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 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 autolanding 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 autolanding 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.
Summary

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 autolanding 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

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

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

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 Deflections194

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 Deflections210

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

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
# TABLE OF CONTENTS

**Appendix B**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1 Simulation Block of the High Performance Fighter Aircraft</td>
<td>223</td>
</tr>
</tbody>
</table>

**Appendix C**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.1 Data of the Lateral and Longitudinal Aircraft Dynamics for BTFC Controller Design</td>
<td>225</td>
</tr>
</tbody>
</table>

**Appendix D**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.1 Results for Parameter Determination</td>
<td>228</td>
</tr>
<tr>
<td>D.2 Results for Chunk-by-Chunk Mode</td>
<td>231</td>
</tr>
<tr>
<td>D.3 Results for Different T-norm Operators</td>
<td>233</td>
</tr>
<tr>
<td>D.4 Results for Different Fuzzy Models</td>
<td>237</td>
</tr>
</tbody>
</table>

**Author’s Publications**

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>241</td>
</tr>
</tbody>
</table>

**Bibliography**

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>242</td>
</tr>
</tbody>
</table>
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

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 autolanding ..................... 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 ($\beta$) 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 ($\beta$) and lateral position ($Y$) under winds and left elevator failure stuck at -10deg . 94
5.1 SAFIS aided BTFC control strategy for autolanding ............... 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

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

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

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

5.9 Failure tolerance under left elevator stuck conditions

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

5.11 Failure tolerance under left aileron stuck conditions

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

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

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

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

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

5.17 Failure tolerance under left elevator and left aileron stuck conditions

5.18 Failure tolerance under left elevator and right aileron stuck conditions

6.1 Structure of Fuzzy Inference Systems
ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library

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 autolanding . . . . 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 (VT ) under the left
elevator stuck at -10deg . . . . . . . . . . . . . . . . . . . . . . . . 178

NANYANG TECHNOLOGICAL UNIVERSITY

SINGAPORE


LIST OF FIGURES

7.5 Comparison of OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC for the sideslip angle ($\beta$) 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 ($\beta$) 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 conditions194
LIST OF FIGURES

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 (\(\beta\)) 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 (\(\beta\)) 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
A.1 Graph Examples of Eight Membership Functions: a) Trimf; b) Trapmf; c) Gaussmf; d) Gauss2mf; e) Cauchymf; f) Pimf; g) Dsigmf; h) Psigmf.

B.1 The high performance fighter aircraft simulation block
### List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Effect of parameter $e_g$ on system performance (number of rules and the testing RMS error) under different $\epsilon_{\text{max}}$ values and $\kappa = 1.0$, $\gamma = 0.997$</td>
<td>47</td>
</tr>
<tr>
<td>3.2</td>
<td>Effect of parameter $e_g$ on system performance (number of rules and the testing RMS error) under different $\kappa$ values and $\epsilon_{\text{max}} = 10.0$, $\gamma = 0.997$</td>
<td>48</td>
</tr>
<tr>
<td>3.3</td>
<td>Effect of parameter $e_g$ on system performance (number of rules and the testing RMS error) under different $\gamma$ values and $\epsilon_{\text{max}} = 10.0$, $\kappa = 1.0$</td>
<td>48</td>
</tr>
<tr>
<td>3.4</td>
<td>Results of nonlinear identification problem 1</td>
<td>50</td>
</tr>
<tr>
<td>3.5</td>
<td>Results of nonlinear identification problem 2</td>
<td>52</td>
</tr>
<tr>
<td>3.6</td>
<td>Results of Mackey-Glass time series prediction</td>
<td>55</td>
</tr>
<tr>
<td>4.1</td>
<td>Specification of control surfaces</td>
<td>68</td>
</tr>
<tr>
<td>4.2</td>
<td>Mass and Geometry Data in aircraft model</td>
<td>69</td>
</tr>
<tr>
<td>4.3</td>
<td>Pillbox conditions at touchdown</td>
<td>83</td>
</tr>
<tr>
<td>4.4</td>
<td>Equilibrium states</td>
<td>86</td>
</tr>
<tr>
<td>4.5</td>
<td>Equilibrium control inputs</td>
<td>86</td>
</tr>
<tr>
<td>5.1</td>
<td>The parameter bounds of SAFIS used in the GA optimization</td>
<td>101</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>5.2</td>
<td>The parameter values of SAFIS after GA optimization</td>
<td>102</td>
</tr>
<tr>
<td>5.3</td>
<td>Performance comparison between SAFIS aided BTFC and EMRAN aided BTFC control schemes under the left elevator stuck at -10deg</td>
<td>105</td>
</tr>
<tr>
<td>5.4</td>
<td>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</td>
<td>114</td>
</tr>
<tr>
<td>6.1</td>
<td>Effect of parameter $\tilde{N}$ on system performance (training and testing accuracies) for TSK fuzzy model</td>
<td>144</td>
</tr>
<tr>
<td>6.2</td>
<td>Effect of parameter $\tilde{N}$ on system performance (training and testing accuracies) for Mamdani fuzzy model</td>
<td>147</td>
</tr>
<tr>
<td>6.3</td>
<td>Specification of Benchmark Datasets</td>
<td>149</td>
</tr>
<tr>
<td>6.4</td>
<td>Performance comparison for nonlinear system identification</td>
<td>153</td>
</tr>
<tr>
<td>6.5</td>
<td>Performance comparison for California housing prediction between different algorithms</td>
<td>156</td>
</tr>
<tr>
<td>6.6</td>
<td>Performance comparison between different algorithms for DNA classification</td>
<td>158</td>
</tr>
<tr>
<td>6.7</td>
<td>Performance comparison implemented in different chunk sizes for nonlinear system identification</td>
<td>160</td>
</tr>
<tr>
<td>6.8</td>
<td>Performance comparison implemented in different chunk sizes for California housing prediction</td>
<td>161</td>
</tr>
<tr>
<td>6.9</td>
<td>Performance comparison implemented in different chunk sizes for DNA classification</td>
<td>162</td>
</tr>
<tr>
<td>6.10</td>
<td>Performance comparison between Product and Min T-norm operators for nonlinear system identification</td>
<td>164</td>
</tr>
<tr>
<td>6.11</td>
<td>Performance comparison between Product and Min T-norm operators for California housing prediction</td>
<td>165</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

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
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.6</td>
<td>Performance comparison between Product and Min T-norm operators for nonlinear system identification</td>
<td>233</td>
</tr>
<tr>
<td>D.7</td>
<td>Performance comparison between Product and Min T-norm operators for California housing prediction</td>
<td>235</td>
</tr>
<tr>
<td>D.8</td>
<td>Performance comparison between Product and Min T-norm operators for DNA classification</td>
<td>236</td>
</tr>
<tr>
<td>D.9</td>
<td>Performance comparison between TSK and Mamdani fuzzy models for nonlinear system identification</td>
<td>237</td>
</tr>
<tr>
<td>D.10</td>
<td>Performance comparison between TSK and Mamdani fuzzy models for California housing prediction</td>
<td>238</td>
</tr>
<tr>
<td>D.11</td>
<td>Performance comparison between TSK and Mamdani fuzzy models for DNA classification</td>
<td>239</td>
</tr>
</tbody>
</table>
Nomenclature for the Aircraft Model

\(X, Y, Z\) \hspace{1cm} \text{aircraft body axes}

\(X_{cg}\) \hspace{1cm} \text{center-of-gravity location, m}

\(X_{cgr}\) \hspace{1cm} \text{reference center-of-gravity location for aerodynamic data, m}

\(\alpha\) \hspace{1cm} \text{angle of attack, deg}

\(\beta\) \hspace{1cm} \text{angle of sideslip, deg}

\(u, v, w\) \hspace{1cm} \text{component of aircraft velocity along X, Y, Z body axes, m/s}

\(\dot{u}, \dot{v}, \dot{w}\) \hspace{1cm} \text{aircraft acceleration along X, Y, Z body axes, m/s}^2

\(p\) \hspace{1cm} \text{aircraft roll rate about X body axis, rad/s}

\(q\) \hspace{1cm} \text{aircraft pitch rate about Y body axis, rad/s}

\(r\) \hspace{1cm} \text{aircraft yaw rate about Z body axis, rad/s}

\(\dot{p}\) \hspace{1cm} \text{aircraft roll acceleration about X body axis, rad/s}^2

\(\dot{q}\) \hspace{1cm} \text{aircraft pitch acceleration about Y body axis, rad/s}^2

\(\dot{r}\) \hspace{1cm} \text{aircraft yaw acceleration about Z body axis, rad/s}^2

\(\phi, \theta, \psi\) \hspace{1cm} \text{aircraft roll, pitch, yaw angle about X, Y, Z body axes respectively, rad}

\(x, y, z\) \hspace{1cm} \text{inertial location of the aircraft, m}

\(I_x, I_y, I_z\) \hspace{1cm} \text{moments of inertia about X, Y, Z body axes respectively, kgm}^2

\(I_{xz}, I_{yz}, I_{xy}\) \hspace{1cm} \text{cross moment of inertia, kgm}^2

\(C_{x,t}, C_{y,t}, C_{z,t}\) \hspace{1cm} \text{total aerodynamic force coefficients in X, Y, Z axes respectively}

\(C_{l,t}, C_{m,t}, C_{n,t}\) \hspace{1cm} \text{total aerodynamic moment coefficients in X, Y, Z axes respectively}

\(\bar{q}\) \hspace{1cm} \text{free-stream dynamic pressure, N/m}^2
Nomenclature for the Aircraft Model

\( \bar{c} \) 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

\( \chi \) 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 sideway acceleration respectively, g

\( \delta_h \) deflection angle of elevator, deg

\( \delta_a \) deflection angle of aileron, deg

\( \delta_r \) deflection angle of rudder respectively, deg

\( \delta_{sb} \) deflection angle of speed brake, deg

\( \delta_{le,f} \) deflection angle of leading edge flap, deg

\( \delta_t \) 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
1.1 Motivation

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.
1.1 Motivation

Among all the aircraft flight phases, autolanding 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 \textit{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 autolanding 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 autolanding 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 autolanding 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 autolanding 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
1.4 Organization of the Thesis

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-
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
2.1 Fuzzy Inference Systems

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,

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

2. **Trapezoid Membership Function**

   Trapezoid membership function (Trapmf) includes four parameters \(\{c_1, a_1, c_2, a_2\}\)
2.1 Fuzzy Inference Systems

and is given as,

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

(3) Gaussian Membership Function

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

\[
\text{Gaussmf}(x; c, a) = \exp \left( -\left( \frac{x - c}{a} \right)^2 \right) \]  \hspace{1cm} (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,

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

(5) Cauchy Membership Function

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

\[
\text{Cauchymf}(x; c, a) = \frac{1}{1 + \left( \frac{x - c}{a} \right)^2} \]  \hspace{1cm} (2.5)

(6) \(\pi\)-shaped Membership Function

\(\pi\)-shaped membership function is the product of S membership function and
2.1 Fuzzy Inference Systems

\( Z \) membership function.

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

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

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

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

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

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

(7) Difference between two Sigmoidally-shaped Membership Functions

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

\[
f(x; c, a) = \frac{1}{1 + \exp(ax + c)}
\] (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,

\[
Dsigmf(x; c_1, a_1, c_2, a_2) = f(x; c_1, a_1) - f(x; c_2, a_2)
\] (2.10)
2.1 Fuzzy Inference Systems

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,

\[
\text{Psigmf}(x; c_1, a_1, c_2, a_2) = f(x; c_1, a_1) \ast f(x; c_2, a_2)
\] (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],

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

where \(A_{ji}(j = 1, 2, \cdots, n; i = 1, 2, \cdots, \tilde{N})\) and \(B_{ki}(k = 1, 2, \cdots, m; i = 1, 2, \cdots, \tilde{N})\) 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 \(\mathbf{x}(\mathbf{x} = [x_1, \cdots, x_n]^T)\), \(m\) is the dimension of the output vector \(\mathbf{y}(\mathbf{y} = [y_1, \cdots, y_m]^T)\), and \(\tilde{N}\) is the number of fuzzy rules.
2.1 Fuzzy Inference Systems

(2) TSK Fuzzy Model

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

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

where \( \omega_{ki} (k = 1, 2, \ldots, m; i = 1, 2, \ldots, \tilde{N}) \) 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 \( \omega_{ki} = q_{k0i} + q_{k1i}x_1 + \cdots + q_{kin}x_n \).

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 \( x' \) into a fuzzy set \( A' \).

One of the most commonly used fuzzifier methods is the singleton fuzzifier, that is \( \mu_{A'}(x') = 1 \) for \( x' = x \) and \( \mu_{A'}(x') = 0 \) for \( x' \neq x \). All the studies in this thesis are based on the singleton fuzzifier.
2.1 Fuzzy Inference Systems

2.1.4 Fuzzy Inference Engine

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

\[
\Psi_i : A_{1i} \otimes A_{2i} \otimes \cdots \otimes A_{ni} \rightarrow B_{1i} + \cdots + B_{mi} \tag{2.12}
\]

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,

\[
R_i = \mu_{A_{1i}}(x_1) \otimes \mu_{A_{2i}}(x_2) \otimes \cdots \otimes \mu_{A_{ni}}(x_n) = \min \{\mu_{A_{1i}}(x_1), \mu_{A_{2i}}(x_2), \cdots, \mu_{A_{ni}}(x_n)\} \tag{2.13}
\]

For the algebraic product it is given as,

\[
R_i = \mu_{A_{1i}}(x_1) \ast \mu_{A_{2i}}(x_2) \ast \cdots \ast \mu_{A_{ni}}(x_n) = \prod_{j=1}^{n} \mu_{A_{ji}}(x_j) \tag{2.14}
\]

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

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

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
2.1 Fuzzy Inference Systems

output space is given by,

\[ \mu_B(y) = \mu_{A'o\Psi_i}(y) \]

\[ = \sup_{x'} [\mu_{A'}(x') \otimes \mu_{\Psi_i}(x', y)] \]

\[ = \sup_{x'} [\mu_{A'}(x') \otimes \mu_{A_1'}(x'_1) \otimes \mu_{A_2'}(x'_2) \otimes \mu_{A_m'}(x'_n) \otimes (\mu_{B_1i}(y_1) + \mu_{B_2i}(y_2) \cdots + \mu_{B_mi}(y_m))] \]

(2.16)

where \( o \) 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,

\[ y = \frac{\sum_{i=1}^{N} \tilde{y}_i \mu_B(\tilde{y}_i)}{\sum_{i=1}^{N} \mu_B(\tilde{y}_i)} \]

(2.17)

where \( \tilde{y}_i = [\tilde{y}_{i_1}, \cdots, \tilde{y}_{i_m}] \) and \( \tilde{y}_{ki} \) is the point at which \( B_{ki} \) achieves its maximum value, that is \( \mu_B(\tilde{y}_{ki}) = 1 \).

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
2.1 Fuzzy Inference Systems

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

$$y = \frac{\sum_{i=1}^{\tilde{N}} \beta_i \mu_{A'\phi_i}(\beta_i)}{\sum_{i=1}^{\tilde{N}} \mu_{A'\phi_i}(\beta_i)}$$  \hspace{1cm} (2.18)

where

$$\mu_{A'\phi_i}(\beta_i) = \sup_{x'} [\mu_A(x') \otimes \mu_{A_1}(x_1') \otimes \mu_{A_2}(x_2') \otimes \mu_{A_n}(x_n') \otimes (\mu_{B_1}(\beta_{1i})$$

$$+ \mu_{B_2}(\beta_{2i}) \cdots + \mu_{B_m}(\beta_{mi})]$$  \hspace{1cm} (2.19)

Due to the singleton fuzzifier, $\mu_A(x') = 1$ for $x' = x$ and because of center average defuzzifier, $\mu_{B_1}(\beta_{1i}) = \mu_{B_2}(\beta_{2i}) = \cdots = \mu_{B_m}(\beta_{mi}) = 1$.

Thus equation 2.18 becomes as,

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

where $\beta_i = [\beta_{1i}, \cdots, \beta_{mi}]$ and $\beta_{ki}$ 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

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

where $R_i$ is the weight and computed based on equation 2.13. $\omega_i = [\omega_{i1}, \omega_{i2}, \cdots, \omega_{ki}]$ ($k = 1, 2, \cdots, m; i = 1, 2, \cdots, \tilde{N}$) is the crisp output of rule $i$ and its elements may
be any function of the input variables. In case of linear function, the kth element \( \omega_{ki} \) 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

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

(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 \( (c, d] \).
2.2 Neural Networks

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 Neural Networks

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 \( \mathbf{x} = [x_1, x_2, \ldots, x_n] \)

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

\[
\mathbf{\hat{y}} = \sum_{i=1}^{\tilde{N}} \beta_i g(a_i \cdot \mathbf{x} + b_i), \quad \mathbf{x} \in \mathbb{R}^n, a_i \in \mathbb{R}^n
\]  \( (2.23) \)

where \( \beta_i \) is the weight connecting the \( i \)th hidden node to the output node; \( a_i \) 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. \( a_i \cdot \mathbf{x} \) denotes the inner product of vectors \( a_i \) and \( \mathbf{x} \)
2.2 Neural Networks

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

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

(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

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

(2.25)

(2) Multiquadric function

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

(2.26)

(3) Inverse multiquadric function

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

(2.27)

(4) Gaussian function

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

(2.28)
where \( r \) represents the Euclidean distance between a center \( a \) and the input data point \( x \), i.e. \( r = \| x - a \| \), \( b \) 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 functions. 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
2.2 Neural Networks

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 $\mathbf{x} = [x_1, x_2, \cdots, x_n]$, $\tilde{N}$ hidden neurons and $m$ output variables $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m]$, the outputs are given by

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

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

$$g_i(x) = \exp\left(-\frac{\|x - a_i\|^2}{b_i^2}\right)$$

where $a_i$ 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 been 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-
2.3 Functional Equivalence between fuzzy inference systems and radial basis function networks

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...
2.4 Fuzzy-Neural Systems

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,

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

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

\[
R_i = \prod_{j=1}^{n} \mu_{A_{ji}}(x_j) = \exp \left( - \left( \frac{x - a_i}{b_i} \right)^2 \right) \tag{2.32}
\]

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
2.4 Fuzzy-Neural Systems

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:
2.4 Fuzzy-Neural Systems

(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:

\[ y(t) = f[y(t-1), y(t-2), \ldots, y(t-k+1); u(t), u(t-1), \ldots, u(t-p+1)] \] (3.1)

where \( y \) is a vector containing \( m \) system outputs, \( u \) is a vector for \( r \) system inputs; \( f \) 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 \([y(t-1), \ldots, y(t-k+1); u(t), u(t-1), \ldots, u(t-p+1)]\) as the fuzzy system’s input-output \( x_t, y_t \) at time \( t \), the above equation becomes

\[ y_t = f(x_t) \] (3.2)

The aim of the new SAFIS algorithm is to approximate \( f \) such that

\[ \hat{y}_t = \hat{f}(x_t) \] (3.3)

where \( \hat{y}_t \) is the output of SAFIS. This means that the objective is to minimize the error between the system output and the output of SAFIS, \( \| y_t - \hat{y}_t \| \). 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 \( (x_1 \text{ is } A_{1i}) \text{ AND } \cdots \text{ AND } (x_n \text{ is } A_{ni}) \), then \( (\hat{y}_1 \text{ is } \omega_{1i}) \cdots (\hat{y}_m \text{ is } \omega_{mi}) \)

where \( \omega_{ik}(k = 1, 2, \ldots, m; i = 1, 2, \ldots, \tilde{N}) \) is a constant consequent parameter in
3.1 Architecture of SAFIS

rule $i$, $A_{ji}(j = 1, 2, \ldots, n)$ is fuzzy set of the $j$th input variable $x_j$ in rule $i$, $n$ is the dimension of the input vector $\mathbf{x}(\mathbf{x} = [x_1, \ldots, x_n]^T)$, $\tilde{N}$ is the number of fuzzy rules, $m$ is the dimension of the output vector $\hat{\mathbf{y}}(\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_m]^T)$. 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](image)

**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

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

(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

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

(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

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

(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

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

(3.7)
where \( \hat{y} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m]^T, \omega_i = [\omega_{i1}, \omega_{i2}, \ldots, \omega_{im}]^T \).

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 \( x_l \) is given by:

\[
E(i, l) = |\omega_i| \frac{R_i(x_l)}{\sum_{i=1}^{N} R_i(x_l)} \tag{3.8}
\]

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

\[
E(i) = |\omega_i| \frac{\sum_{i=1}^{N} R_i(x_l)}{\sum_{i=1}^{N} \sum_{l=1}^{N} R_i(x_l)} \tag{3.9}
\]

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

\[
E(i) = |\omega_i| \frac{\sum_{i=1}^{N} R_i(x_l)/N}{\sum_{i=1}^{N} \sum_{l=1}^{N} R_i(x_l)/N} \tag{3.10}
\]

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 \( N \to \infty \)
3.1 Architecture of SAFIS

the influence of the \( i \)th rule is given by

\[
E_{\inf}(i) = \lim_{N \to \infty} E(i) = \lim_{N \to \infty} \frac{\sum_{l=1}^{N} R_i(x_l)/N}{\sum_{i=1}^{N} \sum_{l=1}^{N} R_i(x_l)/N} \tag{3.11}
\]

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

\[
E_i = \lim_{N \to \infty} \frac{\sum_{l=1}^{N} R_i(x_l)}{N} \tag{3.12}
\]

Assume that the observations \((x_l, y_l), l = 1, \cdots\), are drawn from a sampling range \( X \) with a sampling density function \( p(x) \). 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 \( \Delta_k, k = 1, \cdots, M \). The size of \( \Delta_k \) is represented by \( S(\Delta_k) \). Since the sampling density function is \( p(x) \) there are around \( N \cdot p(x_k) \cdot S(\Delta_k) \) samples in each \( \Delta_k \), where \( x_k \) is any point chosen in \( \Delta_k \). When the number of input observations \( N \) is large and \( \Delta_k \) is small. From equation (3.12), we have

\[
E_i \approx \lim_{M \to \infty} \frac{\sum_{k=1}^{M} R_i(x_k) \cdot Np(x_k) \cdot S(\Delta_k)}{N} \\
= \lim_{M \to \infty} \sum_{k=1}^{M} R_i(x_k) \cdot p(x_k) \cdot S(\Delta_k) \tag{3.13} \\
= \int_X R_i(x)p(x)dx \\
= \int_X \exp \left( -\frac{\|x - c_i\|^2}{a_i^2} \right) p(x)dx
\]
Equation (3.13) shows that the joint density function of the input variable \( p(x) \) 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 \((x_1, \cdots, x_j, \cdots, x_n)^T\) of observations \( x \) are independent from each other. According to the assumption, a simplified model for Equation (3.13) can be obtained where the joint density function \( p(x) \) of \( x \) can be written as: \( p(x) = \prod_{j=1}^{n} p_j(x) \). 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:

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

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

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

\[
E_i = \int_X \exp\left( -\frac{\|x - c_i\|^2}{a_i^2} \right) \frac{1}{S(X)} dx
\] (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

\[
E_i \approx \frac{1}{S(X)} \left( 2 \int_0^{+\infty} \exp\left( -\frac{x^2}{a_i^2} \right) dx \right)^n
\]

\[
= \frac{\pi^{n/2} a_i^n}{S(X)}
\]

\[
\approx \frac{(1.8a_i)^n}{S(X)}
\] (3.16)

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

\[
E_{inf}(i) = |\omega_i| \frac{(1.8a_i)^n}{\sum_{i=1}^{N} (1.8a_i)^n}
\] (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 $x_1, y_1$ is received, it is translated into the first rule whose parameters are given as, $c_1 = x_1, \omega_1 = y_1, a_1 = \kappa \|x_1\|$. Then as new inputs $x_t, y_t$ ($t(>1)$ 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 $\tilde{N} + 1$:

\[
\begin{cases}
\|x_t - c_{nr}\| > \epsilon_t \\
E_{\text{inf}}(\tilde{N} + 1) = |e_t| \left(\frac{(1.8\kappa \|x_t - c_{nr}\|)^n}{\sum_{i=1}^{\tilde{N}+1} (1.8a_i)^n}\right) > e_g
\end{cases}
\]  

(3.18)

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

\[
\epsilon_t = \max\{\epsilon_{\text{max}} \times \gamma^t, \epsilon_{\text{min}}\}
\]  

(3.19)

where $\epsilon_{\text{max}}, \epsilon_{\text{min}}$ are the largest and smallest length of interest, $\gamma$ 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:

\[
\begin{align*}
\omega_{\tilde{N} + 1} & = e_t \\
\mathbf{c}_{\tilde{N} + 1} & = x_t \\
a_{\tilde{N} + 1} & = \kappa \|x_t - c_{nr}\|
\end{align*}
\]  

(3.20)
3.2 SAFIS Algorithm

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

$$\theta_t = \begin{bmatrix} \theta_1 & \cdots & \theta_{nr} & \cdots & \theta_N \end{bmatrix}^T$$

$$= \left[ \omega_1, c_1, a_1, \cdots, \omega_{nr}, c_{nr}, a_{nr}, \cdots, \omega_N, c_N, a_N \right]^T$$

where $\theta_{nr} = [\omega_{nr}, c_{nr}, a_{nr}]$ is the parameter vector of the nearest fuzzy rule and its gradient is derived as follows:

$$\dot{\omega}_{nr} = \frac{\partial \hat{y}_t}{\partial \omega_{nr}} = \frac{\partial \hat{y}_t}{\partial R_{nr}} \frac{\partial R_{nr}}{\partial \omega_{nr}} = \frac{R_{nr}}{\sum_{i=1}^{N} R_i}$$

$$\dot{c}_{nr} = \frac{\partial \hat{y}_t}{\partial c_{nr}} = \frac{\partial \hat{y}_t}{\partial R_{nr}} \frac{\partial R_{nr}}{\partial c_{nr}} = \frac{\omega_{nr} - \hat{y}_t}{\sum_{i=1}^{N} R_i} \frac{\partial R_{nr}}{\partial c_{nr}}$$

$$\dot{a}_{nr} = \frac{\partial \hat{y}_t}{\partial a_{nr}} = \frac{\partial \hat{y}_t}{\partial R_{nr}} \frac{\partial R_{nr}}{\partial a_{nr}} = \frac{\omega_{nr} - \hat{y}_t}{\sum_{i=1}^{N} R_i} \frac{\partial R_{nr}}{\partial a_{nr}}$$

$$\frac{\partial R_{nr}}{\partial c_{nr}} = 2R_{nr} \frac{x_t - c_{nr}}{a_{nr}^2}$$

$$\frac{\partial R_{nr}}{\partial a_{nr}} = 2R_{nr} \frac{||x_t - c_{nr}||^2}{a_{nr}^3}$$
3.2 SAFIS Algorithm

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

$$K_t = P_{t-1}B_t[R_t + B_t^TP_{t-1}B_t]^{-1}$$
$$\theta_t = \theta_{t-1} + K_te_t \quad (3.23)$$
$$P_t = [I - K_tB_t^T]P_{t-1} + Q_0I$$

When a new rule is added, the dimension of $P_t$ increases to

$$
\begin{pmatrix}
P_{t-1} & 0 \\
0 & p_0I_{Z_1 \times Z_1}
\end{pmatrix}
$$

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

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

Considering the Gaussian function $R(x) = \exp(-\frac{x^2}{2\sigma^2})$, 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

$$\left( \frac{R_{nr}, 2(\omega_{nr} - \hat{y}_t)R_{nr} x_{nr} - c_{nr}^2, 2(\omega_{nr} - \hat{y}_t)R_{nr} \|x_{nr} - c_{nr}\|^2}{\sum_{i=1}^{N} R_i, \sum_{i=1}^{N} R_i, \sum_{i=1}^{N} R_i} \right).$$

In this case, one may only need to adjust parameters of the nearest rule without adjusting the parameters of all
3.2 SAFIS Algorithm

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 $(x_t, y_t)$, where $x_t \in \mathbb{R}^n$, $y_t \in \mathbb{R}^m$ and $t = 1, 2, \cdots$, do

**compute** the overall system output:

$$
\hat{y}_t = \frac{\sum_{i=1}^{\tilde{N}} \omega_i R_i(x_t)}{\sum_{i=1}^{\tilde{N}} R_i(x_t)}
$$

(3.26)

$$
R_i(x_t) = \exp \left( -\frac{1}{a_i^2} \|x_t - c_i\|^2 \right)
$$

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

**calculate** the parameters required in the growth criterion:

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

$$
e_t = y_t - \hat{y}_t
$$

(3.27)

**apply** the criterion for adding rules:

*If $\|x_t - c_{nr}\| > \epsilon_t$ and $E_{\text{inf}}(\tilde{N} + 1) = \left| e_{t} \left( \frac{1.8a_{\text{avg}}(\|x_t - c_{\text{avg}}\|^2)_{\text{avg}}}{\sum_{i=1}^{\tilde{N}} (1.8a_i)^n} \right) > e_{g}\right.*
3.2 SAFIS Algorithm

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

$$\omega_{\tilde{N}+1} = \epsilon_t,$$

$$c_{\tilde{N}+1} = x_t,$$

$$a_{\tilde{N}+1} = \kappa \|x_t - c_{nr}\|$$

Else

**adjust** the system parameters $\omega_{nr}$, $c_{nr}$, $a_{nr}$ for the nearest rule only by using the EKF method.

**check** the criterion for pruning the rule:

\[
\text{If } E_{\text{inf}}(nr) = \left| \omega_{nr} \right| + \frac{(1.8a_{nr})^n}{\sum_{i=1}^{n} (1.8a_i)^n} < e_p \\
\text{remove the nr-th rule} \]

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 ($\kappa$) 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: $\epsilon_{\text{max}}$ is set to around the upper bound of input variables; $\epsilon_{\text{min}}$ is set to around 10% of $\epsilon_{\text{max}}$; $\gamma$ 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 ($\kappa$) is utilized to initialize the width of the newly added rule and chosen according to different problems. $\kappa$ 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.
3.2 SAFIS Algorithm

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:

\[
z = \text{sinc}(x, y) = \frac{\sin(x)\sin(y)}{xy}
\]  

(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 \(\gamma\) 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 \(\gamma\) values. From 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 \(\gamma\) 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 \(\kappa = 1.0, \gamma=0.997\)

<table>
<thead>
<tr>
<th>(\epsilon_{max})</th>
<th>(e_g) (0.001)</th>
<th>(0.005)</th>
<th>(0.01)</th>
<th>(0.05)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>(61, 0.0198)</td>
<td>(17, 0.0385)</td>
<td>(11, 0.0535)</td>
<td>(2, 0.0912)</td>
</tr>
<tr>
<td>5.0</td>
<td>(45, 0.0233)</td>
<td>(17, 0.0385)</td>
<td>(11, 0.0535)</td>
<td>(2, 0.0912)</td>
</tr>
<tr>
<td>10.0</td>
<td>(41, 0.0249)</td>
<td>(14, 0.0386)</td>
<td>(9, 0.0461)</td>
<td>(2, 0.0912)</td>
</tr>
</tbody>
</table>
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], SimpLeTS [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], SimpLeTS [5] and SONFIN [55]. For the chaotic time series prediction problem, the comparison is done with MRAN [136], RANEKF [60], eTS [6] and SimpLeTS [5]. In all the studies, the parameters \((r, \Omega)\) for eTS and SimpLeTS where \(r\) is the distance and \(\Omega\) 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

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

<table>
<thead>
<tr>
<th>(\kappa)</th>
<th>(e_g)</th>
<th>Rules</th>
<th>Testing RMS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.001</td>
<td>(41, 0.0249)</td>
<td>(9, 0.0461)</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>(14, 0.0386)</td>
<td>(2, 0.0912)</td>
</tr>
<tr>
<td>1.5</td>
<td>0.001</td>
<td>(50, 0.0350)</td>
<td>(15, 0.0598)</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>(18, 0.0586)</td>
<td>(3, 0.1382)</td>
</tr>
<tr>
<td>2.0</td>
<td>0.001</td>
<td>(52, 0.0557)</td>
<td>(15, 0.1384)</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>(25, 0.0902)</td>
<td>(3, 0.1391)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>(\gamma)</th>
<th>(e_g)</th>
<th>Rules</th>
<th>Testing RMS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>0.001</td>
<td>(58, 0.0167)</td>
<td>(9, 0.0535)</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>(17, 0.0385)</td>
<td>(2, 0.0912)</td>
</tr>
<tr>
<td>0.93</td>
<td>0.001</td>
<td>(58, 0.0167)</td>
<td>(9, 0.0535)</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>(17, 0.0385)</td>
<td>(2, 0.0912)</td>
</tr>
<tr>
<td>0.96</td>
<td>0.001</td>
<td>(41, 0.0229)</td>
<td>(9, 0.0538)</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>(16, 0.0394)</td>
<td>(2, 0.0912)</td>
</tr>
<tr>
<td>0.99</td>
<td>0.001</td>
<td>(45, 0.0201)</td>
<td>(9, 0.0524)</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>(16, 0.0373)</td>
<td>(2, 0.0912)</td>
</tr>
<tr>
<td>0.992</td>
<td>0.001</td>
<td>(38, 0.0229)</td>
<td>(10, 0.0600)</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>(16, 0.0373)</td>
<td>(2, 0.0912)</td>
</tr>
<tr>
<td>0.995</td>
<td>0.001</td>
<td>(40, 0.0247)</td>
<td>(13, 0.0340)</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>(19, 0.0327)</td>
<td>(2, 0.0912)</td>
</tr>
<tr>
<td>0.997</td>
<td>0.001</td>
<td>(41, 0.0249)</td>
<td>(9, 0.0461)</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>(14, 0.0386)</td>
<td>(2, 0.0912)</td>
</tr>
<tr>
<td>0.999</td>
<td>0.001</td>
<td>(30, 0.0357)</td>
<td>(10, 0.0636)</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>(17, 0.0534)</td>
<td>(3, 0.0931)</td>
</tr>
</tbody>
</table>
3.3 Performance Evaluation of SAFIS

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:

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

Model II:

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

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

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

\[ \hat{y}(t) = \hat{f}(y(t - 1), y(t - 2), \cdots, y(t - k), u(t - 1), u(t - 2), \cdots, u(t - p)) \] (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]

\[ y(t) = \frac{y(t - 1)y(t - 2)(y(t - 1) - 0.5)}{1 + y^2(t - 1) + y^2(t - 2)} + u(t - 1) \] (3.33)
3.3 Performance Evaluation of SAFIS

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: \(\gamma = 0.997, \epsilon_{\text{max}} = 1.0, \epsilon_{\text{min}} = 0.1, \kappa = 1.0, e_\gamma = 0.05, e_p = 0.005\).

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

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.
3.3 Performance Evaluation of SAFIS

![Graph showing rule update process between different algorithms for nonlinear identification problem 1 during the whole observation and between 0 and 1000 observation.](image)

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 Performance Evaluation of SAFIS

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]

\[ y(t) = \frac{y(t-1)}{1 + y^2(t-1)} + u^3(t-1) \] (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: \( \gamma = 0.997, \epsilon_{\text{max}} = 2.0, \epsilon_{\text{min}} = 0.2, \kappa = 2.0, \epsilon_{g} = 0.03, \epsilon_{p} = 0.003 \). 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>
<thead>
<tr>
<th>Methods</th>
<th>No. of Rules</th>
<th>Testing RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAFIS</td>
<td>8</td>
<td>0.0116</td>
</tr>
<tr>
<td>MRAN</td>
<td>10</td>
<td>0.0129</td>
</tr>
<tr>
<td>RANEKF</td>
<td>11</td>
<td>0.0184</td>
</tr>
<tr>
<td>Simpl(_e)TS ((r = 0.075, \Omega = 10^6))</td>
<td>18</td>
<td>0.0122</td>
</tr>
<tr>
<td>eTS ((r = 1.0, \Omega = 10^6))</td>
<td>19</td>
<td>0.0082</td>
</tr>
<tr>
<td>SONFIN [55]</td>
<td>10</td>
<td>0.0130</td>
</tr>
</tbody>
</table>

Table 3.5 shows the performance comparison of SAFIS with MRAN, RANEKF, eTS, Simpl\(_e\)TS 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\(_e\)TS 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\(_e\)TS 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.
3.3 Performance Evaluation of SAFIS

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
3.3 Performance Evaluation of SAFIS

![Graph showing rule update process between different algorithms for nonlinear identification problem 2](image)

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]:

\[
\frac{dx(t)}{dt} = \frac{0.2x(t - \tau)}{1 + x^{10}(t - \tau)} - 0.1x(t)
\]  

(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: \( \gamma = 0.98, \epsilon_{\text{max}} = 1.6, \epsilon_{\text{min}} = 0.16, \kappa = 1.68, e_g = 0.0005, e_p = 0.00005 \). The data follow a uniform sample distribution in the range \([0.4, 1.4]\).
3.3 Performance Evaluation of SAFIS

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

<table>
<thead>
<tr>
<th>Methods</th>
<th>No. of Rules</th>
<th>Testing NDEI</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAFIS</td>
<td>6</td>
<td>0.376</td>
</tr>
<tr>
<td>MRAN</td>
<td>14</td>
<td>0.375</td>
</tr>
<tr>
<td>RANEKF</td>
<td>18</td>
<td>0.378</td>
</tr>
<tr>
<td>Simpl_eTS ($r = 0.25$, $\Omega = 750$) [5]</td>
<td>11</td>
<td>0.394</td>
</tr>
<tr>
<td>eTS ($r = 0.25$, $\Omega = 750$) [6]</td>
<td>9</td>
<td>0.380</td>
</tr>
</tbody>
</table>

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...
3.3 Performance Evaluation of SAFIS

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 $\hat{y}$ to approximate the target output $y$. 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
3.3 Performance Evaluation of SAFIS

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.
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 autolanding 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
Aircraft Autolanding Fault-Tolerant Control Problem

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 autolanding 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.

Ghalia 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 $H_{\infty}$ techniques for the F/A-
Aircraft Autolanding Fault-Tolerant Control Problem

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 $H_2/H_\infty$ 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 $H_\infty$ 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 autolanding problem studied in this thesis considers not only severe wind disturbances but also the actuator failures during landing.

The aircraft autolanding 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.
4.1 Aircraft Mathematical Model

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.

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 \((\alpha, \beta)\) where \(\alpha\) is the
angle-of-attack and $\beta$ 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 $\delta_e$ is the deflection of elevator, $\delta_a$ is the deflection of aileron and $\delta_r$ is the deflection of rudder.

<table>
<thead>
<tr>
<th>Control surfaces</th>
<th>Control effectiveness</th>
<th>Sign of convention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevator ($\delta_e$)</td>
<td>$\pm 25^\circ$</td>
<td>Trailing edge down</td>
</tr>
<tr>
<td>Aileron ($\delta_a$)</td>
<td>$\pm 20^\circ$</td>
<td>Right wing trailing edge down</td>
</tr>
<tr>
<td>Rudder ($\delta_r$)</td>
<td>$\pm 30^\circ$</td>
<td>Trailing edge left</td>
</tr>
</tbody>
</table>

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,

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

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
4.1 Aircraft Mathematical Model

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>N</td>
<td>91,188</td>
</tr>
<tr>
<td>$I_x$</td>
<td>kgm$^2$</td>
<td>12,875</td>
</tr>
<tr>
<td>$I_y$</td>
<td>kgm$^2$</td>
<td>75,674</td>
</tr>
<tr>
<td>$I_z$</td>
<td>kgm$^2$</td>
<td>85,552</td>
</tr>
<tr>
<td>$I_{xz}$</td>
<td>kgm$^2$</td>
<td>1,331</td>
</tr>
<tr>
<td>$I_{xy}$</td>
<td>kgm$^2$</td>
<td>0</td>
</tr>
<tr>
<td>$I_{yz}$</td>
<td>kgm$^2$</td>
<td>0</td>
</tr>
<tr>
<td>$H_e$</td>
<td>kgm$^2$</td>
<td>216.9</td>
</tr>
<tr>
<td>$b$</td>
<td>m</td>
<td>9.144</td>
</tr>
<tr>
<td>$S$</td>
<td>m$^2$</td>
<td>27.87</td>
</tr>
<tr>
<td>$\bar{c}$</td>
<td>m</td>
<td>3.45</td>
</tr>
<tr>
<td>$x_{cg}$</td>
<td>m</td>
<td>0.3$\bar{c}$</td>
</tr>
<tr>
<td>$x_{cr}$</td>
<td>m</td>
<td>0.35$\bar{c}$</td>
</tr>
<tr>
<td>$\delta_{sb}$</td>
<td>deg</td>
<td>60</td>
</tr>
<tr>
<td>$\delta_{le}$</td>
<td>deg</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 4.2: Mass and Geometry Data in aircraft model

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 ($\phi, \theta$ and $\psi$) where $\phi$ is the roll angle, $\theta$ is the pitch angle and $\psi$ is the yaw angle and the position vector ($x, y$ and $h$) along $x, y$ and $z$ axis.

Force equations

\[
\dot{u} = rv - qw - g \sin \theta + \frac{\bar{q}}{m} \bar{S}C_{x,t} + \frac{T}{m} \\
\dot{v} = pw - ru + g \cos \theta \sin \phi + \frac{\bar{q}}{m} \bar{S}C_{y,t} \\
\dot{w} = qu - pv + g \cos \theta \cos \phi + \frac{\bar{q}}{m} \bar{S}C_{z,t}
\]

(4.2)

Moment equations

\[
\dot{p} = \frac{I_z - I_x}{I_x} qr + \frac{I_{xz}}{I_x} (\dot{r} + pq) + \frac{\bar{q} \bar{S}b}{I_x} C_{l,t} + \frac{H_e q}{I_x} \\
\dot{q} = \frac{I_z - I_y}{I_y} pr + \frac{I_{xz}}{I_y} (r^2 - p^2) + \frac{\bar{q} \bar{S} \bar{c}}{I_y} C_{m,t} - \frac{H_e r}{I_y} \\
\dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{I_{xz}}{I_z} (\dot{p} - qr) + \frac{\bar{q} \bar{S}b}{I_z} C_{n,t} + \frac{H_e q}{I_z}
\]

(4.3)
4.1 Aircraft Mathematical Model

Kinematics equations

\[
\begin{align*}
\dot{\phi} &= p + \tan \theta (q \sin \phi + r \cos \phi) \\
\dot{\theta} &= q \cos \phi - r \sin \phi \\
\dot{\psi} &= \frac{q \sin \phi + r \cos \phi}{\cos \theta}
\end{align*}
\] (4.4)

Navigation equations

\[
\begin{align*}
\dot{x} &= u \cos \psi \cos \theta + v (\cos \psi \sin \theta \sin \phi - \sin \psi \cos \phi) \\
&\quad + w (\cos \psi \sin \theta \cos \phi + \sin \psi \sin \phi) \\
\dot{y} &= u \sin \psi \cos \theta + v (\sin \psi \sin \theta \sin \phi + \sin \psi \cos \phi) \\
&\quad + w (\sin \psi \sin \theta \cos \phi - \cos \psi \sin \phi) \\
\dot{h} &= u \sin \theta - v \sin \phi \cos \theta - w \cos \phi \cos \theta
\end{align*}
\] (4.5)

Auxiliary equations

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

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

\[
\Delta C_{x,lef} = C_{x,lef}(\alpha, \beta) - C_x(\alpha, \beta, \delta_e = 0^\circ)
\]

(4.8)

For the Y-axis force coefficient \(C_{y,t}\),

\[
C_{y,t} = C_y(\alpha, \beta) + \Delta C_{y,lef}(1 - \frac{\delta_{lef}}{25}) + \left[ \Delta C_{y,\delta_e} + \Delta C_{y,\delta_a,lef}(1 - \frac{\delta_{lef}}{25}) \right] \frac{\delta_a}{20} \\
+ \Delta C_{y,\delta_r} \frac{\delta_r}{30} + \frac{rb}{2VT} \left[ C_{y,r}(\alpha) + \Delta C_{y,r,lef}(1 - \frac{\delta_{lef}}{25}) \right] \\
+ \frac{pb}{2VT} \left[ C_{y,p}(\alpha) + \Delta C_{p,lef}(\alpha)(1 - \frac{\delta_{lef}}{25}) \right]
\]

(4.9)

where

\[
\Delta C_{y,lef} = C_{y,lef}(\alpha, \beta) - C_y(\alpha, \beta)
\]

\[
\Delta C_{y,\delta_e} = C_{y,\delta_e}(\alpha, \beta) - C_y(\alpha, \beta)
\]

\[
\Delta C_{y,\delta_a,lef} = C_{y,\delta_a,lef}(\alpha, \beta) - C_{y,lef}(\alpha, \beta) - \Delta C_{y,\delta_a}
\]

(4.10)

\[
\Delta C_{y,\delta_r} = C_{y,\delta_r}(\alpha, \beta) - C_y(\alpha, \beta)
\]

For the Z-axis force coefficient \(C_{z,t}\),

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

(4.11)

where

\[
\Delta C_{z,lef} = C_{z,lef}(\alpha, \beta) - C_z(\alpha, \beta, \delta_e = 0^\circ)
\]

(4.12)
For the X axis moment coefficient $C_{l,t}$,

$$C_{l,t} = C_l(\alpha, \beta, \delta_e) + \Delta C_{l,lef}(1 - \frac{\delta_{lef}}{25}) + [\Delta C_{l,\delta_a} + \Delta C_{l,\delta_a,lef}(1 - \frac{\delta_{lef}}{25})] \frac{\delta_a}{20}$$

$$+ \Delta C_{l,\delta_r} \frac{r_b}{30} + \frac{r_b}{2V_T} [C_{l,r}(\alpha) + \Delta C_{l,r,lef}(1 - \frac{\delta_{lef}}{25})]$$

$$+ \frac{p_b}{2V_T} [C_{l,p}(\alpha) + \Delta C_{l,p,lef}(1 - \frac{\delta_{lef}}{25})] + \Delta C_{l,\beta}(\alpha)\beta$$

(4.13)

where

$$\Delta C_{l,lef} = C_{l,lef}(\alpha, \beta) - C_l(\alpha, \beta, \delta_e = 0^\circ)$$

$$\Delta C_{l,\delta_a} = C_{l,\delta_a}(\alpha, \beta) - C_l(\alpha, \beta, \delta_e = 0^\circ)$$

$$\Delta C_{l,\delta_a,lef} = C_{l,\delta_a,lef}(\alpha, \beta) - C_{l,lef}(\alpha, \beta) - \Delta C_{l,\delta_a}$$

$$\Delta C_{l,\delta_r} = C_{l,\delta_r}(\alpha, \beta) - C_l(\alpha, \beta, \delta_e = 0^\circ)$$

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

$$C_{m,t} = C_m(\alpha, \beta, \delta_e) + C_{z,t} [x_{cgr} - x_{cg}] + \Delta C_{m,lef}(1 - \frac{\delta_{lef}}{25})$$

$$+ \frac{q\bar{c}}{2V_T} [C_{m,q}(\alpha) + \Delta C_{m,q,lef}(\alpha)](1 - \frac{\delta_{lef}}{25}) + \Delta C_{m}(\alpha) + \Delta C_{m,ds}(\alpha, \delta_e)$$

(4.15)

where

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

(4.16)

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

$$C_{n,t} = C_n(\alpha, \beta, \delta_e) + \Delta C_{n,lef}(1 - \frac{\delta_{lef}}{25}) + [\Delta C_{n,\delta_a} + \Delta C_{n,\delta_a,lef}(1 - \frac{\delta_{lef}}{25})] \frac{\delta_a}{20}$$

$$+ \Delta C_{n,\delta_r} \frac{r_b}{30} + \frac{r_b}{2V_T} [C_{n,r}(\alpha) + \Delta C_{n,r,lef}(\alpha)(1 - \frac{\delta_{lef}}{25})]$$

$$+ \frac{p_b}{2V_T} [C_{n,p}(\alpha) + \Delta C_{n,p,lef}(\alpha)(1 - \frac{\delta_{lef}}{25})]p + \Delta C_{n,\beta}(\alpha)\beta - C_{y,t} [x_{cgr} - x_{cg}] \frac{\bar{c}}{b}$$

(4.17)
4.1 Aircraft Mathematical Model

where

\[
\Delta C_{n,lef} = C_{n,lef}(\alpha, \beta) - C_n(\alpha, \beta, \delta_e = 0^\circ)
\]

\[
\Delta C_{n,\delta_a} = C_{n,\delta_a}(\alpha, \beta) - C_n(\alpha, \beta, \delta_e = 0^\circ)
\]

\[
\Delta C_{n,\delta_a,lef} = C_{n,\delta_a,lef}(\alpha, \beta) - C_{n,lef}(\alpha, \beta) - \Delta C_{n,\delta_a}
\]

\[
\Delta C_{n,\delta_r} = C_{n,\delta_r}(\alpha, \beta) - C_n(\alpha, \beta, \delta_e = 0^\circ)
\]

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 slide 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 slide 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 \(C_n(\alpha, \beta, \delta_{e1}, \delta_{e2})\) where \(\delta_{e1}\) and \(\delta_{e2}\) 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 $-90\text{deg}$ (from East to West). Velocity is maintained at $83\text{m/s}$ for this and all the other segments except the flare (segment 6).

- **Segment 2**: A coordinated right turn with bank angle $40\text{deg}$ at 600m to align with the runway $0\text{deg}$ (heading north).

- **Segment 3**: A level flight at 600m heading $0\text{deg}$ (towards north).

- **Segment 4**: Descent on glide-slope of $-6\text{deg}$ to altitude of 300m.

- **Segment 5**: Descent on glide-slope of $-3\text{deg}$ to altitude of 12m.

- **Segment 6**: Flare and touchdown. Velocity is reduced from $83\text{m/s}$ to $79\text{m/s}$.

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

$$
\mu_g = \mu_g1 + \mu_{gc}
$$

$$
\dot{\mu}_g1 = 0.2|\mu_{gc}| \sqrt{2a_u N_1} - a_p \mu_g1
$$

(4.19)
4.2 Aircraft Landing Task and Wind Model

Figure 4.2: The landing trajectory

where $\mu_{g1}$ is turbulence component and $\mu_{gc}$ is the mean wind.

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

where $\mu_0$ is 6m/sec and $N_1$ is the Gaussian random noises with mean zero and variance 100, and

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

\[ \omega_g = -\omega_0(1 + \ln(h/510)/\ln(51)) \] (4.22)
where $\omega_0$ is $12\text{m/sec}$ above the height of $h_{\text{shear}}$ and $-12\text{m/sec}$ under $h_{\text{shear}}$, $h_{\text{shear}}$ is $91\text{m}$. The wind shear in y-body axis is a step change of magnitude $10\text{m/s}$ and $-20\text{m/s}$ at altitudes $470\text{m}$ and $190\text{m}$ 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](image)
4.3 Failure Types

Not only the winds but also the actuator failures are considered in the aircraft autolanding 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
4.3 Failure Types

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

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.
4.3 Failure Types

Figure 4.4: Feasible range for left elevator stuck failure
4.3 Failure Types

Figure 4.5: Feasible range for left aileron stuck failure
4.3 Failure Types

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

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>
<thead>
<tr>
<th>Condition</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-distance</td>
<td>$-100m \leq x \leq 300m$</td>
</tr>
<tr>
<td>Y-distance</td>
<td>$</td>
</tr>
<tr>
<td>Total Velocity</td>
<td>$V_T \geq 60m/s$</td>
</tr>
<tr>
<td>Sink Rate</td>
<td>$\dot{h} \leq 1.0m/s$</td>
</tr>
<tr>
<td>Bank Angle</td>
<td>$</td>
</tr>
</tbody>
</table>

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.

![Control Architecture](image.png)

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, $r$, which are used as the input to the BTFC controller. The output of the BTFC controller, $u_c$, 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 `$r$`
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](image-url)
4.4 Baseline Trajectory Following Controller

- Straight Line Segment:

\[
q = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\
\delta = \text{sgn}[(x - x_1)(y_2 - y_1) - (x_2 - x_1)(y - y_1)] \\
\text{abs} \left[ \frac{(y_2 - y_1)x - (x_2 - x_1)y + (x_2 - x_1)y_1 - (y_2 - y_1)x_1}{q} \right] \\
l = \sqrt{(x - x_1)^2 + (y - y_1)^2} \\
s = \sqrt{l^2 - \delta^2} \\
\psi_{ref} = \tan^{-1}\left[ \frac{(y_2 - y_1)}{(x_2 - x_1)} \right]
\]

- Turn Segment

\[
q = r \Delta \theta \\
l = \sqrt{(x - x_r)^2 + (y - y_r)^2} \\
\psi_{ref} = \tan^{-1}\left[ \frac{(x - x_r)}{(y - y_r)} \right] \\
s = r(\psi_t - \theta_0)
\]

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:

\[
w_r = s/q \\
w_l = 1 - w_r \\
h_{ref} = h_1 w_l + h_2 w_r \\
V_{ref} = V_1 w_l + V_2 w_r
\]

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

\[
h_{ref} = h_2 + (h_1 - h_2) \cdot e^{-2.5s/q}
\]
4.4 Baseline Trajectory Following Controller

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>
<thead>
<tr>
<th>State Variables</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u$ (m/s)</td>
<td>81.31</td>
</tr>
<tr>
<td>$w$ (m/s)</td>
<td>16.08</td>
</tr>
<tr>
<td>$q$ (rad/s)</td>
<td>0</td>
</tr>
<tr>
<td>$\theta$ (rad)</td>
<td>11.19</td>
</tr>
<tr>
<td>$h$ (m)</td>
<td>600</td>
</tr>
<tr>
<td>$v$ (m/s)</td>
<td>0</td>
</tr>
<tr>
<td>$p$ (rad/s)</td>
<td>0</td>
</tr>
<tr>
<td>$r$ (rad/s)</td>
<td>0</td>
</tr>
<tr>
<td>$\phi$ (rad)</td>
<td>0</td>
</tr>
<tr>
<td>$\psi$ (rad)</td>
<td>0</td>
</tr>
<tr>
<td>$y$ (m)</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control Surfaces</th>
<th>Deflections</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_c$ (deg)</td>
<td>-0.64</td>
</tr>
<tr>
<td>$\delta_a$ (deg)</td>
<td>0</td>
</tr>
<tr>
<td>$\delta_r$ (deg)</td>
<td>0</td>
</tr>
<tr>
<td>$\delta_t$ (deg)</td>
<td>0.22</td>
</tr>
</tbody>
</table>

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

$$
\Delta \dot{x} = A\Delta x + B\Delta u \\
\Delta \dot{y} = C\Delta x + D\Delta u
$$

(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 $\Delta$ 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 $(r - p \cdot \tan \alpha)$ 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)
4.4 Baseline Trajectory Following Controller

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 ($\beta$) and lateral position ($Y$).

Figure 4.12: BTFC controller performance for the altitude ($h$) and velocity ($V_T$) 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 $\beta$ 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.
4.4 Baseline Trajectory Following Controller

Figure 4.14: BTFC controller performance for the altitude (h) and velocity ($V_T$) under winds and left elevator failure stuck at -10deg
4.4 Baseline Trajectory Following Controller

Figure 4.15: BTFC controller performance for the sideslip angle ($\beta$) 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 autolanding 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 autolanding 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.
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

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

(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,

\[ u = f^{-1}(\dot{x}, x) \]  

(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.
5.1 Selection of SAFIS Parameters

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

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 \( (\epsilon_{\text{max}}, \epsilon_{\text{min}}, \gamma) \), the overlap factor for determining the width of the newly added rule \( (\kappa) \), the growing threshold for adding a new rule \( (e_g) \), the
5.1 Selection of SAFIS Parameters

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 autolanding fault-tolerant control problem, its parameters have to been 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**
5.1 Selection of SAFIS Parameters

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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>$\epsilon_{\text{max}}$</th>
<th>$\epsilon_{\text{min}}$</th>
<th>$\gamma$</th>
<th>$\kappa$</th>
<th>$e_p$</th>
<th>$e_g$</th>
<th>$p_0$</th>
<th>$q$</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Bound</td>
<td>0.1</td>
<td>0.01</td>
<td>0.9</td>
<td>1.0</td>
<td>0.0001</td>
<td>0.00001</td>
<td>0.5</td>
<td>0.0001</td>
<td>0.5</td>
</tr>
<tr>
<td>Upper Bound</td>
<td>1.0</td>
<td>0.1</td>
<td>0.999</td>
<td>2.0</td>
<td>0.001</td>
<td>0.00001</td>
<td>1000</td>
<td>1.0</td>
<td>1000</td>
</tr>
</tbody>
</table>

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,
5.1 Selection of SAFIS Parameters

![Diagram showing best and average solutions in the GA population during a typical run.](image)

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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>$\epsilon_{\text{max}}$</th>
<th>$\epsilon_{\text{min}}$</th>
<th>$\gamma$</th>
<th>$\kappa$</th>
<th>$\epsilon_g$</th>
<th>$\epsilon_p$</th>
<th>$p_0$</th>
<th>$q$</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td>0.5081</td>
<td>0.0161</td>
<td>0.9456</td>
<td>1.5103</td>
<td>0.0007</td>
<td>0.000012</td>
<td>551.18</td>
<td>0.6530</td>
<td>795.07</td>
</tr>
</tbody>
</table>

Table 5.2: The parameter values of SAFIS after GA optimization
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 -10deg at the beginning of the turn. Figure 5.5 and Figure 5.6 show the altitude (h), the velocity \( V_T \), sideslip angle \( \beta \) and lateral position \( Y \) during the landing phase for the SAFIS aided BTFC, EMRAN aided BTFC and BTFC.

![Diagram of altitude and velocity](image1)

![Diagram of velocity over time](image2)

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 -10deg
5.2 Single Surface Failure

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
5.2 Single Surface Failure

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

<table>
<thead>
<tr>
<th>Method</th>
<th>SAFIS aided BTFC</th>
<th>EMRAN aided BTFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory error (RMSE)</td>
<td>8.8573</td>
<td>8.9206</td>
</tr>
<tr>
<td>Velocity error (RMSE)</td>
<td>3.4243</td>
<td>3.8146</td>
</tr>
<tr>
<td># of rules/neurons for δe</td>
<td>6</td>
<td>28</td>
</tr>
<tr>
<td># of rules/neurons for δa</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

To analyze the control scheme performance in more detail, Figure 5.7 shows the deflection of the left elevator (δe−left−t) and right elevator (δe−right−t) along with the BTFC (δe−right−c) and SAFIS/EMRAN components (δe−right−f/n) of the control signals. The rule and neuron history based on SAFIS and EMRAN for the elevator δe 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.

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 cannot 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
5.2 Single Surface Failure

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
5.2 Single Surface Failure

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](image)

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.
5.3 Two Surface Failures

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.

![Diagram showing touchdown points for SAFIS aided BTFC under left aileron stuck conditions.](image)

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 Two Surface Failures

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 -10deg and the left aileron is stuck at +10deg. Figure 5.13 and Figure 5.14 show the altitude (h), the velocity ($V_T$), sideslip angle ($\beta$) and lateral position (Y) during the landing phase for the SAFIS aided BTFC, EMRAN aided BTFC and BTFC along with the reference trajectory.

![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.](image)

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 -10deg and the left aileron stuck at 10deg
5.3 Two Surface Failures

Figure 5.14: Comparison of SAFIS aided BTFC, EMRAN aided BTFC and BTFC for the sideslip angle ($\beta$) 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.
5.3 Two Surface Failures

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

<table>
<thead>
<tr>
<th>Method</th>
<th>SAFIS aided BTFC</th>
<th>EMRAN aided BTFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory error (RMSE)</td>
<td>8.8978</td>
<td>8.9127</td>
</tr>
<tr>
<td>Velocity error (RMSE)</td>
<td>3.2761</td>
<td>3.8097</td>
</tr>
<tr>
<td># of rules/neurons for δₑ</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td># of rules/neurons for δₐ</td>
<td>22</td>
<td>25</td>
</tr>
</tbody>
</table>

Figure 5.15 shows the deflection of the left elevator (δₑ−left−t) and right elevator (δₑ−right−t).
5.3 Two Surface Failures

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 $\delta_e$ 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.

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

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](image)

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 ×.
5.3 Two Surface Failures

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 Two Surface Failures

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 ×.

![Figure 5.18: Failure tolerance under left elevator and right aileron stuck conditions](image)

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
5.3 Two Surface Failures

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”
5.3 Two Surface Failures

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.
6.1 Extreme Learning Machine (ELM)

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 Simpl_eTS [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 $\mathbb{R}^n$. 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

$$
\hat{f}_{\tilde{N}}(x) = \sum_{i=1}^{\tilde{N}} \beta_i G(a_i, b_i, x), \quad x \in \mathbb{R}^n, a_i \in \mathbb{R}^n
$$

(6.1)

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

$$
G(a_i, b_i, x) = g(a_i \cdot x + b_i), \quad b_i \in R
$$

(6.2)

where $a_i$ 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. $a_i \cdot x$ denotes the inner product of vectors $a_i$ and $x$ in $\mathbb{R}^n$.

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

$$
G(a_i, b_i, x) = g(b_i \|x - a_i\|), \quad b_i \in \mathbb{R}^+
$$

(6.3)

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

For $N$ arbitrary distinct samples $(x_i, t_i) \in \mathbb{R}^n \times \mathbb{R}^m$, if a SLFN with $\tilde{N}$ hidden nodes can approximate these $N$ samples, it then implies that there exist $\beta_i$, $a_i$ and
6.1 Extreme Learning Machine (ELM)

$b_i$ such that

$$f_{\tilde{N}}(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i G(a_i, b_i, x_j) = t_j, \quad j = 1, \ldots, N. \quad (6.4)$$

Equation (6.4) can be written compactly as:

$$H\beta = T \quad (6.5)$$

where

$$H(a_1, \ldots, a_{\tilde{N}}, b_1, \ldots, b_{\tilde{N}}, x_1, \ldots, x_N) = \begin{bmatrix}
G(a_1, b_1, x_1) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_1) \\
\vdots & \ddots & \vdots \\
G(a_1, b_1, x_N) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_N)
\end{bmatrix}_{\tilde{N} \times \tilde{N}} \quad (6.6)$$

$$\beta = \begin{bmatrix}
\beta_1^T \\
\vdots \\
\beta_{\tilde{N}}^T
\end{bmatrix}_{\tilde{N} \times m} \quad \text{and} \quad T = \begin{bmatrix}
t_1^T \\
\vdots \\
t_N^T
\end{bmatrix}_{N \times m} \quad (6.7)$$

$H$ is called the hidden layer output matrix of the network [39, 40]; the $i$th column of $H$ is the $i$th hidden node’s output vector with respect to inputs $x_1, x_2, \ldots, x_N$ and the $j$th row of $H$ is the output vector of the hidden layer with respect to input $x_j$.

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

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

where $H^\dagger$ is the Moore-Penrose generalized inverse [109] of the hidden layer output matrix $H$ and equal to $(H^TH)^{-1}H^T$. 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 \( \mathcal{D} = \{(x_i, t_i) | x_i \in \mathbb{R}^n, t_i \in \mathbb{R}^m, i = 1, \ldots, N\} \),
activation function \( g \), and hidden node number \( \tilde{N} \).

1. **step 1** Randomly assign hidden node parameters \((a_i, b_i)\), \(i = 1, \ldots, \tilde{N}\).
2. **step 2** Calculate the hidden layer output matrix \( H \).
3. **step 3** Calculate the output weight \( \beta \): \( \beta = H^\dagger T \).

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 \( g \):

\[
\mu_{A_{ji}}(x_j; c_{ji}, a_i) = g(x_j; c_{ji}, a_i)
\]  

(6.9)
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

$$R_i(x; c_i, a_i) = \mu_{A_{i1}}(x_1; c_{i1}, a_i) \otimes \mu_{A_{i2}}(x_2; c_{i2}, a_i) \otimes \cdots \otimes \mu_{A_{ni}}(x_n; c_{ni}, a_i) \quad (6.10)$$

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
6.2 Structure of Fuzzy Inference Systems

as

\[ R_i(x; c_i, a_i) = \prod_{j=1}^{n} \mu_{A_{ji}}(x_j; c_{ji}, a_i) = \prod_{j=1}^{n} \left(1 - \frac{|x_j - c_{ji}|}{a_i}\right) \]  

(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:

\[ G(x; c_i, a_i) = \frac{R_i(x; c_i, a_i)}{\sum_{i=1}^{N} R_i(x; c_i, a_i)} \]  

(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 \(\hat{y}\) for given input \(x\) is calculated by,

\[ \hat{y} = \frac{\sum_{i=1}^{N} \beta_i R_i(x; c_i, a_i)}{\sum_{i=1}^{N} R_i(x; c_i, a_i)} = \sum_{i=1}^{N} \beta_i G(x; c_i, a_i) \]  

(6.13)

where \(\beta_i = [\beta_{1i}, \ldots, \beta_{mi}]\) and \(\beta_{ki}(k = 1, 2, \ldots, m)\) 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 \(\hat{y}\) for given input \(x\) is
where the weight \( \beta_i = [\beta_{i1}, \cdots, \beta_{im}] \) is the consequent crisp outputs of each rule and here it is a linear function on input variables. For \( \beta_{ki} \), it equals to \( \beta_{ki} = q_{ki0} + q_{ki1}x_1 + \cdots + q_{kin}x_n \).

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 \( R_i(\cdot)/\sum_{i=1}^{N} R_i(\cdot) \) represents the normalized firing strength of fuzzy rules while the consequent (then) part \( \beta_i(= [\beta_{i1}, \cdots, \beta_{im}]^T) \) is same in the form but represents different meaning. For the Mamdani fuzzy model, the \( \beta_{ki}(k = 1, \cdots, m) \) contains the linguistic information since it is related with the linguistic variable \( B_{ki} \) while the \( \beta_{ki} \) 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 \( \beta \) 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 \( x = [x_1, \cdots, x_n]^T \) for the
fuzzy inference system (Figure 6.1) is:

\[
G(x; c_i, a_i) = \frac{R_i(x; c_i, a_i)}{\sum_{i=1}^N R_i(x; c_i, a_i)} = \frac{\prod_{j=1}^n \left(1 - \frac{|x_j - c_{ji}|}{a_i}\right)}{\sum_{i=1}^N \prod_{j=1}^n \left(1 - \frac{|x_j - c_{ji}|}{a_i}\right)}
\]  

(6.15)

For \( N \) arbitrary distinct training samples \((x_i, t_i)\), where \( x_i = [x_{i1}, x_{i2}, \cdots, x_{im}]^T \in \mathbb{R}^n \) and \( t_i = [t_{i1}, t_{i2}, \cdots, t_{im}]^T \in \mathbb{R}^m \), 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


6.2 Structure of Fuzzy Inference Systems

$\beta_i$, $c_i$ and $a_i$ as shown in Figure 6.1 is given as,

$$f_{\tilde{N}}(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i G(x_j; c_i, a_i) = t_j, \quad j = 1, \cdots, N.$$  \hspace{1cm} (6.16)

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

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

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

$$H\beta = T \hspace{1cm} (6.18)$$

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

$$H(c_1, \cdots, c_{\tilde{N}}, a_1, \cdots, a_{\tilde{N}}; x_1, \cdots, x_N) = \begin{bmatrix} G(x_1; c_1, a_1) & \cdots & G(x_1; c_{\tilde{N}}, a_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ G(x_N; c_1, a_1) & \cdots & G(x_N; c_{\tilde{N}}, a_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}} \hspace{1cm} (6.19)$$

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

$$\beta_i = x_e^T q_i \hspace{1cm} (6.20)$$

where $x_e^T$ is the extended input vector by appending the input vector $x$ with 1, that is $[1, x^T]^T$; $q_i$ is the parameters existing in the TSK model for the $i$th fuzzy
6.2 Structure of Fuzzy Inference Systems

rule and given by

\[ q_i = \begin{bmatrix}
q_{i0} & \cdots & q_{i0} \\
\vdots & \ddots & \vdots \\
q_{im} & \cdots & q_{im}
\end{bmatrix}_{(n+1) \times m} \quad (6.21) \]

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

\[ f_N(x_j) = \sum_{i=1}^{\hat{N}} x_j^T q_i G(x_j; c_i, a_i) = t_j, \quad j = 1, \ldots, N. \quad (6.22) \]

The equation further is written in the following compact form,

\[ Hq = T \quad (6.23) \]

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

\[ q = \begin{bmatrix}
q_{10} & \cdots & q_{m0} \\
\vdots & \ddots & \vdots \\
q_{1n} & \cdots & q_{mn}
\end{bmatrix}_{\hat{N}(n+1) \times m} \quad (6.24) \]
6.2 Structure of Fuzzy Inference Systems

\[
H(c_1, \cdots, c_N, a_1, \cdots, a_N; x_1, \cdots, x_N) = \\
\begin{bmatrix}
G(x_1; c_1, a_1) & \cdots & G(x_N; c_1, a_1) \\
\vdots & \ddots & \vdots \\
G(x_1; c_1, a_1)x_1 & \cdots & G(x_N; c_1, a_1)x_N \\
G(x_1; c_2, a_2) & \cdots & G(x_N; c_2, a_2) \\
\vdots & \ddots & \vdots \\
G(x_1; c_N, a_N) & \cdots & G(x_N; c_N, a_N) \\
\vdots & \ddots & \vdots \\
G(x_1; c_N, a_N)x_1 & \cdots & G(x_N; c_N, a_N)x_N \\
\end{bmatrix}^T \\
_{N(n+1) \times N (6.25)}
\]

To simplify the following analysis, the parameter matrix \( \beta \) and \( q \) for the Mamdani and TSK models are identically expressed using the symbol \( \theta \).

ELM has demonstrated that for SLFNs with any bounded nonconstant piecewise continuous membership function one may randomly choose and fix the hidden node parameters and then analytically determine the output weights of SLFNs and such a SLFN can approximate any continuous target function. It should be noted that in fact this is also valid for fuzzy inference systems. Since we have shown that a fuzzy inference system is equivalent to a SLFN, we can easily show that the SLFNs with activation function \( G(\cdot) \) (cf. equation (6.15)) can approximate any continuous target function as long as the parameters of the membership function \( g \) are randomly generated and the membership function \( g \) is bounded, nonconstant and piecewise continuous. The universal approximation capabilities of such equivalent SLFNs for both Mamdani and TSK models are proved for any bounded nonconstant piecewise continuous membership function using arguments similar to [41]. Thus we have:

**Theorem 6.2.1.** Given any bounded nonconstant piecewise continuous membership function \( g \), if \( \text{span}\{G(x; c, a) : (c, a) \in R^n \times R\} \) is dense in \( L^p \) for every \( p \in [1, \infty) \), then for any continuous target function \( f \) and any randomly generated...
6.2 Structure of Fuzzy Inference Systems

\[ \text{sequence } \{ (c_i, a_i) \}_{i=1}^{\mathcal{N}}, \lim_{\mathcal{N} \to \infty} \| f(x) - f_\mathcal{N}(x) \| = 0 \text{ holds with probability one.} \]

**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 \( \Delta = \| e_{\mathcal{N}-1} \|^2 - \| e_\mathcal{N} \|^2 \), where \( e_{\mathcal{N}-1} = f(x) - f_\mathcal{N}(x) \), then we have

\[
\Delta = \| e_{\mathcal{N}-1} \|^2 - \| f(x) - (f_{\mathcal{N}-1}(x) + \beta_\mathcal{N} G(a_\mathcal{N}, b_\mathcal{N}, x)) \|^2 \\
= \| e_{\mathcal{N}-1} \|^2 - \| e_{\mathcal{N}-1} - \beta_\mathcal{N} G(a_\mathcal{N}, b_\mathcal{N}, x) \|^2 \\
= 2\beta_\mathcal{N} \langle e_{\mathcal{N}-1}, G(a_\mathcal{N}, b_\mathcal{N}, x) \rangle - \beta_\mathcal{N}^2 \| G(a_\mathcal{N}, b_\mathcal{N}, x) \|^2 \\
= \| G(a_\mathcal{N}, b_\mathcal{N}, x) \|^2 \left( \frac{\langle e_{\mathcal{N}-1}, G(a_\mathcal{N}, b_\mathcal{N}, x) \rangle}{\| G(a_\mathcal{N}, b_\mathcal{N}, x) \|^4} \right)^2 - \left( \beta_\mathcal{N} - \frac{\langle e_{\mathcal{N}-1}, G(a_\mathcal{N}, b_\mathcal{N}, x) \rangle}{\| G(a_\mathcal{N}, b_\mathcal{N}, x) \|^2} \right)^2 \quad (6.26)
\]

When \( \beta_\mathcal{N} = \langle e_{\mathcal{N}-1}, G(a_\mathcal{N}, b_\mathcal{N}, x) \rangle \| G(a_\mathcal{N}, b_\mathcal{N}, x) \|^2 \), \( \Delta \) achieves its maximum, that is \( \Delta_{\text{max}} = \langle e_{\mathcal{N}-1}, G(a_\mathcal{N}, b_\mathcal{N}, x) \rangle \| G(a_\mathcal{N}, b_\mathcal{N}, x) \|^2 \geq 0 \). Equivalently when \( \beta_\mathcal{N} = \langle e_{\mathcal{N}-1}, G(a_\mathcal{N}, b_\mathcal{N}, x) \rangle \| G(a_\mathcal{N}, b_\mathcal{N}, x) \|^2 \), \( \| e_\mathcal{N} \| = \| f(x) - (f_{\mathcal{N}-1}(x) + \beta_\mathcal{N} G(a_\mathcal{N}, b_\mathcal{N}, x)) \| \) is minimized. Further when \( \beta_\mathcal{N} = \langle e_{\mathcal{N}-1}, G(a_\mathcal{N}, b_\mathcal{N}, x) \rangle \| G(a_\mathcal{N}, b_\mathcal{N}, x) \|^2 \), we have \( e_{\mathcal{N}-1} G(a_\mathcal{N}, b_\mathcal{N}, x) \) derived as,

\[
\langle e_{\mathcal{N}-1}, G(a_\mathcal{N}, b_\mathcal{N}, x) \rangle = \langle e_{\mathcal{N}-1} - \beta_\mathcal{N} G(a_\mathcal{N}, b_\mathcal{N}, x), G(a_\mathcal{N}, b_\mathcal{N}, x) \rangle \\
= \langle e_{\mathcal{N}-1}, G(a_\mathcal{N}, b_\mathcal{N}, x) \rangle - \beta_\mathcal{N} \langle G(a_\mathcal{N}, b_\mathcal{N}, x), G(a_\mathcal{N}, b_\mathcal{N}, x) \rangle \\
= 0 \quad (6.27)
\]

As such the sequence \( \{ \| e_\mathcal{N} \| \} \) converges and is bound below by zero. Actually when \( \mathcal{N} \to +\infty, \| e_\mathcal{N} \| \to 0 \), that is \( \lim_{\mathcal{N} \to +\infty} \| e_\mathcal{N} \| = 0 \). We can prove this by contradiction method. The sequence \( \{ \| e_\mathcal{N} \| \} \) converges and thus there exists \( r \geq 0 \) such as \( \lim_{\mathcal{N} \to -\infty} \| e_\mathcal{N} \| = r \). If \( r > 0 \), the sequence \( \{ \| e_\mathcal{N} \| \} \) is decreasing and bound below by \( r \), that is \( \| e_\mathcal{N} \| \geq r \) for all \( \mathcal{N} > 0 \). Thus there exists \( \tilde{\mathcal{N}} \) for \( \forall \delta > 0 \) so that
6.2 Structure of Fuzzy Inference Systems

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

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

**Proof.** The validity of the theorem is straightforward based on theorem 6.2.1.

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 \( \text{span}\{G(x; c, a) : (c, a) \in \mathbb{R}^n \times \mathbb{R}\} \) is dense in \( L^p \) for every \( p \in [1, \infty) \), then there exists a finite \( \bar{N} \) (< \( \infty \)) such that for any continuous target function \( f \) and any randomly generated sequence \( \{(c_i, a_i)\}_{i=1}^{\bar{N}} \), \( \|f(x) - f_{\bar{N}}(x)\| < \epsilon \) 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 \( \mathbb{N} = \{(x_i, t_i) | x_i \in \mathbb{R}^n, t_i \in \mathbb{R}^m, i = 1, \cdots, N\} \), membership function \( g \), and fuzzy rule number \( \bar{N} \),

1. **step 1** Randomly assign membership function parameters \( (c_i, a_i), i = 1, \cdots, \bar{N} \).
2. **step 2** Calculate the matrix \( H \) (cf. equation (6.19) and equation (6.25)) for Mamdani and TSK fuzzy models.
3. **step 3** Calculate the parameter matrix \( \theta: \theta = H^\dagger T \).

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 \( \theta \) as,

\[
\theta = H^\dagger T
\]  

(6.28)

where \( H^\dagger \) is the Moore-Penrose generalized inverse of the matrix \( H \) and given by,

\[
H^\dagger = (H^T H)^{-1} H^T
\]  

(6.29)

The equation (6.28) is referred to as the least-squares solution to \( H\theta = T \). 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 \( \mathbb{X}_0 = \{ (x_i, t_i) \}_{i=1}^{N_0}, N_0 \geq \tilde{N} \) is given, the batch Fuzzy-ELM algorithm is used to estimate the parameter matrix \( \theta \) by considering the problem of minimizing \( \| H_0 \theta - T_0 \| \).

For Mamdani fuzzy model,

\[
H_0(\mathbf{c}_1, \cdots, \mathbf{c}_{\tilde{N}}, a_1, \cdots, a_{\tilde{N}}; \mathbf{x}_1, \cdots, \mathbf{x}_{N_0}) = \begin{bmatrix}
G(x_1; c_1, a_1) & \cdots & G(x_1; c_{\tilde{N}}, a_{\tilde{N}}) \\
\vdots & \ddots & \vdots \\
G(x_{N_0}; c_1, a_1) & \cdots & G(x_{N_0}; c_{\tilde{N}}, a_{\tilde{N}})
\end{bmatrix}_{N_0 \times \tilde{N}}
\]  

(6.30)
For TSK fuzzy model, 

\[
H_0(c_1, \cdots, c_N, a_1, \cdots, a_N; x_1, \cdots, x_N) = \begin{bmatrix}
G(x_1; c_1, a_1) & \cdots & G(x_N; c_1, a_1) \\
\vdots & \ddots & \vdots \\
G(x_1; c_1, a_1)x_{1n} & \cdots & G(x_N; c_N, a_N)x_{N0n} \\
G(x_1; c_2, a_2) & \cdots & G(x_N; c_2, a_2) \\
\vdots & \ddots & \vdots \\
G(x_1; c_N, a_N)x_{1n} & \cdots & G(x_N; c_N, a_N)x_{N0n} \\
G(x_1; c_N, a_N) & \cdots & G(x_N; c_N, a_N)
\end{bmatrix}^{T}_{N(N+1) \times N0}
\]

According to theorem 6.2.3, the solution to minimizing \(\|H_0 \theta - T_0\|\) is given by \(\theta^{(0)} = K_0^{-1}H_0^{T}T_0\), where \(K_0^{-1} = H_0^{T}H_0\).

Next when another chunk of data \(N_1 = \{(x_i, t_i)\}_{i=N0+1}^{N_0+N1}\) where \(N_1\) denotes the number of observations in this chunk is given, the minimization problem then becomes as,

\[
\left\| \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} \theta - \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} \right\| = \left(6.32\right)
\]

For Mamdani fuzzy model, 

\[
H_1(c_1, \cdots, c_N, a_1, \cdots, a_N; x_{N0+1}, \cdots, x_{N1}) = \begin{bmatrix}
G(x_{N0+1}; c_1, a_1) & \cdots & G(x_{N0+1}; c_N, a_N) \\
\vdots & \ddots & \vdots \\
G(x_{N1}; c_1, a_1) & \cdots & G(x_{N1}; c_N, a_N)
\end{bmatrix}_{N_1 \times N}
\]

\[\text{ATTENTION: The Singapore Copyright Act applies to the use of this document. Nanyang Technological University Library}\]
6.3 Online Sequential Fuzzy Extreme Learning Machine

For TSK fuzzy model,

\[
H_1(c_1, \cdots, c_{\tilde{N}}, a_1, \cdots, a_{\tilde{N}}; x_{N_0+1}, \cdots, x_{N_1}) = \begin{bmatrix}
G(x_{N_0+1}; c_1, a_1) & \cdots & G(x_{N_1}; c_1, a_1) \\
\vdots & \ddots & \vdots \\
G(x_{N_0+1}; c_1, a_1)x_{(N_0+1)n} & \cdots & G(x_{N_1}; c_1, a_1)x_{N_1n} \\
G(x_{N_0+1}; c_2, a_2) & \cdots & G(x_{N_1}; c_2, a_2) \\
\vdots & \ddots & \vdots \\
G(x_{N_0+1}; c_{\tilde{N}}, a_{\tilde{N}}) & \cdots & G(x_{N_1}; c_{\tilde{N}}, a_{\tilde{N}}) \\
\vdots & \ddots & \vdots \\
G(x_{N_0+1}; c_{\tilde{N}}, a_{\tilde{N}})x_{(N_0+1)n} & \cdots & G(x_{N_1}; c_{\tilde{N}}, a_{\tilde{N}})x_{N_1n}
\end{bmatrix}^{T} N(n+1)_{(n+1)N_1}
\]

Based on both chunks of training data sets $\mathcal{N}_0$ and $\mathcal{N}_1$, the output weight $\theta$ becomes as,

\[
\theta^{(1)} = K_1^{-1} \begin{bmatrix}
H_0 \\
H_1
\end{bmatrix}^{T} \begin{bmatrix}
T_0 \\
T_1
\end{bmatrix} \tag{6.35}
\]

where

\[
K_1 = \begin{bmatrix}
H_0 \\
H_1
\end{bmatrix}^{T} \begin{bmatrix}
H_0 \\
H_1
\end{bmatrix} \tag{6.36}
\]

Furthermore $K_1$ can be written as

\[
K_1 = [H_0^T \ H_1^T] \begin{bmatrix}
H_0 \\
H_1
\end{bmatrix} \tag{6.37}
\]

\[
= K_0 + H_1^T H_1
\]
6.3 Online Sequential Fuzzy Extreme Learning Machine

and

\[
\begin{bmatrix}
H_0 \\
H_1
\end{bmatrix}^T \begin{bmatrix}
T_0 \\
T_1
\end{bmatrix} = H_0^T T_0 + H_1^T T_1
= K_0 K_0^{-1} H_0^T T_0 + H_1^T T_1
= K_0 \theta^{(0)} + H_1^T T_1
= (K_1 - H_1^T H_1) \theta^{(0)} + H_1^T T_1
= K_1 \theta^{(0)} - H_1^T H_1 \theta^{(0)} + H_1^T T_1
\]

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

\[
\theta^{(1)} = K_1^{-1} \begin{bmatrix}
H_0 \\
H_1
\end{bmatrix}^T \begin{bmatrix}
T_0 \\
T_1
\end{bmatrix} = K_1^{-1} (H_0^T T_0 + H_1^T T_1)
= K_1^{-1} (K_1 \theta^{(0)} - H_1^T H_1 \theta^{(0)} + H_1^T T_1)
= \theta^{(0)} + K_1^{-1} H_1^T (T_1 - H_1 \theta^{(0)})
\]

where

\[
K_1 = K_0 + H_1^T H_1
\]

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 \( \theta \), which is similar to the recursive least squares algorithm [23]. Given \((k + 1)\)-th chunk of data set \( S_{k+1} = \{(x_i, t_i)\}_{(\sum_{j=0}^{k+1} N_j)+1} \), where \( k \geq 0 \) and \( N_{k+1} \) denotes the number of observations in the \((k + 1)\)-th chunk, we have

\[
K_{k+1} = K_k + H_{k+1}^T H_{k+1}
\theta^{(k+1)} = \theta^{(k)} + K_{k+1}^{-1} H_{k+1}^T (T_{k+1} - H_{k+1} \theta^{(k)})
\]

\[\text{(6.41)}\]
6.3 Online Sequential Fuzzy Extreme Learning Machine

For Mamdani fuzzy model,

\[
H_{k+1}(c_1, \cdots, c_{\tilde{N}}, a_1, \cdots, a_{\tilde{N}}; x_{\Delta_1}, \cdots, x_{\Delta_2})
= \begin{bmatrix}
G(x_{\Delta_1}; c_1, a_1) & \cdots & G(x_{\Delta_1}; c_{\tilde{N}}, a_{\tilde{N}}) \\
\vdots & \ddots & \vdots \\
G(x_{\Delta_2}; c_1, a_1) & \cdots & G(x_{\Delta_2}; c_{\tilde{N}}, a_{\tilde{N}})
\end{bmatrix}_{N_{k+1} \times \tilde{N}}
\]  

(6.42)

For TSK fuzzy model,

\[
H_{k+1}(c_1, \cdots, c_{\tilde{N}}, a_1, \cdots, a_{\tilde{N}}; x_{\Delta_1}, \cdots, x_{\Delta_2})
= \begin{bmatrix}
G(x_{\Delta_1}; c_1, a_1) & \cdots & G(x_{\Delta_1}; c_1, a_1) \\
\vdots & \ddots & \vdots \\
G(x_{\Delta_1}; c_{n-1}, a_{n-1})x_{\Delta_1} & \cdots & G(x_{\Delta_1}; c_{n-1}, a_{n-1})x_{\Delta_1} \\
\vdots & \ddots & \vdots \\
G(x_{\Delta_1}; c_{n-1}, a_{n-1})x_{\Delta_1} & \cdots & G(x_{\Delta_1}; c_{n-1}, a_{n-1})x_{\Delta_1} \\
\vdots & \ddots & \vdots \\
G(x_{\Delta_1}; c_1, a_1) & \cdots & G(x_{\Delta_1}; c_{\tilde{N}}, a_{\tilde{N}}) \\
\vdots & \ddots & \vdots \\
G(x_{\Delta_1}; c_{\tilde{N}}, a_{\tilde{N}}) & \cdots & G(x_{\Delta_1}; c_{\tilde{N}}, a_{\tilde{N}})
\end{bmatrix}^T_{\tilde{N}(n+1) \times N_{k+1}}
\]  

(6.43)

where \(\Delta_1\) and \(\Delta_2\) denote \(\sum_{j=0}^{k} N_j + 1\) and \(\sum_{j=0}^{k+1} N_j\).

\[
T_1 = \begin{bmatrix}
t^T_{(\sum_{j=0}^{k} N_j)+1} \\
\vdots \\
t^T_{\sum_{j=0}^{k+1} N_j}
\end{bmatrix}_{N_{k+1} \times m}
\]  

(6.44)

In equation 6.41, \(K_{k+1}^{-1}\) is used to compute \(\theta^{(k+1)}\) from \(\theta^{(k)}\) instead of \(K_{k+1}\). The
6.3 Online Sequential Fuzzy Extreme Learning Machine

The update formula for $K_{k+1}^{-1}$ is achieved using the Sherman-Morrison formula [30]

$$
K_{k+1}^{-1} = (K_k + H_{k+1}^T H_{k+1})^{-1}
= K_k^{-1} - K_k^{-1} H_{k+1}^T (I + K_k^{-1} H_{k+1}^T) K_k^{-1} H_{k+1}
$$

(6.45)

By making $P_{k+1} = K_{k+1}^{-1}$, we have the following equations for updating $\theta^{(k+1)}$,

$$
P_{k+1} = P_k - P_k H_{k+1}^T (I + H_{k+1} K_k^{-1} H_{k+1}^T) K_k^{-1} H_{k+1} P_k
$$

$$
\theta^{(k+1)} = \theta^{(k)} + P_{k+1} H_{k+1}^T (T_{k+1} - H_{k+1} \theta^{(k)})
$$

(6.46)

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

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 Žak [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 $\mathbb{R} = \{(x_i, t_i) | x_i \in \mathbb{R}^n, t_i \in \mathbb{R}^m, i = 1, \cdots \}$ arrives sequentially.

**step 1 Initialization Phase:** Initialize the learning using a small chunk of initial training data $\mathbb{R}_0 = \{(x_i, t_i)\}_{i=1}^{N_0}$ from the given training set $\mathbb{R} = \{(x_i, t_i) | x_i \in \mathbb{R}^n, t_i \in \mathbb{R}^m, i = 1, \cdots \}$, $N_0 \geq \tilde{N}$.

1. Assign random centers $c_i$ and width $a_i$ for membership function $g$, $i = 1, \cdots, \tilde{N}$.
2. Calculate the initial matrix $H_0$ (cf. equation (6.30) and equation (6.31)) for Mamdani and TSK models.
3. Estimate the initial parameter matrix $\theta^{(0)} = P_0 H_0^T T_0$, where $P_0 = (H_0^T H_0)^{-1}$ and $T_0 = [t_1, \cdots, t_{N_0}]^T$.
4. Set $k = 0$.

**step 2 Sequential Learning Phase:**

Present the $(k+1)$-th chunk of new observations: $\mathbb{R}_{k+1} = \{(x_i, t_i)\}_{j=(\sum_{j=0}^{k} N_j) + 1}^{\sum_{j=0}^{k+1} N_j}$, where $N_{k+1}$ denotes the number of observations in the $(k+1)$-th chunk, do

1. Calculate the partial matrix $H_{k+1}$ for the $(k+1)$-th chunk of data $\mathbb{R}_{k+1}$ (cf. equation (6.42) and equation (6.43)) for the Mamdani and TSK models.
   
   Set $T_{k+1} = [t_{(\sum_{j=0}^{k} N_j) + 1}, \cdots, t_{\sum_{j=0}^{k+1} N_j}]^T$.

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

   \[
   \begin{align*}
   P_{k+1} &= P_k - P_k H_{k+1}^T (I + H_{k+1} P_k H_{k+1}^T)^{-1} H_{k+1} P_k \\
   \theta^{(k+1)} &= \theta^{(k)} + P_{k+1} H_{k+1}^T (T_{k+1} - H_{k+1} \theta^{(k)})
   \end{align*}
   \] (6.47)


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
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,

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.6482</td>
<td>0.4769</td>
<td>0.6861</td>
<td>0.5289</td>
<td>0.5696</td>
<td>0.4543</td>
</tr>
<tr>
<td>10</td>
<td>0.3272</td>
<td>0.2881</td>
<td>0.3982</td>
<td>0.3044</td>
<td>0.2795</td>
<td>0.2531</td>
</tr>
<tr>
<td>15</td>
<td>0.1863</td>
<td>0.1346</td>
<td>0.2001</td>
<td>0.1456</td>
<td>0.1828</td>
<td>0.1370</td>
</tr>
<tr>
<td>20</td>
<td>0.1672</td>
<td>0.0901</td>
<td>0.1724</td>
<td>0.1031</td>
<td>0.1439</td>
<td>0.0753</td>
</tr>
<tr>
<td>25</td>
<td>0.1459</td>
<td>0.0647</td>
<td>0.1333</td>
<td>0.0638</td>
<td>0.1266</td>
<td>0.0455</td>
</tr>
<tr>
<td>30</td>
<td>0.1368</td>
<td>0.0505</td>
<td>0.1229</td>
<td>0.0495</td>
<td>0.1200</td>
<td>0.0356</td>
</tr>
<tr>
<td>35</td>
<td>0.1218</td>
<td>0.0426</td>
<td>0.1187</td>
<td>0.0414</td>
<td>0.1177</td>
<td>0.0306</td>
</tr>
<tr>
<td>40</td>
<td>0.1150</td>
<td>0.0344</td>
<td>0.1160</td>
<td>0.0353</td>
<td>0.1160</td>
<td>0.0272</td>
</tr>
<tr>
<td>45</td>
<td>0.1139</td>
<td>0.0307</td>
<td>0.1124</td>
<td>0.0311</td>
<td>0.1158</td>
<td>0.0262</td>
</tr>
<tr>
<td>50</td>
<td>0.1127</td>
<td>0.0275</td>
<td>0.1119</td>
<td>0.0289</td>
<td>0.1152</td>
<td>0.0257</td>
</tr>
</tbody>
</table>
6.4 Parameter Determination of OS-Fuzzy-ELM

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
6.4 Parameter Determination of OS-Fuzzy-ELM

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 $N$ 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

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

Figure 6.6: The tendency of training and testing accuracies for Gaussmf and Mamdani fuzzy model according to the varying number of rules
6.4 Parameter Determination of OS-Fuzzy-ELM

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Input Attributes</th>
<th># Classes</th>
<th># Training Data</th>
<th># Testing Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonlinear System Identification</td>
<td>3</td>
<td>-</td>
<td>5,000</td>
<td>200</td>
</tr>
<tr>
<td>California Housing</td>
<td>8</td>
<td>-</td>
<td>10,320</td>
<td>10,320</td>
</tr>
<tr>
<td>DNA</td>
<td>180</td>
<td>3</td>
<td>2,000</td>
<td>1,186</td>
</tr>
</tbody>
</table>

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.
6.5 Performance Evaluation of OS-Fuzzy-ELM

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 \((\epsilon_{\text{max}}, \epsilon_{\text{min}}, \gamma)\), the overlap factor \((\kappa)\) 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 \((\Omega)\). 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 Performance Evaluation of OS-Fuzzy-ELM

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 Simpl\text{e}TS. 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, Simpl_eTS 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, Simpl_eTS 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>
<thead>
<tr>
<th>Algorithms</th>
<th>Gaussmf</th>
<th>Trimf</th>
<th>Cauchymf</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training (RMSE)</td>
<td>Testing (RMSE)</td>
<td># Rules</td>
</tr>
<tr>
<td>OS-Fuzzy-ELM</td>
<td>0.1279</td>
<td>0.0490</td>
<td>30</td>
</tr>
<tr>
<td>ANFIS</td>
<td>0.1311</td>
<td>0.0588</td>
<td>27</td>
</tr>
<tr>
<td>SAFIS</td>
<td>0.1493</td>
<td>0.0533</td>
<td>30</td>
</tr>
<tr>
<td>eTS</td>
<td>0.1620</td>
<td>0.0638</td>
<td>31</td>
</tr>
<tr>
<td>SimpleTS</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>DENFIS</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Figure 6.9: Training time comparison for nonlinear system identification problem based on the same membership functions between different algorithms.
6.5 Performance Evaluation of OS-Fuzzy-ELM

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, Simpl_eTS 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

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Gaussmf</th>
<th>Trimf</th>
<th>Cauchymf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training (RMSE)</td>
<td>Testing (RMSE)</td>
<td># Training Times (Secs)</td>
<td>Training (RMSE)</td>
</tr>
<tr>
<td>OS-Fuzzy-ELM</td>
<td>0.1311</td>
<td>0.1344</td>
<td>5</td>
</tr>
<tr>
<td>ANFIS</td>
<td>0.1391</td>
<td>0.1395</td>
<td>8</td>
</tr>
<tr>
<td>SAFIS</td>
<td>0.1346</td>
<td>0.1355</td>
<td>8</td>
</tr>
<tr>
<td>eTS</td>
<td>0.1334</td>
<td>0.1358</td>
<td>8</td>
</tr>
<tr>
<td>Simpl.eTS</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>DENSIS</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
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, Simpl_eTS 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>
<thead>
<tr>
<th>Algorithms</th>
<th>Training Accuracy (%)</th>
<th>Testing Accuracy (%)</th>
<th># Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS-Fuzzy-ELM</td>
<td>96.61</td>
<td>94.28</td>
<td>5</td>
</tr>
<tr>
<td>FSVMNN [78]</td>
<td>93.60</td>
<td>94.20</td>
<td>334</td>
</tr>
<tr>
<td>FNN [69]</td>
<td>N/A</td>
<td>89.75</td>
<td>N/A</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Chunk Sizes</th>
<th>Gaussmf</th>
<th>Trimf</th>
<th>Cauchymf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training (RMSE)</td>
<td>Testing (RMSE)</td>
<td># Rules</td>
<td>Training Times (Secs)</td>
</tr>
<tr>
<td>Batch</td>
<td>0.1278</td>
<td>0.0491</td>
<td>30</td>
</tr>
<tr>
<td>1-by-1</td>
<td>0.1279</td>
<td>0.0490</td>
<td>30</td>
</tr>
<tr>
<td>20-by-20</td>
<td>0.1281</td>
<td>0.0491</td>
<td>30</td>
</tr>
<tr>
<td>[10, 30]</td>
<td>0.1269</td>
<td>0.0492</td>
<td>30</td>
</tr>
</tbody>
</table>
Table 6.8: Performance comparison implemented in different chunk sizes for California housing prediction

<table>
<thead>
<tr>
<th>Chunk Sizes</th>
<th>Gaussmf</th>
<th></th>
<th>Trimf</th>
<th></th>
<th>Cauchymf</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training (RMSE)</td>
<td>Testing (RMSE)</td>
<td># Rules</td>
<td>Training Times (Secs)</td>
<td>Testing (RMSE)</td>
<td># Rules</td>
</tr>
<tr>
<td>Batch</td>
<td>0.1311</td>
<td>0.1340</td>
<td>5</td>
<td>0.4410</td>
<td>0.1321</td>
<td>0.1357</td>
</tr>
<tr>
<td>1-by-1</td>
<td>0.1311</td>
<td>0.1344</td>
<td>5</td>
<td>3.7616</td>
<td>0.1317</td>
<td>0.1363</td>
</tr>
<tr>
<td>20-by-20</td>
<td>0.1312</td>
<td>0.1344</td>
<td>5</td>
<td>0.5390</td>
<td>0.1321</td>
<td>0.1365</td>
</tr>
<tr>
<td>[10, 30]</td>
<td>0.1314</td>
<td>0.1340</td>
<td>5</td>
<td>0.6092</td>
<td>0.1313</td>
<td>0.1365</td>
</tr>
</tbody>
</table>
Table 6.9: Performance comparison implemented in different chunk sizes for DNA classification

<table>
<thead>
<tr>
<th>Chunk Sizes</th>
<th>Gaussmf</th>
<th></th>
<th></th>
<th>Trmf</th>
<th></th>
<th></th>
<th>Cauchymf</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training (%)</td>
<td>Testing (%)</td>
<td># Rules</td>
<td>Training Times (Secs)</td>
<td>Training (%)</td>
<td>Testing (%)</td>
<td># Rules</td>
<td>Training Times (Secs)</td>
</tr>
<tr>
<td>Batch</td>
<td>96.72</td>
<td>94.33</td>
<td>5</td>
<td>24.419</td>
<td>96.61</td>
<td>94.29</td>
<td>5</td>
<td>23.062</td>
</tr>
<tr>
<td>1-by-1</td>
<td>96.61</td>
<td>94.28</td>
<td>5</td>
<td>2164.8</td>
<td>96.56</td>
<td>94.28</td>
<td>5</td>
<td>2177.8</td>
</tr>
<tr>
<td>20-by-20</td>
<td>96.65</td>
<td>94.14</td>
<td>5</td>
<td>128.515</td>
<td>96.33</td>
<td>94.15</td>
<td>5</td>
<td>127.85</td>
</tr>
<tr>
<td>[10,30]</td>
<td>96.66</td>
<td>94.16</td>
<td>5</td>
<td>126.487</td>
<td>96.43</td>
<td>94.12</td>
<td>5</td>
<td>123.92</td>
</tr>
</tbody>
</table>
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 | | | Cauchynmf | | |
|-------------|--------------|---------|---------|---------|---------|---------|---------|
|             | Training (RMSE) | Testing (RMSE) | # Rules | Training Times (Secs) | Testing (RMSE) | # Rules | Training Times (Secs) | 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 | 13.114  | 0.1271  | 0.0525  | 30 | 13.195  | 0.1217  | 0.0401  | 30 | 1.2004 |
|             | Min          | 0.1416  | 0.0624  | 30 | 17.278  | 0.1500  | 0.0629  | 30 | 1.6980  | 0.1414  | 0.0682  | 30 | 1.6927 |
| [10, 30]    | Product      | 0.1269  | 0.0492  | 30 | 13.344  | 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

<table>
<thead>
<tr>
<th>Chunk Sizes</th>
<th>T-norm Types</th>
<th>Gaussmf</th>
<th>Trimf</th>
<th>Cauchymf</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training (RMSE)</td>
<td>Testing (RMSE)</td>
<td># Rules</td>
<td>Training (RMSE)</td>
</tr>
<tr>
<td>Batch</td>
<td>Product</td>
<td>0.1311</td>
<td>0.1340</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1330</td>
<td>0.1365</td>
<td>5</td>
</tr>
<tr>
<td>1-by-1</td>
<td>Product</td>
<td>0.1311</td>
<td>0.1344</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1330</td>
<td>0.1364</td>
<td>5</td>
</tr>
<tr>
<td>20-by-20</td>
<td>Product</td>
<td>0.1312</td>
<td>0.1344</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1330</td>
<td>0.1363</td>
<td>5</td>
</tr>
<tr>
<td>[10, 30]</td>
<td>Product</td>
<td>0.1314</td>
<td>0.1340</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1334</td>
<td>0.1352</td>
<td>5</td>
</tr>
</tbody>
</table>
Table 6.12: Performance comparison between Product and Min T-norm operators for DNA classification

<table>
<thead>
<tr>
<th>Chunk Sizes</th>
<th>T-norm Types</th>
<th>Gaussmf</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Cauchymf</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Training (%)</td>
<td>Testing (%)</td>
<td># Rules</td>
<td>Training Times (Secs)</td>
<td></td>
<td>Training (%)</td>
<td>Testing (%)</td>
</tr>
<tr>
<td>Batch</td>
<td>Product</td>
<td>96.72</td>
<td>94.33</td>
<td>5</td>
<td>24.419</td>
<td></td>
<td>96.61</td>
<td>94.29</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>97.35</td>
<td>92.49</td>
<td>5</td>
<td>24.712</td>
<td></td>
<td>97.05</td>
<td>92.68</td>
</tr>
<tr>
<td>1-by-1</td>
<td>Product</td>
<td>96.61</td>
<td>94.28</td>
<td>5</td>
<td>2164.8</td>
<td></td>
<td>96.56</td>
<td>94.28</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>97.52</td>
<td>92.59</td>
<td>5</td>
<td>2192.2</td>
<td></td>
<td>97.15</td>
<td>92.41</td>
</tr>
<tr>
<td>20-by-20</td>
<td>Product</td>
<td>96.65</td>
<td>94.14</td>
<td>5</td>
<td>128.515</td>
<td></td>
<td>96.33</td>
<td>94.15</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>97.96</td>
<td>92.91</td>
<td>5</td>
<td>129.21</td>
<td></td>
<td>97.50</td>
<td>92.40</td>
</tr>
<tr>
<td>[10, 30]</td>
<td>Product</td>
<td>96.66</td>
<td>94.16</td>
<td>5</td>
<td>126.487</td>
<td></td>
<td>96.43</td>
<td>94.12</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>97.01</td>
<td>92.99</td>
<td>5</td>
<td>126.09</td>
<td></td>
<td>97.97</td>
<td>92.41</td>
</tr>
</tbody>
</table>
6.5 Performance Evaluation of OS-Fuzzy-ELM

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 Mamdani models for these problems.

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

<table>
<thead>
<tr>
<th>Problems</th>
<th>Fuzzy models</th>
<th># Input Dimension</th>
<th># Output Dimension</th>
<th># Rules</th>
<th># Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonlinear System Identification</td>
<td>TSK</td>
<td>3</td>
<td>1</td>
<td>30</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td></td>
<td></td>
<td>110</td>
<td>110</td>
</tr>
<tr>
<td>California Housing Prediction</td>
<td>TSK</td>
<td>8</td>
<td>1</td>
<td>5</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td></td>
<td></td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>DNA Classification</td>
<td>TSK</td>
<td>180</td>
<td>3</td>
<td>5</td>
<td>2715</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td></td>
<td></td>
<td>200</td>
<td>600</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Chunk Sizes</th>
<th>Fuzzy Models</th>
<th>Gaussmf</th>
<th></th>
<th>Cauchymf</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training (RMSE)</td>
<td>Testing (RMSE)</td>
<td># Rules</td>
<td>Training Times (Secs)</td>
</tr>
<tr>
<td>Batch</td>
<td>TSK</td>
<td>0.1278</td>
<td>0.0491</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td>0.1285</td>
<td>0.0494</td>
<td>110</td>
</tr>
<tr>
<td>1-by-1</td>
<td>TSK</td>
<td>0.1279</td>
<td>0.0490</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td>0.1286</td>
<td>0.0495</td>
<td>110</td>
</tr>
<tr>
<td>20-by-20</td>
<td>TSK</td>
<td>0.1281</td>
<td>0.0491</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td>0.1289</td>
<td>0.0493</td>
<td>110</td>
</tr>
<tr>
<td>[10, 30]</td>
<td>TSK</td>
<td>0.1269</td>
<td>0.0492</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td>0.1282</td>
<td>0.0483</td>
<td>110</td>
</tr>
<tr>
<td>Chunk Sizes</td>
<td>Fuzzy Models</td>
<td>Gaussmf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>--------------</td>
<td>---------</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(RMSE)</td>
<td>(RMSE)</td>
<td>(Secs)</td>
</tr>
<tr>
<td>Batch</td>
<td>TSK</td>
<td>0.1311</td>
<td>0.1340</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td>0.1330</td>
<td>0.1351</td>
<td>110</td>
</tr>
<tr>
<td>1-by-1</td>
<td>TSK</td>
<td>0.1311</td>
<td>0.1344</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td>0.1338</td>
<td>0.1348</td>
<td>35</td>
</tr>
<tr>
<td>20-by-20</td>
<td>TSK</td>
<td>0.1312</td>
<td>0.1344</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td>0.1332</td>
<td>0.1351</td>
<td>35</td>
</tr>
<tr>
<td>[10, 30]</td>
<td>TSK</td>
<td>0.1314</td>
<td>0.1340</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td>0.1336</td>
<td>0.1350</td>
<td>35</td>
</tr>
</tbody>
</table>

Table 6.15: Performance comparison between TSK and Mamdani fuzzy models for California housing prediction
6.5 Performance Evaluation of OS-Fuzzy-ELM

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

<table>
<thead>
<tr>
<th>Chunk Sizes</th>
<th>Fuzzy Models</th>
<th>Gaussmf</th>
<th>Trimf</th>
<th>Cauchymf</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training (%)</td>
<td>Testing (%)</td>
<td># Rules</td>
<td>Training Times (Secs)</td>
</tr>
<tr>
<td>Batch</td>
<td>TSK</td>
<td>96.72</td>
<td>94.33</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td>96.73</td>
<td>94.33</td>
<td>200</td>
</tr>
<tr>
<td>1-by-1</td>
<td>TSK</td>
<td>96.61</td>
<td>94.28</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td>96.55</td>
<td>94.62</td>
<td>200</td>
</tr>
<tr>
<td>20-by-20</td>
<td>TSK</td>
<td>96.65</td>
<td>94.14</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td>96.44</td>
<td>94.43</td>
<td>200</td>
</tr>
<tr>
<td>[10, 30]</td>
<td>TSK</td>
<td>96.66</td>
<td>94.16</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Mamdani</td>
<td>96.45</td>
<td>94.60</td>
<td>200</td>
</tr>
</tbody>
</table>
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](image-url)
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.

\[ q \quad Q \quad \theta \quad \dot{h} \quad h \quad \dot{v} \quad v \]

\[ k_e \rightarrow \delta_e \]

\[ k_r \rightarrow \delta_r \]

Figure 7.2: OS-Fuzzy-ELM implementation block for the longitudinal 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)
7.1 Single Surface Failure

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>OS-Fuzzy-ELM</th>
<th>EMRAN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gaussmf</td>
<td>Cauchymf</td>
</tr>
<tr>
<td>Trajectory error (RMSE)</td>
<td>8.8195</td>
<td>8.8364</td>
</tr>
<tr>
<td>Velocity error (RMSE)</td>
<td>3.0087</td>
<td>3.1929</td>
</tr>
<tr>
<td># of rules/neurons for $\delta_e$</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td># of rules/neurons for $\delta_a$</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

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 ($\beta$) and lateral position (Y) during the landing phase for the OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC.
7.1 Single Surface Failure

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 -10deg
Figure 7.5: Comparison of OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC for the sideslip angle ($\beta$) 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 \((\beta)\). 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 \((\delta_{c-left-t})\) and right elevator \((\delta_{c-right-t})\) along with the BTFC \((\delta_{c-right-c})\) and OS-Fuzzy-ELM/EMRAN components \((\delta_{c-right-f/n})\) of the control signals. The rule and neuron history based on OS-Fuzzy-ELM and EMRAN for the elevator \(\delta_c\) 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.
7.1 Single Surface Failure

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.

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

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.
7.1 Single Surface Failure

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
7.1 Single Surface Failure

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

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](image)

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 Two Surface Failures

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>
<thead>
<tr>
<th>Algorithms</th>
<th>OS-Fuzzy-ELM</th>
<th>EMRAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory error (RMSE)</td>
<td>8.8947</td>
<td>8.8926</td>
</tr>
<tr>
<td>Velocity error (RMSE)</td>
<td>2.8508</td>
<td>2.8663</td>
</tr>
<tr>
<td># of rules/neurons for $\delta_e$</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td># of rules/neurons for $\delta_a$</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

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 ($\beta$) 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.
7.2 Two Surface Failures

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.
7.2 Two Surface Failures

Figure 7.13: Comparison of OS-Fuzzy-ELM aided BTFC, EMRAN aided BTFC and BTFC for the sideslip angle ($\beta$) 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 ($\beta$). Table 7.2 shows the RMS trajectory errors along with the number of rules/neurons for OS-Fuzzy-ELM aided
7.2 Two Surface Failures

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 ($\delta_{e-right-t}$).

![Graph showing deflection and rules/neurons comparison]

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

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 $\delta_e$ 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.
7.2 Two Surface Failures

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

![Graph showing control signals for left and right ailerons under specific conditions.](image)

**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
Two Surface Failures

○, + and ×.

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 Two Surface Failures

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 ◦, + and ×.

![Graph showing fault tolerance under left elevator and right aileron stuck conditions](image)

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.
7.2 Two Surface Failures

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 ($\beta$) (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 (\(\beta\)) and lateral position (Y) during the landing phase for the OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC.

![Chart showing altitude and velocity comparison](chart.png)

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
8.1 Single Surface Failure

Figure 8.2: Comparison of OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC for the sideslip angle ($\beta$) 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

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>OS-Fuzzy-ELM aided BTFC</th>
<th>SAFIS aided BTFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory error (RMSE)</td>
<td>8.8364</td>
<td>8.8573</td>
</tr>
<tr>
<td>Velocity error (RMSE)</td>
<td>3.1929</td>
<td>3.4243</td>
</tr>
<tr>
<td># of rules for $\delta_e$</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td># of rules for $\delta_a$</td>
<td>6</td>
<td>14</td>
</tr>
</tbody>
</table>

Figure 8.3 shows the deflection of the left elevator ($\delta_{e-left}$) and right elevator ($\delta_{e-right}$) 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.
8.1 Single Surface Failure

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.
8.1 Single Surface Failure

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
8.1 Single Surface Failure

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).
8.2 Two Surface Failures

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).

![Diagram of failure tolerance under left elevator stuck conditions]

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 Two Surface Failures

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 -10deg and the left aileron is stuck at +10deg. Figure 8.7 and Figure 8.8 give the altitude (h), the velocity ($V_T$), sideslip angle ($\beta$) and lateral position (Y) during the landing phase for the OS-Fuzzy-ELM aided BTFC and SAFIS aided BTFC.

![Graph showing altitude and velocity comparison](image)

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 -10deg and the left aileron stuck at 10deg
8.2 Two Surface Failures

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.
8.2 Two Surface Failures

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>
<thead>
<tr>
<th>Algorithms</th>
<th>OS-Fuzzy-ELM aided BTFC</th>
<th>SAFIS aided BTFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory error (RMSE)</td>
<td>8.8926</td>
<td>8.8978</td>
</tr>
<tr>
<td>Velocity error (RMSE)</td>
<td>2.8663</td>
<td>3.2761</td>
</tr>
<tr>
<td># of rules for $\delta_e$</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td># of rules for $\delta_a$</td>
<td>6</td>
<td>22</td>
</tr>
</tbody>
</table>

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.
8.2 Two Surface Failures

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.

Figure 8.9: Left and right elevator control signals under the left elevator stuck at -10deg and the left aileron stuck at 10deg
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-
8.2 Two Surface Failures

down pillbox requirements is represented by ◊ and o.

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
8.2 Two Surface Failures

tolerant regions each point which indicates a successful landing meeting the touchdown pillbox requirements is represented by ♦ and ○.

![Diagram showing failure tolerance under left elevator and right aileron stuck conditions](image)

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 autolanding fault-tolerant control problem under actuator failures and severe winds. The following points are achieved by comparing OS-Fuzzy-ELM with SAFIS.
8.2 Two Surface Failures

(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 autolanding 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 autolanding 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.
9.1 Conclusions

- 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 autolanding 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 ($\beta$).

- 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.
9.2 Recommendations for Future Work

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 autolanding 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].

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.
9.2 Recommendations for Future Work

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 Cacuhy 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 $H_{\infty}$ 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 $H_2/H_{\infty}$ control method instead of $H_{\infty}$ 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.
A.1 Graphs for the Membership Functions

(c) Gaussmf

(d) Gauss2mf

(e) Cauchymf

(f) Pimf
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

\[
C = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
-0.0152 & 0.0195 & 0.0666 & 0 & 0 & 0 \\
0.0121 & 0 & 0 & -0.1940 & 1 & 0
\end{bmatrix}
\]

\[
D = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
-0.0002 & 0.0034 \\
0 & 0
\end{bmatrix}
\]

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

\[ D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \]
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

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th># rules $\tilde{N}$</th>
<th>Training RMSE</th>
<th>Testing RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psigmf</td>
<td>5</td>
<td>0.6287</td>
<td>0.4889</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.2981</td>
<td>0.2622</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.1813</td>
<td>0.1325</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.1482</td>
<td>0.0793</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.1285</td>
<td>0.0482</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.1206</td>
<td>0.0352</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>0.1170</td>
<td>0.0284</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.1151</td>
<td>0.0260</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>0.1140</td>
<td>0.0254</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.1131</td>
<td>0.0251</td>
</tr>
<tr>
<td>Dsigmf</td>
<td>5</td>
<td>0.6133</td>
<td>0.4334</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.3325</td>
<td>0.2768</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.2062</td>
<td>0.1628</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.1605</td>
<td>0.1044</td>
</tr>
</tbody>
</table>

Continued on next page
### Table D.1: (continued)

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th># (N) Rules</th>
<th>Training (RMSE)</th>
<th>Testing (RMSE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gauss2mf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.5733</td>
<td>0.4600</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.2897</td>
<td>0.2631</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.1878</td>
<td>0.1343</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.1423</td>
<td>0.0688</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.1268</td>
<td>0.0483</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.1212</td>
<td>0.0431</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>0.1171</td>
<td>0.0360</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.1151</td>
<td>0.0338</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>0.1138</td>
<td>0.0307</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.1129</td>
<td>0.0294</td>
</tr>
<tr>
<td>Trapmf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.6890</td>
<td>0.5329</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.3703</td>
<td>0.3312</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.2485</td>
<td>0.2157</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.1728</td>
<td>0.1128</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.1475</td>
<td>0.0782</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.1353</td>
<td>0.0467</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>0.1251</td>
<td>0.0409</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.1199</td>
<td>0.0366</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>0.1168</td>
<td>0.0324</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.1146</td>
<td>0.0306</td>
</tr>
<tr>
<td>Pimf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.7117</td>
<td>0.5118</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.3696</td>
<td>0.3187</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.2306</td>
<td>0.1885</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.1647</td>
<td>0.1071</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.1431</td>
<td>0.0764</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.1206</td>
<td>0.0363</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>0.1173</td>
<td>0.0338</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.1153</td>
<td>0.0323</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>0.1136</td>
<td>0.0312</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.1127</td>
<td>0.0292</td>
</tr>
</tbody>
</table>
Table D.2: Effect of parameter $\tilde{N}$ on system performance (training and testing accuracies) for Mamdani model

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Chunk Sizes</th>
<th>Training (RMSE)</th>
<th>Testing (RMSE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psigmf</td>
<td>20</td>
<td>0.6006</td>
<td>0.4523</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.3546</td>
<td>0.2887</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.2666</td>
<td>0.2198</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.2032</td>
<td>0.1530</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>0.1747</td>
<td>0.1142</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>0.1531</td>
<td>0.0838</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.1433</td>
<td>0.0695</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>0.1347</td>
<td>0.0541</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.1291</td>
<td>0.0456</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>0.1252</td>
<td>0.0412</td>
</tr>
<tr>
<td>Dsigmf</td>
<td>20</td>
<td>0.5459</td>
<td>0.3939</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.3466</td>
<td>0.2974</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.2487</td>
<td>0.2082</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.2328</td>
<td>0.1920</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>0.2118</td>
<td>0.1707</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>0.1882</td>
<td>0.1367</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.1579</td>
<td>0.1010</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>0.1398</td>
<td>0.0718</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.1343</td>
<td>0.0564</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>0.1298</td>
<td>0.0471</td>
</tr>
<tr>
<td>Gauss2mf</td>
<td>20</td>
<td>0.5320</td>
<td>0.4294</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.3167</td>
<td>0.2757</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.2364</td>
<td>0.1833</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.1863</td>
<td>0.1228</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>0.1592</td>
<td>0.0903</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>0.1401</td>
<td>0.0680</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.1312</td>
<td>0.0560</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>0.1257</td>
<td>0.0500</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.1215</td>
<td>0.0450</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>0.1191</td>
<td>0.0418</td>
</tr>
<tr>
<td>Trapmf</td>
<td>20</td>
<td>0.5609</td>
<td>0.4415</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.4206</td>
<td>0.3725</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.3196</td>
<td>0.2591</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.2372</td>
<td>0.1927</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>0.1796</td>
<td>0.1109</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>0.1550</td>
<td>0.0896</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.1463</td>
<td>0.0779</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>0.1352</td>
<td>0.0609</td>
</tr>
</tbody>
</table>
D.2 Results for Chunk-by-Chunk Mode

Table D.2: (continued)

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th># rules N</th>
<th>Training (RMSE)</th>
<th>Testing (RMSE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pimf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.5527</td>
<td>0.4219</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>0.3408</td>
<td>0.2920</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>0.2394</td>
<td>0.1858</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0.1961</td>
<td>0.1362</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>0.1695</td>
<td>0.1056</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>0.1517</td>
<td>0.0844</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>0.1426</td>
<td>0.0747</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>0.1330</td>
<td>0.0654</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0.1276</td>
<td>0.0508</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>0.1192</td>
<td>0.0464</td>
<td></td>
</tr>
</tbody>
</table>

Table D.3: Performance comparison implemented in different chunk sizes for nonlinear system identification

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Chunk Sizes</th>
<th>Training (RMSE)</th>
<th>Testing (RMSE)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psigmf</td>
<td>batch</td>
<td>0.1271</td>
<td>0.0525</td>
<td>30</td>
<td>1.2438</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>0.1271</td>
<td>0.0525</td>
<td>30</td>
<td>14.750</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>0.1271</td>
<td>0.0525</td>
<td>30</td>
<td>1.3195</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>0.1271</td>
<td>0.0533</td>
<td>30</td>
<td>1.3896</td>
</tr>
<tr>
<td>Dsigmf</td>
<td>batch</td>
<td>0.1258</td>
<td>0.0513</td>
<td>30</td>
<td>1.2738</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>0.1260</td>
<td>0.0519</td>
<td>30</td>
<td>14.775</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>0.1271</td>
<td>0.0525</td>
<td>30</td>
<td>1.3195</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>0.1281</td>
<td>0.0518</td>
<td>30</td>
<td>1.4223</td>
</tr>
<tr>
<td>Gauss2mf</td>
<td>batch</td>
<td>0.1346</td>
<td>0.0533</td>
<td>30</td>
<td>1.1704</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>0.1319</td>
<td>0.0534</td>
<td>30</td>
<td>14.942</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>0.1320</td>
<td>0.0532</td>
<td>30</td>
<td>1.2710</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>0.1345</td>
<td>0.0537</td>
<td>30</td>
<td>1.3432</td>
</tr>
<tr>
<td>Trapmf</td>
<td>batch</td>
<td>0.1308</td>
<td>0.0528</td>
<td>30</td>
<td>1.1351</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>0.1308</td>
<td>0.0528</td>
<td>30</td>
<td>14.969</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>0.1308</td>
<td>0.0528</td>
<td>30</td>
<td>1.2166</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>0.1305</td>
<td>0.0527</td>
<td>30</td>
<td>1.3252</td>
</tr>
</tbody>
</table>
### Table D.3: (continued)

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Chunk Sizes</th>
<th>Training (RMSE)</th>
<th>Testing (RMSE)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pimf</td>
<td>batch</td>
<td>0.1328</td>
<td>0.0548</td>
<td>30</td>
<td>1.1351</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>0.1328</td>
<td>0.0548</td>
<td>30</td>
<td>15.029</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>0.1328</td>
<td>0.0548</td>
<td>30</td>
<td>1.2331</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>0.1331</td>
<td>0.0562</td>
<td>30</td>
<td>1.3228</td>
</tr>
</tbody>
</table>

Table D.4: Performance comparison implemented in different chunk sizes for California housing prediction

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Chunk Sizes</th>
<th>Training (RMSE)</th>
<th>Testing (RMSE)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psigmf</td>
<td>batch</td>
<td>0.1308</td>
<td>0.1334</td>
<td>5</td>
<td>0.5200</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>0.1308</td>
<td>0.1334</td>
<td>5</td>
<td>3.8778</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>0.1309</td>
<td>0.1333</td>
<td>5</td>
<td>0.5891</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>0.1311</td>
<td>0.1338</td>
<td>5</td>
<td>0.6852</td>
</tr>
<tr>
<td>Dsigmf</td>
<td>batch</td>
<td>0.1303</td>
<td>0.1335</td>
<td>5</td>
<td>0.5506</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>0.1313</td>
<td>0.1340</td>
<td>5</td>
<td>3.9187</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>0.1315</td>
<td>0.1342</td>
<td>5</td>
<td>0.6385</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>0.1318</td>
<td>0.1347</td>
<td>5</td>
<td>0.7219</td>
</tr>
<tr>
<td>Gauss2mf</td>
<td>batch</td>
<td>0.1314</td>
<td>0.1336</td>
<td>5</td>
<td>0.4475</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>0.1316</td>
<td>0.1339</td>
<td>5</td>
<td>3.9991</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>0.1315</td>
<td>0.1342</td>
<td>5</td>
<td>0.6385</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>0.1310</td>
<td>0.1337</td>
<td>5</td>
<td>0.6146</td>
</tr>
<tr>
<td>Trapmf</td>
<td>batch</td>
<td>0.1305</td>
<td>0.1333</td>
<td>5</td>
<td>0.4322</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>0.1312</td>
<td>0.1328</td>
<td>5</td>
<td>3.9528</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>0.1309</td>
<td>0.1331</td>
<td>5</td>
<td>0.4991</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>0.1305</td>
<td>0.1330</td>
<td>5</td>
<td>0.5943</td>
</tr>
<tr>
<td>Pimf</td>
<td>batch</td>
<td>0.1318</td>
<td>0.1350</td>
<td>5</td>
<td>0.4431</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>0.1312</td>
<td>0.1348</td>
<td>5</td>
<td>4.0578</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>0.1316</td>
<td>0.1345</td>
<td>5</td>
<td>0.6097</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>0.1316</td>
<td>0.1345</td>
<td>5</td>
<td>0.6097</td>
</tr>
</tbody>
</table>
D.3 Results for Different T-norm Operators

Table D.5: Performance comparison implemented in different chunk sizes for DNA classification

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Chunk Sizes</th>
<th>Training (%)</th>
<th>Testing (%)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psigmf</td>
<td>batch</td>
<td>96.98</td>
<td>94.38</td>
<td>5</td>
<td>25.671</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>96.55</td>
<td>94.94</td>
<td>5</td>
<td>2537.2</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>96.85</td>
<td>94.34</td>
<td>5</td>
<td>128.99</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>96.75</td>
<td>94.35</td>
<td>5</td>
<td>123.35</td>
</tr>
<tr>
<td>Dsigmf</td>
<td>batch</td>
<td>97.11</td>
<td>94.28</td>
<td>5</td>
<td>25.025</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>96.94</td>
<td>94.50</td>
<td>5</td>
<td>2468.0</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>96.45</td>
<td>94.15</td>
<td>5</td>
<td>129.08</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>96.49</td>
<td>94.13</td>
<td>5</td>
<td>126.40</td>
</tr>
<tr>
<td>Gauss2mf</td>
<td>batch</td>
<td>96.80</td>
<td>94.37</td>
<td>5</td>
<td>23.274</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>96.60</td>
<td>94.25</td>
<td>5</td>
<td>2182.8</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>96.37</td>
<td>94.19</td>
<td>5</td>
<td>128.14</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>96.56</td>
<td>94.16</td>
<td>5</td>
<td>123.88</td>
</tr>
<tr>
<td>Trapmf</td>
<td>batch</td>
<td>96.85</td>
<td>94.31</td>
<td>5</td>
<td>22.766</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>96.79</td>
<td>94.26</td>
<td>5</td>
<td>2180.9</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>96.68</td>
<td>94.30</td>
<td>5</td>
<td>122.38</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>96.68</td>
<td>94.30</td>
<td>5</td>
<td>122.38</td>
</tr>
<tr>
<td>Pimf</td>
<td>batch</td>
<td>96.85</td>
<td>94.26</td>
<td>5</td>
<td>22.431</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>96.84</td>
<td>94.77</td>
<td>5</td>
<td>2185.0</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>96.74</td>
<td>94.23</td>
<td>5</td>
<td>124.45</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>96.74</td>
<td>94.23</td>
<td>5</td>
<td>124.45</td>
</tr>
</tbody>
</table>

D.3 Results for Different T-norm Operators

Table D.6: Performance comparison between Product and Min T-norm operators for nonlinear system identification

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Chunk Sizes</th>
<th>T-norm Types</th>
<th>Training RMSE</th>
<th>Testing RMSE</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psigmf</td>
<td>batch</td>
<td>Product</td>
<td>0.1271</td>
<td>0.0525</td>
<td>30</td>
<td>1.2438</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1410</td>
<td>0.0689</td>
<td>30</td>
<td>1.7109</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>Product</td>
<td>0.1271</td>
<td>0.0525</td>
<td>30</td>
<td>14.750</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1440</td>
<td>0.0616</td>
<td>30</td>
<td>15.201</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>Product</td>
<td>0.1271</td>
<td>0.0525</td>
<td>30</td>
<td>1.3195</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1449</td>
<td>0.0649</td>
<td>30</td>
<td>1.8054</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>Product</td>
<td>0.1271</td>
<td>0.0533</td>
<td>30</td>
<td>1.3896</td>
</tr>
</tbody>
</table>

Continued on next page
### Table D.6: (continued)

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Chunk Sizes</th>
<th>T-norm Types</th>
<th>Training (RMSE)</th>
<th>Testing (RMSE)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dsigmf</td>
<td>batch</td>
<td>Min</td>
<td>0.1431</td>
<td>0.0667</td>
<td>30</td>
<td>1.9198</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1408</td>
<td>0.0674</td>
<td>30</td>
<td>1.7544</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>Product</td>
<td>0.1260</td>
<td>0.0519</td>
<td>30</td>
<td>14.775</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1420</td>
<td>0.0683</td>
<td>30</td>
<td>15.167</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>Product</td>
<td>0.1271</td>
<td>0.0525</td>
<td>30</td>
<td>1.3195</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1427</td>
<td>0.0695</td>
<td>30</td>
<td>1.8400</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>Product</td>
<td>0.1281</td>
<td>0.0518</td>
<td>30</td>
<td>1.4223</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1405</td>
<td>0.0685</td>
<td>30</td>
<td>1.9404</td>
<td></td>
</tr>
<tr>
<td>Gauss2mf</td>
<td>batch</td>
<td>Product</td>
<td>0.1346</td>
<td>0.0533</td>
<td>30</td>
<td>1.1704</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1422</td>
<td>0.0614</td>
<td>30</td>
<td>1.6522</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>Product</td>
<td>0.1319</td>
<td>0.0534</td>
<td>30</td>
<td>14.942</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1424</td>
<td>0.0649</td>
<td>30</td>
<td>15.392</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>Product</td>
<td>0.1320</td>
<td>0.0532</td>
<td>30</td>
<td>1.2710</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1430</td>
<td>0.0660</td>
<td>30</td>
<td>1.7496</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>Product</td>
<td>0.1345</td>
<td>0.0537</td>
<td>30</td>
<td>1.3432</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1440</td>
<td>0.0634</td>
<td>30</td>
<td>1.8243</td>
<td></td>
</tr>
<tr>
<td>Trapmf</td>
<td>batch</td>
<td>Product</td>
<td>0.1308</td>
<td>0.0528</td>
<td>30</td>
<td>1.1351</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1408</td>
<td>0.0637</td>
<td>30</td>
<td>1.5503</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>Product</td>
<td>0.1308</td>
<td>0.0528</td>
<td>30</td>
<td>14.969</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1403</td>
<td>0.0616</td>
<td>30</td>
<td>15.264</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>Product</td>
<td>0.1308</td>
<td>0.0528</td>
<td>30</td>
<td>1.2166</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1404</td>
<td>0.0679</td>
<td>30</td>
<td>1.7150</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>Product</td>
<td>0.1305</td>
<td>0.0527</td>
<td>30</td>
<td>1.3252</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1406</td>
<td>0.0664</td>
<td>30</td>
<td>1.8478</td>
<td></td>
</tr>
<tr>
<td>Pimf</td>
<td>batch</td>
<td>Product</td>
<td>0.1328</td>
<td>0.0548</td>
<td>30</td>
<td>1.1351</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1424</td>
<td>0.0671</td>
<td>30</td>
<td>1.6154</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>Product</td>
<td>0.1328</td>
<td>0.0548</td>
<td>30</td>
<td>15.029</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1438</td>
<td>0.0697</td>
<td>30</td>
<td>15.090</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>Product</td>
<td>0.1328</td>
<td>0.0548</td>
<td>30</td>
<td>1.2331</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1435</td>
<td>0.0685</td>
<td>30</td>
<td>1.7509</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>Product</td>
<td>0.1331</td>
<td>0.0562</td>
<td>30</td>
<td>1.3228</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.1439</td>
<td>0.0638</td>
<td>30</td>
<td>1.8434</td>
<td></td>
</tr>
</tbody>
</table>
**Table D.7: Performance comparison between Product and Min T-norm operators for California housing prediction**

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Chunk Sizes</th>
<th>T-norm Types</th>
<th>Training RMSE</th>
<th>Testing RMSE</th>
<th># Training rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Psigmf</strong></td>
<td>batch</td>
<td>Product</td>
<td>0.1308</td>
<td>0.1334</td>
<td>5</td>
<td>0.5200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1341</td>
<td>0.1379</td>
<td>5</td>
<td>0.6893</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-by-1</td>
<td>0.1308</td>
<td>0.1334</td>
<td>5</td>
<td>3.8778</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1341</td>
<td>0.1379</td>
<td>5</td>
<td>3.9413</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20-by-20</td>
<td>0.1309</td>
<td>0.1333</td>
<td>5</td>
<td>0.5891</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1341</td>
<td>0.1379</td>
<td>5</td>
<td>0.7506</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>Product</td>
<td>0.1311</td>
<td>0.1338</td>
<td>5</td>
<td>0.6852</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1337</td>
<td>0.1359</td>
<td>5</td>
<td>0.8581</td>
</tr>
<tr>
<td><strong>Dsigmf</strong></td>
<td>batch</td>
<td>Product</td>
<td>0.1303</td>
<td>0.1335</td>
<td>5</td>
<td>0.5506</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1338</td>
<td>0.1359</td>
<td>5</td>
<td>0.7393</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-by-1</td>
<td>0.1313</td>
<td>0.1340</td>
<td>5</td>
<td>3.9187</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1338</td>
<td>0.1359</td>
<td>5</td>
<td>4.0548</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20-by-20</td>
<td>0.1315</td>
<td>0.1342</td>
<td>5</td>
<td>0.6385</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1338</td>
<td>0.1360</td>
<td>5</td>
<td>0.7872</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>Product</td>
<td>0.1318</td>
<td>0.1347</td>
<td>5</td>
<td>0.7219</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1338</td>
<td>0.1391</td>
<td>5</td>
<td>0.9025</td>
</tr>
<tr>
<td><strong>Gauss2mf</strong></td>
<td>batch</td>
<td>Product</td>
<td>0.1314</td>
<td>0.1336</td>
<td>5</td>
<td>0.4475</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1333</td>
<td>0.1362</td>
<td>5</td>
<td>0.6256</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-by-1</td>
<td>0.1316</td>
<td>0.1339</td>
<td>5</td>
<td>3.9991</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1333</td>
<td>0.1362</td>
<td>5</td>
<td>4.0535</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20-by-20</td>
<td>0.1315</td>
<td>0.1342</td>
<td>5</td>
<td>0.6385</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1333</td>
<td>0.1360</td>
<td>5</td>
<td>0.7683</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>Product</td>
<td>0.1310</td>
<td>0.1337</td>
<td>5</td>
<td>0.6146</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1336</td>
<td>0.1367</td>
<td>5</td>
<td>0.8047</td>
</tr>
<tr>
<td><strong>Trapmf</strong></td>
<td>batch</td>
<td>Product</td>
<td>0.1305</td>
<td>0.1333</td>
<td>5</td>
<td>0.4322</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1341</td>
<td>0.1392</td>
<td>5</td>
<td>0.6040</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-by-1</td>
<td>0.1312</td>
<td>0.1328</td>
<td>5</td>
<td>3.9528</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1341</td>
<td>0.1392</td>
<td>5</td>
<td>3.9988</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20-by-20</td>
<td>0.1309</td>
<td>0.1331</td>
<td>5</td>
<td>0.4991</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1341</td>
<td>0.1393</td>
<td>5</td>
<td>0.6698</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>Product</td>
<td>0.1305</td>
<td>0.1330</td>
<td>5</td>
<td>0.5943</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1341</td>
<td>0.1396</td>
<td>5</td>
<td>0.7939</td>
</tr>
<tr>
<td><strong>Pimf</strong></td>
<td>batch</td>
<td>Product</td>
<td>0.1318</td>
<td>0.1350</td>
<td>5</td>
<td>0.4431</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1336</td>
<td>0.1368</td>
<td>5</td>
<td>0.6153</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-by-1</td>
<td>0.1312</td>
<td>0.1348</td>
<td>5</td>
<td>4.0578</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1336</td>
<td>0.1367</td>
<td>5</td>
<td>4.1593</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20-by-20</td>
<td>0.1316</td>
<td>0.1345</td>
<td>5</td>
<td>0.6097</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>0.1336</td>
<td>0.1366</td>
<td>5</td>
<td>0.6829</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>Product</td>
<td>0.1316</td>
<td>0.1345</td>
<td>5</td>
<td>0.6097</td>
</tr>
</tbody>
</table>

Continued on next page
Table D.7: (continued)

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Chunk Sizes</th>
<th>T-norm Types</th>
<th>Training (RMSE)</th>
<th>Testing (RMSE)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>0.1342</td>
<td>0.1361</td>
<td>5</td>
<td>0.8222</td>
<td></td>
</tr>
</tbody>
</table>

Table D.8: Performance comparison between Product and Min T-norm operators for DNA classification

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Chunk Sizes</th>
<th>T-norm Types</th>
<th>Training (%)</th>
<th>Testing (%)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psigmf batch</td>
<td></td>
<td>Product</td>
<td>96.98</td>
<td>94.38</td>
<td>5</td>
<td>25.671</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.55</td>
<td>92.06</td>
<td>5</td>
<td>26.593</td>
</tr>
<tr>
<td>1-by-1</td>
<td></td>
<td>Product</td>
<td>96.55</td>
<td>94.94</td>
<td>5</td>
<td>2537.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.07</td>
<td>92.66</td>
<td>5</td>
<td>2591.6</td>
</tr>
<tr>
<td>20-by-20</td>
<td></td>
<td>Product</td>
<td>96.85</td>
<td>94.34</td>
<td>5</td>
<td>128.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.02</td>
<td>92.53</td>
<td>5</td>
<td>129.11</td>
</tr>
<tr>
<td>[10, 30]</td>
<td></td>
<td>Product</td>
<td>96.75</td>
<td>94.35</td>
<td>5</td>
<td>123.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.76</td>
<td>92.96</td>
<td>5</td>
<td>127.93</td>
</tr>
<tr>
<td>Dsigmf batch</td>
<td></td>
<td>Product</td>
<td>97.11</td>
<td>94.28</td>
<td>5</td>
<td>25.025</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.95</td>
<td>92.24</td>
<td>5</td>
<td>26.750</td>
</tr>
<tr>
<td>1-by-1</td>
<td></td>
<td>Product</td>
<td>96.94</td>
<td>94.50</td>
<td>5</td>
<td>2468.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.23</td>
<td>92.32</td>
<td>5</td>
<td>2492.9</td>
</tr>
<tr>
<td>20-by-20</td>
<td></td>
<td>Product</td>
<td>96.45</td>
<td>94.15</td>
<td>5</td>
<td>129.08</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.19</td>
<td>92.47</td>
<td>5</td>
<td>129.37</td>
</tr>
<tr>
<td>[10, 30]</td>
<td></td>
<td>Product</td>
<td>96.49</td>
<td>94.13</td>
<td>5</td>
<td>126.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.93</td>
<td>92.24</td>
<td>5</td>
<td>126.87</td>
</tr>
<tr>
<td>Gauss2mf batch</td>
<td></td>
<td>Product</td>
<td>96.80</td>
<td>94.37</td>
<td>5</td>
<td>23.274</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.35</td>
<td>92.34</td>
<td>5</td>
<td>23.872</td>
</tr>
<tr>
<td>1-by-1</td>
<td></td>
<td>Product</td>
<td>96.60</td>
<td>94.25</td>
<td>5</td>
<td>2182.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.10</td>
<td>92.61</td>
<td>5</td>
<td>2196.3</td>
</tr>
<tr>
<td>20-by-20</td>
<td></td>
<td>Product</td>
<td>96.37</td>
<td>94.19</td>
<td>5</td>
<td>128.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.29</td>
<td>92.72</td>
<td>5</td>
<td>128.14</td>
</tr>
<tr>
<td>[10, 30]</td>
<td></td>
<td>Product</td>
<td>96.56</td>
<td>94.16</td>
<td>5</td>
<td>123.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.95</td>
<td>92.31</td>
<td>5</td>
<td>124.03</td>
</tr>
<tr>
<td>Trapmf batch</td>
<td></td>
<td>Product</td>
<td>96.85</td>
<td>94.31</td>
<td>5</td>
<td>22.766</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.75</td>
<td>92.68</td>
<td>5</td>
<td>26.531</td>
</tr>
<tr>
<td>1-by-1</td>
<td></td>
<td>Product</td>
<td>96.79</td>
<td>94.26</td>
<td>5</td>
<td>2180.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.35</td>
<td>92.32</td>
<td>5</td>
<td>2189.2</td>
</tr>
<tr>
<td>20-by-20</td>
<td></td>
<td>Product</td>
<td>96.68</td>
<td>94.30</td>
<td>5</td>
<td>122.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.12</td>
<td>92.63</td>
<td>5</td>
<td>128.17</td>
</tr>
<tr>
<td>[10, 30]</td>
<td></td>
<td>Product</td>
<td>96.68</td>
<td>94.30</td>
<td>5</td>
<td>122.38</td>
</tr>
</tbody>
</table>

Continued on next page
### D.4 Results for Different Fuzzy Models

#### Table D.8: (continued)

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Chunk Sizes</th>
<th>T-norm Types</th>
<th>Training (%)</th>
<th>Testing (%)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pimf</td>
<td>batch</td>
<td>Product</td>
<td>96.85</td>
<td>94.26</td>
<td>5</td>
<td>22.431</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Product</td>
<td>96.74</td>
<td>94.23</td>
<td>5</td>
<td>124.45</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>Product</td>
<td>97.97</td>
<td>92.04</td>
<td>5</td>
<td>124.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>97.40</td>
<td>92.17</td>
<td>5</td>
<td>24.050</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>Product</td>
<td>96.84</td>
<td>94.77</td>
<td>5</td>
<td>2185.0</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>Product</td>
<td>97.20</td>
<td>92.66</td>
<td>5</td>
<td>2188.1</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>Product</td>
<td>96.74</td>
<td>94.23</td>
<td>5</td>
<td>124.45</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>Product</td>
<td>97.82</td>
<td>92.50</td>
<td>5</td>
<td>128.32</td>
</tr>
<tr>
<td></td>
<td>[10,30]</td>
<td>Product</td>
<td>96.74</td>
<td>94.23</td>
<td>5</td>
<td>124.45</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>Product</td>
<td>97.03</td>
<td>92.67</td>
<td>5</td>
<td>125.20</td>
</tr>
</tbody>
</table>

#### Table D.9: Performance comparison between TSK and Mamdani fuzzy models for nonlinear system identification

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Chunk Sizes</th>
<th>Fuzzy Models</th>
<th>Training RMSE</th>
<th>Testing RMSE</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psigmf</td>
<td>batch</td>
<td>TSK</td>
<td>0.1271</td>
<td>0.0525</td>
<td>30</td>
<td>1.2438</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1317</td>
<td>0.0520</td>
<td>110</td>
<td>1.4110</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>TSK</td>
<td>0.1271</td>
<td>0.0525</td>
<td>30</td>
<td>14.750</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1326</td>
<td>0.0526</td>
<td>110</td>
<td>13.051</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>TSK</td>
<td>0.1271</td>
<td>0.0525</td>
<td>30</td>
<td>1.3195</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1331</td>
<td>0.0538</td>
<td>110</td>
<td>1.5876</td>
</tr>
<tr>
<td></td>
<td>[10,30]</td>
<td>TSK</td>
<td>0.1271</td>
<td>0.0533</td>
<td>30</td>
<td>1.3896</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1329</td>
<td>0.0536</td>
<td>110</td>
<td>1.6544</td>
</tr>
<tr>
<td>Dsigmf</td>
<td>batch</td>
<td>TSK</td>
<td>0.1258</td>
<td>0.0513</td>
<td>30</td>
<td>1.2738</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1270</td>
<td>0.0500</td>
<td>110</td>
<td>1.5728</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>TSK</td>
<td>0.1260</td>
<td>0.0519</td>
<td>30</td>
<td>14.775</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1279</td>
<td>0.0506</td>
<td>110</td>
<td>13.175</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>TSK</td>
<td>0.1271</td>
<td>0.0525</td>
<td>30</td>
<td>1.3195</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1331</td>
<td>0.0538</td>
<td>110</td>
<td>1.5876</td>
</tr>
<tr>
<td></td>
<td>[10,30]</td>
<td>TSK</td>
<td>0.1281</td>
<td>0.0518</td>
<td>30</td>
<td>14.223</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1273</td>
<td>0.0504</td>
<td>110</td>
<td>1.7716</td>
</tr>
<tr>
<td>Gauss2mf</td>
<td>batch</td>
<td>TSK</td>
<td>0.1346</td>
<td>0.0533</td>
<td>30</td>
<td>1.1704</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1304</td>
<td>0.0515</td>
<td>110</td>
<td>1.2012</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>TSK</td>
<td>0.1319</td>
<td>0.0534</td>
<td>30</td>
<td>14.942</td>
</tr>
</tbody>
</table>

Continued on next page
### Table D.9: (continued)

<table>
<thead>
<tr>
<th>Membership Types</th>
<th>Chunk Sizes</th>
<th>Fuzzy Models</th>
<th>Training RMSE</th>
<th>Testing RMSE</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1318</td>
<td>0.0509</td>
<td>110</td>
<td>14.378</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSK</td>
<td>0.1320</td>
<td>0.0532</td>
<td>30</td>
<td>1.2710</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>Mamdani</td>
<td>0.1350</td>
<td>0.0506</td>
<td>110</td>
<td>1.3525</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSK</td>
<td>0.1345</td>
<td>0.0537</td>
<td>30</td>
<td>1.3432</td>
</tr>
<tr>
<td></td>
<td>[10,30]</td>
<td>Mamdani</td>
<td>0.1343</td>
<td>0.0521</td>
<td>110</td>
<td>1.4065</td>
</tr>
<tr>
<td>Trapmf</td>
<td>batch</td>
<td>TSK</td>
<td>0.1308</td>
<td>0.0528</td>
<td>30</td>
<td>1.1351</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1324</td>
<td>0.0529</td>
<td>110</td>
<td>1.1022</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>TSK</td>
<td>0.1308</td>
<td>0.0528</td>
<td>30</td>
<td>14.969</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1336</td>
<td>0.0537</td>
<td>110</td>
<td>13.780</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>TSK</td>
<td>0.1308</td>
<td>0.0528</td>
<td>30</td>
<td>1.2166</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1334</td>
<td>0.0540</td>
<td>110</td>
<td>1.2475</td>
</tr>
<tr>
<td></td>
<td>[10,30]</td>
<td>TSK</td>
<td>0.1305</td>
<td>0.0527</td>
<td>30</td>
<td>1.3252</td>
</tr>
<tr>
<td>Pimf</td>
<td>batch</td>
<td>TSK</td>
<td>0.1328</td>
<td>0.0548</td>
<td>30</td>
<td>1.1351</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1283</td>
<td>0.0546</td>
<td>110</td>
<td>1.1359</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>TSK</td>
<td>0.1328</td>
<td>0.0548</td>
<td>30</td>
<td>15.029</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1332</td>
<td>0.0553</td>
<td>110</td>
<td>13.683</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>TSK</td>
<td>0.1328</td>
<td>0.0548</td>
<td>30</td>
<td>1.2331</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1332</td>
<td>0.0555</td>
<td>110</td>
<td>1.2693</td>
</tr>
<tr>
<td></td>
<td>[10,30]</td>
<td>TSK</td>
<td>0.1331</td>
<td>0.0562</td>
<td>30</td>
<td>1.3228</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1321</td>
<td>0.0551</td>
<td>110</td>
<td>1.3260</td>
</tr>
</tbody>
</table>

### Table D.10: Performance comparison between TSK and Mamdani fuzzy models for California housing prediction

<table>
<thead>
<tr>
<th>Membership Types</th>
<th>Chunk Sizes</th>
<th>Fuzzy Models</th>
<th>Training RMSE</th>
<th>Testing RMSE</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psigmf</td>
<td>batch</td>
<td>TSK</td>
<td>0.1308</td>
<td>0.1334</td>
<td>5</td>
<td>0.5200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1339</td>
<td>0.1352</td>
<td>35</td>
<td>0.8522</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>TSK</td>
<td>0.1308</td>
<td>0.1334</td>
<td>5</td>
<td>3.8778</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1355</td>
<td>0.1365</td>
<td>35</td>
<td>3.8522</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>TSK</td>
<td>0.1309</td>
<td>0.1333</td>
<td>5</td>
<td>0.5891</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1366</td>
<td>0.1376</td>
<td>35</td>
<td>0.9285</td>
</tr>
<tr>
<td></td>
<td>[10,30]</td>
<td>TSK</td>
<td>0.1311</td>
<td>0.1338</td>
<td>5</td>
<td>0.6852</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1361</td>
<td>0.1369</td>
<td>35</td>
<td>0.9394</td>
</tr>
<tr>
<td>Dsigmf</td>
<td>batch</td>
<td>TSK</td>
<td>0.1303</td>
<td>0.1335</td>
<td>5</td>
<td>0.5506</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>0.1337</td>
<td>0.1344</td>
<td>35</td>
<td>0.8985</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>TSK</td>
<td>0.1313</td>
<td>0.1340</td>
<td>5</td>
<td>3.9187</td>
</tr>
</tbody>
</table>

Continued on next page
### D.4 Results for Different Fuzzy Models

#### Table D.10: (continued)

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Fuzzy Models</th>
<th>Training (RMSE)</th>
<th>Testing (RMSE)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSK</td>
<td>0.1315</td>
<td>0.1364</td>
<td>35</td>
<td>0.6385</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1357</td>
<td>0.1364</td>
<td>35</td>
<td>0.9335</td>
<td></td>
</tr>
<tr>
<td>[10, 30] TSK</td>
<td>0.1315</td>
<td>0.1347</td>
<td>5</td>
<td>0.7219</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1363</td>
<td>0.1366</td>
<td>35</td>
<td>0.9612</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>membership Types</th>
<th>Fuzzy Models</th>
<th>Training (RMSE)</th>
<th>Testing (RMSE)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSK</td>
<td>0.1314</td>
<td>0.1336</td>
<td>5</td>
<td>0.4475</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1329</td>
<td>0.1344</td>
<td>35</td>
<td>0.7180</td>
<td></td>
</tr>
<tr>
<td>1-by-1 TSK</td>
<td>0.1316</td>
<td>0.1339</td>
<td>5</td>
<td>3.9991</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1331</td>
<td>0.1346</td>
<td>35</td>
<td>3.6613</td>
<td></td>
</tr>
<tr>
<td>20-by-20 TSK</td>
<td>0.1315</td>
<td>0.1342</td>
<td>5</td>
<td>0.6385</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1332</td>
<td>0.1347</td>
<td>35</td>
<td>0.8077</td>
<td></td>
</tr>
<tr>
<td>[10, 30] TSK</td>
<td>0.1310</td>
<td>0.1337</td>
<td>5</td>
<td>0.6146</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1338</td>
<td>0.1344</td>
<td>35</td>
<td>0.9196</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>membership Types</th>
<th>Fuzzy Models</th>
<th>Training (RMSE)</th>
<th>Testing (RMSE)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSK</td>
<td>0.1305</td>
<td>0.1333</td>
<td>5</td>
<td>0.4322</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1340</td>
<td>0.1349</td>
<td>35</td>
<td>0.5928</td>
<td></td>
</tr>
<tr>
<td>1-by-1 TSK</td>
<td>0.1312</td>
<td>0.1328</td>
<td>5</td>
<td>3.9528</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1351</td>
<td>0.1361</td>
<td>35</td>
<td>3.7247</td>
<td></td>
</tr>
<tr>
<td>20-by-20 TSK</td>
<td>0.1309</td>
<td>0.1331</td>
<td>5</td>
<td>0.4991</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1345</td>
<td>0.1352</td>
<td>35</td>
<td>0.7344</td>
<td></td>
</tr>
<tr>
<td>[10, 30] TSK</td>
<td>0.1305</td>
<td>0.1330</td>
<td>5</td>
<td>0.5943</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1344</td>
<td>0.1350</td>
<td>35</td>
<td>0.8255</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>membership Types</th>
<th>Fuzzy Models</th>
<th>Training (RMSE)</th>
<th>Testing (RMSE)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSK</td>
<td>0.1318</td>
<td>0.1350</td>
<td>5</td>
<td>0.4431</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1349</td>
<td>0.1363</td>
<td>35</td>
<td>0.7350</td>
<td></td>
</tr>
<tr>
<td>1-by-1 TSK</td>
<td>0.1312</td>
<td>0.1348</td>
<td>5</td>
<td>4.0578</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1340</td>
<td>0.1356</td>
<td>35</td>
<td>3.5285</td>
<td></td>
</tr>
<tr>
<td>20-by-20 TSK</td>
<td>0.1316</td>
<td>0.1345</td>
<td>5</td>
<td>0.6097</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1355</td>
<td>0.1363</td>
<td>35</td>
<td>0.8478</td>
<td></td>
</tr>
<tr>
<td>[10, 30] TSK</td>
<td>0.1316</td>
<td>0.1345</td>
<td>5</td>
<td>0.6097</td>
<td></td>
</tr>
<tr>
<td>Mamdani</td>
<td>0.1349</td>
<td>0.1363</td>
<td>35</td>
<td>0.9350</td>
<td></td>
</tr>
</tbody>
</table>

#### Table D.11: Performance comparison between TSK and Mamdani fuzzy models for DNA classification

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Fuzzy Models</th>
<th>Training (%)</th>
<th>Testing (%)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psigmf</td>
<td>TSK</td>
<td>96.98</td>
<td>94.38</td>
<td>5</td>
<td>25.671</td>
</tr>
<tr>
<td>Mamdani</td>
<td>96.63</td>
<td>94.36</td>
<td>200</td>
<td>23.056</td>
<td></td>
</tr>
<tr>
<td>1-by-1</td>
<td>TSK</td>
<td>96.55</td>
<td>94.94</td>
<td>5</td>
<td>2537.2</td>
</tr>
</tbody>
</table>

Continued on next page
### Table D.11 (continued)

<table>
<thead>
<tr>
<th>Membership Function Types</th>
<th>Chunk Sizes</th>
<th>Fuzzy Models</th>
<th>Training (%)</th>
<th>Testing (%)</th>
<th># rules</th>
<th>Training Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mamdani</td>
<td>20-by-20</td>
<td>TSK</td>
<td>96.85</td>
<td>94.34</td>
<td>5</td>
<td>128.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.27</td>
<td>94.26</td>
<td>200</td>
<td>48.136</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>TSK</td>
<td>96.75</td>
<td>94.35</td>
<td>5</td>
<td>123.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.30</td>
<td>94.29</td>
<td>200</td>
<td>27.652</td>
</tr>
<tr>
<td>Dsgmf</td>
<td>batch</td>
<td>TSK</td>
<td>97.11</td>
<td>94.28</td>
<td>5</td>
<td>25.025</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.67</td>
<td>94.35</td>
<td>200</td>
<td>23.598</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>TSK</td>
<td>96.94</td>
<td>94.50</td>
<td>5</td>
<td>2468.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.46</td>
<td>94.27</td>
<td>200</td>
<td>71.946</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>TSK</td>
<td>96.45</td>
<td>94.15</td>
<td>5</td>
<td>129.08</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.38</td>
<td>94.24</td>
<td>200</td>
<td>58.531</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>TSK</td>
<td>96.49</td>
<td>94.13</td>
<td>5</td>
<td>126.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.42</td>
<td>94.30</td>
<td>200</td>
<td>30.819</td>
</tr>
<tr>
<td>Gauss2mf</td>
<td>batch</td>
<td>TSK</td>
<td>96.80</td>
<td>94.37</td>
<td>5</td>
<td>23.274</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.74</td>
<td>94.51</td>
<td>200</td>
<td>12.213</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>TSK</td>
<td>96.60</td>
<td>94.25</td>
<td>5</td>
<td>2182.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.67</td>
<td>94.30</td>
<td>200</td>
<td>56.052</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>TSK</td>
<td>96.37</td>
<td>94.19</td>
<td>5</td>
<td>128.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.69</td>
<td>94.54</td>
<td>200</td>
<td>25.822</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>TSK</td>
<td>96.56</td>
<td>94.16</td>
<td>5</td>
<td>123.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.04</td>
<td>94.33</td>
<td>200</td>
<td>14.352</td>
</tr>
<tr>
<td>Trapmf</td>
<td>batch</td>
<td>TSK</td>
<td>96.85</td>
<td>94.31</td>
<td>5</td>
<td>22.766</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.70</td>
<td>94.41</td>
<td>200</td>
<td>10.464</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>TSK</td>
<td>96.79</td>
<td>94.26</td>
<td>5</td>
<td>2180.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.56</td>
<td>94.84</td>
<td>200</td>
<td>53.775</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>TSK</td>
<td>96.68</td>
<td>94.30</td>
<td>5</td>
<td>122.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.72</td>
<td>94.54</td>
<td>200</td>
<td>22.178</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>TSK</td>
<td>96.68</td>
<td>94.30</td>
<td>5</td>
<td>122.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.57</td>
<td>94.31</td>
<td>200</td>
<td>12.420</td>
</tr>
<tr>
<td>Pimf</td>
<td>batch</td>
<td>TSK</td>
<td>96.85</td>
<td>94.26</td>
<td>5</td>
<td>22.431</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.85</td>
<td>94.42</td>
<td>200</td>
<td>12.703</td>
</tr>
<tr>
<td></td>
<td>1-by-1</td>
<td>TSK</td>
<td>96.84</td>
<td>94.77</td>
<td>5</td>
<td>2185.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.64</td>
<td>94.26</td>
<td>200</td>
<td>56.626</td>
</tr>
<tr>
<td></td>
<td>20-by-20</td>
<td>TSK</td>
<td>96.74</td>
<td>94.23</td>
<td>5</td>
<td>124.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.60</td>
<td>94.29</td>
<td>200</td>
<td>26.877</td>
</tr>
<tr>
<td></td>
<td>[10, 30]</td>
<td>TSK</td>
<td>96.74</td>
<td>94.23</td>
<td>5</td>
<td>124.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mamdani</td>
<td>96.63</td>
<td>94.25</td>
<td>200</td>
<td>16.407</td>
</tr>
</tbody>
</table>
Author’s Publications

Journal Paper


Conference Paper


Bibliography


Extreme learning machine for predicting HLA-peptide binding. Lecture Notes 


[37] W. E. Holley. Wind modeling and lateral control for automatic landing. In 
American Institute of Aeronautics and Astronautics, Guidance and Control 

[38] C. R. Houck, J. A. Joines, and M. G. Kay. A genetic algorithm for function 


[40] G.-B. Huang and H. A. Babri. Upper bounds on the number of hidden 
neurons in feedforward networks with arbitrary bounded nonlinear activation 

[41] G.-B. Huang, L. Chen, and C.-K. Siew. Universal approximation using incre-
mental constructive feedforward networks with random hidden nodes. IEEE 

[42] G.-B. Huang, P. Saratchandran, and N. Sundararajan. An efficient se-
quential learning algorithm for growing and pruning RBF (GAP-RBF) net-
works. IEEE Transactions on Systems, Man, Cybernetics-Part B: Cybernet-


[135] L. Yingwei, N. Sundararajan, and P. Saratchandran. Identification of time-varying nonlinear systems using minimal radial basis function neural net-


