It can be seen that the training time for both these models are of the same order of magnitude although Mamdani produces larger number of rules. This is because the number of update parameters (as given in Table 6.13) for Mamdani is of the same order of magnitude as that of TSK even though its number of rules is larger.

Table 6.14: Performance comparison between TSK and Mamdani fuzzy models for nonlinear system identification

Chunk Sizes	Fuzzy Models	Gaussmf				Trimf				Cauchymf			
		Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)
Batch	TSK	0.1278	0.0491	30	1.1434	0.1321	0.0530	30	1.1116	0.1217	0.0402	30	1.1103
	Mamdani	0.1285	0.0494	110	1.1531	0.1318	0.0528	110	1.0629	0.1224	0.0415	110	1.0297
1-by-1	TSK	0.1279	0.0490	30	16.261	0.1321	0.0530	30	15.772	0.1217	0.0402	30	14.592
	Mamdani	0.1286	0.0495	110	13.249	0.1325	0.0541	110	13.977	0.1223	0.0421	110	12.609
20-by-20	TSK	0.1281	0.0491	30	1.3114	0.1271	0.0525	30	1.3195	0.1217	0.0401	30	1.2004
	Mamdani	0.1289	0.0493	110	1.2928	0.1318	0.0528	110	1.1887	0.1227	0.0428	110	1.1530
[10, 30]	TSK	0.1269	0.0492	30	1.3344	0.1333	0.0536	30	1.2972	0.1226	0.0406	30	1.2822
	Mamdani	0.1282	0.0483	110	1.3526	0.1327	0.0544	110	1.2491	0.1230	0.0433	110	1.2112

Table 6.15: Performance comparison between TSK and Mamdani fuzzy models for California housing prediction

Chunk Sizes	Fuzzy Models	Gaussmf				Trimf				Cauchymf			
		Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)	Training (RMSE)	Testing (RMSE)	# Rules	Training Times (Secs)
Batch	TSK	0.1311	0.1340	5	0.4410	0.1321	0.1357	5	0.4263	0.1301	0.1319	5	0.4253
	Mamdani	0.1330	0.1351	110	0.6331	0.1346	0.1351	35	0.5640	0.1330	0.1338	35	0.5066
1-by-1	TSK	0.1311	0.1344	5	3.7616	0.1317	0.1363	5	3.7441	0.1302	0.1320	5	3.7515
	Mamdani	0.1338	0.1348	35	3.4572	0.1346	0.1351	35	3.4110	0.1331	0.1340	35	3.3027
20-by-20	TSK	0.1312	0.1344	5	0.5390	0.1321	0.1365	5	0.4919	0.1303	0.1321	5	0.5247
	Mamdani	0.1332	0.1353	35	0.7735	0.1346	0.1351	35	0.1329	0.1340	35	0.6441	
[10, 30]	TSK	0.1314	0.1340	5	0.6092	0.1313	0.1365	5	0.5897	0.1310	0.1329	5	0.5863
	Mamdani	0.1336	0.1350	35	0.8672	0.1343	0.1346	35	0.1335	0.1349	35	0.7940	

Table 6.16 shows the accuracies for the DNA problem for both TSK and Mamdani models and they are similar. However the training time taken by Mamdani is generally two order of magnitude lesser than that of TSK for one-by-one implementation mode although Mamdani needs larger number of fuzzy rules. This is because of the smaller number of update parameters required by Mamdani as compared to TSK (Table 6.13) for this problem.

Table 6.16: Performance comparison between TSK and Mamdani fuzzy models for DNA classification

Chunk Sizes	Fuzzy Models	Gaussmf			Trimf			Cauchymf		
		Training (%)	Testing (%)	# Rules	Training Times (Secs)	Testing (%)	# Rules	Training Times (Secs)	Testing (%)	# Rules
Batch	TSK	96.72	94.33	5	24.419	96.61	5	23.062	96.72	5
	Mamdani	96.73	94.33	200	10.732	96.60	200	9.8509	96.75	200
1-by-1	TSK	96.61	94.28	5	2164.8	96.56	5	2177.8	96.51	5
	Mamdani	96.55	94.62	200	54.448	96.25	200	53.497	96.58	200
20-by-20	TSK	96.65	94.14	5	128.515	96.33	5	127.85	96.40	5
	Mamdani	96.44	94.43	200	22.531	96.57	200	20.844	96.43	200
[10, 30]	TSK	96.66	94.16	5	126.487	96.43	5	123.92	94.63	5
	Mamdani	96.45	94.60	200	12.705	96.49	200	11.692	96.54	200

In this chapter, an online sequential fuzzy extreme learning machine (OS-Fuzzy-ELM) has been developed. Its advantages are summarized as follows:

- OS-Fuzzy-ELM can be applied for both the TSK and Mamdani fuzzy models for any bounded nonconstant piecewise continuous membership function and any type of T-norm fuzzy logic operation in a unified way.
- In OS-Fuzzy-ELM the parameters in the membership functions are randomly assigned and then the consequent parameters are analytically determined.
- OS-Fuzzy-ELM can learn the data sequentially in a one-by-one or chunk-by-chunk mode and then discard the data for which the training has already been done.

Furthermore the performance evaluation of OS-Fuzzy-ELM has been carried out on benchmark problems of nonlinear system identification, California Housing prediction and DNA pattern classification problem. The conclusions based on these problems are summarized as follows:

- Compared with other algorithms the proposed OS-Fuzzy-ELM produces similar or better accuracies. It also produces a significantly lower training time than all the other algorithms.
- Implemented in the chunk-by-chunk learning mode the results indicate that training and testing accuracies are similar for all the membership functions. However, the training time for 1-by-1 learning mode is higher than chunk-by-chunk mode.
- As to another commonly used ‘Min’ T-norm the results using different implementation modes and membership functions are similar. As to the comparison between Product T-norm with Min T-norm, their results are also similar.

- As far as TSK and Mamdani fuzzy models based on OS-Fuzzy-ELM are concerned, Mamdani fuzzy model requires more fuzzy rules than TSK fuzzy model when similar training and testing accuracies are achieved. However, when the number of the input variables is large, the training time taken by Mamdani model is much less than that by TSK although the required fuzzy rules are more.

In the next chapter OS-Fuzzy-ELM is utilized for aircraft autolanding fault-tolerant control problem suffering from actuator failures and severe winds described in Chapter 4.

Chapter 7

Aircraft Autolanding Fault-Tolerant Control with OS-Fuzzy-ELM Algorithm

In this chapter the proposed OS-Fuzzy-ELM is used for the aircraft autolanding problem described in Chapter 4. Using the feedback-error-learning control strategy described in Chapter 3, the proposed OS-Fuzzy-ELM is utilized as the fuzzy controller to aid the BTFC controller which is shown in Figure 7.1.

Figure 7.1: OS-Fuzzy-ELM aided BTFC control strategy for autolanding

The control scheme makes use of the BTFC conventional controller existing in the inner-loop to stabilize the system dynamics, and the OS-Fuzzy-ELM fuzzy controller acting as an aid to the inner-loop controller through online learning. Similar to SAFIS aided BTFC introduced in Chapter 5, two OS-Fuzzy-ELM controllers are designed for both the longitudinal and lateral direction motion given by Figure 7.2 and Figure 7.3.

Figure 7.2: OS-Fuzzy-ELM implementation block for the longitudinal direction

Figure 7.3: OS-Fuzzy-ELM implementation block for the lateral direction

Moreover both Gaussian form and Cauchy form membership functions are studied for the problem based on TSK fuzzy model and Product T-norm under one-by-one implementation mode. In the following sections, the performance of OS-Fuzzy-ELM aided BTFC control strategy is evaluated on the fault scenarios described in Chapter 4 and is compared with those of EMRAN aided BTFC and BTFC controllers [98]. The results show that OS-Fuzzy-ELM aided BTFC improves the fault tolerant capabilities compared to the other two schemes.

7.1 Single Surface Failure

In this section, a single failure of elevator or aileron is studied along with severe winds. First, we present the results for elevator failure. Here, left elevator is stuck at -10deg at the beginning of the turn. Table 7.1 shows the trajectory errors (RMS)

along with the number of rules for OS-Fuzzy-ELM aided BTFC based on the two types of membership functions.

Table 7.1: Performance comparison between OS-Fuzzy-ELM and EMRAN

Algorithms	OS-Fuzzy-ELM		EMRAN
	Gaussmf	Cauchymf	
Trajectory error (RMSE)	8.8195	8.8364	8.9206
Velocity error (RMSE)	3.0087	3.1929	3.8146
# of rules/neurons for δ_e	6	6	28
# of rules/neurons for δ_a	6	6	15

As seen from the table OS-Fuzzy-ELM produces similar performance for the two kinds of membership functions. Thus, in the following figures comparing OS-Fuzzy-ELM and EMRAN only the results achieved from the Cauchy form membership function are illustrated.

Figure 7.4 and Figure 7.5 show the altitude (h), the velocity (V_T), sideslip angle (β) and lateral position (Y) during the landing phase for the OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC.

Figure 7.4: Comparison of OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC for the altitude (h) and velocity (V_T) under the left elevator stuck at -10°

Figure 7.5: Comparison of OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC for the sideslip angle (β) and lateral position (Y) under the left elevator stuck at -10deg

The numbers at the top of the two figures represent the different segments of the trajectory. As presented in Chapter 4 BTFC alone is unable to cope with this failure and achieve a safe landing as the altitude drops around 50sec whereas OS-Fuzzy-ELM aided BTFC and EMRAN aided BTFC controllers are not only able to land the aircraft but also satisfy the touchdown performance requirements. Furthermore, it can be seen from the two figures that OS-Fuzzy-ELM aided BTFC and EMRAN aided BTFC control schemes are able to follow the reference trajectory closely but OS-Fuzzy-ELM produces lesser deviations than EMRAN for the

velocity (V_T) and sideslip angle (β). As described in Chapter 4 the large sideslip seen in Figure 7.5 around 110s is due to the abrupt step inputs in the side gust profile but these excursions are quickly damped out by the controllers. Table 7.1 shows the RMS trajectory errors along with the number of rules/neurons for OS-Fuzzy-ELM aided BTFC and EMRAN aided BTFC schemes. From the table, it can be seen that the trajectory error and the number of rules for OS-Fuzzy-ELM are smaller than those of EMRAN.

To analyze the control scheme performance in more detail, Figure 7.6 shows the deflection of the left elevator ($\delta_{e-left-t}$) and right elevator ($\delta_{e-right-t}$) along with the BTFC ($\delta_{e-right-c}$) and OS-Fuzzy-ELM/EMRAN components ($\delta_{e-right-f/n}$) of the control signals. The rule and neuron history based on OS-Fuzzy-ELM and EMRAN for the elevator δ_e is also shown. In OS-Fuzzy-ELM the number of fuzzy rules is maintained constant during the whole learning whereas in EMRAN the number of neurons is varied according to the learning.

Figure 7.6: Left and right elevator control signals under the left elevator stuck at -10deg

In the figure, the solid line represents the results for OS-Fuzzy-ELM aided BTFC and the dotted line represents the results from EMRAN aided BTFC. From the figure it is to be noted that the control signals for OS-Fuzzy-ELM aided BTFC and EMRAN aided BTFC are similar. However, the control signal for OS-Fuzzy-ELM aided BTFC is less oscillatory than that of EMRAN aided BTFC. Also, OS-Fuzzy-ELM only needs 6 rules for learning while EMRAN needs 28 neurons for learning.

The left and right aileron signals are given in Figure 7.7 and they are similar.

Figure 7.7: Left and right aileron control signals under the left elevator stuck at -10deg

The fault tolerant capabilities of OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC for a single elevator surface failure are given in Figure 7.8.

Figure 7.8: Failure tolerance under left elevator stuck conditions

Each point in the figure indicates a successful landing meeting the touchdown pillbox requirements. From the figure, it can be seen that OS-Fuzzy-ELM aided BTFC is able to meet the pillbox requirements for a wider range of deflections (-12 to 18 degrees) compared to EMRAN aided BTFC (-12 to 12 degrees). No points corresponding to BTFC are shown in the figure because it can not meet the pillbox requirements for the entire range of failures from -18deg to +25deg. Furthermore, OS-Fuzzy-ELM aided BTFC is able to meet the pillbox requirements during continuous elevator deflections without “gaps” but EMRAN aided BTFC is unable to meet the pillbox requirements at -8, 0 and 8 degrees of elevator deflections.

The pillbox in the x-y plane is shown in Figure 7.9 along with the touchdown points for OS-Fuzzy-ELM. It can be seen that all these touchdown points lie inside the pillbox.

Figure 7.9: Touchdown points for OS-Fuzzy-ELM aided BTFC under left elevator stuck conditions

Similar results for the case of single aileron failure are shown in Figure 7.10.

Figure 7.10: Failure tolerance under left elevator stuck conditions

From this figure, one can note that the OS-Fuzzy-ELM aided BTFC has a wider aileron failure tolerance range (-20 to 9 degrees) compared to BTFC (-7 to 4 degrees) and a slightly wider failure tolerance range than EMRAN aided BTFC (-7 to 20 degrees) with the exception of failure at 2 degree (for EMRAN). The figure further indicates that OS-Fuzzy-ELM is able to tolerate very large negative aileron stuck failures while EMRAN is able to tolerant very large positive aileron stuck failures. As to the range from 10 degree to 20 degree where EMRAN can tolerant the failures, the OS-Fuzzy-ELM is capable of landing the aircraft successfully but violates the touchdown pillbox requirements in terms of the y-distance criterion.

Figure 7.11 gives the touchdown points for OS-Fuzzy-ELM together with the pill-box in the x-y plane and we can observe that these touchdown points are inside the pillbox.

Figure 7.11: Touchdown points for OS-Fuzzy-ELM aided BTFC under left elevator stuck conditions

It should be noted that all the results from OS-Fuzzy-ELM in Figure 7.8 and Figure 7.10 are achieved by using the same number of fuzzy rules listed in Table 7.1.

7.2 Two Surface Failures

In this section, two failures are considered. Specifically, for the first case the left elevator and left aileron are stuck at different deflections. For the second case the left elevator and right aileron are considered.

7.2.1 Left Elevator and Left Aileron Stuck at Different Deflections

In this section, we first present a typical trajectory result when the left elevator is stuck at -10deg and the left aileron is stuck at +10deg. Table 7.2 shows the trajectory errors (RMS) along with the number of rules for OS-Fuzzy-ELM aided BTFC based on the Gaussian and Cauchy form membership functions.

Table 7.2: Performance comparison between OS-Fuzzy-ELM and EMRAN

Algorithms	OS-Fuzzy-ELM		EMRAN
	Gaussmf	Cauchymf	
Trajectory error (RMSE)	8.8947	8.8926	8.9127
Velocity error (RMSE)	2.8508	2.8663	3.8097
# of rules/neurons for δ_e	6	6	32
# of rules/neurons for δ_a	6	6	25

As seen from the table OS-Fuzzy-ELM produces similar performance for the two kinds of membership functions. Thus, in the following figures comparing OS-Fuzzy-ELM and EMRAN only the results achieved from the Cauchy form membership function are illustrated.

Figure 7.12 and Figure 7.13 show the altitude (h), the velocity (V_T), sideslip angle (β) and lateral position (Y) during the landing phase for the OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC along with the reference trajectory.

Figure 7.12: Comparison of OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC for the altitude (h) and velocity (V_T) under the left elevator stuck at -10deg and the left aileron stuck at 10deg

Figure 7.13: Comparison of OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC for the sideslip angle (β) and lateral position (Y) under the left elevator stuck at -10deg and the left aileron stuck at 10deg

From the two figures, it can be seen that BTFC alone is unable to cope with this failure and achieve a safe landing as the altitude drops around 30sec whereas OS-Fuzzy-ELM aided BTFC and EMRAN aided BTFC controllers are not only able to land the aircraft but also satisfy the touchdown performance requirements. The two figures demonstrate that OS-Fuzzy-ELM and EMRAN both follow the reference trajectory closely but OS-Fuzzy-ELM produces lesser deviations than EMRAN for the velocity (V_T) and sideslip angle (β). Table 7.2 shows the RMS trajectory errors along with the number of rules/neurons for OS-Fuzzy-ELM aided

BTFC and EMRAN aided BTFC schemes. It can be seen from the table that the trajectory error and the number of rules for OS-Fuzzy-ELM are smaller than those of EMRAN.

Figure 7.14 shows the deflection of the left elevator ($\delta_{e-left-t}$) and right elevator ($\delta_{e-right-t}$).

Figure 7.14: Left and right elevator control signals under the left elevator stuck at -10deg and the left aileron stuck at 10deg

In the figure, the solid line represents the results from OS-Fuzzy-ELM aided BTFC and the dotted line represents the results from EMRAN aided BTFC. The figure also shows the control signals for the BTFC component ($\delta_{e-right-c}$) along with the OS-Fuzzy-ELM/EMRAN components ($\delta_{e-right-f/n}$). The rule and neuron update process based on OS-Fuzzy-ELM and EMRAN for the elevator δ_e is also shown. Similarly in OS-Fuzzy-ELM the number of fuzzy rules is constant while the number of neurons in EMRAN is varied during the whole learning process. It is seen that EMRAN scheme needs around 30 neurons for learning whereas OS-Fuzzy-ELM requires only 6 rules indicating that OS-Fuzzy-ELM can do the job with a compact network.

Similar results for the left and right ailerons are given in Figure 7.15.

Figure 7.15: Left and right aileron control signals under the left elevator stuck at -10deg and the left aileron stuck at 10deg

The fault tolerant capabilities of OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC for the whole range of deflections are given in Figure 7.16. The edge of the fault tolerance envelopes achieved by OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC is given by the solid line, dashed line and dash-dot line. In their respective fault tolerant regions each point which indicates a successful landing meeting the touchdown pillbox requirements is represented by

$\diamond$, $+$ and $\times$.

Figure 7.16: Failure tolerance under left elevator and left aileron stuck conditions

From the figure, it can be seen that OS-Fuzzy-ELM aided BTFC is able to meet the pillbox requirements for a wider range of deflections compared to EMRAN aided BTFC as well as BTFC. It can be seen from the figure that the fault tolerance region achieved by OS-Fuzzy-ELM aided BTFC covers those of EMRAN aided BTFC and BTFC. But EMRAN aided BTFC controller does not cover the entire BTFC controller region due to its inability to meet the tight pillbox requirements.

It can also be observed from the figure that OS-Fuzzy-ELM aided BTFC has no “gaps” in the whole region whereas the regions covered by EMRAN aided BTFC and BTFC have “gaps” indicating their inability to meet the tight pillbox requirements at those stuck deflections.

7.2.2 Left Elevator and Right Aileron Stuck at Different Deflections

The fault tolerant envelope achieved by OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC under both left elevator and right aileron failures is given in Figure 7.17. Their fault tolerance envelope edge is given by the solid line, dashed line and dashdot line respectively. In their respective fault tolerant regions each point which indicates a successful landing meeting the touchdown pillbox requirements is represented by $\diamond$, $+$ and $\times$.

Figure 7.17: Failure tolerance under left elevator and right aileron stuck conditions

From the figure, it can also be seen that OS-Fuzzy-ELM aided BTFC is able to meet the pillbox requirements for a wider range of deflections compared to EMRAN aided BTFC as well as BTFC. It can be seen from the figure that the

fault tolerance region achieved by OS-Fuzzy-ELM aided BTFC covers those of EMRAN aided BTFC and BTFC. But EMRAN aided BTFC controller does not cover the entire BTFC controller region due to its inability to meet the tight pillbox requirements. It can be further observed from the figure that OS-Fuzzy-ELM aided BTFC has no “gaps” in the whole region whereas the regions covered by EMRAN aided BTFC and BTFC have “gaps” indicating their inability to meet the tight pillbox requirements at those stuck deflections.

The results from OS-Fuzzy-ELM in Figure 7.16 and Figure 7.17 are achieved by using the same number of fuzzy rules listed in Table 7.2.

We have also looked at the pitch angle at touch down for all these cases as it is critical to ensure that the aircraft tail does not touch the ground when it lands. For the normal and all the failure cases, the pitch angle at touch down is in the range of 9-10 degrees. A typical value of 15 degree is safe for most aircraft (depending on the take-off AOA plus some margin). Since we are specifically looking for touch down speeds higher than 60m/s, the pitch angle criteria is satisfied.

In this chapter, a control strategy based on OS-Fuzzy-ELM aided BTFC has been developed for a high performance fighter aircraft during the landing phase when aircraft is subject to actuator stuck faults and severe winds. Based on the study, the following points can be observed.

- For single surface failure cases, the BTFC controller is unable to meet the touchdown dispersions under elevator failures and is able to meet the requirements under aileron stuck failures only for small deflections (-8 and 4 degrees for aileron). The EMRAN aided BTFC controller enlarges the fault tolerant envelope for both the aileron and elevator failures (-12 to 12 degrees for elevator and -8 to 20 degrees for aileron) based on the same case of single surface failure. However, in the fault tolerant envelope achieved by EMRAN aided BTFC there are some “gaps” where the touchdown pillbox is not met

under those stuck degrees (2 degree for aileron and -8, 0 and 8 degrees for elevator). The proposed OS-Fuzzy-ELM aided BTFC controller achieves not only a continuous fault tolerant range but also a larger fault tolerant envelope (-12 to 18 degrees for elevator and -20 to 9 degrees for aileron) for single elevator and aileron failures compared with BTFC controller and EMRAN aided BTFC controller.

- For two surface failure cases like the combined elevator and aileron failures, the fault tolerant envelope achieved by OS-Fuzzy-ELM aided BTFC controller is enlarged and furthermore there are no “gaps” in the whole region compared with BTFC controller and EMRAN aided BTFC controller whose regions have “gaps” indicating their inability to meet the tight pillbox requirements at those stuck deflections (see Figure 7.16 and Figure 7.17).
- Compared with EMRAN, OS-Fuzzy-ELM produces lesser deviations for the the velocity (V_T), sideslip angle (β) (see Figure 7.4, Figure 7.5, Figure 7.12 and Figure 7.13).

From Chapter 5 and this chapter we can observe that when SAFIS and OS-Fuzzy-ELM are utilized as fuzzy controllers to aid the existing BTFC controller, both improve the fault-tolerant capabilities compared with EMRAN aided BTFC and BTFC controllers. In the next chapter, we will give a comparison between SAFIS and OS-Fuzzy-ELM for the aircraft autolanding problem and analyse the same.

Chapter 8

Comparison of OS-Fuzzy-ELM and SAFIS for Aircraft Autolanding Fault-Tolerant Control Problem

In this chapter, the proposed OS-Fuzzy-ELM and SAFIS algorithms are further compared based on the aircraft autolanding fault-tolerant control problem as described in Chapter 4. The comparison is based on the same fault scenarios introduced in Chapter 4 and the simulation results introduced in Chapter 5 and Chapter 7. Besides, as described in Chapter 7 OS-Fuzzy-ELM aided BTFC has been studied based on the two types of membership functions, viz., the Gaussian membership function and Cauchy membership function, and similar results from the two kinds of membership functions have been obtained. Thus, for convenience in the following sections for comparing OS-Fuzzy-ELM and SAFIS, only the results obtained from the Cauchy form membership function are presented.

8.1 Single Surface Failure

In this section, a single failure of elevator or aileron is considered. First, we present the results for elevator failure. Here, left elevator is stuck at -10deg at the beginning of the turn. Figure 8.1 and Figure 8.2 give the altitude (h), the velocity (V_T), sideslip angle (β) and lateral position (Y) during the landing phase for the OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC.

Figure 8.1: Comparison of OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC for the altitude (h) and velocity (V_T) under the left elevator stuck at -10deg

Figure 8.2: Comparison of OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC for the sideslip angle (β) and lateral position (Y) under the left elevator stuck at -10deg

The numbers at the top of the figure represent the different segments of the trajectory. As observed from the two figures, OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC are able to handle the failure and land the aircraft successfully for they utilize online learning to learn the desired signals quickly and generate a larger control signal to drive the aircraft to follow the desired outputs. The two figures further illustrate that OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC are able to follow the reference trajectory closely. OS-Fuzzy-ELM and SAFIS produce the large sideslip seen in Figure 8.2 around 110s due to the abrupt step inputs in

the side gust profile but these excursions are quickly damped out by the controllers. Table 8.1 shows the RMS trajectory error together with the number of rules for OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC and we can observe that the trajectory error of OS-Fuzzy-ELM is smaller than that of SAFIS and the number of rules for OS-Fuzzy-ELM is lesser than that of SAFIS for the lateral direction motion. In the longitudinal direction motion, they require the same number of fuzzy rules.

Table 8.1: Performance comparison between OS-Fuzzy-ELM and SAFIS

Algorithms	OS-Fuzzy-ELM aided BTFC	SAFIS aided BTFC
Trajectory error (RMSE)	8.8364	8.8573
Velocity error (RMSE)	3.1929	3.4243
# of rules for δ_e	6	6
# of rules for δ_a	6	14

Figure 8.3 shows the deflection of the left elevator ($\delta_{e-left-t}$) and right elevator ($\delta_{e-right-t}$) along with the BTFC ($\delta_{e-right-c}$) and OS-Fuzzy-ELM/SAFIS components ($\delta_{e-right-f}$) of the control signals achieved by OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC control schemes.

Figure 8.3: Left and right elevator control signals under the left elevator stuck at -10deg

In the figure, the solid line represents the results from OS-Fuzzy-ELM aided BTFC and the dot line represents the results from SAFIS aided BTFC. As observed from Figure 8.3 control signals achieved from OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC are very similar.

The left and right aileron signals are given in Figure 8.4 and they are very similar.

Figure 8.4: Left and right aileron control signals under the left elevator stuck at -10deg

The fault tolerant capabilities of OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC for a single elevator surface failure are given in Figure 8.5.

Figure 8.5: Failure tolerance under left elevator stuck conditions

Each point in the figure indicates a successful landing meeting the touchdown pillbox requirements. From the figure, it can be seen that OS-Fuzzy-ELM aided BTFC is able to tolerate a same continuous range of elevator failures (-12 to 18 degrees) compared to SAFIS aided BTFC (-12 to 18 degrees).

Figure 8.6 gives the case of single aileron failure and from this figure one can note that the OS-Fuzzy-ELM aided BTFC has a slightly wider continuous aileron failure tolerance range (-20 to 9 degrees) than SAFIS aided BTFC (-14 to 14 degrees).

Figure 8.6: Failure tolerance under left elevator stuck conditions

8.2 Two Surface Failures

In this section, two failures are considered. Specifically, for the first case the left elevator and left aileron are stuck at different deflections. For the second case the left elevator and right aileron are considered.

8.2.1 Left Elevator and Left Aileron Stuck at Different Deflections

In this section, we first present a typical trajectory result when the left elevator is stuck at -10° and the left aileron is stuck at $+10^\circ$. Figure 8.7 and Figure 8.8 give the altitude (h), the velocity (V_T), sideslip angle (β) and lateral position (Y) during the landing phase for the OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC.

Figure 8.7: Comparison of OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC for the altitude (h) and velocity (V_T) under the left elevator stuck at -10° and the left aileron stuck at 10°

Figure 8.8: Comparison of OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC for the sideslip angle (β) and lateral position (Y) under the left elevator stuck at -10deg and the left aileron stuck at 10deg

The numbers at the top of the figure represent the different segments of the trajectory. As observed from the two figures, OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC both are able to handle the failure and land the aircraft successfully by utilizing their online learning ability. The two figures further illustrate that OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC are able to follow the reference trajectory closely.

Table 8.2 shows the RMS trajectory error together with the number of rules for OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC and we can observe that the trajectory error and the number of rules for OS-Fuzzy-ELM are smaller than those of SAFIS.

Table 8.2: Performance comparison between OS-Fuzzy-ELM and SAFIS

Algorithms	OS-Fuzzy-ELM aided BTFC	SAFIS aided BTFC
Trajectory error (RMSE)	8.8926	8.8978
Velocity error (RMSE)	2.8663	3.2761
# of rules for δ_e	6	8
# of rules for δ_a	6	22

Figure 8.9 shows the deflection of the left elevator ($\delta_{e-left-t}$) and right elevator ($\delta_{e-right-t}$) along with the BTFC ($\delta_{e-right-c}$) and OS-Fuzzy-ELM/SAFIS components ($\delta_{e-right-f}$) of the control signals achieved by OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC control schemes.

Figure 8.9: Left and right elevator control signals under the left elevator stuck at -10deg and the left aileron stuck at 10deg

In the figure, the solid line represents the results from OS-Fuzzy-ELM aided BTFC and the dot line represents the results from SAFIS aided BTFC. As observed from Figure 8.9 control signals achieved from OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC are very similar.

The left and right aileron signals are given in Figure 8.10 and they are very similar.

Figure 8.10: Left and right aileron control signals under the left elevator stuck at -10deg and the left aileron stuck at 10deg

The fault tolerant capabilities of OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC for the whole range of deflections are given in Figure 8.11. The edge of the fault tolerance envelopes achieved by OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC is given by the solid line and dashdot line. In their respective fault tolerant regions each point which indicates a successful landing meeting the touch-

down pillbox requirements is represented by $\diamond$ and $\circ$.

Figure 8.11: Failure tolerance under left elevator and left aileron stuck conditions

From the figure, it can be seen that OS-Fuzzy-ELM aided BTFC is able to meet the pillbox requirements in a slightly wider range of deflections than SAFIS aided BTFC and their fault tolerant envelopes largely overlap.

8.2.2 Left Elevator and Right Aileron Stuck at Different Deflections

The fault tolerant capabilities of OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC for the whole range of deflections are given in Figure 8.12. The edge of the fault tolerance envelopes achieved by OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC is given by the solid line and dashdot line. In their respective fault

tolerant regions each point which indicates a successful landing meeting the touch-down pillbox requirements is represented by $\diamond$ and $\circ$.

Figure 8.12: Failure tolerance under left elevator and right aileron stuck conditions

From the figure, it can be seen that OS-Fuzzy-ELM aided BTFC is able to meet the pillbox requirements in a slightly wider range of deflections than SAFIS aided BTFC and its fault tolerant envelope has much overlap with that of SAFIS aided BTFC.

To summarize in this chapter the proposed OS-Fuzzy-ELM and SAFIS algorithms are compared by means of the aircraft autoland fault-tolerant control problem under actuator failures and severe winds. The following points are achieved by comparing OS-Fuzzy-ELM with SAFIS.

- (1) OS-Fuzzy-ELM can achieve similar or slightly better fault tolerant envelope for single failure surface and two failure surfaces compared with SAFIS.
- (2) In SAFIS only Gaussian membership function can be utilized to handle the problem while in OS-Fuzzy-ELM other membership functions besides Gaussian membership function can be utilized to handle the problem. Here, cauchy form membership function is utilized to verify this.
- (3) In OS-Fuzzy-ELM the number of rules is fixed whereas in SAFIS the number of rules is determined online. This may lead to a compact system.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

In this thesis, we have described an in-depth investigation on the development and application of efficient sequential fuzzy-neural system learning algorithms for aircraft autoland fault-tolerant control under actuator failures and severe winds. The conclusions from the study are summarized below:

- A sequential fuzzy inference system called SAFIS has been developed to automatically construct a FIS using the training data during learning. Specifically SAFIS algorithm implements the structure identification and parameter adjustment for a FIS using the concepts from GAP-RBF algorithm. SAFIS algorithm utilizes the influence of a fuzzy rule to add and remove the fuzzy rules during learning. At the same time the SAFIS algorithm utilizes the EKF to update the parameters of the nearest rule instead of all the rules without losing the approximation performance.

Its performance has been evaluated on benchmark problems which include two nonlinear system identification problems and the Mackey-Glass time series prediction problem. The simulation results from the benchmark problems indicate that compared with other algorithms, SAFIS produces similar

or better testing accuracies with lesser number of rules.

- The proposed SAFIS algorithm has been utilized as the fuzzy controller to aid the existing BTFC controller based on the feedback-error-learning scheme to solve the aircraft autoland fault-tolerant control problem under actuator failures and severe winds. When SAFIS is utilized as the fuzzy controller to aid the existing conventional BTFC controller, SAFIS not only learns online but also automatically determines the number of fuzzy rules during learning. The proposed SAFIS aided BTFC control scheme has been evaluated with a single fault of either aileron or elevator stuck at certain deflections and also double fault cases where one aileron and one elevator at the same or opposite direction are stuck at different deflections. Based on the failure scenarios the performance of the proposed SAFIS aided BTFC scheme is compared with EMRAN aided BTFC and BTFC controllers. From the study, the following conclusions can be summarized.
 - For single surface failure cases, the BTFC controller is unable to meet the touchdown dispersions under elevator failures and is able to meet the dispersions under aileron stuck failures only for small deflections (-8 and 4 degrees for aileron). The EMRAN aided BTFC controller enlarges the fault tolerant envelope (-12 to 12 degrees for elevator and -8 to 20 degrees for aileron) for the same case of single surface failure. However, in the fault tolerant envelopes achieved by EMRAN aided BTFC there are some “gaps” where the touchdown pillbox is not met under those stuck degrees (2 degree for aileron and -8, 0 and 8 degrees for elevator). The proposed SAFIS aided BTFC controller achieves not only a continuous fault tolerant range but also a larger fault tolerant envelope (-12 to 18 degrees for elevator and -14 to 14 degrees for aileron) for single elevator and aileron failures compared with BTFC controller and EMRAN aided BTFC controller.
 - For two surface failure cases like the combined elevator and aileron

failures, the fault tolerant envelope achieved by SAFIS aided BTFC controller is enlarged and furthermore there are no “gaps” in the whole region compared with BTFC controller and EMRAN aided BTFC controller whose regions have “gaps” indicating their inability to meet the tight pillbox requirements at those stuck deflections.

- Compared with EMRAN aided BTFC, SAFIS aided BTFC has a smooth rule generation process and requires lesser number of fuzzy rules during learning.
- An Online Sequential Fuzzy Extreme Learning Machine (OS-Fuzzy-ELM) has been developed based on the ELM developed by Huang, *et al.* Its advantages are summarized as follows:
 - OS-Fuzzy-ELM can be applied for both the TSK and Mamdani fuzzy models for any bounded nonconstant piecewise continuous membership function and any type of T-norm fuzzy logic operation in a unified way.
 - In OS-Fuzzy-ELM the parameters in the membership functions are randomly assigned and then the consequent parameters are analytically determined.
 - OS-Fuzzy-ELM can learn the data sequentially in a one-by-one or chunk-by-chunk mode and then discard the data for which the training has already been done.

Furthermore the performance evaluation of OS-Fuzzy-ELM has been carried out on benchmark problems of nonlinear system identification, California Housing prediction and DNA pattern classification problem. The conclusions based on these problems are summarized as follows:

- Compared with other algorithms the proposed OS-Fuzzy-ELM produces similar or better accuracies. It also produces a significantly lower training time than all the other algorithms.

- Implemented in the chunk-by-chunk learning mode the results indicate that training and testing accuracies are similar for all the membership functions. However, the training time for 1-by-1 learning mode is higher than chunk-by-chunk mode.
- As to another commonly used ‘Min’ T-norm the results using different implementation modes and membership functions are similar. As to the comparison between Product T-norm with Min T-norm, their results are also similar.
- As far as TSK and Mamdani fuzzy models based on OS-Fuzzy-ELM are concerned, Mamdani fuzzy model requires more fuzzy rules than TSK fuzzy model when similar training and testing accuracies are achieved. However, when the number of the input variables is large, the training time taken by Mamdani model is much less than that by TSK although the required fuzzy rules are more.
- The proposed OS-Fuzzy-ELM algorithm also has been utilized as the fuzzy controller to solve the aircraft autoland fault-tolerant control problem by using its online adaptation capability to aid the existing BTFC controller based on feedback-error-learning scheme. Based on the same failure scenarios used by SAFIS, the OS-Fuzzy-ELM aided BTFC scheme is also compared with EMRAN aided BTFC and BTFC controllers. By the simulation study, the following conclusions are achieved.
 - The proposed OS-Fuzzy-ELM aided BTFC controller achieves continuous and larger fault tolerant envelopes (-12 to 18 degrees for elevator and -20 to 9 degrees for aileron) for single elevator and aileron failures compared with BTFC controller (no points for elevator and -8 and 4 degrees for aileron) and EMRAN aided BTFC controller (-12 to 12 degrees for elevator except -8, 0 and 8 degrees and -8 to 20 degrees for aileron except 2 degree).
 - For two surface failure cases like the combined elevator and aileron fail-

ures, OS-Fuzzy-ELM aided BTFC controller obtains a larger fault tolerant envelope and furthermore there are no “gaps” in the whole region compared with BTFC controller and EMRAN aided BTFC controller whose regions have “gaps” indicating their inability to meet the tight pillbox requirements at those stuck deflections.

- Compared with EMRAN, OS-Fuzzy-ELM produces lesser deviations for the velocity (V_T), sideslip angle (β).
- The performance of the proposed OS-Fuzzy-ELM and SAFIS algorithms have also been compared on the aircraft autolanding fault-tolerant control problem. In summary,
 - OS-Fuzzy-ELM can achieve similar or slightly better fault tolerant envelope for single failure surface and two failure surfaces compared with SAFIS.
 - In SAFIS only Gaussian membership function can be utilized to handle the problem while in OS-Fuzzy-ELM other membership functions besides Gaussian membership function can be utilized to handle the problem. Here, cauchy form membership function is utilized to verify this.
 - In OS-Fuzzy-ELM the number of rules is fixed whereas in SAFIS the number of rules is determined online. This may lead to a compact system.
- Also a comparison between the fault-tolerant control scheme proposed in the thesis with the existing state-of-the-art fault tolerant schemes used in the aircraft autolanding problems is given below:
 - Based on the review for other aircraft autolanding fault-tolerant schemes described in Chapter 4, it is easy to note that most of these work consider only wind disturbances and not actuator failures as considered in this thesis.

- In [77], the authors have proposed a reliable H_2 controller that can tolerate actuator stuck faults and wind disturbances simultaneously. The approach used the polytopic fault systems to model the aircraft with actuator stuck faults and was verified based on the same autoland problem considered in this thesis. As analysed in [77], the polytopic fault system can accommodate only single actuator stuck fault. Even here, the failure range for a single aileron is $[-20, +20]$ and that is also satisfying the lateral error of ± 10 m. For the case of single elevator stuck fault, it is not able to handle and the system becomes unstable.
- Compared with the fault-tolerant control scheme in [77], the proposed fault-tolerant control schemes in this thesis achieve better failure tolerance performance. Firstly the proposed fault-tolerant control method in this thesis can not only tolerate the single actuator stuck faults but also double actuator faults. Besides a larger single elevator failure tolerance range is achieved and by enlarging the lateral error within 10 m instead of within 5 m considered in the thesis, the single aileron tolerance failure can be enlarged to the range of $[-20, +20]$.

9.2 Recommendations for Future Work

Possible areas of future work that emerge from this thesis include:

- In SAFIS, the calculation for the influence of a fuzzy rule is based on the radial basis function with the same width of the Gaussian memberships for all input variables in a rule and the constant consequent parameters. In the future work, an improved way to calculate the influence of the fuzzy rule can be developed for extending SAFIS to be suitable to linear consequent parameters and also ellipsoidal basis function with variable width of the Gaussian membership functions for all input variables in a rule. Also we can develop other update algorithms to tune the parameters in the fuzzy rules

so that the system performance is further improved.

- In OS-Fuzzy-ELM algorithm, the number of fuzzy rules is fixed a priori and can not be determined automatically during learning. In the future work we can further develop incremental OS-Fuzzy-ELM algorithm where the number of fuzzy rules is determined according to the learning process without selecting the number of fuzzy rules before hand.
- When OS-Fuzzy-ELM is utilized for aircraft autolaning fault-tolerant control problem, Only Cacuchy form membership function is utilized to achieve the fault-tolerant envelops. Future work can be undertaken by using other membership functions in the problem and based on this a comparison between their fault-tolerant envelops can be made.
- Recently H_∞ has been used as the baseline controller for the aircraft autolanding application to minimize the effect of the wind disturbance without considering the trajectory optimization [74]. In the future work, a mixed H_2/H_∞ control method instead of H_∞ can be used as the baseline controller together with the proposed fuzzy-neural algorithms for considering the trajectory optimization and wind disturbance simultaneously.
- The proposed algorithms are mainly employed in the fault tolerant control under actuator failures. In the future work they can be evaluated by the Aviation Safety (AvS) program including Intelligent Vehicle Health Monitoring (IVHM) [1,8] and Integrated Resilient Aircraft Control (IRAC) [28,33]. The AvS focuses on developing tools and technologies to enable multifold increase in aviation safety. The objective of the IVHM is to develop methods for reliably detecting and isolating faults in sensors so as to monitor and manage the structural status and operation of space transportation systems while IRAC is to investigate control concepts and architectures that will enable effective use of the propulsion system as an actuator for flight control in the presence of damage to aircraft or flight control surfaces.

Appendix A

A.1 Graphs for the Membership Functions

The appendix presents the graphs for the eight membership functions described in Chapter 2. The example for each membership function is given in Figure A.1.

(c)

(d)

(e)

(f)

Figure A.1: Graph Examples of Eight Membership Functions: a) Trimf; b) Trapmf; c) Gaussmf; d) Gauss2mf; e) Cauchymf; f) Pimf; g) Dsigmf; h) Psigmf.

Appendix B

B.1 Simulation Block of the High Performance Fighter Aircraft

The appendix presents the simulation block of the high performance aircraft model described in Chapter 4. Figure B.1 shows the simulation block details constructed using Simulink blocks in the Matlab environment. As observed in the figure, the dynamic equations of aircraft consisting of the force block to the auxiliary block as in equation (4.2) to equation (4.6) are included in the block. Since the calculations of all the aerodynamic coefficients ($C_{x,t}$, $C_{y,t}$, etc.) are based on the experiment data which are listed in tabular forms, a lookup-table program is included in the block.

Figure B.1: The high performance fighter aircraft simulation block

Appendix C

C.1 Data of the Lateral and Longitudinal Aircraft Dynamics for BTFC Controller Design

The appendix presents the data for the lateral and longitudinal aircraft dynamics which are used in designing the BTFC controller and described in the state space.

The corresponding A, B, C, D matrices for the lateral dynamics are as follows:

$$A = \begin{bmatrix} -0.1487 & 16.2718 & -80.6519 & 9.6216 & 0 & 0 \\ -0.1708 & -1.7533 & 0.8792 & 0 & 0 & 0 \\ 0.0227 & -0.0482 & -0.2424 & 0 & 0 & 0 \\ 0 & 1 & 0.1978 & 0 & 0 & 0 \\ 0 & 0 & 1.0194 & 0 & 0 & 0 \\ 1 & 0 & 0 & -16.0768 & 82.8632 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.0024 & 0.0336 \\ -0.1684 & 0.0340 \\ -0.0014 & -0.0169 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

C.1 Data of the Lateral and Longitudinal Aircraft Dynamics for BTFC Controller Design 226

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -0.0152 & 0.0195 & 0.0666 & 0 & 0 & 0 \\ 0.0121 & 0 & 0 & -0.1940 & 1 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.0002 & 0.0034 \\ 0 & 0 \end{bmatrix}$$

The corresponding A, B, C, D matrices for the longitudinal dynamics are given as,

$$A = \begin{bmatrix} -0.0183 & 0.1023 & -15.3342 & -9.6192 & -0.0002 \\ -0.1060 & -0.6485 & 73.6202 & -1.9024 & 0.0009 \\ -0.0025 & 0.0098 & -0.6491 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0.1940 & -0.9810 & 0 & 82.8421 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.0049 & 7.1522 \\ -0.1359 & 0 \\ -0.0598 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.0023 & -0.0118 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0.9810 & 0.1938 & 0 & 0 & 0 \end{bmatrix}$$

C.1 Data of the Lateral and Longitudinal Aircraft Dynamics for BTFC Controller Design **227**

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Appendix D

This appendix presents the simulation results achieved from OS-Fuzzy-ELM for other membership functions which include Psigmf, Dsigmf, Gauss2mf, Trapmf and Pimf for parameter determination, chunk-by-chunk mode, different T-norm operators and fuzzy models.

D.1 Results for Parameter Determination

Table D.1: Effect of parameter $\tilde{N}$ on system performance (training and testing accuracies) for TSK model

Membership Function Types	# rules $\tilde{N}$	Training (RMSE)	Testing (RMSE)
Psigmf	5	0.6287	0.4889
	10	0.2981	0.2622
	15	0.1813	0.1325
	20	0.1482	0.0793
	25	0.1285	0.0482
	30	0.1206	0.0352
	35	0.1170	0.0284
	40	0.1151	0.0260
	45	0.1140	0.0254
	50	0.1131	0.0251
Dsigmf	5	0.6133	0.4334
	10	0.3325	0.2768
	15	0.2062	0.1628
	20	0.1605	0.1044

Continued on next page

Table D.1: (continued)

Membership Function Types	# rules $\tilde{N}$	Training (RMSE)	Testing (RMSE)
	25	0.1354	0.0608
	30	0.1271	0.0477
	35	0.1219	0.0353
	40	0.1191	0.0309
	45	0.1170	0.0281
	50	0.1160	0.0275
Gauss2mf	5	0.5733	0.4600
	10	0.2897	0.2631
	15	0.1878	0.1343
	20	0.1423	0.0688
	25	0.1268	0.0483
	30	0.1212	0.0431
	35	0.1171	0.0360
	40	0.1151	0.0338
	45	0.1138	0.0307
	50	0.1129	0.0294
Trapmf	5	0.6890	0.5329
	10	0.3703	0.3312
	15	0.2485	0.2157
	20	0.1728	0.1128
	25	0.1475	0.0782
	30	0.1353	0.0467
	35	0.1251	0.0409
	40	0.1199	0.0366
	45	0.1168	0.0324
	50	0.1146	0.0306
Pimf	5	0.7117	0.5118
	10	0.3696	0.3187
	15	0.2306	0.1885
	20	0.1647	0.1071
	25	0.1431	0.0764
	30	0.1206	0.0363
	35	0.1173	0.0338
	40	0.1153	0.0323
	45	0.1136	0.0312
	50	0.1127	0.0292

Table D.2: Effect of parameter $\tilde{N}$ on system performance (training and testing accuracies) for Mamdani model

Membership Function Types	Chunk Sizes	Training (RMSE)	Testing (RMSE)
Psigmf	20	0.6006	0.4523
	30	0.3546	0.2887
	40	0.2666	0.2198
	50	0.2032	0.1530
	60	0.1747	0.1142
	70	0.1531	0.0838
	80	0.1433	0.0695
	90	0.1347	0.0541
	100	0.1291	0.0456
	110	0.1252	0.0412
Dsigmf	20	0.5459	0.3939
	30	0.3466	0.2974
	40	0.2487	0.2082
	50	0.2328	0.1920
	60	0.2118	0.1707
	70	0.1882	0.1367
	80	0.1579	0.1010
	90	0.1398	0.0718
	100	0.1343	0.0564
	110	0.1298	0.0471
Gauss2mf	20	0.5320	0.4294
	30	0.3167	0.2757
	40	0.2364	0.1833
	50	0.1863	0.1228
	60	0.1592	0.0903
	70	0.1401	0.0680
	80	0.1312	0.0560
	90	0.1257	0.0500
	100	0.1215	0.0450
	110	0.1191	0.0418
Trapmf	20	0.5609	0.4415
	30	0.4206	0.3725
	40	0.3196	0.2591
	50	0.2372	0.1927
	60	0.1796	0.1109
	70	0.1550	0.0896
	80	0.1463	0.0779
	90	0.1352	0.0609

Continued on next page

Table D.2: (continued)

Membership Function Types	# rules $\tilde{N}$	Training (RMSE)	Testing (RMSE)
Pimf	100	0.1296	0.0493
	110	0.1237	0.0447
	20	0.5527	0.4219
	30	0.3408	0.2920
	40	0.2394	0.1858
	50	0.1961	0.1362
	60	0.1695	0.1056
	70	0.1517	0.0844
	80	0.1426	0.0747
	90	0.1330	0.0654
	100	0.1276	0.0508
	110	0.1192	0.0464

D.2 Results for Chunk-by-Chunk Mode

Table D.3: Performance comparison implemented in different chunk sizes for nonlinear system identification

Membership Function Types	Chunk Sizes	Training (RMSE)	Testing (RMSE)	# rules	Training Time (Secs)
Psigmf	batch	0.1271	0.0525	30	1.2438
	1-by-1	0.1271	0.0525	30	14.750
	20-by-20	0.1271	0.0525	30	1.3195
	[10, 30]	0.1271	0.0533	30	1.3896
Dsigmf	batch	0.1258	0.0513	30	1.2738
	1-by-1	0.1260	0.0519	30	14.775
	20-by-20	0.1271	0.0525	30	1.3195
	[10, 30]	0.1281	0.0518	30	1.4223
Gauss2mf	batch	0.1346	0.0533	30	1.1704
	1-by-1	0.1319	0.0534	30	14.942
	20-by-20	0.1320	0.0532	30	1.2710
	[10, 30]	0.1345	0.0537	30	1.3432
Trapmf	batch	0.1308	0.0528	30	1.1351
	1-by-1	0.1308	0.0528	30	14.969
	20-by-20	0.1308	0.0528	30	1.2166
	[10, 30]	0.1305	0.0527	30	1.3252

Continued on next page

Table D.3: (continued)

Membership Function Types	Chunk Sizes	Training (RMSE)	Testing (RMSE)	# rules	Training Time (Secs)
Pimf	batch	0.1328	0.0548	30	1.1351
	1-by-1	0.1328	0.0548	30	15.029
	20-by-20	0.1328	0.0548	30	1.2331
	[10, 30]	0.1331	0.0562	30	1.3228

Table D.4: Performance comparison implemented in different chunk sizes for California housing prediction

Membership Function Types	Chunk Sizes	Training (RMSE)	Testing (RMSE)	# rules	Training Time (Secs)
Psigmf	batch	0.1308	0.1334	5	0.5200
	1-by-1	0.1308	0.1334	5	3.8778
	20-by-20	0.1309	0.1333	5	0.5891
	[10, 30]	0.1311	0.1338	5	0.6852
Dsigmf	batch	0.1303	0.1335	5	0.5506
	1-by-1	0.1313	0.1340	5	3.9187
	20-by-20	0.1315	0.1342	5	0.6385
	[10, 30]	0.1318	0.1347	5	0.7219
Gauss2mf	batch	0.1314	0.1336	5	0.4475
	1-by-1	0.1316	0.1339	5	3.9991
	20-by-20	0.1315	0.1342	5	0.6385
	[10, 30]	0.1310	0.1337	5	0.6146
Trapmf	batch	0.1305	0.1333	5	0.4322
	1-by-1	0.1312	0.1328	5	3.9528
	20-by-20	0.1309	0.1331	5	0.4991
	[10, 30]	0.1305	0.1330	5	0.5943
Pimf	batch	0.1318	0.1350	5	0.4431
	1-by-1	0.1312	0.1348	5	4.0578
	20-by-20	0.1316	0.1345	5	0.6097
	[10, 30]	0.1316	0.1345	5	0.6097

Table D.5: Performance comparison implemented in different chunk sizes for DNA classification

Membership Function Types	Chunk Sizes	Training (%)	Testing (%)	# rules	Training Time (Secs)
Psigmf	batch	96.98	94.38	5	25.671
	1-by-1	96.55	94.94	5	2537.2
	20-by-20	96.85	94.34	5	128.99
	[10, 30]	96.75	94.35	5	123.35
Dsigmf	batch	97.11	94.28	5	25.025
	1-by-1	96.94	94.50	5	2468.0
	20-by-20	96.45	94.15	5	129.08
	[10, 30]	96.49	94.13	5	126.40
Gauss2mf	batch	96.80	94.37	5	23.274
	1-by-1	96.60	94.25	5	2182.8
	20-by-20	96.37	94.19	5	128.14
	[10, 30]	96.56	94.16	5	123.88
Trapmf	batch	96.85	94.31	5	22.766
	1-by-1	96.79	94.26	5	2180.9
	20-by-20	96.68	94.30	5	122.38
	[10, 30]	96.68	94.30	5	122.38
Pimf	batch	96.85	94.26	5	22.431
	1-by-1	96.84	94.77	5	2185.0
	20-by-20	96.74	94.23	5	124.45
	[10, 30]	96.74	94.23	5	124.45

D.3 Results for Different T-norm Operators

Table D.6: Performance comparison between Product and Min T-norm operators for nonlinear system identification

Membership Function Types	Chunk Sizes	T-norm Types	Training (RMSE)	Testing (RMSE)	# rules	Training Time (Secs)
Psigmf	batch	Product	0.1271	0.0525	30	1.2438
		Min	0.1410	0.0689	30	1.7109
	1-by-1	Product	0.1271	0.0525	30	14.750
		Min	0.1440	0.0616	30	15.201
	20-by-20	Product	0.1271	0.0525	30	1.3195
		Min	0.1449	0.0649	30	1.8054
	[10, 30]	Product	0.1271	0.0533	30	1.3896
		Min				

Continued on next page

Table D.6: (continued)

Membership Function Types	Chunk Sizes	T-norm Types	Training (RMSE)	Testing (RMSE)	# rules	Training Time (Secs)
Dsigmf	batch	Min	0.1431	0.0667	30	1.9198
		Product	0.1258	0.0513	30	1.2738
	1-by-1	Min	0.1408	0.0674	30	1.7544
		Product	0.1260	0.0519	30	14.775
	20-by-20	Min	0.1420	0.0683	30	15.167
		Product	0.1271	0.0525	30	1.3195
	[10, 30]	Min	0.1427	0.0695	30	1.8400
		Product	0.1281	0.0518	30	1.4223
		Min	0.1405	0.0685	30	1.9404
		Product	0.1346	0.0533	30	1.1704
Gauss2mf	batch	Min	0.1422	0.0614	30	1.6522
		Product	0.1319	0.0534	30	14.942
	1-by-1	Min	0.1424	0.0649	30	15.392
		Product	0.1320	0.0532	30	1.2710
	20-by-20	Min	0.1430	0.0660	30	1.7496
		Product	0.1345	0.0537	30	1.3432
	[10, 30]	Min	0.1440	0.0634	30	1.8243
		Product	0.1308	0.0528	30	1.1351
		Min	0.1408	0.0637	30	1.5503
		Product	0.1308	0.0528	30	14.969
Trapmf	batch	Min	0.1403	0.0616	30	15.264
		Product	0.1308	0.0528	30	1.2166
	1-by-1	Min	0.1404	0.0679	30	1.7150
		Product	0.1308	0.0528	30	1.2166
	20-by-20	Min	0.1404	0.0679	30	1.7150
		Product	0.1308	0.0528	30	1.2166
	[10, 30]	Min	0.1404	0.0679	30	1.7150
		Product	0.1308	0.0528	30	1.2166
		Min	0.1404	0.0679	30	1.7150
		Product	0.1308	0.0528	30	1.2166
Pimf	batch	Min	0.1404	0.0679	30	1.7150
		Product	0.1308	0.0528	30	1.2166
	1-by-1	Min	0.1404	0.0679	30	1.7150
		Product	0.1308	0.0528	30	1.2166
	20-by-20	Min	0.1404	0.0679	30	1.7150
		Product	0.1308	0.0528	30	1.2166
	[10, 30]	Min	0.1404	0.0679	30	1.7150
		Product	0.1308	0.0528	30	1.2166
		Min	0.1404	0.0679	30	1.7150
		Product	0.1308	0.0528	30	1.2166

Table D.7: Performance comparison between Product and Min T-norm operators for California housing prediction

Membership Function Types	Chunk Sizes	T-norm Types	Training (RMSE)	Testing (RMSE)	# rules	Training Time (Secs)
Psigmf	batch	Product	0.1308	0.1334	5	0.5200
		Min	0.1341	0.1379	5	0.6893
	1-by-1	Product	0.1308	0.1334	5	3.8778
		Min	0.1341	0.1379	5	3.9413
	20-by-20	Product	0.1309	0.1333	5	0.5891
		Min	0.1341	0.1379	5	0.7506
	[10, 30]	Product	0.1311	0.1338	5	0.6852
		Min	0.1337	0.1359	5	0.8581
	batch	Product	0.1303	0.1335	5	0.5506
		Min	0.1338	0.1359	5	0.7393
	1-by-1	Product	0.1313	0.1340	5	3.9187
		Min	0.1338	0.1359	5	4.0548
Dsigmf	20-by-20	Product	0.1315	0.1342	5	0.6385
		Min	0.1338	0.1360	5	0.7872
	[10, 30]	Product	0.1318	0.1347	5	0.7219
		Min	0.1338	0.1391	5	0.9025
Gauss2mf	batch	Product	0.1314	0.1336	5	0.4475
		Min	0.1333	0.1362	5	0.6256
	1-by-1	Product	0.1316	0.1339	5	3.9991
		Min	0.1333	0.1362	5	4.0535
	20-by-20	Product	0.1315	0.1342	5	0.6385
		Min	0.1333	0.1360	5	0.6783
	[10, 30]	Product	0.1310	0.1337	5	0.6146
		Min	0.1336	0.1367	5	0.8047
Trapmf	batch	Product	0.1305	0.1333	5	0.4322
		Min	0.1341	0.1392	5	0.6040
	1-by-1	Product	0.1312	0.1328	5	3.9528
		Min	0.1341	0.1392	5	3.9988
	20-by-20	Product	0.1309	0.1331	5	0.4991
		Min	0.1341	0.1393	5	0.6698
	[10, 30]	Product	0.1305	0.1330	5	0.5943
		Min	0.1341	0.1396	5	0.7939
Pimf	batch	Product	0.1318	0.1350	5	0.4431
		Min	0.1336	0.1368	5	0.6153
	1-by-1	Product	0.1312	0.1348	5	4.0578
		Min	0.1336	0.1367	5	4.1593
	20-by-20	Product	0.1316	0.1345	5	0.6097
		Min	0.1336	0.1366	5	0.6829
	[10, 30]	Product	0.1316	0.1345	5	0.6097

Continued on next page

Table D.7: (continued)

Membership Function Types	Chunk Sizes	T-norm Types	Training (RMSE)	Testing (RMSE)	# rules	Training Time (Secs)
		Min	0.1342	0.1361	5	0.8222

Table D.8: Performance comparison between Product and Min T-norm operators for DNA classification

Membership Function Types	Chunk Sizes	T-norm Types	Training (%)	Testing (%)	# rules	Training Time (Secs)
Psgmf	batch	Product	96.98	94.38	5	25.671
		Min	97.55	92.06	5	26.593
	1-by-1	Product	96.55	94.94	5	2537.2
		Min	97.07	92.66	5	2591.6
	20-by-20	Product	96.85	94.34	5	128.99
		Min	97.02	92.53	5	129.11
	[10, 30]	Product	96.75	94.35	5	123.35
		Min	97.76	92.96	5	127.93
Dsgmf	batch	Product	97.11	94.28	5	25.025
		Min	97.95	92.24	5	26.750
	1-by-1	Product	96.94	94.50	5	2468.0
		Min	97.23	92.32	5	2492.9
	20-by-20	Product	96.45	94.15	5	129.08
		Min	97.19	92.47	5	129.37
	[10, 30]	Product	96.49	94.13	5	126.40
		Min	97.93	92.24	5	126.87
Gauss2mf	batch	Product	96.80	94.37	5	23.274
		Min	97.35	92.34	5	23.872
	1-by-1	Product	96.60	94.25	5	2182.8
		Min	97.10	92.61	5	2196.3
	20-by-20	Product	96.37	94.19	5	128.14
		Min	97.29	92.72	5	128.14
	[10, 30]	Product	96.56	94.16	5	123.88
		Min	97.95	92.31	5	124.03
Trapmf	batch	Product	96.85	94.31	5	22.766
		Min	97.75	92.68	5	26.531
	1-by-1	Product	96.79	94.26	5	2180.9
		Min	97.35	92.32	5	2189.2
	20-by-20	Product	96.68	94.30	5	122.38
		Min	97.12	92.63	5	128.17
	[10, 30]	Product	96.68	94.30	5	122.38

Continued on next page

Table D.8: (continued)

Membership Function Types	Chunk Sizes	T-norm Types	Training (%)	Testing (%)	# rules	Training Time (Secs)
Pimf	batch	Min	97.97	92.04	5	124.65
		Product	96.85	94.26	5	22.431
	1-by-1	Min	97.40	92.17	5	24.050
		Product	96.84	94.77	5	2185.0
	20-by-20	Min	97.20	92.66	5	2188.1
		Product	96.74	94.23	5	124.45
	[10, 30]	Min	97.82	92.50	5	128.32
		Product	96.74	94.23	5	124.45
		Min	97.03	92.67	5	125.20

D.4 Results for Different Fuzzy Models

Table D.9: Performance comparison between TSK and Mamdani fuzzy models for nonlinear system identification

Membership Function Types	Chunk Sizes	Fuzzy Models	Training (RMSE)	Testing (RMSE)	# rules	Training Time (Secs)
Psigmf	batch	TSK	0.1271	0.0525	30	1.2438
		Mamdani	0.1317	0.0520	110	1.4110
	1-by-1	TSK	0.1271	0.0525	30	14.750
		Mamdani	0.1326	0.0526	110	13.051
	20-by-20	TSK	0.1271	0.0525	30	1.3195
		Mamdani	0.1331	0.0538	110	1.5876
	[10, 30]	TSK	0.1271	0.0533	30	1.3896
		Mamdani	0.1329	0.0536	110	1.6544
Dsigmf	batch	TSK	0.1258	0.0513	30	1.2738
		Mamdani	0.1270	0.0500	110	1.5728
	1-by-1	TSK	0.1260	0.0519	30	14.775
		Mamdani	0.1279	0.0506	110	13.175
	20-by-20	TSK	0.1271	0.0525	30	1.3195
		Mamdani	0.1331	0.0538	110	1.5876
	[10, 30]	TSK	0.1281	0.0518	30	1.4223
		Mamdani	0.1273	0.0504	110	1.7716
Gauss2mf	batch	TSK	0.1346	0.0533	30	1.1704
		Mamdani	0.1304	0.0515	110	1.2012
	1-by-1	TSK	0.1319	0.0534	30	14.942

Continued on next page

Table D.9: (continued)

Membership Function Types	Chunk Sizes	Fuzzy Models	Training (RMSE)	Testing (RMSE)	# rules	Training Time (Secs)
Trapmf	20-by-20	Mamdani	0.1318	0.0509	110	14.378
		TSK	0.1320	0.0532	30	1.2710
	[10, 30]	Mamdani	0.1350	0.0506	110	1.3525
		TSK	0.1345	0.0537	30	1.3432
	batch	Mamdani	0.1343	0.0521	110	1.4065
		TSK	0.1308	0.0528	30	1.1351
	1-by-1	Mamdani	0.1324	0.0529	110	1.1022
		TSK	0.1308	0.0528	30	14.969
	20-by-20	Mamdani	0.1336	0.0537	110	13.780
		TSK	0.1308	0.0528	30	1.2166
	[10, 30]	Mamdani	0.1334	0.0540	110	1.2475
		TSK	0.1305	0.0527	30	1.3252
Pimf	batch	Mamdani	0.1342	0.0544	110	1.2630
		TSK	0.1328	0.0548	30	1.1351
	1-by-1	Mamdani	0.1283	0.0546	110	1.1359
		TSK	0.1328	0.0548	30	15.029
	20-by-20	Mamdani	0.1332	0.0553	110	13.683
		TSK	0.1328	0.0548	30	1.2331
	[10, 30]	Mamdani	0.1332	0.0555	110	1.2693
		TSK	0.1331	0.0562	30	1.3228
		Mamdani	0.1321	0.0551	110	1.3260
		TSK				

Table D.10: Performance comparison between TSK and Mamdani fuzzy models for California housing prediction

Membership Function Types	Chunk Sizes	Fuzzy Models	Training (RMSE)	Testing (RMSE)	# rules	Training Time (Secs)
Psigmf	batch	TSK	0.1308	0.1334	5	0.5200
		Mamdani	0.1339	0.1352	35	0.8522
	1-by-1	TSK	0.1308	0.1334	5	3.8778
		Mamdani	0.1355	0.1365	35	3.8522
	20-by-20	TSK	0.1309	0.1333	5	0.5891
		Mamdani	0.1366	0.1376	35	0.9285
	[10, 30]	TSK	0.1311	0.1338	5	0.6852
		Mamdani	0.1361	0.1369	35	0.9394
Dsigmf	batch	TSK	0.1303	0.1335	5	0.5506
		Mamdani	0.1337	0.1344	35	0.8985
	1-by-1	TSK	0.1313	0.1340	5	3.9187

Continued on next page

Table D.10: (continued)

Membership Function Types	Chunk Sizes	Fuzzy Models	Training (RMSE)	Testing (RMSE)	# rules	Training Time (Secs)
Gauss2mf	20-by-20	Mamdani	0.1348	0.1364	35	3.9010
		TSK	0.1315	0.1342	5	0.6385
	[10, 30]	Mamdani	0.1357	0.1364	35	0.9335
		TSK	0.1318	0.1347	5	0.7219
	batch	Mamdani	0.1363	0.1366	35	0.9612
		TSK	0.1314	0.1336	5	0.4475
	1-by-1	Mamdani	0.1329	0.1344	35	0.7180
		TSK	0.1316	0.1339	5	3.9991
	20-by-20	Mamdani	0.1331	0.1346	35	3.6613
		TSK	0.1315	0.1342	5	0.6385
	[10, 30]	Mamdani	0.1332	0.1347	35	0.8077
		TSK	0.1310	0.1337	5	0.6146
Trapmf	batch	Mamdani	0.1338	0.1344	35	0.9196
		TSK	0.1305	0.1333	5	0.4322
	1-by-1	Mamdani	0.1340	0.1349	35	0.5928
		TSK	0.1312	0.1328	5	3.9528
	20-by-20	Mamdani	0.1351	0.1361	35	3.7247
		TSK	0.1309	0.1331	5	0.4991
	[10, 30]	Mamdani	0.1345	0.1352	35	0.7344
		TSK	0.1305	0.1330	5	0.5943
	batch	Mamdani	0.1344	0.1350	35	0.8255
		TSK	0.1318	0.1350	5	0.4431
	1-by-1	Mamdani	0.1349	0.1363	35	0.7350
		TSK	0.1312	0.1348	5	4.0578
Pimf	20-by-20	Mamdani	0.1340	0.1356	35	3.5285
		TSK	0.1316	0.1345	5	0.6097
	[10, 30]	Mamdani	0.1355	0.1363	35	0.8478
		TSK	0.1316	0.1345	5	0.6097
	batch	Mamdani	0.1349	0.1363	35	0.9350
		TSK	0.1316	0.1345	5	0.6097
	1-by-1	Mamdani	0.1355	0.1363	35	0.8478
		TSK	0.1316	0.1345	5	0.6097

Table D.11: Performance comparison between TSK and Mamdani fuzzy models for DNA classification

Membership Function Types	Chunk Sizes	Fuzzy Models	Training (%)	Testing (%)	# rules	Training Time (Secs)
Psigmf	batch	TSK	96.98	94.38	5	25.671
		Mamdani	96.63	94.36	200	23.056
	1-by-1	TSK	96.55	94.94	5	2537.2

Continued on next page

Table D.11: (continued)

Membership Function Types	Chunk Sizes	Fuzzy Models	Training (%)	Testing (%)	# rules	Training Time (Secs)
	20-by-20	Mamdani	96.61	94.30	200	66.900
		TSK	96.85	94.34	5	128.99
	[10, 30]	Mamdani	96.27	94.26	200	48.136
		TSK	96.75	94.35	5	123.35
		Mamdani	96.30	94.29	200	27.652
		TSK	97.11	94.28	5	25.025
	batch	Mamdani	96.67	94.35	200	23.598
		TSK	96.94	94.50	5	2468.0
Dsigmf	1-by-1	Mamdani	96.46	94.27	200	71.946
		TSK	96.45	94.15	5	129.08
	20-by-20	Mamdani	96.38	94.24	200	58.531
		TSK	96.49	94.13	5	126.40
	[10, 30]	Mamdani	96.42	94.30	200	30.819
		TSK	96.80	94.37	5	23.274
	batch	Mamdani	96.74	94.51	200	12.213
		TSK	96.60	94.25	5	2182.8
Gauss2mf	1-by-1	Mamdani	96.67	94.30	200	56.052
		TSK	96.37	94.19	5	128.14
	20-by-20	Mamdani	96.69	94.54	200	25.822
		TSK	96.56	94.16	5	123.88
	[10, 30]	Mamdani	96.04	94.33	200	14.352
		TSK	96.85	94.31	5	22.766
	batch	Mamdani	96.70	94.41	200	10.464
		TSK	96.79	94.26	5	2180.9
Trapmf	1-by-1	Mamdani	96.56	94.84	200	53.775
		TSK	96.68	94.30	5	122.38
	20-by-20	Mamdani	96.72	94.54	200	22.178
		TSK	96.68	94.30	5	122.38
	[10, 30]	Mamdani	96.57	94.31	200	12.420
		TSK	96.85	94.26	5	22.431
	batch	Mamdani	96.85	94.42	200	12.703
		TSK	96.84	94.77	5	2185.0
Pimf	1-by-1	Mamdani	96.64	94.26	200	56.626
		TSK	96.74	94.23	5	124.45
	20-by-20	Mamdani	96.60	94.29	200	26.877
		TSK	96.74	94.23	5	124.45
	[10, 30]	Mamdani	96.63	94.25	200	16.407
		TSK	96.74	94.23	5	124.45
	batch	Mamdani	96.85	94.26	5	22.431
		TSK	96.85	94.42	200	12.703

Author's Publications

Journal Paper

- (1) Hai-Jun Rong, N. Sundararajan, Guang-Bin Huang, and P. Saratchandran, "Sequential Adaptive Fuzzy Inference System (SAFIS) for nonlinear system identification and prediction", *Fuzzy Sets and Systems*, vol. 157, no. 9, pp. 1260-1275, 2006.
- (2) Hai-Jun Rong, N. Sundararajan, P. Saratchandran, and Guang-Bin Huang, "Adaptive Fuzzy Fault-Tolerant Controller for Aircraft Autolanding Under Failures", Accepted by *IEEE Transactions on Aerospace and Electronic Systems*.

Conference Paper

- (1) Guang-Bin Huang, Nan-Ying Liang, Hai-Jun Rong, P. Saratchandran, and N. Sundararajan, "On-line sequential extreme learning machine", in the *IASTED International Conference on Computational Intelligence (CI 2005)*, Calgary, Canada, July 4-6, 2005.
- (2) Hai-Jun Rong, Guang-Bin Huang, N. Sundararajan, and P. Saratchandran, "Fuzzy Fault Tolerant Controller for Actuator Failures During Aircraft Autolanding", in *2006 IEEE International Conference on Fuzzy Systems*, Vancouver, BC, Canada, July 16-21, 2006.

Bibliography

- [1] D. Abbott, B. Doyle, J. Dunlop, A. Farmer, M. Hedley, J. Herrmann, G. James, M. Johnson, B. Joshi, G. Poulton, D. Price, M. Prokopenko, T. Reda, D. Rees, D. Scott, P. Valencia, D. Ward, and J. Winter. Concepts for an integrated vehicle health monitoring system. In D. Thompson and D. Chimenti, editors, *Review of Progress in Quantitative Nondestructive Evaluation, American Institute of Physics Conference, 2003*, volume 22, pages 1606–1614, 2003.
- [2] M. Agarwal. A systematic classification of neural-network-based control. *IEEE Transactions on Control Systems Magazine*, 17(2):75–93, 1997.
- [3] F. Ahmed-Zaid, P. Ioannou, K. Gousman, and R. Rooney. Accommodation of failures in the F-16 aircraft using adaptive control. *IEEE Transactions on Control Systems Magazine*, 11(1):73–78, 1991.
- [4] A. Alonge and F. D’ppolito. Takeoff and landing robust control system for a tandem canard UAV. In *AIAA/CIRA 13th International Space Planes and Hypersonics Systems and Technologies Conference*. Capua, Italy, 2005.
- [5] P. Angelov and D. Filev. Simpl.eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models. In *The 14th IEEE International Conference on Fuzzy Systems*, pages 1068–1073, 2005.

- [6] P. Angelov, J. Victor, A. Dourado, and D. Filev. On-line evolution of Takagi-Sugeno fuzzy models. In *Proceedings of the 2nd IFAC Workshop on Advanced Fuzzy/Neural Control*, pages 67–72. Oulu, Finland, 2004.
- [7] P. P. Angelov and D. P. Filev. An approach to online identification of Takagi-Sugeno fuzzy models. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 34(1):484–498, 2004.
- [8] E. Baroth, W. Powers, J. Fox, B. Prosser, J. Pallix, K. Schweikard, and J. Zakrajsek. IVHM (Integrated Vehicle Health Management) techniques for future space vehicles. In *37th Joint Propulsion Conference & Exhibit*. Salt Lake City, Utah, 2001.
- [9] J. C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Plenum Press, New York, 1981.
- [10] A. Bhattacharya, A. G. Parlos, and A. F. Atiya. Video source traffic flow prediction using neural networks. In *Proceedings of the 46th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS'03)*, pages 33–38. Cairo, Egypt, 2003.
- [11] D. Bischoff, R. Duffy, and H. Kaufmant. Automatic control of adverse yaw in the landing environment using optimal control theory. *Journal of Aircraft*, 11(9):499–500, 1974.
- [12] C. Blake and C. Merz. UCI repository of machine learning databases. <http://www.ics.uci.edu/mllearn/MLRepository.html>, Department of Information and Computer Sciences, University of California, Irvine, USA, 1998.
- [13] M. Bodson and J. E. Groszkiewicz. Multivariable adaptive algorithms for reconfigurable flight control. *IEEE Transactions on Control Systems Technology*, 5(2):217–229, 1997.
- [14] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, (2):321–355, 1988.

- [15] D. J. Bugajki and D. F. Enns. Nonlinear control law with application to high angle-of-attack flight. *Journal of Guidance, Control, and Dynamics*, 15(3):761 – 767, 1992.
- [16] J. L. Castro and M. Delgado. Fuzzy systems with defuzzification are universal approximators. *IEEE Transactions on Systems, Man, and Cybernetic-Part B: Cybernetics*, 26(1):149–152, 1996.
- [17] P.-R. Chang and J.-T. Hu. Optimal nonlinear adaptive prediction and modeling of MPEG video in ATM networks using pipelined recurrent neural networks. *IEEE Journal on Selected Areas in Communications*, 15(6):1087–1100, 1997.
- [18] J. Chen and R. J. Patton. *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Kluwer Academic Publishers, 1998.
- [19] O. T.-C. Chen and B. J. Sheu. Optimization schemes for neural network training. In *IEEE World Congress on Computational Intelligence (WCCI'94)*, pages 817–822. Orlando Florida, USA, 1994.
- [20] S. Chen, S. A. Billings, and P. M. Grant. Nonlinear system identification using neural networks. *International Journal of Control*, 51:1191–1214, 1990.
- [21] S. L. Chiu. Selecting input variables for fuzzy models. *Journal of Intelligent and Fuzzy Systems*, 4:243–256, 1996.
- [22] 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*, 83:325–339, 1996.
- [23] E. K. P. Chong and S. H. Zak. *An Introduction to Optimization*. New York: JohnWiley, 2001.
- [24] F. Da and W. Song. Fuzzy neural networks for direct adaptive control. *IEEE Transactions on Industrial Electronics*, 50(3):507–513, 2003.

- [25] Y. Diao and K. M. Passino. Stable fault-tolerant adaptive fuzzy/neural control for a turbine engine. *IEEE Transactions on Control Systems Technology*, 9(3):494–509, 2001.
- [26] J. L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7:195–224, 1991.
- [27] A. Fujimori, M. Kurozumi, P. N. Nikiforuk, and M. M. Gupta. Flight control design of an automatic landing flight experiment vehicle. *Journal of Guidance, Control, and Dynamics*, 23(2):373–376, 2000.
- [28] S. Garg. Propulsion controls and diagnostics research at NASA Glenn. In *43rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*. Cincinnati, OH, 2007.
- [29] M. B. Ghalia and A. T. Alouani. Robust control design of an autoland system. In *Proceedings of the Twenty-Fifth Southeastern System Theory*, pages 248–252. Tuscaloosa, AL, 1993.
- [30] G. H. Golub and C. F. V. Loan. *Matrix Computations (3rd ed.)*. The Johns Hopkins University Press, 1996.
- [31] H. Gomi and M. Kawato. Neural network control for a closed-loop system using feedback-error-learning. *Neural Networks*, 6(7):933–946, 1992.
- [32] S. Gopal, B. Karthikeyan, and D. Kavitha. Partial discharge pattern classification using fuzzy expert system. In *Proceedings of the 2004 IEEE International Conference on Solid Dielectrics*, pages 653–656. Toulouse, France, 2004.
- [33] T.-H. Guo. Resilient propulsion control research for the NASA integrated resilient aircraft control (IRAC) project. In *2007 AIAA Conference and Exhibit*. Rohnert Park, California, 2007.

- [34] M. M. Gupta and D. H. Rao. On the principles of fuzzy neural networks. *Fuzzy Sets and Systems*, 61(1):1–18, 1994.
- [35] S. D. Handoko, K. C. Keong, O. Y. Soon, G. L. Zhang, and V. Brusic. Extreme learning machine for predicting HLA-peptide binding. *Lecture Notes in Computer Science*, 3973:716–721, 2006.
- [36] S. Haykin. *Neural networks: A comprehensive foundation*. New Jersey: Prentice Hall, 1999.
- [37] W. E. Holley. Wind modeling and lateral control for automatic landing. In *American Institute of Aeronautics and Astronautics, Guidance and Control Conference*. Boston, Mass., 1975.
- [38] C. R. Houck, J. A. Joines, and M. G. Kay. A genetic algorithm for function optimization: A matlab implementation. NCSU-IE TR 95-09, 1995.
- [39] G.-B. Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, 14(2):274–281, 2003.
- [40] G.-B. Huang and H. A. Babri. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. *IEEE Transactions on Neural Networks*, 9(1):224–229, 1998.
- [41] G.-B. Huang, L. Chen, and C.-K. Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17(4):879–892, 2006.
- [42] G.-B. Huang, P. Saratchandran, and N. Sundararajan. An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. *IEEE Transactions on Systems, Man, Cybernetics-Part B: Cybernetics*, 34(6):2284–2292, 2004.

- [43] G.-B. Huang, P. Saratchandran, and N. Sundararajan. A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. *IEEE Transactions on Neural Networks*, 16(1):57–67, 2005.
- [44] G.-B. Huang and C.-K. Siew. Extreme learning machine: RBF network case. In *Proceedings of the Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV 2004)*, pages 6–9. Kunming, China, 2004.
- [45] G.-B. Huang, Q.-Y. Zhu, K. Z. Mao, C.-K. Siew, P. Saratchandran, and N. Sundararajan. Can threshold networks be trained directly? *IEEE Transactions on Circuits and Systems II*, 53(3):187–191, 2006.
- [46] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70:489–501, 2006.
- [47] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Real-time learning capability of neural networks. *IEEE Transactions on Neural Networks*, 17(4):863–878, 2006.
- [48] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: A new learning scheme of feedforward neural networks. In *Proceedings of International Joint Conference on Neural Networks (IJCNN2004)*, volume 2, pages 985–990, Budapest, Hungary, 25-29 July, 2004.
- [49] K. J. Hunt, D. Sbarbaro, R. Zbikowski, and P. J. Gawthrop. Neural networks for control systems - a survey. *Automatica*, 28(6):1083–1112, 1992.
- [50] R. Iqdour and A. Zeroual. A rule based fuzzy model for the prediction of daily solar radiation. In *2004 IEEE International Conference on Industrial Technology (ICIT)*, pages 1482–1487. Hammamet, Tunisia, 2004.
- [51] J.-S. R. Jang. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3):665–685, 1993.

- [52] J.-S. R. Jang and C.-T. Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 4(1):156–159, 1993.
- [53] J.-S. R. Jang, C.-T. Sun, and E. Mizutani. *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence*. Prentice Hall, 1997.
- [54] J.-T. Jeng, C.-C. Chuang, and Y. C. Lee. Annealing robust nonlinear adaptive inverse control with FNNBSVR for magnetic bearing systems. In *Proceedings of 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 1276–1281. Kobe, Japan, 2003.
- [55] C.-F. Juang and C.-T. Lin. An on-line self-constructing neural fuzzy inference network and its applications. *IEEE Transactions on Fuzzy Systems*, 10(2):144–154, 2002.
- [56] J.-G. Juang, K.-C. Chin, and J.-Z. Chio. Intelligent automatic landing system using fuzzy neural networks and genetic algorithm. In *Proceedings of the 2004 American Control Conference*, volume 6, pages 5790–5795. Boston, Massachusetts, 2004.
- [57] J.-G. Juang and J.-Z. Chio. Aircraft landing control based on fuzzy modeling networks. In *Proceedings of the 2002 IEEE International Conference on Control Applications*, volume 1, pages 144–149. Glasgow, Scotland, U.K., 2002.
- [58] J.-G. Juang and J.-Z. Chio. Fuzzy modelling control for aircraft automatic landing system. *International Journal of Systems Science*, 26(2):77–87, 2005.
- [59] J.-G. Juang, B.-S. Lin, and K.-C. Chin. Intelligent fuzzy systems for aircraft landing control. *Lecture Notes in Computer Science*, 3613:851–861, 2005.

- [60] V. Kadirkamanathan and M. Niranjan. A function estimation approach to sequential learning with neural networks. *Neural Computation*, 5(6):954–975, 1993.
- [61] A. Kandel. *Fuzzy expert systems*. Boca Raton, FL CRC Press, 1992.
- [62] N. Kasabov. Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 31(6):902–918, 2001.
- [63] N. K. Kasabov and Q. Song. DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time series prediction. *IEEE Transactions on Fuzzy Systems*, 10(2):144–154, 2002.
- [64] J. Khan, J. S. Wei, M. Ringner, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, C. Peterson, and P. S. Meltzer. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7(6):673–679, 2001.
- [65] B. S. Kim. *Nonlinear Flight Control Using Neural Networks*. Ph.D Thesis, Department of Aerospace Engineering, Georgia Institute of Technology, Georgia, USA, 1993.
- [66] D. Kim, Y. Choi, and J. Suk. A glidepath tracking algorithm for autoland of a UAV. In *Infotech@Aerospace*. Arlington, Virginia, 2005.
- [67] T. Konjic, V. Miranda, and I. Kapetanovic. Prediction of LV substation load curves with fuzzy inference systems. In *Proceedings of the 5th International Conference on Probabilistic Methods Applied to Power Systems*, pages 129–134. Ames, Iowa, 2004.
- [68] W. A. Kwong, K. M. Passino, E. G. Laukonen, and S. Yurkovich. Expert supervision of fuzzy learning systems for fault tolerant aircraft control. *Proceedings of the IEEE*, 83(3):466–483, 1995.

- [69] H.-M. Lee, C.-M. Chen, J.-M. Chen, and Y.-L. Jou. An efficient fuzzy classifier with feature selection based on fuzzy entropy. *IEEE Transactions on Systems, Man, and Cybernetic-Part B: Cybernetics*, 31(3):426–432, 2001.
- [70] G. Leng, T. M. McGinnity, and G. Prasad. An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network. *Fuzzy Sets and Systems*, 150(2):211–243, 2005.
- [71] Y.-G. Leu, W.-Y. Wang, and T.-T. Lee. Observer-based direct adaptive fuzzy-neural control for nonaffine nonlinear systems. *IEEE Transactions on Neural Networks*, 16(4):853–861, 2005.
- [72] F. Lewis, A. Yesildirek, and K. Liu. Multilayer neural-net robot controller with guaranteed tracking performance. *IEEE Transactions Neural Networks*, 7(2):388–399, 1996.
- [73] F. L. Lewis. Nonlinear network structures for feedback control. *Asian Journal of Control*, 1(4):205–228, 1999.
- [74] Y. Li, N. Sundararajan, P. Saratchandran, and Z. Wang. Robust neuro- H_∞ controller design for aircraft auto-landing. *IEEE Transactions on Aerospace and Electronic Systems*, 40(1):158–167, 2004.
- [75] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan. A fast and accurate on-line sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, 17(6):1411–1423, 2006.
- [76] N.-Y. Liang, P. Saratchandran, G.-B. Huang, and N. Sundararajan. Classification of mental tasks from EEG signals using extreme learning machine. *International Journal of Neural Systems*, 16(1):29–38, 2006.
- [77] F. Liao, J. L. Wang, E. K. Poh, and D. Li. Fault-tolerant robust automatic landing control design. *Journal of Guidance, Control, and Dynamics*, 28(5):854–871, 2005.

- [78] C.-T. Lin, C.-M. Yeh, S.-F. Liang, J.-F. Chung, and N. Kumar. Support-vector-based fuzzy neural network for pattern classification. *IEEE Transactions on Fuzzy Systems*, 14(1):31–41, 2006.
- [79] C. K. Ling. Quasi-optimum design of an aircraft landing control system. *Journal of Aircraft*, 7(1):38–43, 1970.
- [80] D. A. Linkensand and H. O. Nyongesa. Learning systems in intelligent control: an appraisal of fuzzy, neural and genetic algorithm control applications. *IEE Proceedings of Control Theory and Applications*, 143(4):367–386, 1996.
- [81] S. M. B. Malaek, N. Sadati, H. Izadi, and M. Pakmehr. Intelligent autoland-ing controller design using neural networks and fuzzy logic. In *Proceedings of the 5th Asian Control Conference*, volume 1, pages 365–373, 2004.
- [82] E. Mamdani. Applications of fuzzy algorithms for simple dynamic plant. In *Proceedings of IEE*, volume 121, pages 1585–1588, 1974.
- [83] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.
- [84] P. S. Maybeck and R. D. Stevens. Reconfigurable flight control via multiple model adaptive control methods. *IEEE Transactions on Aerospace and Electronic Systems*, 27(3):470–480, 1991.
- [85] T. E. Menke and P. S. Maybeck. Sensor/Actuator failure detection in the Vista F -16 by multiple model adaptive estimation. *IEEE Transactions on Aerospace and Electronic Systems*, 31(4):1218–1229, 1995.
- [86] W. T. Miller, R. S. Sutton, and P. J. Werbos. *Neural Networks for Control*. MIT Press: Cambridge, MA, 1990.
- [87] M. L. Minsky and S. A. Papert. *Perceptrons: expanded edition*. Cambridge: MIT Press, 1988.

- [88] J. Moody and C. J. Darken. Fast learning in locally-tuned processing units. *Neural Computation*, 1:281–294, 1989.
- [89] W. D. Morse and K. A. Ossman. Model following reconfigurable flight control system for the AFTI/F-16. *Journal of Guidance, Control, and Dynamics*, 13(6):969–976, 1990.
- [90] K. S. Narendra. Neural networks for control theory and practice. *Proceedings of the IEEE*, 84(10):1385–1406, 1996.
- [91] K. S. Narendra and S. Mukhopadhyay. Adaptive control using neural networks and approximate models. *IEEE Transactions on Neural Networks*, 8(3):475–485, 1997.
- [92] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.
- [93] L. T. Nguyen, M. E. Ogburn, W. P. Gilbert, K. S. Kibler, P. W. Brown, and P. L. Deal. Simulator study of stall/post-stall characteristics of a fighter airplane with relaxed longitudinal static stability. NASA Technical Paper 1538, Dec. 1979.
- [94] K. Nho and R. K. Agarwal. Glideslope capture in wind gust via fuzzy logic controller. In *37th AIAA Aerospace Sciences Meeting and Exhibit*, pages 1–11. Reno, NV, 1999.
- [95] K. Nho and R. K. Agarwal. Automatic landing system design using fuzzy logic. *Journal of Guidance, Control, and Dynamics*, 23(2):298–304, 2000.
- [96] V. Olej and J. Krupka. Prediction of gross domestic product development by Takagi-Sugeno fuzzy inference systems. In *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications (ISDA '05)*, pages 186–191. Wroclaw, Poland, 2005.

- [97] M. Pachter, P. R. Chandler, and M. Mears. Reconfigurable tracking control with saturation. *Journal of Guidance, Control, and Dynamics*, 18(5):1016–1022, 1995.
- [98] A. A. Pashilkar, N. Sundararajan, and P. Saratchandran. A fault-tolerant neural aided controller for aircraft auto-landing. *Aerospace Science and Technology*, 10(1):49–61, 2006.
- [99] R. J. Patton. Fault detection and diagnosis in aerospace systems using analytical redundancy. *Computing & Control Engineering Journal*, 2(3):127–136, 1991.
- [100] R. J. Patton. Fault tolerant control: the 1997 situation. In *Proceedings of the IFAC Symposium SAFEPROCESS'97*, pages 1033–1055. Hull, England, 1997.
- [101] W. Pedrycz. *Fuzzy Control and Fuzzy Systems*. New York: Wiley, 1993.
- [102] W. Pedrycz and A. F. Rocha. Fuzzy-set based models of neurons and knowledge-based networks. *IEEE Transactions on Fuzzy Systems*, 1(4):254–266, 1993.
- [103] H. Peng, T. Ozaki, V. Haggan-Ozaki, and Y. Toyoda. A parameter optimization method for radial basis function type models. *IEEE Transactions on Neural Networks*, 14(2):432–438, 2003.
- [104] J. Platt. A resource allocating network for function interpolation. *Neural Computation*, 3(2):213–225, 1991.
- [105] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.
- [106] M. M. Polycarpou. An on-line approximation approach to fault monitoring, diagnosis and accommodation. SAE Technical Paper 941217, 1994.

- [107] M. J. D. Powell. Radial basis functions for multivariable interpolation: A review. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation*, pages 143–168. Oxford: Clarendon, 1987.
- [108] J. R. Quinlan. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
- [109] C. Rao and S. K. Mitra. *Generalized Inverse of Matrices and Its Applications*. John Wiley and Sons Inc, 1971.
- [110] P. Rodriguez, J. Wiles, and J. Elman. A recurrent neural network that learns to count. *Connection Science*, 11(1):5–40, 1999.
- [111] H.-J. Rong, N. Sundararajan, G.-B. Huang, and P. Saratchandran. Sequential Adaptive Fuzzy Inference System (SAFIS) for nonlinear system identification and prediction. *Fuzzy Sets and Systems*, 157(9):1260–1275, 2006.
- [112] D. Sadhuban and S. Feteih. F8 neurocontroller based on dynamic inversion. *Journal of Guidance, Control, and Dynamics*, 19(1):150–156, 1996.
- [113] A. Saffiotti, E. H. Ruspini, and K. Konolige. Blending reactivity and goal-directedness in a fuzzy controller. In *Proceedings of the Second IEEE International Conference on Fuzzy Systems*, pages 134–139. San Francisco, CA, USA, 1993.
- [114] R. M. Sanner and J.-J. E. Slotine. Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks*, 3(6):837–863, 1992.
- [115] S.-P. Shue and R. K. Agarwal. Design of automatic landing systems using mixed H_2/H_∞ control. *Journal of Guidance, Control, and Dynamics*, 22(1):103–114, 1999.
- [116] B. L. Stevens and F. L. Lewis. *Aircraft Control and Simulation*. John Wiley and Sons, 1992.

- [117] M. B. Subrahmanyam. H-infinity design of F/A-18A automatic carrier landing system. *Journal of Guidance, Control, and Dynamics*, 17(1):187–191, 1994.
- [118] M. Sugeno. *Industrial Applications of Fuzzy Control*. New York: Elsevier, 1985.
- [119] M. Sugeno and T. Yasukawa. A fuzzy-logic based approach to qualitative modeling. *IEEE Transactions on fuzzy systems*, 1(1):7–31, 1993.
- [120] N. Sundararajan, P. Saratchandran, and L. Yan. *Fully tuned radial basis function neural networks for flight control*. Boston: Kluwer academic publishers, 2001.
- [121] T. Tagaki and M. Sugeno. Fuzzy identification of systems and its application to modelling and control. *IEEE Transactions on Systems, Man, and Cybernetic*, 15(1):116–132, 1985.
- [122] E. Tzirkel-Hancock and F. Fallside. *Stable Control of Nonlinear Systems using Neural Networks*. Ph.D Thesis, Cambridge University Engineering Department, Cambridge, U.K., 1992.
- [123] M. N. Wagdi. An active control system for aircraft during landing approach in wind shear. In *22th Aerospace Sciences Meeting*. Reno, NV, 1984.
- [124] L. Wang. *Adaptive fuzzy systems and control : design and stability analysis*. PTR Prentice Hall, 1994.
- [125] L. Wang and J. Yen. Extracting fuzzy rules for system modeling using a hybrid of genetic algorithm and Kalman Filter. *Fuzzy Sets and Systems*, 101:353–362, 1999.
- [126] W. Wei and J. M. Mendel. A fuzzy logic method for modulation classification in nonideal environments. *IEEE Transactions on Fuzzy Systems*, 7(3):333–344, 1999.

- [127] D. A. White and D. A. Sofge. *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*. Van Nostrand Reinhold: New York, 1992.
- [128] B. Widrow and E. Walach. *Adaptive Inverse Control*. Prentice Hall, NJ, USA, 1996.
- [129] B. M. Wilamowski, S. Iplikci, O. Kaynak, and M. O. Efe. An algorithm for fast convergence in training neural networks. In *Proceedings of the 2001 International Joint Conference on Neural Networks (IJCNN'01)*, pages 1778–1782. Washington, D.C., 2001.
- [130] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (2nd ed.)*. Morgan Kaufmann, San Francisco, 2005.
- [131] S. Wu and M. J. Er. Dynamic fuzzy neural networks - a novel approach to function approximation. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 30(2):358–364, 2000.
- [132] J.-X. Xu, W. Wang, J. C. H. Goh, and G. Lee. Internal model approach for gait modeling and classification. In *The 27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, Shanghai, China, September 1-4, 2005.
- [133] C.-W. T. Yeu, M.-H. Lim, G.-B. Huang, A. Agarwal, and Y.-S. Ong. A new machine learning paradigm for terrain reconstruction. *IEEE Geoscience And Remote Sensing Letters*, 3(3):382–386,, 2006.
- [134] H. Ying. Sufficient conditions on uniform approximation of multivariate functions by general Takagi-Sugeno fuzzy systems with linear rule consequent. *IEEE Transactions on Systems, Man, and Cybernetic - Part A: Systems and Humans*, 28(4):515–520, 1998.
- [135] L. Yingwei, N. Sundararajan, and P. Saratchandran. Identification of time-varying nonlinear systems using minimal radial basis function neural net-

- works. *IEE Proceedings on Control Theory and Applications*, 144(2):202–208, 1997.
- [136] L. Yingwei, N. Sundararajan, and P. Saratchandran. A sequential learning scheme for function approximation and using minimal radial basis neural networks. *Neural Computation*, 9(2):461–478, 1997.
- [137] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.
- [138] X.-J. Zeng and M. G. Singh. Approximation theory of fuzzy systems-SISO case. *IEEE Transactions on Fuzzy Systems*, 2(2):162–176,, 1994.
- [139] X.-J. Zeng and M. G. Singh. Approximation theory of fuzzy systems-MIMO case. *IEEE Transactions on Fuzzy Systems*, 3(2):219–235, 1995.
- [140] R. Zhang, G.-B. Huang, N. Sundararajan, and P. Saratchandran. Multi-category classification using an extreme learning machine for microarray gene expression cancer diagnosis. (*in press*) *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2006.
- [141] W.-Q. Zhang. Control of a high performance fighter aircraft by using H_∞ theory and neural-fuzzy concepts. M.Eng Thesis, School of EEE, NTU, Singapore, 1997.