KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
AFDELING ESAT
Kasteelpark Arenberg 10 — 3001 Leuven (Heverlee)

Fuzzy Modeling and Control

Jury :
Prof. E. Aernoudt, voorzitter
Prof. J. Vandewalle, promotor
Prof. V. Wertz (U.C.L.), promotor
Prof. B. De Moor
Prof. A. Barbé
Prof. J. De Schutter
Dr. Da Ruan (SCK-CEN Research)

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Jairo J. ESPINOSA OVIEDO

©Katholieke Universiteit Leuven - Faculteit Toegepaste Wetenschappen Arenbergkasteel, B-3001 Heverlee (Belgium)
Alle rechten voorbehouden. Niets uit deze uitgave mag vermenigvuldigd en /of openbaar gemaakt worden door middel van druk, fotocopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toesteming van de uitgever.
All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the publisher.
D/2001/7515/14
ISBN 90-5682-303-5
UDC:681.5

Acknowledgements

It is always hard to start something but it is even harder to finish it. Along the years of this Ph.D. many people contributed in one way or another to accomplish the work reflected on these pages. So many that I will probably need another two hundred pages to make justice to each of them. However, if I expect someone to read these lines I should restrict myself to one or two short pages.
First I would like to thank Prof. Joos Vandewalle. Sometime in 1996 I was hesitating to continue with the doctoral studies or to start a carrier somewhere else. During a course in Cartagena (Colombia), he talked to me about the positive and negative things about the doctoral studies and by this conversation I realized that research was the motor of my carrier and the doctoral studies the perfect way to improve my research skills. He was already a good support during the Master studies. At the beginning of my studies he provided me with a scholarship and at the end he supported my applications for grants of the Research Council of the University. His support has been important not only for me but also for the well being of my wife and kids during my studies.
The SISTA group as such provides all the means to achieve a good research. The active interactions of this group with other universities and research groups play a very important role in what this thesis has achieved. The IUAP project and their regular meetings gave me the possibility to meet Prof. Vincent Wertz. He was all the time very attentive to my questions and every trip I made to Louvain La Neuve has been full of suggestions and knowledge. When I was starting my research he read one of my early publications and provide me with the feedback and the confidence to start showing my results to the scientific community. Afterwards he played an important role in the corrections of this document and some other publications.
Prof. Bart De Moor played a key role in the last phase of this Ph.D. He provided with an extensive feedback during the reading phase of this thesis and afterwards his frequent e-mails were the stimulus needed to complete the corrections.
I like to thank Prof. André Barbé, during the Master Studies he showed to me the beauty of nonlinear systems and played a role in the definition of my research subject. As member of the reading committee his recommendations were important to improve this text.

I am grateful with the other members of the jury Prof. Etienne Aernoudt for being the chairman of the jury. Prof. Joris De Schutter and Dr. Da Ruan for taking the time to read and judge this thesis.
The Ph.D. studies demands a good working environment and at SISTA I did have it. From the beginning to the end Bart Motmans was sharing office with me but we share not only that space but also the daily difficulties. He started first as a researcher with this topic and he gave me lots of vital information to start. Afterwards he helped me to solve minor and big problems from the everyday life. Bedankt Señor Menheer!
Ida, Lut, Elvira, Anne-Marie you deserve my gratitude because your job make the life as a Ph.D. student very easy.
The completion of this thesis will not be possible without the support of ISMC. During the last months Peter and Christiaan helped me by being flexible with my working hours and the use of the ISMC equipment.
Parallel to my Ph.D. studies I have been teaching at the Postgraduate Program on Automation from the Coruniversitaria. This has been a valuable opportunity to show my research to the student community and to receive valuable feedback. To them I ought my initial contact with the Katholieke Universiteit Leuven and with Prof. Vandewalle. I am sure we will continue having a fruitful cooperation.
Claudia, my wife she is part of this too. She has been the daily support in the difficult and in the easy times and during this period she sacrificed her family and friends to follow me more than half a world away. Laura Patricia and Jairo Ernesto were born during my studies and help me to overcome the difficult situations with the relief caused by their ingenuous happiness.
My family and their blind faith and joy for science and culture provide me the seeds to grow this Ph.D. Padre, Madre, Jorgín gracias por el apoyo incondicional que solo ustedes saben brindar. Madre, hoy tengo la satisfacción que me prometiste aquel día cuando dudé en continuar. Abuelos sus oraciones, ejemplo y fe construyeron esta estirpe y con ella este logro. Tío Juan has sabido estar ahípresente, me has hecho sentir acompaado y esto es un poco tuyo también. Estos años se llevaron a personas que apreciaba Abuelita María, Tío Medardo, Tío Humberto y Doña Ruby. Tío Humberto hoy te rememoro como ejemplo de inteligencia, rectitud, nobleza y perseverancia tu memoria est presente en los logros de quienes aprendimos algo de tí, es una pena que no estés, sé que lo hubieras disfrutado.
Friends your manifest interest on the progress of my research work has been motivating and that makes you part of this. Amigos sus mensajes, cartas, visitas e interés en el progreso de este trabajo los hace parte de este logro. Pilar, Carlos gracias por su tiempo e insistencia.
Pilar! ya me puedes invitar a comer arroz.

Jairo José

Abstract

This thesis explores several issues related to the use of fuzzy set theory in modeling, identification and control of dynamic systems. The thesis presents new algorithms, analyses and examples. The thesis has four sections.
The first section explores the issues of function approximation using fuzzy models. Issues such as generalization and consequence initialization are analyzed and an algorithm to improve these features is presented. The importance of linguistic integrity is explored and the AFRELI (Automatic Fuzzy Rule Extractor with Linguistic Integrity) algorithm is presented. Examples of the approximation capabilities of the AFRELI algorithm are presented.
The second section is dedicated to the analysis of the use of fuzzy models on system identification. Experiment design, regressor selection, parameter estimation and validation are discussed on this chapter. On each of these subjects recommendations and guidelines to build dynamic fuzzy models are included.
The third section is an overview of the classical fuzzy control methods. The analysis is complemented with a practical application where these techniques are applied into an automotive transmission system.
The fourth section presents the use of fuzzy models on predictive control. The thesis explores the problem of unconstrained, constrained predictive control and robust predictive control based on fuzzy models. Four algorithms are proposed and they are illustrated with two simulation examples on a chemical reactor and a power plant.

Samenvatting

Voorliggende thesis behandelt verschillende onderwerpen die verband houden met het gebruik van de theorie van Vage Sets in modellering, identificatie en regeling van dynamische systemen. In voorliggende thesis worden een aantal nieuwe algoritmes, analyses en voorbeelden uitgewerkt. De thesis kan onderverdeeld worden in vier delen.
Het eerste deel onderzoekt de benadering van functies met behulp van vage modellen. Aspecten zoals veralgemening en het initialiseren van de conclusies van regels worden geanalyseerd, en er wordt een algoritme voorgesteld dat deze aspecten verbetert. Het belang van lingustische integriteit wordt onderzocht en het AFRELI (Automatische Fuzzy Regel Extractor met Lingustische Integriteit) algoritme wordt voorgesteld. Een aantal voorbeelden tonen de mogelijkheden van het AFRELI algoritme bij de benadering van functies.
Het tweede deel is gewijd aan de analyse van het gebruik van vage modellen bij systeemidentificatie. In dit hoofdstuk worden het ontwerpen van een experiment, de selectie van regressoren, het schatten van parameters en de validatie van modellen beschreven. Om dynamische vage modellen op te stellen, worden er voor elk van deze onderwerpen een aantal aanbevelingen en richtlijnen opgesteld.
Het derde deel omvat een overzicht van de klassieke vage regelaars. Dit overzicht wordt aangevuld met een praktische toepassing van deze technieken, bij het regelen van een continu variabele transmissie.
Het vierde en laatste deel behandelt het gebruik van vage modellen bij predictieve controle. We onderzoeken de problematiek van predictieve controle met of zonder beperkingen, en van robuuste predictieve controle gebaseerd op vage modellen. Er worden vier algoritmes voorgesteld die toegelicht worden aan de hand van twee gesimuleerde systemen (een chemische reactor en een elektriciteitscentrale).

Contents

1 Introduction 1
- 1.1 Chapter Description . . . 1
  - 1.1.1 Chapter 2-Fuzzy Modeling . . . 1
  - 1.1.2 Chapter 3-Nonlinear Identification Using Fuzzy Models . . . 2
  - 1.1.3 Chapter 4-Fuzzy Control . . . 3
  - 1.1.4 Chapter 5-Predictive Control Based on Fuzzy Models . . . 3
  - 1.1.5 Chapter 6-Robust Nonlinear Predictive Control Using Fuzzy Models . . . 4
  - 1.1.6 Chapter 7-Conclusions and Future Perspectives . . . 4
  - 1.1.7 Appendices . . . 5
- 1.2 Main Contributions . . . 5
- 1.3 Published Results . . . 7

2 Fuzzy Modeling 11
- 2.1 Function Approximation . . . 12
  - 2.1.1 System description . . . 12
  - 2.1.2 Approximation Error . . . 13
  - 2.1.3 Constructing Units in the Fuzzy Models . . . 16
- 2.2 Constructing fuzzy models from Input-Output data . . . 25
  - 2.2.1 Mosaic or table look-up scheme . . . 27
  - 2.2.2 Using gradient descent . . . 29
  - 2.2.3 Using Clustering and Gradient Descent . . . 35
  - 2.2.4 Using Evolutionary Strategies . . . 36
- 2.3 Generalization and Consequences Estimation . . . 41
  - 2.3.1 Consequence Initialization . . . 41
  - 2.3.2 Consequence Estimation . . . 42
  - 2.3.3 Example . . . 43
- 2.4 Fuzzy Modeling with Linguistic Integrity . . . 47
  - 2.4.1 Introduction . . . 47
  - 2.4.2 Structure of the fuzzy model . . . 49

  - 2.4.3 The AFRELI algorithm . . . 51
  - 2.4.4 The FuZion algorithm . . . 57
  - 2.4.5 Examples . . . 59
  - 2.4.6 Example 1: Modeling a two input nonlinear function . . . 60
  - 2.4.7 Example 2:Modeling of a three input nonlinear function . . . 67
  - 2.4.8 Example 3:Predicting Chaotic Time Series . . . 70
  - 2.4.9 Remarks on the AFRELI algorithm . . . 75
- 2.5 Conclusions . . . 75

3 Nonlinear Identification Using Fuzzy Models 77
- 3.1 System Identification . . . 78
- 3.2 The basic structure of the Fuzzy System . . . 79
- 3.3 Experiment Design . . . 81
- 3.4 Choosing the Regressors . . . 82
  - 3.4.1 Search methods . . . 85
  - 3.4.2 Regressors Evaluation . . . 86
- 3.5 Choosing the Structure . . . 90
- 3.6 Calculating the Parameters . . . 91
  - 3.6.1 Example . . . 93
- 3.7 Validation . . . 93
- 3.8 Example . . . 94
- 3.9 Conclusions . . . 105

4 Fuzzy Control 107
- 4.1 Model Free fuzzy control . . . 107
  - 4.1.1 Heuristic-Trial and Error Design . . . 107
  - 4.1.2 Design of PID-like fuzzy controllers . . . 108
- 4.2 Model Based Fuzzy Control . . . 111
  - 4.2.1 Using Adaptive Methods . . . 113
  - 4.2.2 Using Direct Synthesis . . . 119
  - 4.2.3 Conclusions and Future Perspectives . . . 128

5 Predictive Control Based on Fuzzy Models 131
- 5.1 The Predictive Control Strategy . . . 132
- 5.2 Unconstrained Nonlinear Predictive Control . . . 134
  - 5.2.1 Estimation of the step response to construct $\mathbf{G}(t)$ . . . 137
  - 5.2.2 Predictive Control of a CSTR Using a Fuzzy Model . . . 138
  - 5.2.3 Fuzzy identification of the CSTR . . . 139
  - 5.2.4 Implementation of the controller and performance results 140
- 5.3 Constrained Nonlinear Predictive Control . . . 150
  - 5.3.1 The Constrained Non-Linear Predictive Control Problem 150
  - 5.3.2 Approach using estimated step response . . . 155

5.3.3 Approach using Takagi-Sugeno Fuzzy Models 159
5.3.4 Approach using Takagi Sugeno Fuzzy models and multiple models in the predictor 161
5.3.5 Predictive Control of a Steam Generator Using a Fuzzy Model 164
5.4 Conclusions 172

6 Robust Nonlinear Predictive Control using Fuzzy Models 175
6.1 Introduction 175
6.2 Robust Quadratic Programming 176
6.3 Problem Description 179
6.4 Nominal Solution 179
6.5 Formulation of the MPC problem as a Robust QP 181
6.6 The Control Algorithm 182
6.7 Uncertainty Description in Fuzzy Models 183
6.7.1 Local uncertainty described on each rule 183
6.7.2 Using the Active Rules 184
6.7.3 Using All the Rules 184
6.7.4 Using the Reachable Set 186
6.8 Conclusions and Perspectives 187

7 Conclusions and Future Perspectives 189
7.1 Conclusions and Contributions 189
7.2 Perspectives and Future Work 192

A Fuzzy Set Theory 195
A.1 Introduction 195
A.2 Fuzzy Sets 195
A.2.1 Some examples of membership functions 196
A.3 Basic Definitions on Fuzzy Sets 196
A.3.1 Support 196
A.3.2 Core 197
A.3.3 Height 197
A.3.4 Normal fuzzy set 197
A.3.5 α-cut 197
A.3.6 Strict α-cut 197
A.3.7 Convexity 197
A.4 Operations on fuzzy sets 197
A.4.1 A is contained in B 197
A.4.2 Complement, Negation 198
A.4.3 Intersection 198
A.4.4 Union 199

A.5 Fuzzy relations 200
A.5.1 Projection of fuzzy relations 200
A.5.2 Composition of relations 201
A.6 Approximate Reasoning 201
A.6.1 Introduction 201
A.6.2 Linguistic Variables 201
A.7 General Structure of a Fuzzy Inference System 202
A.7.1 Control rules as a knowledge representation 202
A.7.2 Defuzzification 203

B Clustering Methods 205
B.1 Fuzzy C- Means [8] 205
B.2 Mountain Clustering [84] 207

C Gradients Used in Identification with Fuzzy Models 209
C.1 Gradient for the singleton consequences 210
C.1.1 with Trapezoidal Membership Functions 211
C.2 with Polynomial Membership Functions 211
C.2.1 with Gaussian Membership Functions 212
C.3 Gradient for the parameters of the membership functions 212
C.3.1 with Trapezoidal Membership Functions 212
C.3.2 with Polynomial Membership Functions 214
C.3.3 with Gaussian Membership Functions 215
C.3.4 with Triangular Membership Functions with 0.5 overlap . 216
C.3.5 with Polynomial Membership Functions with 0.5 overlap . 217

D Discrete Lin. Dyn. Approximation Theorem 221

E Fuzzy Control for a CVT 225
E.1 Introduction and process description 225
E.2 Performance Specifications 227
E.3 A Physical Model for the CVT 227
E.4 Design of the Controller 228
E.5 Stability Analysis 232
E.6 Conclusions 234

List of Figures

2.1 *Membership function definition for the i-th input* 12
2.2 *Input Point in the Rule Space* 15
2.3 *Triangular Membership functions with overlap* $\frac{1}{2}$ 17
2.4 *Bilinear Surface generated with triangular membership functions with overlap* $\frac{1}{2}$ 18
2.5 *Polynomial membership functions* 19
2.6 *Polynomial membership functions with overlap* $\frac{1}{2}$ 20
2.7 *Polynomial interpolation generated with polynomial membership functions with overlap* $\frac{1}{2}$ 22
2.8 *Polynomial surface generated with polynomial membership functions with overlap* $\frac{1}{2}$ *in a system with two inputs* 22
2.9 *Gaussian interpolation generated with Gaussian membership functions with* $\sigma = 0.6(m^i_{j_i+1} - m^i_{j_i})$ 23
2.10 *Gaussian surface generated with Gaussian membership functions with* $\sigma = 0.6(m^i_{j_i+1} - m^i_{j_i})$ 24
2.11 *(a) Triangular membership functions with overlap different to* $\frac{1}{2}$ *(b) Polynomial membership functions with overlap different to* $\frac{1}{2}$ *(c) Gaussian membership functions with different* σ 25
2.12 *Interpolation and Surfaces generated with triangular membership functions with overlap different to* $\frac{1}{2}$ 26
2.13 *Interpolation and Surfaces generated with polynomial membership functions with overlap different to* $\frac{1}{2}$ 26
2.14 *Interpolation and Surfaces generated with Gaussian membership functions with different* σ 27
2.15 *Codification of the membership functions* 38
2.16 *Mutation operation in a fuzzy partitions* 39
2.17 *Crossover operation between two fuzzy partitions* 40
2.18 *Function to be approximated* $f(x,y) = 6x+4y+\cos(\pi x)+\cos(\pi y)+50$ 43
2.19 *Sampled Points to Approximate the function* $f(x,y) = 6x + 4y + \cos(\pi x) + \cos(\pi y) + 50$ 44

2.20 *(a) Approximation obtained with the consequences calculated using LS (b)Approximation obtained with the consequences calculated using RLS* 45
2.21 *Approximation generated by the "smallest" fuzzy model* $\hat{f}(x,y)$ *with only 4 rules* 45
2.22 *Approximation generated by the fuzzy model initialized with* $\hat{f}(x,y)$ *and with the consequences calculated using RLS* 46
2.23 *Traditional Knowledge Acquisition* 48
2.24 *Knowledge Acquisition using AFRELI* 49
2.25 *Flow Diagram of AFRELI algorithm* 52
2.26 *Input membership functions for the smallest fuzzy model* 55
2.27 *Flow Diagram of FuZion algorithm* 58
2.28 *Effect of the FuZion algorithm* 59
2.29 *Example 1:Function* $f(x,y) = \sin\left(\frac{\pi x}{10}\right)\sin\left(\frac{\pi y}{10}\right)$ 60
2.30 *Example 1:Extracted clusters ('x') from the data (o)* 61
2.31 *Example 1:Projection of the centers of the clusters* 62
2.32 *Example 1:Projected membership functions* 62
2.33 *Example 1:Membership functions after FuZion with M =10%* . . 63
2.34 *Example 1:Membership functions after FuZion with M =7%* . . . 63
2.35 *Example 1:(a) Singletons (b) Membership functions with linguistic meaning* 64
2.36 *Example 1:Surface Generated by the Fuzzy System* 66
2.37 *Example 2:Projected membership functions* 68
2.38 *Example 2: Membership functions after FuZion* 69
2.39 *Example 2:(a) Singletons (b) Membership functions with linguistic meaning* 69
2.40 *Example 3:Membership functions after projection and FuZion* . . 71
2.41 *Example 3:(a)Singletons (b) Membership functions with linguistic meaning* 72
2.42 *Example 3:(a) Mackey-Glass time series (solid line) from* $t = 618$ *to 1117 and six steps ahead prediction (dashed line) (b) Prediction errors* 73
2.43 *Example 3:(a) Mackey-Glass time series (solid line) from* $t = 118$ *to 1117 and 84 steps ahead prediction (dashed line) (b) Prediction errors* 74

3.1 *Heuristically Designed Excitation Signal with four discrete levels between 0 and 1. (a) Slow random signal with 4 discrete levels (b) Wide band signal (c) Combined signals.* 83
3.2 *Sequence for regressors selection* 84
3.3 *Heuristic Search for regressors selection* 86

3.4 *Regressors selection using Genetic Algorithms. Effect of the crossover operator* . . . 87
3.5 *Regularity Criterion. (a) Estimation of models A and B (b) Cross-validation of the models* . . . 89
3.6 *Input Selection Using Lipschitz Quotients* . . . 95
3.7 *RC index* . . . 96
3.8 *Membership function after Clustering for the model using* $y(t-1), y(t-2), u(t-3)$. . . . 97
3.9 *Membership function after Clustering for the model using* $y(t-1), y(t-4), u(t-3)$. . . . 97
3.10 *Membership function after Clustering and FuZion for the model using* $y(t-1), y(t-2), u(t-3)$. . . . 98
3.11 *Membership function after Clustering and FuZion for the model using* $y(t-1), y(t-4), u(t-3)$. . . . 98
3.12 *Prediction error one step ahead for the NARX model with regressors* $y(t-1), y(t-2), u(t-3)$. . . . 99
3.13 *Prediction error one step ahead for the NOE model with regressors* $y(t-1), y(t-2), u(t-3)$. . . . 100
3.14 *Prediction error one step ahead for the NARX model with regressors* $y(t-1), y(t-4), u(t-3)$. . . . 100
3.15 *Prediction error one step ahead for the NOE model with regressors* $y(t-1), y(t-4), u(t-3)$. . . . 101
3.16 *Correlation analysis for the NARX model with regressors* $y(t-1), y(t-2), u(t-3)$. . . . 102
3.17 *Correlation analysis for the NOE model with regressors* $y(t-1), y(t-2), u(t-3)$. . . . 103
3.18 *Correlation analysis for the NARX model with regressors* $y(t-1), y(t-4), u(t-3)$. . . . 103
3.19 *Correlation analysis for the NOE model with regressors* $y(t-1), y(t-4), u(t-3)$. . . . 104

4.1 *Classification of the fuzzy controllers* . . . 108
4.2 *Fuzzy PD controller* . . . 109
4.3 *Fuzzy PI controller* . . . 109
4.4 *Fuzzy System with linear relations* . . . 111
4.5 *Fuzzy PID controller* . . . 112
4.6 *Fuzzy PD+I controller* . . . 112
4.7 *Scheme for design of inverse controller (a) Plant (b)Learning Phase (c) Scheme of operation* . . . 114
4.8 *Direct adaptive control* . . . 115
4.9 *Tuning of the gain constants using nonlinear optimization* . . . 118

4.10 *Tuning of the internal parameter of the fuzzy controller (Membership functions, Consequences, etc)* . . . 118
4.11 *Scheme for Adaptive Feedback Linearization* . . . 122
4.12 *Relation between the position of the membership functions and the shape of* $K_{fuzz}(\mathbf{x}, t, s, \Phi)$ . . . 126

5.1 Predictive Controller . . . 133
5.2 Prediction and Control Horizons . . . 134
5.3 Continuous Stirred Tank Reactor . . . 139
5.4 Identification Experiments . . . 141
5.5 Validation Experiment . . . 142
5.6 Membership function of the fuzzy model. . . . 143
5.7 Reference and Perturbation . . . 144
5.8 Response of the system . . . 145
5.9 Detailed comparison for two different operating points . . . 146
5.10 Comparison against the "optimal" strategy . . . 147
5.11 Detailed comparison against the "optimal" strategy for two different operating points . . . 148
5.12 *Execution time (msec.) vs. Horizons* . . . 149
5.13 *Steam Generator* . . . 164
5.14 *Stabilization Scheme for the Steam Generator* . . . 167
5.15 *Validation Experiment* . . . 168
5.16 *Comparative result: (-) MPC with Fuzzy Model using step response, (.-) MPC using Takagi-Sugeno Fuzzy Model, (.) MPC with Linear Model-(a) Pressure (b) Oxygen Level (c) Water Level (d) Steam Flow* . . . 171

6.1 *Diagram with the stages of the control algorithm* . . . 183
6.2 *Uncertainty description on each of the rules* . . . 184
6.3 *Uncertainty description using the "polytope" of the active rules* . 185
6.4 *Uncertainty description using the "polytope" of the complete set of rules* . . . 185
6.5 *Graphic representation of the reachable set* . . . 186

A.1 *Fuzzy sets for temperature* . . . 203

E.1 *Power train of a car with a CVT and wet-plate clutch. FD=Final reduction.* . . . 226
E.2 *Diagram of the system including the high level controller.* . . . 227
E.3 *Representation of the process by means of a static nonlinearity* $f(V_{in}, N_{engine})$ *and a dynamic linear system* $G(s)$ *with variable parameters.* . . . 228

E.4 *Static nonlinearity f for different values of engine speed N_{engine} and constant temperature.* 229
E.5 *Membership functions of the Fuzzy Inference Systems that generates the parameters P and T_i for the PI controller.* 230
E.6 *Block diagram of the process with the nonlinear controller.* . . . 232
E.7 *Comparative response of the systems continuous line (_) reference, dashed line () linear controller with feedforward action and dotted line (..) fuzzy controller. Observe that the overshoot and the settling time of the fuzzy with the fuzzy controller is significantly reduced.* 233

Chapter 1

Introduction

Since the idea of fuzzy set was proposed in 1965, many developments have occurred in this area. Applications have been made in such diverse areas as medicine, engineering, management, behavioral science, just to mention some. The application of the fuzzy sets involves different technologies, such as fuzzy clustering on image processing, classification, identification and fault detection, fuzzy controllers to map expert knowledge into control systems, fuzzy modeling combining expert knowledge, fuzzy optimization to solve design problems.
The fuzzy systems are used in the area of Artificial Intelligence as a way to represent knowledge. This representation belongs to the paradigm of behavioral representation in opposition to the structural representation (Neural Networks). The foundation of this paradigm is that intelligent behavior can be obtained by the use of structures which not necessarily resemble the human brain.
A very interesting characteristic of the fuzzy systems is their capability to handle in the same framework numeric and linguistic information. This characteristic made these systems very useful to handle expert control tasks.
This thesis is oriented in the first part to the construction of static and dynamic fuzzy models from numerical information. The second part shows how to exploit these models to design controllers. The next sections show a general overview of the content of the chapters included in the present thesis.

1.1 Chapter Description

1.1.1 Chapter 2-Fuzzy Modeling

Chapter 2 is entirely dedicated to the problem of function approximation and modeling. The first part of the chapter shows the approximation capabilities of

fuzzy systems with triangular, polynomial and Gaussian membership functions. In this part, this thesis contributes an analytical study of the approximation capabilities of the different types of membership functions.
The second part of the chapter describes different techniques to construct fuzzy models. Gradient descent techniques, clustering and evolutionary techniques are explained in this chapter. Some gradient expressions are derived to illustrate the expressions used to adjust the fuzzy models. This section is complemented with the appendix B where the main clustering techniques used in modeling are explained.
In the third part of the chapter, the problems of generalization and consequence estimation are studied. An initialization and training method for the consequences is the contribution of this thesis to the solution of this problem. The method improves the generalization capabilities of the fuzzy models. It is illustrated by means of a graphical example. Finally the fourth part of the chapter introduces the concept of linguistic integrity and presents an algorithm to build fuzzy models with linguistic meaning. The contributed algorithm is named AFRELI (Autonomous Fuzzy Rule Extractor with Linguistic Integrity). This algorithm is complemented with an algorithm to reduce the complexity of the fuzzy models (FuZion algorithm). Several examples are presented where complex nonlinear functions and chaotic nonlinear systems are modeled using this algorithm.

1.1.2 Chapter 3-Nonlinear Identification Using Fuzzy Models

Using the tools developed in chapter 2 a framework for system identification using fuzzy models is presented in chapter 3.
The chapter begins by formulating the problem of systems identification using fuzzy models. The thesis contributes an analysis of the structure of the fuzzy models, which are more suitable to be applied in system identification. Thereafter the thesis studies the problem of experiment design and proposes types of signals which are considered to deliver "sufficient excitation" to guarantee the reliable construction of the model.
The regressors selection is consider a complex problem also analyzed in this thesis. The main methods are reviewed and the contribution to this subject is the discussion of the advantages and disadvantages of the methods.
The possible structures of the models for nonlinear systems are enumerated and a short analysis of the applicability of some of these structures to identification using fuzzy models is included.
The parameter calculation for different type of structure is studied under the assumption that gradient descent methods are used in this calculation. The issue of dynamic calculation of the gradients is emphasized and the thesis contributes

the derivation of the gradient expressions, which are included in the appendix C. The theoretical analysis of this chapter closes with a short discussion of the validation issues. The discussion remarks the fact that fuzzy models can be validated not only using quantitative criteria but also qualitative criteria based on the information given by the linguistic rules.
The chapter closes over an example of identification of a gas furnace process proposed by Box and Jenkins in [11]. In this example, most of the elements presented in the chapter are included.

1.1.3 Chapter 4-Fuzzy Control

Chapter 4 presents an overview of different techniques that have been designed to construct fuzzy controllers. The chapter starts by making a classification of the different methods where fuzzy sets are applied to control. The chapter explains some of the methods starting with the first early ideas employing pure expert knowledge.
Then the discussion focuses on a very practical method, which is the design of PID-like fuzzy controllers. This section is complemented with the theorem presented in the appendix D. This contributed theorem guarantees that any discrete linear controller can be copied exactly by a fuzzy system. This property is exploited to initialize fuzzy controllers such that their initial performance (before tuning) at least equals the performance of a given linear controller.
Then, the overview presented in this chapter focus on the adaptive control techniques based on fuzzy models. Inverse learning and direct learning are the two methods studied in this section.
The chapter then pays attention to methods based on direct synthesis and the method of feedback linearization is proposed where the models used to linearize the affine system are fuzzy models. The main drawbacks of this method are analyzed and the sliding mode fuzzy control method is explained as an alternative. Finally, the chapter is completed by a description of the fuzzy gain scheduling method. The advantages of the method including the existence of methods to directly design stable controllers and test controllers for stability are studied and discussed. The section is complemented with the contribution of appendix E where a stable fuzzy scheduling controller is designed for a Continuous Variable Transmission (CVT) system.

1.1.4 Chapter 5-Predictive Control Based on Fuzzy Models

In chapter 5, the use of fuzzy models in predictive control is explored. The chapter begins with the simplest idea of unconstrained predictive fuzzy control. The problem is formulated and a method to reduce the problem to a quadratic

program is presented. The method includes the formulation of a predictor based on the concept of free and forced response; the estimation of the forced response is improved by a method proposed in this chapter. The chapter includes an application example where the control strategy is applied to a Continuous Stirred Tank Reactor (CSTR). In this example, the strategy is shown to perform quite similarly to the optimal strategy.
In the second part of the chapter the constrained predictive control problem is studied, a new algorithm based on the strategy proposed for unconstrained systems is presented. In this section, two other control strategies are presented. These strategies exploit the information provided by the Takagi-Sugeno fuzzy models. One strategy is a fast one step strategy, which can be applied to monotonic nonlinear systems with success. The second strategy is iterative and is created for applications where the plant has deep changes in the prediction horizon.
The chapter closes with an example where the control methods are applied to a steam generator model of a power plant. The strategies are compared with classical linear predictive control strategies and the improvement in performance can be clearly observed.

1.1.5 Chapter 6-Robust Nonlinear Predictive Control Using Fuzzy Models

Chapter 6 presents a novel extension to the concepts presented in chapter 5. In this chapter, the problem of robust nonlinear predictive control based on fuzzy models is studied. The chapter begins with the formulation of the problem. Then the problem is reduced to a robust quadratic program. The robust quadratic program is written as a second order cone program using new elements contributed in this thesis. Advantages of the use of these algorithms are mentioned including the computational complexity.
The use of this robust optimization technique guarantees a minimum performance despite the mismatch of the models or the linearization errors introduced by the control algorithm.
The chapter is completed with a list of possible ways to describe the uncertainty of the models.

1.1.6 Chapter 7-Conclusions and Future Perspectives

This last chapter condenses the main contributions of this thesis and proposes new challenges for future research.

1.1.7 Appendices

This thesis presents five appendices, which complement the subjects studied in the chapters.

Appendix A- Fuzzy Set Theory

In this appendix, the definitions and the main operations among fuzzy sets are explained as well as most of the notation used in the thesis.

Appendix B- Clustering Methods

The appendix makes a description of the clustering algorithms fuzzy C-Means [8] and Mountain Clustering [84]. This appendix is included as a complement to the algorithms discussed in chapter 2.

Appendix C- Gradient Expressions Used in Identification with Fuzzy Models

In this appendix the expression for the gradient of the parameters of the dynamic fuzzy models is derived. Some of the gradient expressions are given in a static form and others in the form of a discrete dynamic system. This is one of the contributions of this thesis to the subject of system identification using fuzzy models.

Appendix D- Proof of the theorem connecting linear discrete SISO controllers and fuzzy controllers

This appendix includes the proof of the theorem, which establishes that any linear discrete SISO controller with bounded input and output signals can be represented by an equivalent fuzzy controller. This contribution complements the discussion over the tuning of PID-like fuzzy controllers presented in chapter 4.

Appendix E- Fuzzy Control for a Continuously Variable Transmission

The content of this appendix is an application example of the fuzzy gain-scheduling controller to a system used in the automotive industry. The design of the system is explained and a test for stability is shown.

1.2 Main Contributions

The main contributions of this thesis are:

- The analysis of the approximation properties of fuzzy systems.
- An initialization and estimation method for the consequences of a fuzzy systems with improved generalization capabilities.
- A modeling algorithm AFRELI (Automatic Fuzzy Rule Extractor with Linguistic Integrity) based on the concepts of linguistic integrity.
- The analysis of the problem of system identification using fuzzy models.
- The guideline for the application of fuzzy models to system identification.
- The derivation of the gradient expressions to optimize dynamic fuzzy models.
- The proof of the theorem connecting linear discrete SISO controllers and fuzzy controllers.
- An algorithm for unconstrained predictive control based on fuzzy models including the method for improving the estimation of the step response.
- Three algorithms for constrained predictive control based on fuzzy models. One method is base on the estimation of the step response, another exploits the structure of the Takagi-Sugeno fuzzy models and the third one formulates the nonlinear optimization problem as a predictive control problem with a linear time variant plant.
- An algorithm to calculate solutions for the robust constrained predictive control based on Takagi-Sugeno fuzzy models.
- A formulation of a Robust Quadratic Program with uncertainties in the quadratic and in the linear term, as a Second Order Cone Program. Previous formulations only consider the uncertainty in the quadratic term.

The above contributions are illustrated with several examples. The most important examples are:

- Function approximation of functions in $\Re^2 \rightarrow \Re$ and $\Re^3 \rightarrow \Re$ in chapter 2.
- Identification of a model for the Mackey-Glass Chaotic time series presented in chapter 2.
- Identification of models using the Gas Furnace data from Box and Jenkins [11] in chapter 3.
- Fuzzy Control for a Continuously Variable Transmission with a test of stability presented in appendix E.

- Control of a Continuous Stirred Tank Reactor (CSTR) using the Unconstrained predictive fuzzy control strategy proposed in chapter 5.

- Control of a Steam Generator for a Power Plant using the Constrained Predictive Control Strategies proposed in chapter 5.

1.3 Published Results

The main results included on this thesis have been published in journals, magazines and conferences. A list of the publications is included below.

Espinosa J., Vanvuchelen P., Vandewalle J., "Fuzzy scheduling to control a hydraulic servo system", *Journal A*, Special Issue on Fuzzy Control, vol. 36, no. 3, Oct. 1995, pp. 51-55.

Espinosa J., Vanvuchelen P., Vandewalle J., "Fuzzy Control for a Continously Variable Transmission", in *Proc. of the 5th European Congress on Intelligent Techniques and Soft Computing (EUFIT-97)*, Aachen, Germany, Sep. 1997, pp. 1413-1417.

Gutierrez A., De Keyser R., Espinosa J., "Control predictivo basado en una red neuronal aplicado a un reactor para deposicion de silicio", in *Memorias segundo congreso de la asociacion colombiana de automatica*, Bucaramanga, Colombia, Mar. 1997, pp. 182-189.

Espinosa J., Vandewalle J., "Fuzzy modeling and identification. A guide for the user", in *Proc. of the IEEE Singapore International Symposium on Control Theory and Applications (SISCTA 97)*, Singapore, Singapore, Jul. 1997, pp. 437-443.

Aedo J.E., Espinosa J., Van Noije W.A., Vandewalle J., "Modeling and design of digital complex CMOS gates using neuro-fuzzy logic systems", in *Proc. of the X Brazilian Symposium on Integrated Circuit Design (SBCCI 97)* Porto Alegre, Brazil, Aug. 1997, pp. 67-76.

Espinosa J., Vandewalle J., "Constructing Fuzzy Models with Linguistic Integrity-AFRELI Algorithm", *IEEE Transactions on Fuzyy Systems*, vol. 8, no. 5, Oct. 2000, pp. 591-599.

Espinosa J., Vandewalle J., "Predictive Control Using Fuzzy Models", in *5th contribution on Part 3 Advances in Soft Computing - Engineering Design and Manufacturing*, (R. Roy, and T. Furuhashi and P.K. Chawdhry, eds.), Springer

Verlag, 1998, pp. 187-200.

Espinosa J., Vandewalle J., "Predictive Control Using Fuzzy Models Applied to a Steam Generating Unit", in *Fuzzy Logic and Intelligent Technologies for Nuclear Science and Industry*, (Ruan D., Abderrahim H.A., D'hondt P., and Kerre E. eds.), Proceedings of the third International FLINS Workshop, vol. 3 of *Fuzzy Logic and Intelligent Technologies for Nuclear Science and Industry*, World Scientific (Singapore, Singapore), 1998, pp. 151-160.

Espinosa J.,Vandewalle J., "Fuzzy Modeling and System Identification", in *Fuzzy Logic and Intelligent Technologies for Nuclear Science and Industry*, (Ruan D., Abderrahim H.A., D'Hondt P., and Kerre E., eds.), Proceedings of the third International FLINS Workshop, vol. 3 of *Fuzzy Logic and Intelligent Technologies for Nuclear Science and Industry*, World Scientific (Singapore, Singapore), 1998, pp. 111-120.

Espinosa J., Vandewalle J., "Fuzzy Modeling with Linguistic Integrity", in *Proc. of the International Workshop on Advanced Black Box Techniques for Nonlinear Modeling*, Leuven, Belgium, Jul. 1998, pp. 197-202.

Espinosa J., Vandewalle J., "Fuzzy Modeling and Identification, Using AFRELI and FuZion Algorithms", in *Proc. of the 5th. International Conference on Soft Computing and Information Intelligent Systems (IIZUKA)*, Iizuka, Japan, Oct. 1998, pp. 535-540.

Espinosa J., Hadjili M., Wertz V., Vandewalle J., "Predictive Control Using Fuzzy Models-Comparative Study", in *Proc. of the European Control Conference 1999 (ECC-99)*, Karlsruhe, Germany, Aug. 1999, 547 p.

Espinosa J., Vandewalle J., "Constrained Predictive Control using Fuzzy Models", in *Proc. of the Eight International Fuzzy Systems Association World Congress (IFSA-99)*, Taipei, Taiwan, Aug. 1999, pp. 649-654.

Espinosa J., Vandewalle J., "Linguistic Integrity : A framework for fuzzy modeling - AFRELI algorithm", in *FLSI Soft Computing Series*, Vol. 5, Chapter 2, Ed. World Scientific, Singapore, 2001.

Espinosa J., Vandewalle J., "Fuzzy Control and PID", in *Lecture Notes of the Studienamiddag + Tentoonstelling PID-regelaars : Moderne afstellingsmethodes*, Antwerp, Belgium, Mar. 1999.

Espinosa J., Vandewalle J., "Nonlinear predictive control using fuzzy models

and semidefinite programming", in *Proc. of the 18th International Conference of the North American Fuzzy Information Processing Society (NAFIPS-99)*, New York, USA, Jul. 1999, pp. 174-178.

Espinosa J., Vandewalle J., "The use of convex programming on fuzzy model based predictive control", in *Proc. of the 1999 IEEE International Symposium on Intelligent Control Intelligent Systems and Semiotics (ISIC-ISAS-1999)*, Cambridge, MA, Sep. 1999, pp. 411-416.

Espinosa J., Hadjili M., Wertz V., Vandewalle J., "Unconstrained Predictive Control Using Fuzzy Models-Comparative Study", Internal Report 99-71, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 1999.

Gutierrez A., Espinosa J., De Keyser R., "An application of neural network based predictive control", *Journal A*, vol. 41, no. 1, Mar. 2000, pp. 52-59.

Espinosa J., "Parametric Modeling and control of sliding and pre-sliding friction in a pneumatic servo system", Internal Report 99-73, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 1999.

Chapter 2

Fuzzy Modeling

Fuzzy set theory can be used in the modeling of systems. Different types of fuzzy models have been proposed in the literature. Perhaps the most used type of model is the so-called *rule base fuzzy system* [87][21]. These models are characterized by having rules with fuzzy propositions as antecedents and consequences of the rules. Other type of models are the so called *Takagi-Sugeno fuzzy models* [73] where the consequences of the rules are crisp functions of the antecedents.
After the model structures were proposed many models were developed based on "pure" *empiric knowledge*, although in many applications this proved to be insufficient and non-very efficient because most of the quantitative information was not used. Several data driven techniques are mentioned on the literature. Some of them, attempt to tune the parameters of the fuzzy systems once the structure was selected [81] [43] [39] [71]; others try to use the data to tune not only the parameters but also the structure [84] [50] [42] [74].
This chapter contributes with an extensive analysis over the capabilities for function approximation of different types of rule base fuzzy systems. The analysis includes a proof of the universal approximation capabilities of a class of fuzzy systems. Later on the chapter shows the flexibility and the geometrical properties of the approximating units that can be constructed with the fuzzy rule based systems. The constructing units can be compared with flexible tiles assembled to build a "mosaic". The chapter enumerates the most important techniques to tune fuzzy models: in this section important contributions are the gradient expressions to tune triangular and trapezoidal membership functions while keeping the $(\frac{1}{2})$ overlap. It is also important to remark the contribution presented in section 2.3 where the importance of the consequences estimation for the purpose of generalization is established. A systematic method is proposed to improve the generalization especially under circumstances where the rules are

not completely excited by the data. Finally the chapter is closed with, perhaps, the most important contribution in this chapter where an algorithm to construct fuzzy models using data is presented. The algorithm is based on the concepts of linguistic integrity [62] and optimal interface design [61]. The AFRELI (Autonomous Fuzzy Rule Extractor with Linguistic Integrity) algorithm select the number of rules and the location of the membership functions to guarantee a trade off between accuracy and comprehensibility of the rules.

2.1 Function Approximation

This section shows the approximation capabilities of some fuzzy systems. Initially the analysis is oriented to the approximation error and then to the geometric properties of the rules approximating a function.

2.1.1 System description

For this analysis, a class of fuzzy systems will be used. The class will be described as fuzzy systems with trapezoidal (or triangular) and normal ($\max \mu_i(x) = 1$) membership functions, with an overlap of maximum 0.5 with its neighboring membership functions[1].

$$\text{hgt}(\mu_i \cap \mu_{i\pm1}) \leq \frac{1}{2}.$$

The center of the fuzzy set (modal value for triangular membership functions) will be given by m_i^j as it is shown in the figure 2.1 Using the product as AND .

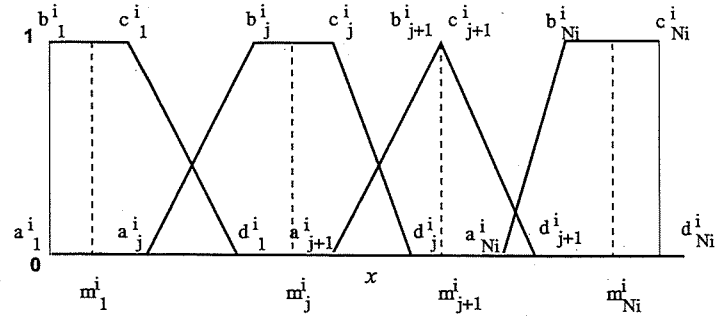


Figure 2.1: *Membership function definition for the i-th input*

[1]For notation and a basic introduction to the fuzzy set theory please see appendix A.

operation, singleton consequences and center average defuzzifier the inference process for a system with two inputs can be represented as:

$$f(x) = \frac{\sum_{j_1=1}^{N_1}\sum_{j_2=1}^{N_2}\bar{y}^{j_1 j_2}(\mu_{j_1}^1(x_1)\mu_{j_2}^2(x_2))}{\sum_{j_1=1}^{N_1}\sum_{j_2=1}^{N_2}(\mu_{j_1}^1(x_1)\mu_{j_2}^2(x_2))} \tag{2.1}$$

where $x \in \Re^2$ and x_1 and x_2 are the components, μ_j^i is the membership function of the set A_j^i defined on the universe of the i-th entry on the vector x and $\bar{y}^{j_1 j_2}$ represents a singleton consequence of the rule:

$$\text{IF } x_1 \text{ is } A_{j_1}^1 \text{ AND } x_2 \text{ is } A_{j_2}^2 \text{ THEN } f(x) \text{ is } \bar{y}^{j_1 j_2}$$

A more general representation is given by:

$$f(x) = \frac{\sum_{l=1}^{L}\bar{y}^l(\prod_{i=1}^{N}\mu_l^i(x_i))}{\sum_{l=1}^{L}(\prod_{i=1}^{N}\mu_l^i(x_i))} \tag{2.2}$$

where $x \in \Re^N$, L is the number of rules, and x_i represents each component of x. The membership functions are trapezoidal membership functions and for the case of the i-th input are parameterized as follows (see fig.2.1):

$$\mu_j^i(x) = \min\left[\max\left(\frac{x_i - a_j^i}{b_j^i - a_j^i}, 0\right), \max\left(1 - \frac{x_i - c_j^i}{d_j^i - c_j^i}, 0\right), 1\right] \tag{2.3}$$

2.1.2 Approximation Error

This section shows the approximation capabilities of the fuzzy systems by proving the following theorem:

Theorem 1 *Let $f(x)$ be a fuzzy inference system with an arbitrary number of normal membership functions (triangular or trapezoidal) with centers m_j^i, distributed over the intervals $[a_i, b_i] \forall i = 1, \ldots, N$ and covering the interval such that at least one and at most two membership functions are different from zero for a given value x_i and let $g(x) : \Re^N \to \Re$ be an unknown function. If $g(x)$ is continuously differentiable in the interval $U = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_N, b_N]$ then the fuzzy system $f(x)$ can approximate the function $g(x)$ with an arbitrary bounded error ϵ*

$$||g(x) - f(x)||_\infty \leq \epsilon \tag{2.4}$$

where $||.||_\infty$ is defined as $||e(x)||_\infty = \sup_{x \in U}|e(x)|$.

Proof:
Let the consequences $\bar{y}$ of the rules of the fuzzy system $f(x)$ be calculated as:

$$\bar{y}^{j_1 j_2 \cdots j_N} = g(m_{j_1}^1, m_{j_2}^2, \ldots, m_{j_N}^N)$$

Without lost of generality assume $g(x) : \Re^2 \to \Re$ defined and continuously differentiable in the interval $U = [a_1, b_1] \times [a_2, b_2]$. Define $U^{j_1 j_2} = [m_{j_1}^1, m_{j_1+1}^1] \times [m_{j_2}^2, m_{j_2+1}^2]$ where $j_1 = 1, 2, \ldots, N_1 - 1$ and $j_2 = 1, 2, \ldots, N_2 - 1$. The intervals $[a_i, b_i] = [m_1^i m_2^i] \cup [m_2^i m_3^i] \cup \ldots \cup [m_{N_i-1}^i m_{N_i}^i]$ with $i = 1, 2$. Then the input domain can be defined as:

$$U = [a_1, b_1] \times [a_2, b_2] = \bigcup_{j_1=1}^{N_1-1}\bigcup_{j_2=1}^{N_2-1} U^{j_1 j_2} \tag{2.5}$$

This implies that for any $x \in U$ there is a "region" $U^{j_1 j_2}$ such that $x \in U^{j_1 j_2}$, so $x_1 \in [m_{j_1}^1 m_{j_1+1}^1]$ and $x_2 \in [m_{j_2}^2 m_{j_2+1}^2]$ (see figure 2.2). Due to the selected description for the input membership functions, at least one and at most two membership functions will have a membership degree different from zero. The expression for the fuzzy system will be simplified as:

$$f(x) = \frac{\sum_{i_1=j_1}^{j_1+1}\sum_{i_2=j_2}^{j_2+1}\bar{y}^{i_1 i_2}(\mu_{i_1}^1(x_1)\mu_{i_2}^2(x_2))}{\sum_{i_1=j_1}^{j_1+1}\sum_{i_2=j_2}^{j_2+1}(\mu_{i_1}^1(x_1)\mu_{i_2}^2(x_2))} \tag{2.6}$$

$$f(x) = \frac{\sum_{i_1=j_1}^{j_1+1}\sum_{i_2=j_2}^{j_2+1} g(m_{i_1}^1, m_{i_2}^2)(\mu_{i_1}^1(x_1)\mu_{i_2}^2(x_2))}{\sum_{i_1=j_1}^{j_1+1}\sum_{i_2=j_2}^{j_2+1}(\mu_{i_1}^1(x_1)\mu_{i_2}^2(x_2))} \tag{2.7}$$

the approximation error will be given by:

$$\begin{aligned} |g(x) - f(x)| &\leq \left| g(x) - \frac{\sum_{i_1=j_1}^{j_1+1}\sum_{i_2=j_2}^{j_2+1} g(m_{i_1}^1, m_{i_2}^2)(\mu_{i_1}^1(x_1)\mu_{i_2}^2(x_2))}{\sum_{i_1=j_1}^{j_1+1}\sum_{i_2=j_2}^{j_2+1}(\mu_{i_1}^1(x_1)\mu_{i_2}^2(x_2))} \right| \\ &\leq \max_{i_1=j_1, j_1+1; i_2=j_2, j_2+1} |g(x) - g(m_{i_1}^1, m_{i_2}^2)| \end{aligned} \tag{2.8}$$

Applying the Mean Value Theorem

$$|g(x) - f(x)| \leq \max_{i_1=j_1, j_1+1; i_2=j_2, j_2+1}(||\frac{\partial g}{\partial x_1}||_\infty |x_1 - m_{i_1}^1| + ||\frac{\partial g}{\partial x_2}||_\infty |x_2 - m_{i_2}^2|) \tag{2.9}$$

since $|x_1 - m_{i_1}^1| \leq |m_{j_1+1}^1 - m_{j_1}^1|$ and $|x_2 - m_{i_2}^2| \leq |m_{j_2+1}^2 - m_{j_2}^2|$ the local error becomes:

$$|g(x) - f(x)| \leq ||\frac{\partial g}{\partial x_1}||_\infty |m_{j_1+1}^1 - m_{j_1}^1| + ||\frac{\partial g}{\partial x_2}||_\infty |m_{j_2+1}^2 - m_{j_2}^2| \tag{2.10}$$

The maximum global error will be given by:

$$||g(x) - f(x)||_\infty = \sup_{x \in U}|g(x) - f(x)|$$

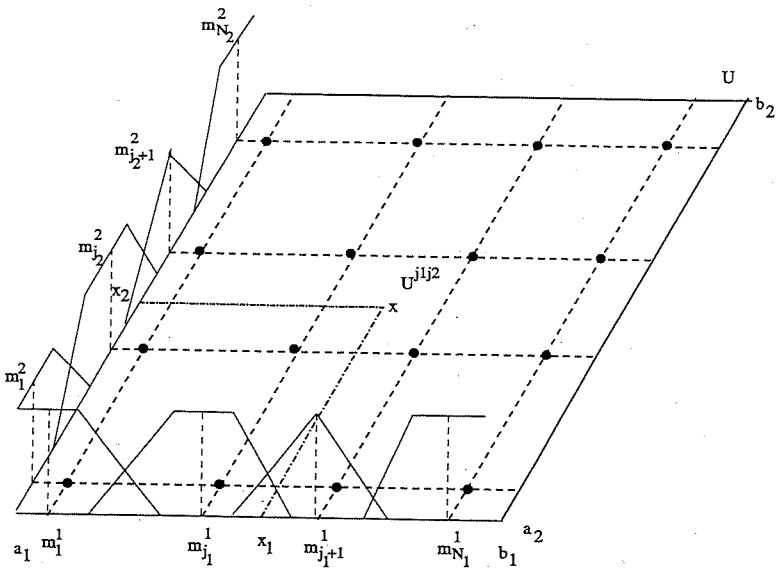


Figure 2.2: *Input Point in the Rule Space*

$$\begin{aligned} &\leq \|\frac{\partial g}{\partial x_1}\|_\infty \max_{1\leq j_1\leq N_1-1} |m^1_{j_1+1} - m^1_{j_1}| \\ &+\|\frac{\partial g}{\partial x_2}\|_\infty \max_{1\leq j_2\leq N_2-1} |m^2_{j_2+1} - m^2_{j_2}| \\ &= \|\frac{\partial g}{\partial x_1}\|_\infty e_1 + \|\frac{\partial g}{\partial x_2}\|_\infty e_2 \end{aligned} \tag{2.11}$$

where $e_i = \max_{1\leq j_i\leq N_i-1} |m^i_{j_2+1} - m^i_{j_2}|$. Extending this result to N dimensions:

$$\|g(x) - f(x)\|_\infty \leq \|\frac{\partial g}{\partial x_1}\|_\infty e_1 + \|\frac{\partial g}{\partial x_2}\|_\infty e_2 + \cdots + \|\frac{\partial g}{\partial x_N}\|_\infty e_N \tag{2.12}$$

This expression means that the maximum error in the approximation is bounded and the approximation error can be reduced to a certain $\|g - f\|_\infty \leq \epsilon$ value by playing with the distance e_i□
In other words the described fuzzy system $f(x)$ is an *universal approximator* to the function $g(x)$ within a finite domain $x \in U$ [82].

2.1.3 Constructing Units in the Fuzzy Models

A more clear perspective on the approximation capabilities of fuzzy systems can be gained by making an analysis of the interpolation properties of neighboring rules. These elements constitute the constructing units of the fuzzy systems. These units work as "patches" that can be used to approximate a given function. The study of these units is important to analyze properties such as generalization and smoothness. It is important to remark that these constructing units are the "smallest" fuzzy model that can be defined on a given interval.

Triangular Membership functions with overlap $\frac{1}{2}$

The membership functions are plotted in figure 2.3. There are two important properties on this description:

- Overlap is equal to $\frac{1}{2}$
- $\mu^i_{j_i}(x_i) = 1 - \mu^i_{j_i+1}(x_i)$

Initially, a one input one-output system will be studied to simplify the analysis. The rules for the interval $x \in [m_{j_1}, m_{j_1+1}]$ are:

$$\begin{array}{c} \text{IF } x \text{ is } A_{j_1} \text{ THEN } \bar{y}^{j_1} \\ \text{IF } x \text{ is } A_{j_1+1} \text{ THEN } \bar{y}^{j_1+1} \end{array}$$

The expression for the function defined by the fuzzy system when the input lies on the interval $x \in [m_{j_1}, m_{j_1+1}]$ will be:

$$\begin{aligned} f(x) &= \frac{\bar{y}^{j_1}\mu_{j_1}(x) + \bar{y}^{j_1+1}\mu_{j_1+1}(x)}{\mu_{j_1}(x) + \mu_{j_1+1}(x)} \\ &= \bar{y}^{j_1}\mu_{j_1}(x) + \bar{y}^{j_1+1}\mu_{j_i+1}(x) \end{aligned} \tag{2.13}$$

where the membership functions are parameterized as:

$$\mu_{j_1}(x) = \frac{m_{j_1+1} - x}{m_{j_1+1} - m_{j_1}} \tag{2.14}$$

$$\mu_{j_1+1}(x) = \frac{x - m_{j_1}}{m_{j_1+1} - m_{j_1}} \tag{2.15}$$

replacing this parameterization in (2.13):

$$\begin{aligned} f(x) &= \bar{y}^{j_1}\left(\frac{m^1_{j_1+1} - x}{m_{j_1+1} - m_{j_1}}\right) + \bar{y}^{j_1+1}\left(\frac{x - m_{j_1}}{m_{j_1+1} - m_{j_1}}\right) \\ &= \left(\frac{\bar{y}^{j_1+1} - \bar{y}^{j_1}}{m_{j_1+1} - m_{j_1}}\right)x + \left(\frac{m_{j_1+1}\bar{y}^{j_1} - m_{j_1}\bar{y}^{j_1+1}}{m_{j_1+1} - m_{j_1}}\right) \end{aligned} \tag{2.16}$$

The equation (2.16) is clearly an affine function of x. For the one-dimensional case, the fuzzy system using the described fuzzy rules works as a linear interpolation between the consequences of the rules.

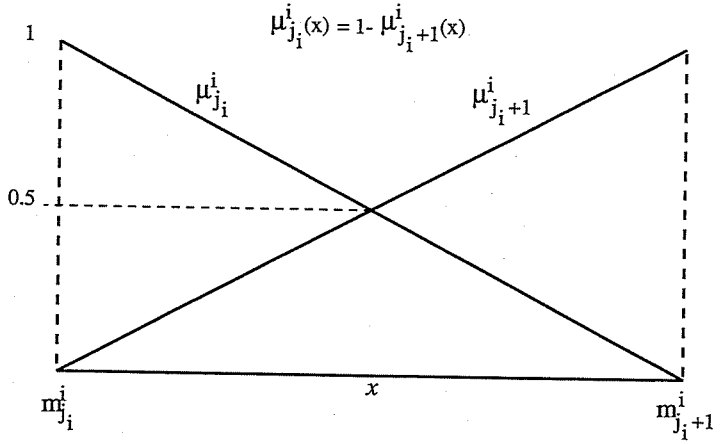


Figure 2.3: *Triangular Membership functions with overlap* $\frac{1}{2}$

Extending the analysis to the case of functions with two inputs ($x \in \Re^2$): The function defined by the fuzzy system in the interval $x \in [m^1_{j_1}, m^1_{j_1+1}] \times [m^2_{j_2}, m^2_{j_1+1}]$ is given by:

$$
\begin{aligned}
f(x) = & \\
& \frac{\bar{y}^{j_1 j_2}(\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2)) + \bar{y}^{(j_1+1)j_2}(\mu^1_{j_1+1}(x)\mu^2_{j_2}(x_2))}{\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2) + \mu^1_{j_1+1}(x)\mu^2_{j_2}(x_2) + \mu^1_{j_1}(x_1)\mu^2_{j_2+1}(x_2) + \mu^1_{j_1+1}(x)\mu^2_{j_2+1}(x_2)} \\
+ & \frac{\bar{y}^{j_1 j_2+1}(\mu^1_{j_1}(x_1)\mu^2_{j_2+1}(x_2)) + \bar{y}^{(j_1+1)(j_2+1)}(\mu^1_{j_1+1}(x)\mu^2_{j_2+1}(x_2))}{\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2) + \mu^1_{j_1+1}(x)\mu^2_{j_2}(x_2) + \mu^1_{j_1}(x_1)\mu^2_{j_2+1}(x_2) + \mu^1_{j_1+1}(x)\mu^2_{j_2+1}(x_2)} \\
= & \ \bar{y}^{j_1 j_2}(\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2)) + \bar{y}^{(j_1+1)j_2}(\mu^1_{j_1+1}(x)\mu^2_{j_2}(x_2)) + \\
& + \bar{y}^{j_1 j_2+1}(\mu^1_{j_1}(x_1)\mu^2_{j_2+1}(x_2)) + \bar{y}^{(j_1+1)(j_2+1)}(\mu^1_{j_1+1}(x)\mu^2_{j_2+1}(x_2)) \qquad (2.17)
\end{aligned}
$$

where x_1 and x_2 are the components of x, and μ^i_j represents the j-th membership function defined on the domain of x_i. Replacing the parameterization of the membership functions given by equation (2.14) and (2.15) the function of the

system can be expressed as:

$$
\begin{aligned}
f(x) = & \frac{-m^2_{j_2+1}\bar{y}^{j_1 j_2} + m^2_{j_2+1}\bar{y}^{(j_1+1)j_2} + m^2_{j_2}\bar{y}^{j_1(j_2+1)} - m^2_{j_2}\bar{y}^{(j_1+1)(j_2+1)}}{(m^1_{(j_1+1)} - m^1_{j_1})(m^2_{(j_2+1)} - m^2_{j_2})} x_1 \\
+ & \frac{-m^1_{(j_1+1)}\bar{y}^{j_1 j_2} + m^1_{j_1}\bar{y}^{(j_1+1)j_2} + m^1_{j_1+1}\bar{y}^{j_1(j_2+1)} - m^1_{j_1}\bar{y}^{(j_1+1)(j_2+1)}}{(m^1_{(j_1+1)} - m^1_{j_1})(m^2_{(j_2+1)} - m^2_{j_2})} x_2 \\
+ & \frac{\bar{y}^{j_1 j_2} - \bar{y}^{(j_1+1)j_2} - \bar{y}^{j_1(j_2+1)} + \bar{y}^{(j_1+1)(j_2+1)}}{(m^1_{(j_1+1)} - m^1_{j_1})(m^2_{(j_2+1)} - m^2_{j_2})} x_1 x_2 \\
+ & \frac{m^1_{(j_1+1)}m^2_{(j_2+1)}\bar{y}^{j_1 j_2} - m^2_{j_2+1}m^1_{j_1}\bar{y}^{(j_1+1)j_2}}{(m^1_{(j_1+1)} - m^1_{j_1})(m^2_{(j_2+1)} - m^2_{j_2})} \\
+ & \frac{-m^1_{j_1+1}m^2_{j_2}\bar{y}^{j_1(j_2+1)} + m^1_{j_1}m^2_{j_2}\bar{y}^{(j_1+1)(j_2+1)}}{(m^1_{(j_1+1)} - m^1_{j_1})(m^2_{(j_2+1)} - m^2_{j_2})} \qquad (2.18)
\end{aligned}
$$

Observe that even if the equation (2.18) looks very complex, it is nothing but a bi-linear expression such as:

$$f(x_1, x_2) = ax_1 + bx_2 + cx_1x_2 + d \qquad (2.19)$$

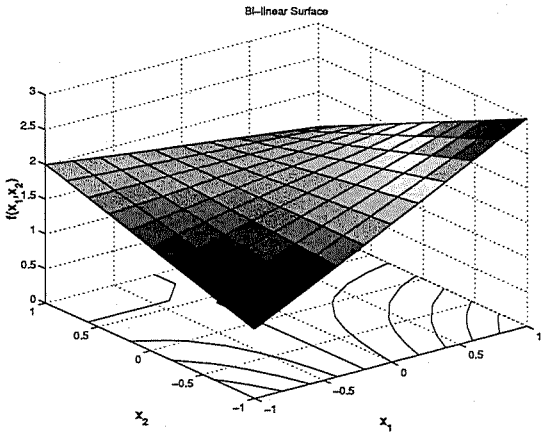


Figure 2.4: *Bilinear Surface generated with triangular membership functions with overlap* $\frac{1}{2}$

If the procedure is extended to higher dimensions, similar expressions will be found. In general this description of the fuzzy systems produces a multi-linear element of interpolation between the centers of the rules; in other words the fundamental constructing element of this kind of fuzzy systems is a finite multi-linear hyper-surface defined in the interval $U^{j_1 j_2 \cdots j_N} = [m^1_{j_1}, m^1_{j_1+1}] \times [m^2_{j_2}, m^2_{j_2+1}] \times \ldots \times [m^N_{j_N}, m^N_{j_N+1}]$.

Polynomial Membership Function with overlap $\frac{1}{2}$

The membership functions are shown in the figures 2.5 and 2.6. Observe that these membership functions preserve the properties described in the previous section and two new additional conditions are included. For the input domain x_i the conditions are:

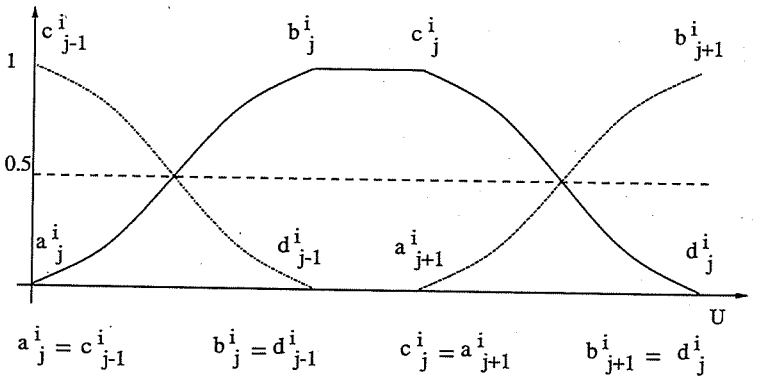


Figure 2.5: *Polynomial membership functions*

- Overlap is equal to $\frac{1}{2}$
- $\mu^i_{j_i}(x_i) = 1 - \mu^i_{j_i+1}(x_i)$
- $\frac{\partial \mu^i_{j_i}(m_{j_i})}{\partial x_i} = 0$
- $\frac{\partial \mu^i_{j_i}(m_{j_i+1})}{\partial x_i} = 0$

With these conditions the polynomial describing the membership functions in the interval $[m_{j_i}, m_{j_i+1}]$ can be constructed. The membership function can be

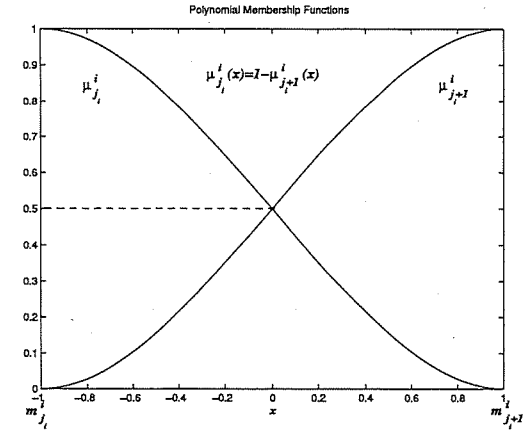


Figure 2.6: *Polynomial membership functions with overlap* $\frac{1}{2}$

described as follows:

$$\mu^i_{j_i}(x_i) = \begin{cases} 0 & \text{if } x_i < a^i_{j_i} \\ \alpha^i_{j_i} x_i^3 + \beta^i_{j_i} x_i^2 + \gamma^i_{j_i} x_i + \delta^i_{j_i} & \text{if } a^i_{j_i} < x_i < b^i_{j_i} \\ 1 & \text{if } b^i_{j_i} \leq x_i \leq c^i_{j_i} \\ \zeta^i_{j_i} x_i^3 + \eta^i_{j_i} x_i^2 + \theta^i_{j_i} x_i + \vartheta^i_{j_i} & \text{if } c^i_{j_i} < x_i < d^i_{j_i} \\ 0 & \text{if } x_i > d^i_{j_i} \end{cases} \tag{2.20}$$

This parameterization is simplified by the fact that the parameters $\alpha, \beta, \gamma, \delta, \zeta, \eta, \theta$ and ϑ can be obtained from the parameters a, b, c and d. The overlap properties between the membership functions simplify the parameterization even more as it is shown in (2.21).

$$\begin{aligned} \alpha^i_{j_i} &= -\zeta^i_{j_i-1} = \frac{2}{(a^i_{j_i} - b^i_{j_i})^3} \\ \beta^i_{j_i} &= -\eta^i_{j_i-1} = \frac{-3(a^i_{j_i} + b^i_{j_i})}{(a^i_{j_i} - b^i_{j_i})^3} \\ \gamma^i_{j_i} &= -\theta^i_{j_i-1} = \frac{6 a^i_{j_i} b^i_{j_i}}{(a^i_{j_i} - b^i_{j_i})^3} \\ \delta^i_{j_i} + \vartheta^i_{j_i-1} &= 1 \end{aligned}$$

$$\delta^i_{j_i-1} = \frac{a^{i\,2}_{j_i}(a^i_{j_i} - 3b^i_{j_i})}{(a^i_{j_i} - b^i_{j_i})^3} \tag{2.21}$$

Initially, a one input one-output system will be studied to simplify the analysis. The rules for the interval $x \in [m_{j_1}, m_{j_1+1}]$ are:

IF x is A_{j_1} THEN $\bar{y}^{j_1}$
IF x is A_{j_1+1} THEN $\bar{y}^{j_1+1}$

The expression for the function generated by a system with one input defined on the interval $x \in [m_{j_1}, m_{j_1+1}]$ and membership functions as shown in figure 2.6.

$$\begin{aligned} f(x) &= \frac{\bar{y}^{j_1}\mu_{j_1}(x) + \bar{y}^{j_1+1}\mu_{j_1+1}(x)}{\mu^1_{j_1}(x) + \mu_{j_1+1}(x)} \\ &= \bar{y}^{j_1}\mu_{j_1}(x) + \bar{y}^{j_1+1}\mu_{j_i+1}(x) \\ &= \bar{y}^{j_1}(\zeta_{j_1}x^3 + \eta_{j_1}x^2 + \theta_{j_1}x + \vartheta_{j_1}) \\ &+ \bar{y}^{j_1+1}(\alpha_{j_i+1}x^3 + \beta_{j_1+1}x^2 + \gamma_{j_1+1}x + \delta_{j_1+1}) \end{aligned} \tag{2.22}$$

replacing (2.21) in (2.22)

$$f(x) = (\bar{y}^{j_1+1} - \bar{y}^{j_1})(\alpha_{j_i+1}x^3 + \beta_{j_1+1}x^2 + \gamma_{j_1+1}x + \delta_{j_1+1}) + \bar{y}^{j_1} \tag{2.23}$$

It is clear from (2.23) that the interpolation given by this type of membership functions is a third order polynomial. A very interesting property of this interpolation is that it has a continuous derivative. If the analysis is extended to more dimensions in the domain of the function the interpolating unit will be a third order polynomial on each of the inputs. A graphical representation of this constructing unit is shown in figure 2.8.

Gaussian Membership Functions with $\sigma = 0.6(m^i_{j_i+1} - m^i_{j_i})$

For this type of membership functions the parameterization is given as:

$$\mu^i_{j_i}(x_i) = e^{-\left(\frac{x(i)-m^i_{j_i}}{\sigma^i_{j_i}}\right)^2} \tag{2.24}$$

The expression for a fuzzy system with one input and one output and only two membership functions will generate the following interpolation in the interval $x \in [m_{j_1}, m_{j_1+1}]$

$$f(x) = \frac{\bar{y}^{j_1} e^{-\left(\frac{x-m_{j_1}}{\sigma}\right)^2} + \bar{y}^{j_1+1} e^{-\left(\frac{x-m_{j_1+1}}{\sigma}\right)^2}}{e^{-\left(\frac{x-m_{j_1}}{\sigma}\right)^2} + e^{-\left(\frac{x-m_{j_1+1}}{\sigma}\right)^2}} \tag{2.25}$$

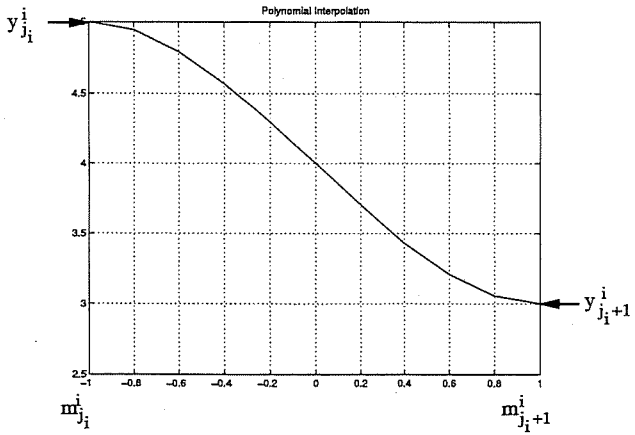


Figure 2.7: *Polynomial interpolation generated with polynomial membership functions with overlap* $\frac{1}{2}$

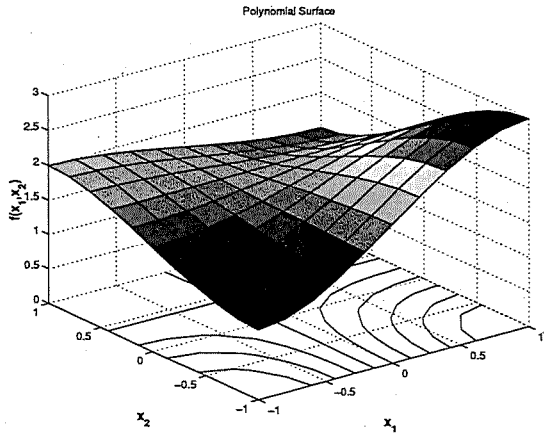


Figure 2.8: *Polynomial surface generated with polynomial membership functions with overlap* $\frac{1}{2}$ *in a system with two inputs*

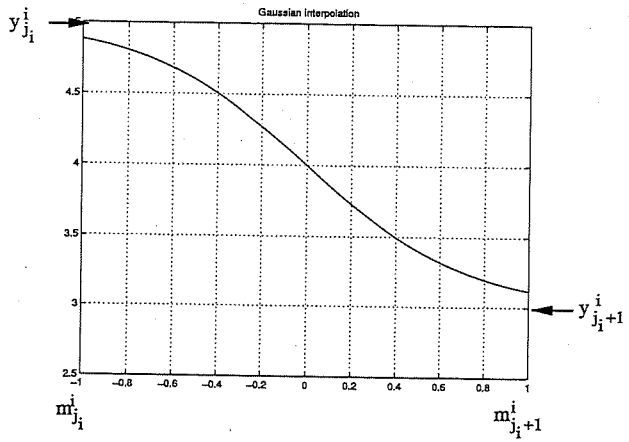


Figure 2.9: *Gaussian interpolation generated with Gaussian membership functions with* $\sigma = 0.6(m^i_{j_i+1} - m^i_{j_i})$

The interpolation generated by this expression is shown in figure 2.9. Observe that this interpolation is smooth but the membership functions are not complementary. For this reason, the denominator in (2.25) is not equal to one. Additionally this fact will make the interpolation less accurate because more than one rule will be activated when the inputs correspond to a modal value. The expression for the function generated by the fuzzy system when the domain of $x \in U$ is defined as $U = [m^1_{j_1}, m^1_{j_1+1}] \times [m^2_{j_2}, m^2_{j_2+1}]$

$$f(x) = \frac{\bar{y}^{j_1 j_2}(\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2)) + \bar{y}^{(j_1+1)j_2}(\mu^1_{j_1+1}(x)\mu^2_{j_2}(x_2))}{\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2) + \mu^1_{j_1+1}(x)\mu^2_{j_2}(x_2) + \mu^1_{j_1}(x_1)\mu^2_{j_2+1}(x_2) + \mu^1_{j_1+1}(x)\mu^2_{j_2+1}(x_2)} + \frac{\bar{y}^{j_1 j_2+1}(\mu^1_{j_1}(x_1)\mu^2_{j_2+1}(x_2)) + \bar{y}^{(j_1+1)(j_2+1)}(\mu^1_{j_1+1}(x)\mu^2_{j_2+1}(x_2))}{\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2) + \mu^1_{j_1+1}(x)\mu^2_{j_2}(x_2) + \mu^1_{j_1}(x_1)\mu^2_{j_2+1}(x_2) + \mu^1_{j_1+1}(x)\mu^2_{j_2+1}(x_2)} \quad (2.26)$$

The product of two membership functions of this type can be parameterized as follows:

$$\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2) = e^{-\left(\frac{\sigma_2^2(x(1)-m^1_{j_1})^2+\sigma_1^2(x(2)-m^2_{j_2})^2}{\sigma_1^2\sigma_2^2}\right)}$$

$$\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2) = \mu_{j_1 j_2}(x) = e^{-\frac{d^{j_1 j_2 T}\Sigma d^{j_1 j_2}}{(\sigma_1\sigma_2)^2}} \quad (2.27)$$

with $\Sigma = \text{diag}(\sigma_1, \sigma_2)$ and $d^{j_1 j_2} = [x_1 - m^1_{j_1}, x_2 - m^2_{j_2}]^T$. Then the expression for the function will be:

$$f(x) = \frac{\bar{y}^{j_1 j_2}\mu_{j_1 j_2}(x) + \bar{y}^{(j_1+1)j_2}\mu_{(j_1+1)j_2}(x)}{\mu_{j_1 j_2}(x) + \mu_{(j_1+1)j_2}(x) + \mu_{j_1 j_2+1}(x) + \mu_{(j_1+1)(j_2+1)}(x)} + \frac{\bar{y}^{j_1 j_2+1}\mu_{j_1 j_2+1}(x) + \bar{y}^{(j_1+1)(j_2+1)}\mu_{(j_1+1)(j_2+1)}(x)}{\mu_{j_1 j_2}(x) + \mu_{(j_1+1)j_2}(x) + \mu_{j_1 j_2+1}(x) + \mu_{(j_1+1)(j_2+1)}(x)} \quad (2.28)$$

The surface generated by this type of systems is shown in figure 2.10. The surface generated is a weighted sum of gaussians. For a domain of n dimensions the function is a weighted sum of gaussians calculated using n dimensional distances.

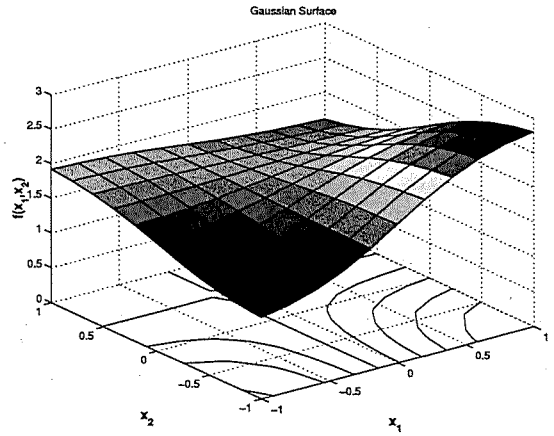


Figure 2.10: *Gaussian surface generated with Gaussian membership functions with* $\sigma = 0.6(m^i_{j_i+1} - m^i_{j_i})$

Other approximating units

Some other approximating units can be obtained with different distribution of the membership functions. Because the analytical expression for such "units" is very complex, only the graphical representation is presented here. The figures show the bi-dimensional and tri-dimensional representation in the figures

2.12, 2.13 and 2.14. The membership functions used to generate these graphics are shown in figure 2.11

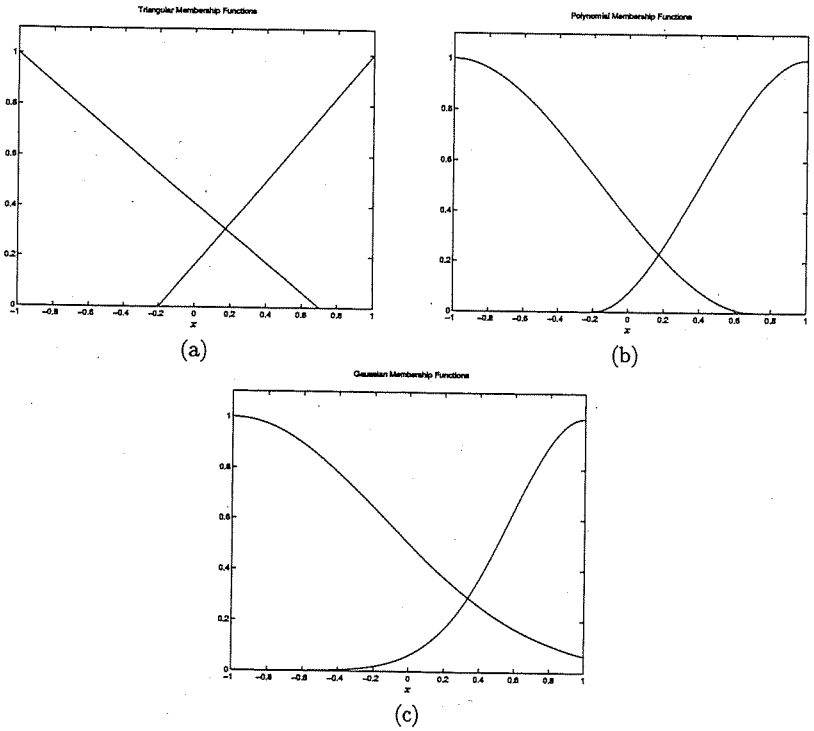


Figure 2.11: *(a) Triangular membership functions with overlap different to* $\frac{1}{2}$ *(b) Polynomial membership functions with overlap different to* $\frac{1}{2}$ *(c) Gaussian membership functions with different* σ

2.2 Constructing fuzzy models from Input-Output data

The construction of fuzzy models involves the selection of several parameters: the position, the shape and the distribution of the membership functions, the

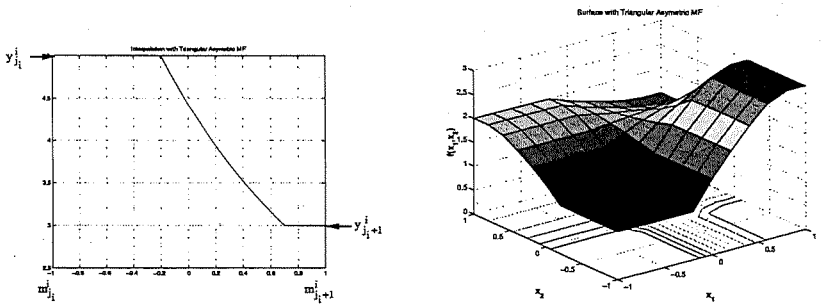


Figure 2.12: *Interpolation and Surfaces generated with triangular membership functions with overlap different to* $\frac{1}{2}$

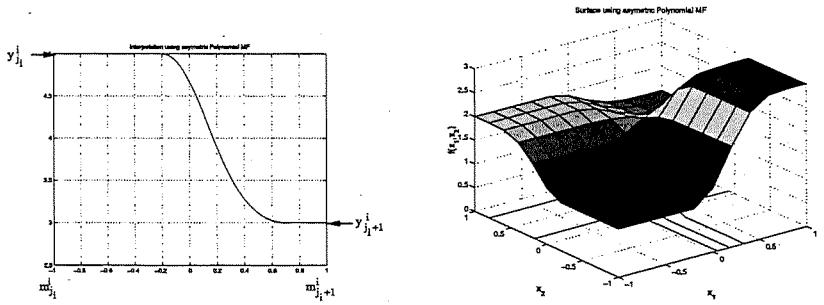


Figure 2.13: *Interpolation and Surfaces generated with polynomial membership functions with overlap different to* $\frac{1}{2}$

rule base construction, the selection of the logical operations, the consequences of the rules, etc. A typical approach is to set in advance the logical operations and the type of membership functions using different criteria. According to the tuned parameters and the strategies different methods have been proposed in the literature. This section presents the following strategies mainly as a survey, but some contributions are made in the gradient descent method where some tuning formulae were derived for some types of membership functions. Some of the most popular methods are:

- Mosaic or table look-up scheme [80]

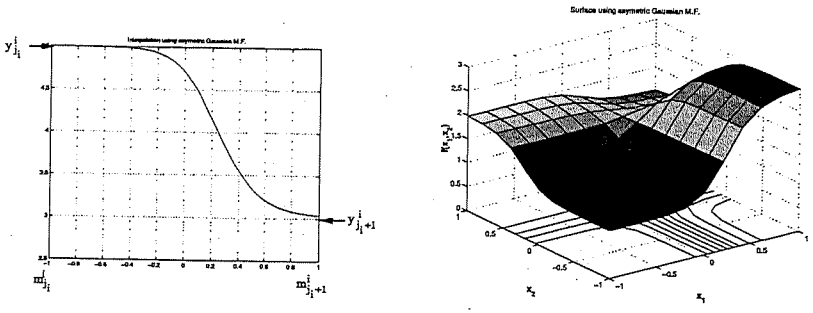


Figure 2.14: *Interpolation and Surfaces generated with Gaussian membership functions with different σ*

- Using gradient descent [80] [40]
- Using clustering and gradient descent [50] [84]
- Using evolutionary strategies [10]

2.2.1 Mosaic or table look-up scheme

The basic scheme of method was proposed by Wang [80]. Here some simple modifications are introduced and these modifications are related to the consequence calculation. In this method the position, the shape and the distribution of the membership functions are choices for the designer. The rule base is composed and the method finds only the consequences of the rules.
Assume a sequence of input-output $\{x^i, y^i\}\, i = 1, \ldots, N$ data are collected, the inputs $x^i \in U \subset \Re^p$ and the output $y^i \in V \subset \Re$. The subset U is a portion of the space $\Re^p$ and it is defined as $U = [a_1, b_1] \times \ldots \times [a_p, b_p]$. The procedure to construct the model proceeds as follows.

- For each of the p inputs of the system distribute over the interval $[a_i, b_i]$ N_i membership functions. The shape, the position and the distribution is a user's choice, the only condition is that the full interval is covered and at least two membership functions are placed on each input. As it was shown in the previous sections, the shape and the distribution affect the smoothness and the accuracy of the approximation.
- Generate the rule base using all possible combinations among the antecedents and the AND operator (choosing in advance "min" or "product" operator).

- Calculate the inference of each rule. For rule l:

$$\mu_l(x^i) = \min\{\mu_l^1(x_1^i), \mu_l^2(x_2^i), \ldots, \mu_l^p(x_p^i)\} \tag{2.29}$$

or

$$\mu_l(x^i) = \mu_l^1(x_1^i).\mu_l^2(x_2^i).\ldots.\mu_l^p(x_p^i) \tag{2.30}$$

so the general expression for this fuzzy system with L rules will be given by

$$f(x^i) = \frac{\sum_{l=1}^{L} \bar{y}^l \mu_l(x^i)}{\sum_{l=1}^{L} \mu_l(x^i)} \tag{2.31}$$

- Calculate the consequence parameters $\bar{y}^l\, l = 1, \ldots, L$ such that $f(x^i) \approx y^i$. Observe that the expression (2.31) can be written as:

$$f(x^i) = \sum_{l=1}^{L} \bar{y}^l w_l(x^i) \tag{2.32}$$

where

$$w_l(x^i) = \frac{\mu_l(x^i)}{\sum_{l=1}^{L} \mu_l(x^i)} = w_l^i \tag{2.33}$$

The N output values can be represented as the vector Y in terms of the inference process:

$$\underbrace{\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} w_1^1 & w_2^1 & \ldots & w_L^1 \\ w_1^2 & w_2^2 & \ldots & w_L^2 \\ \vdots & \vdots & \ddots & \vdots \\ w_1^N & w_2^N & \ldots & w_L^N \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} \bar{y}^1 \\ \bar{y}^2 \\ \vdots \\ \bar{y}^L \end{bmatrix}}_{\theta} + \underbrace{\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix}}_{E} \tag{2.34}$$

In this case the vector E is the approximation error and the aim is to reduce as much as possible the norm of this vector. Using quadratic norm:

$$\min_{\theta} ||E||_2 = \min_{\theta} ||Y - W\theta||_2 \tag{2.35}$$

It is a least squares problem and the consequences can be calculated using least squares. The solution to this least squares problem is:

$$\theta = \arg\min_{\theta} ||E||_2 = (W^T W)^{-1} Y^T W \tag{2.36}$$

This solution is applicable as far as the $\text{rank}(W^T W) = \dim(\theta)$, otherwise other methods must be applied to guarantee a reliable set of consequences for the rules. In section 2.3 a method based Recursive Least Squares is detailed.

2.2.2 Using gradient descent

This method requires the definition of the number of membership functions and their shape. Normally the AND function is fixed to be the "product" because an analytical expression for the gradient of the cost function is needed. The initial position of the membership functions is another element that must be chosen. The method proceeds as follows:

- For each of the p inputs of the system distribute over the interval $[a_i, b_i]$ N_i membership functions. The shape, the initial positions and the distribution is a user's choice. The membership functions must cover the input interval and at least two membership functions should be placed on each input domain.
- Generate the rule base using all possible combinations among the antecedents and the AND operator using "product".
- Initialize the value of the consequences using prior knowledge, least squares or recursive least squares.
- Optimize the value of the consequences $\bar{y}^l$ and the parameters of the membership functions. The criteria will be to minimize the cost function described in previous section but now the optimization will adjust the membership functions of the antecedents. The cost function can be described as:

$$J = \frac{1}{2}\sum_{i=1}^{N}(y^i - f(x^i, \theta))^2 \tag{2.37}$$

where θ is a vector representing all the "adjustable" parameters (consequences, parameters of the membership functions) of the fuzzy system $f(.,.)$. The problem will be the minimization of the cost function J, this minimization is a nonlinear non-convex optimization problem. The objective is to obtain an "acceptable" solution and not necessarily "the global minima" of this cost function. Different schemes for optimization can be applied to find this solution. Probably the simplest one will be the gradient descent method. This method consists in an iterative calculation of the parameters oriented to the negative direction of the gradient. The explanation behind this method is that by taking the negative direction of the gradient the steepest route towards the minimum will be taken. This

descent direction does not guarantee convergence of the scheme, for this reason the α parameter is introduced and it can be modified to improve the convergence rate and properties. Some choices of α are given by Newton and Quasi-Newton methods [36].

$$\theta(k+1) = \theta(k) + \alpha\frac{\partial J}{\partial \theta} \tag{2.38}$$

α is sometimes called the "learning rate". The gradient descent method can be modified; for example, by calculating the consequences by means of least squares (see ANFIS scheme [40]).

The gradient of the cost function will be in general:

$$\frac{\partial J}{\partial \theta} = \sum_{i=1}^{N}(y^i - f(x^i,\theta))\frac{\partial f(x^i,\theta)}{\partial \theta} = \sum_{i=1}^{N} e^i \frac{\partial f(x^i,\theta)}{\partial \theta} \tag{2.39}$$

Using the general expression of the fuzzy system can be parameterized as:

$$f(x^i) = \frac{\sum_{l=1}^{L}\bar{y}^l\mu_l(x^i)}{\sum_{l=1}^{L}\mu_l(x^i)} = \frac{A}{B} \tag{2.40}$$

The updating of the consequence parameters will be independent of the parameterization of the membership functions and will be given by:

$$\bar{y}^l(k+1) = \bar{y}^l(k) + \alpha\sum_{i=1}^{N}(y - f(x^i))\frac{\mu_l(x^i)}{B} \tag{2.41}$$

The expressions to update the parameters of the membership functions are different for each parameterization. Special attention is devoted to the gradient calculation to guarantee a 0.5 overlap between contiguous membership functions. The updating formulas for some of the membership functions are:

Gradient updating for trapezoidal membership functions

Assuming the parameterization given in the expression

$$\mu_j^i(x_i, a_j^i, b_j^i, c_j^i, d_j^i) = \min\left[\max\left(\frac{x_i - a_j^i}{b_j^i - a_j^i}, 0\right), \max\left(1 - \frac{x_i - c_j^i}{d_j^i - c_j^i}, 0\right), 1\right] \tag{2.42}$$

the updating formulas will be:

$$\begin{aligned} a_j^i(k+1) &= a_j^i(k) \\ &+ \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial a_j^i} \end{aligned} \quad (2.43)$$

$$\begin{aligned} b_j^i(k+1) &= b_j^i(k) \\ &+ \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial b_j^i} \end{aligned} \quad (2.44)$$

$$\begin{aligned} c_j^i(k+1) &= c_j^i(k) \\ &+ \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial c_j^i} \end{aligned} \quad (2.45)$$

$$\begin{aligned} d_j^i(k+1) &= d_j^i(k) \\ &+ \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial d_j^i} \end{aligned} \quad (2.46)$$

where the set $\mathcal{U}$ is the set of rules that includes the function $\mu_j^i(x)$ in the antecedents and

$$\frac{\partial \mu_j^i(x_i^t)}{\partial a_j^i} = \begin{cases} 0 & \text{if } x_i^t < a_j^i \\ \frac{x_i^t - b_j^i}{(b_j^i - a_j^i)^2} & \text{if } a_j^i < x_i^t < b_j^i \\ 0 & \text{if } x_i^t > b_j^i \end{cases} \quad (2.47)$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial b_j^i} = \begin{cases} 0 & \text{if } x_i^t < a_j^i \\ \frac{a_j^i - x_i^t}{(b_j^i - a_j^i)^2} & \text{if } a_j^i < x_i^t < b_j^i \\ 0 & \text{if } x_i^t > b_j^i \end{cases} \quad (2.48)$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial c_j^i} = \begin{cases} 0 & \text{if } x_i^t < c_j^i \\ \frac{d_j^i - x_i^t}{(d_j^i - c_j^i)^2} & \text{if } c_j^i < x_i^t < d_j^i \\ 0 & \text{if } x_i^t > d_j^i \end{cases} \quad (2.49)$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial d_j^i} = \begin{cases} 0 & \text{if } x_i^t < c_j^i \\ \frac{x_i^t - c_j^i}{(d_i^i - c_j^i)^2} & \text{if } c_j^i < x_i^t < d_j^i \\ 0 & \text{if } x_i^t > d_j^i \end{cases} \quad (2.50)$$

It is important to remark that the updating should preserve the condition $a_j^i \leq b_j^i \leq c_j^i \leq d_j^i$.
The adaptation rule can be applied to triangular membership functions by just making $b_j^i = c_j^i$.

Gradient updating for triangular membership functions with overlap $\frac{1}{2}$

To calculate the updating rules to tune this type of membership functions and to preserve the overlap, the membership functions are parameterized by using only their modal values. This parameterization preserves not only the overlap but also reduces the number of parameters to be tuned. Triangular membership functions are parameterized by the position of their three vertices; but the condition of overlap $\frac{1}{2}$ makes the lower right vertex of one membership function to be at the same position as the modal value of the next membership function. So, instead of tuning three parameters (the vertices) only one parameter is tuned for each membership function.
The parameterization for a triangular membership function using the modal values as parameters is:

$$\mu_j^i(x_i, m_{j-1}^i, m_j^i, m_{j+1}^i) = \max\left[0, \min\left(\frac{x_i - m_{j-1}}{m_j^i - m_{j-1}^i}, 1 - \frac{x_i - m_j^i}{m_{j+1}^i - m_j^i}\right)\right] \quad (2.51)$$

the updating formula will be:

$$\begin{aligned} m_j^i(k+1) = m_j^i(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \\ \left[\sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_{j-1}^i(x_i^t)} \frac{\partial \mu_{j-1}^i(x_i^t)}{\partial m_j^i} + \sum_{l \in \mathcal{V}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial m_j^i} \right. \\ \left. + \sum_{l \in \mathcal{W}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_{j+1}^i(x_i^t)} \frac{\partial \mu_{j+1}^i(x_i^t)}{\partial m_j^i}\right] \end{aligned} \quad (2.52)$$

where the sets $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{W}$ are the set of rules that includes the functions $\mu_{j-1}^i(.)$, $\mu_j^i(.)$ and $\mu_{j+1}^i(.)$ respectively, and with

$$\frac{\partial \mu_{j-1}^i(x_i^t)}{\partial m_j^i} = \begin{cases} 0 & \text{if } x_i^t < m_{j-1}^i \\ \frac{x_i^t - m_j}{(m_j^i - m_{j-1}^i)^2} & \text{if } m_{j-1}^i < x_i^t < m_j^i \\ 0 & \text{if } x_i^t > m_j^i \end{cases} \quad (2.53)$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial m_j^i} = \begin{cases} 0 & \text{if } x_i^t < m_{j-1}^i \\ \frac{m_{j-1} - x_i^t}{(m_j^i - m_{j-1}^i)^2} & \text{if } m_{j-1}^i < x_i^t < m_j^i \\ \frac{m_{j+1} - x_i^t}{(m_{j+1}^i - m_j^i)^2} & \text{if } m_j^i < x_i^t < m_{j+1}^i \\ 0 & \text{if } x_i^t > m_{j+1}^i \end{cases} \quad (2.54)$$

$$\frac{\partial \mu_{j+1}^i(x_i^t)}{\partial m_j^i} = \begin{cases} 0 & \text{if } x_i^t < m_j^i \\ \frac{x_i^t - m_{j+1}}{(m_{j+1}^i - m_j^i)^2} & \text{if } m_j^i < x_i^t < m_{j+1}^i \\ 0 & \text{if } x_i^t > m_{j+1}^i \end{cases} \quad (2.55)$$

Here the adaptation must be constrained such that the condition $m_j^i \leq m_{j+1}^i$ is preserved.

Gradient updating for polynomial membership functions

Assuming the parameterization given in the expression

$$\mu_j^i(x_i) = \begin{cases} 0 & \text{if } x_i < a_j^i \\ \alpha_j^i x_i^3 + \beta_j^i x_i^2 + \gamma_j^i x_i + \delta_j^i & \text{if } a_j^i < x_i < b_j^i \\ 1 & \text{if } b_j^i \leq x_i \leq c_j^i \\ \zeta_j^i x_i^3 + \eta_j^i x_i^2 + \theta_j^i x_i + \vartheta_j^i & \text{if } c_j^i < x_i < d_j^i \\ 0 & \text{if } x_i > d_j^i \end{cases} \quad (2.56)$$

The expressions to update the parameters a_j^i, b_j^i, c_j^i and d_j^i are similar to the ones used for the trapezoidal membership functions, only the expression for the gradient of the membership functions changes.

$$a_j^i(k+1) = a_j^i(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial a_j^i} \quad (2.57)$$

$$b_j^i(k+1) = b_j^i(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial b_j^i} \quad (2.58)$$

$$c_j^i(k+1) = c_j^i(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial c_j^i} \quad (2.59)$$

$$d_j^i(k+1) = d_j^i(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial d_j^i} \quad (2.60)$$

with

$$\frac{\partial \mu_j^i(x_i^t)}{\partial a_j^i} = \begin{cases} 0 & \text{if } x_i^t < a_j^i \\ 6\frac{(b_j^i - x_i^t)^2 (a_j^i - x_i^t)}{(a_j^i - b_j^i)^4} & \text{if } a_j^i < x_i^t < b_j^i \\ 0 & \text{if } x_i^t > b_j^i \end{cases} \quad (2.61)$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial b_j^i} = \begin{cases} 0 & \text{if } x_i^t < a_j^i \\ -6\frac{(a_j^i - x_i^t)^2 (b_j^i - x_i^t)}{(a_j^i - b_j^i)^4} & \text{if } a_j^i < x_i^t < b_j^i \\ 0 & \text{if } x_i^t > b_j^i \end{cases} \quad (2.62)$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial c_j^i} = \begin{cases} 0 & \text{if } x_i^t < c_j^i \\ -6\frac{(d_j^i - x_i^t)^2 (c_j^i - x_i^t)}{(c_j^i - d_j^i)^4} & \text{if } c_j^i < x_i^t < d_j^i \\ 0 & \text{if } x_i^t > d_j^i \end{cases} \quad (2.63)$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial d_j^i} = \begin{cases} 0 & \text{if } x_i^t < c_j^i \\ 6\frac{(c_j^i - x_i^t)^2 (d_j^i - x_i^t)}{(c_j^i - d_j^i)^4} & \text{if } c_j^i < x_i^t < d_j^i \\ 0 & \text{if } x_i^t > d_j^i \end{cases} \quad (2.64)$$

the adaptation algorithm should preserve the condition $a_j^i \leq b_j^i \leq c_j^i \leq d_j^i$.

Gradient updating for polynomial membership functions with overlap $\frac{1}{2}$ and $b_j^i = c_j^i = m_j^i$

To guarantee the overlap $\frac{1}{2}$ the parameterization of the membership functions is made using only their modal values. The number of adjusted parameters is reduced, instead of adjusting four parameters $(a_j^i, b_j^i, c_j^i, d_j^i)$, for each membership function, only one parameter m_j^i is adjusted. Observe that the overlap $\frac{1}{2}$ is preserved only if the parameters $a_{j+1}^i, b_j^i, c_j^i, d_{j-1}^i$ describing the polynomial membership function are equal among each other and equal to the modal value m_j^i. The parameterization using the modal values is as follows:

$$\mu_j^i(x_i) = \begin{cases} 0 & \text{if } x_i < m_{j-1}^i \\ \alpha_j^i x_i^3 + \beta_j^i x_i^2 + \gamma_j^i x_i + \delta_j^i & \text{if } m_{j-1}^i < x_i < m_j^i \\ -\alpha_{j+1}^i x_i^3 - \beta_{j+1}^i x_i^2 - \gamma_{j+1}^i x_i - \delta_{j+1}^i + 1 & \text{if } m_j^i < x_i < m_{j+1}^i \\ 0 & \text{if } x_i > m_{j+1}^i \end{cases} \quad (2.65)$$

The parameters m_j^i are updated with a similar formulas as the one used for the triangular membership functions with overlap $\frac{1}{2}$

$$m_j^i(k+1) = m_j^i(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \left[\sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_{j-1}^i(x_i^t)} \frac{\partial \mu_{j-1}^i(x_i^t)}{\partial m_j^i} + \sum_{l \in \mathcal{V}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial m_j^i} + \sum_{l \in \mathcal{W}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_{j+1}^i(x_i^t)} \frac{\partial \mu_{j+1}^i(x_i^t)}{\partial m_j^i} \right] \quad (2.66)$$

where the sets $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{W}$ are the set of rules that includes the functions $\mu^i_{j-1}(.)$, $\mu^i_j(.)$ and $\mu^i_{j+1}(.)$ respectively, and with

$$\frac{\partial \mu^i_{j-1}(x^t_i)}{\partial m^i_j} = \begin{cases} 0 & \text{if } x^t_i < m^i_{j-1} \\ 6\frac{(m_{j-1}-x^t_i)^2(m_j-x^t_i)}{(m^i_{j-1}-m^i_j)^4} & \text{if } m^i_{j-1} < x^t_i < m^i_j \\ 0 & \text{if } x^t_i > m^i_j \end{cases} \quad (2.67)$$

$$\frac{\partial \mu^i_j(x^t_i)}{\partial m^i_j} = \begin{cases} 0 & \text{if } x^t_i < m^i_{j-1} \\ -6\frac{(m_{j-1}-x^t_i)^2(m_j-x^t_i)}{(m^i_{j-1}-m^i_j)^4} & \text{if } m^i_{j-1} < x^t_i < m^i_j \\ -6\frac{(m_{j+1}-x^t_i)^2(m_j-x^t_i)}{(m^i_j-m^i_{j+1})^4} & \text{if } m^i_j < x^t_i < m^i_{j+1} \\ 0 & \text{if } x^t_i > m^i_{j+1} \end{cases} \quad (2.68)$$

$$\frac{\partial \mu^i_{j+1}(x^t_i)}{\partial m^i_j} = \begin{cases} 0 & \text{if } x^t_i < m^i_j \\ 6\frac{(m_{j+1}-x^t_i)^2(m_j-x^t_i)}{(m^i_j-m^i_{j+1})^4} & \text{if } m^i_j < x^t_i < m^i_{j+1} \\ 0 & \text{if } x^t_i > m^i_{j+1} \end{cases} \quad (2.69)$$

Observe that the adaptation must preserve the condition $m^i_j \leq m^i_{j+1}$.

Gradient updating for Gaussian membership functions

The parameterization of the membership functions is given by:

$$\mu^i_j(x^t_i) = \exp(-(\frac{x^t_i - \bar{x}^i_j}{\sigma^i_j})^2) \quad (2.70)$$

the updating formula for the parameters $\bar{x}^i_j$ and σ^i_j will be given by:

$$\bar{x}^i_j(k+1) = \bar{x}^i_j(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} 2(\bar{y}^l - f(x^t))\mu_l(x^t)\frac{x^t_i - \bar{x}^i_j}{\sigma^{i\,2}_j} \quad (2.71)$$

$$\sigma^i_j(k+1) = \sigma^i_j(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} 2(\bar{y}^l - f(x^t))\mu_l(x^t)\frac{(x^t_i - \bar{x}^i_j)^2}{\sigma^{i\,3}_j} \quad (2.72)$$

where $\mathcal{U}$ is the set of rules with the antecedent term $\mu^i_j(.)$.

2.2.3 Using Clustering and Gradient Descent

This method is a data driven method. The main idea is to use clustering algorithms to find clusters among the data. The clusters are assimilated as multidimensional fuzzy sets. The fuzzy system is constructed by means of projecting

the clusters into the input space and approximating the projected cluster with a fuzzy set. The advantage of this method is that it generates automatically the membership functions, giving as user's choices only the parameters of the clustering algorithms (number of clusters and distance function). Therefore, a typical procedure will be:

- Collect the data and construct a set of vectors $Z^t = \{x^{t^T}, y^t\}$
- Search for clusters using the Fuzzy C-means algorithm [8]or mountain-clustering algorithm [84] if the dimension of the input is small. The appendix B includes a description of the mentioned algorithms.
- Project the membership functions in the input space.
- Construct the rules with the projected membership functions.
- Calculate the consequences using Recursive Least Squares.
- Adjust the parameters of the antecedents (if needed) using gradient descent.

2.2.4 Using Evolutionary Strategies

The evolutionary strategies are computational algorithms that use methods derived from the concept of "Natural Evolution". Some of the methods include reproduction, mutation and selection. The use of these algorithms has been oriented to the search of parameters such that a certain computational entity can achieve some goals.
In this case the computational entity will be a fuzzy system, the goal will be to approximate a function with certain accuracy and a limited complexity and the parameters could be the number of membership functions, their distribution, etc.
Basic methods in these strategies are the so-called genetic algorithms [34]. In genetic algorithms, the data are represented as binary strings. The parameters are encoded on these binary strings. It is important to remark that the efficiency of these techniques is strongly affected by "code book" used to construct the strings [54]. A group of these strings is generated as initial "population". The fulfillment of the goal is tested for each element of the population (cost evaluation) and a "fitness" value is generated such that, if the value is larger, the objective is better achieved. The procedure can be outlined as follows.

- Take the initial population N and evaluate the "fitness" of the individuals (binary strings).

- Reproduce the population according to the "fitness", such that those individuals with higher values of fitness will have higher probability of being reproduced.

- Make random couples among the individuals of the reproduced population and apply the "crossover" operation. The crossover operation takes two individuals and generates a random number $l \leq L$ where L is the length of the string. This operation generates two new individuals by taking the first l elements of one string and the remaining $L-l$ element from the other string. For example, take the first string $A_1A_2A_3A_4A_5A_6$ and the second string $B_1B_2B_3B_4B_5B_6$ in this case $L = 6$ suppose $l = 2$ the crossover will be represented as:

$$\begin{array}{c} A_1A_2A_3A_4A_5A_6 \\ B_1B_2B_3B_4B_5B_6 \\ \hline A_1A_2B_3B_4B_5B_6 \\ B_1B_2A_3A_4A_5A_6 \end{array}$$

- Finally some members of the population are selected for "mutation", a random number l is generated such that $0 < l \leq L$ for each selected member and the bit l is flipped. Suppose the string $A_1A_2A_3A_4A_5A_6 = 101100$ is selected for mutation and $l = 4$ the string after mutation is: $A_1A_2A_3A_4A_5A_6 = 101000$.

- The "fitness" of the generated population is evaluated and the procedures of reproduction, crossover and mutation are repeated for a given number of times (generations).

These algorithms are very powerful for the search of "global" solutions in the search space, and there is a probability equal to one that the algorithm will find the "global solution" after a number of generations given by [10]:

$$\frac{1}{1-(1-p_M\frac{N_{opt}}{2^L})^N} \tag{2.73}$$

where p_M is the probability of mutation, N_{opt} is the number of global solutions in the final population, L is the length of the strings and N is the number of strings in the population.
The application of these algorithms to the design of fuzzy systems is mainly oriented to the generation of the number and distribution of the membership functions. One example of codification is: Assume a number of triangular, trapezoidal or polynomial membership functions with overlap $\frac{1}{2}$, has been fixed

for each input. Then the string will represent the distance from the previous point as shown in figure 2.15. The length of the string is $L = 28$, four groups of seven bits. An example of mutation is shown in figure 2.16 where $l = 10$.

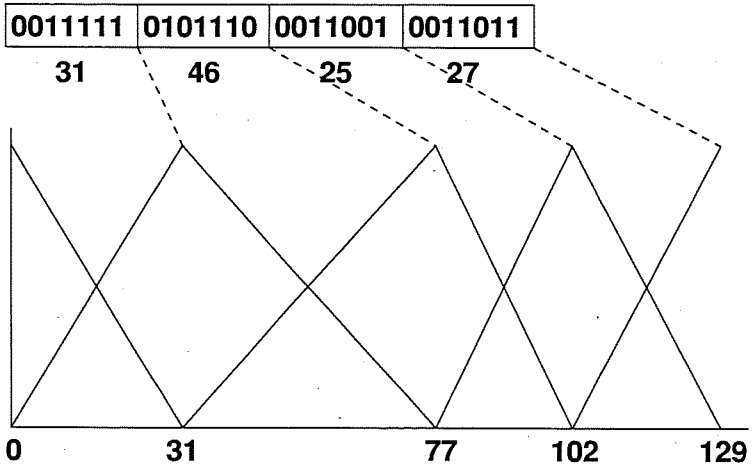


Figure 2.15: *Codification of the membership functions*

Finally an example of the effect of the crossover operation is shown in figure 2.17 where $l = 10$. Other codification methods and details can be seen in [10].

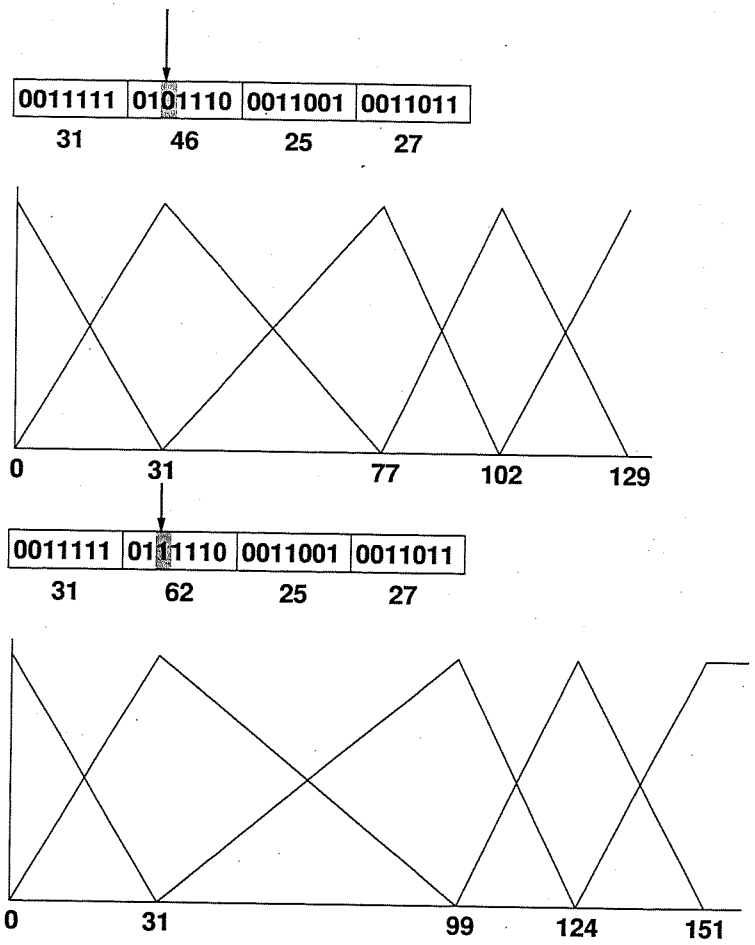


Figure 2.16: *Mutation operation in a fuzzy partitions*

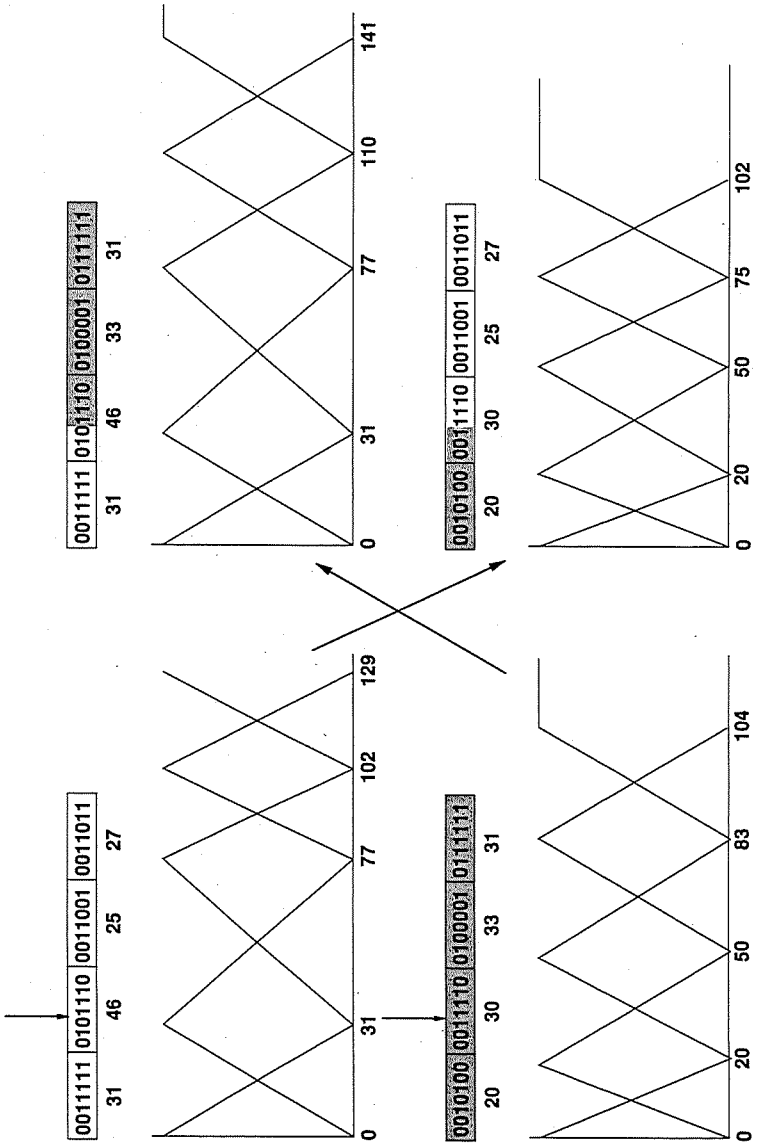


Figure 2.17: *Crossover operation between two fuzzy partitions*

2.3 Generalization and Consequences Estimation

The issue of generalization is quite related with the issue of consequence estimation in fuzzy systems, as it will be shown in the following lines. Generalization is the capacity of the designed system (the fuzzy inference system) to generate "good" output values when new inputs are presented to the system. Two issues limit the capacity of generalization:

- Lack of excitation during model construction.
- Too many degrees of freedom in the model.

These two issues are strongly related because as the degrees of freedom grow the data must excite all the new modes introduced by the new degrees of freedom. There are two ways to improve the generalization,

- Reducing the degrees of freedom with the drawback of reducing the accuracy of the approximation.
- Generating many data for all possible operating modes. For some practical cases, this is almost impossible to do.

Assuming the input-output data are given in advance the challenge is to design a system with good approximation properties and good generalization. The application of the methods reviewed in previous sections postulates the generation of the consequences of the rules by means of Least Squares. As mentioned in section 2.2.1 the calculation of the Least Squares using the equation (2.36) will be possible only if $\text{rank}(W^TW) = \dim\theta$. In cases where $\text{rank}(W^TW) \leq \dim\theta$, the estimation will be very poor and the consequences of those unexcited rules will be very far from their real value. A reasonable solution is to initialize the rules using information given by a simpler model (with very few degrees of freedom) and to improve the estimation of the consequences of those rules that have been excited using Recursive Least Squares. The advantage of the Recursive Least Squares algorithm is that it only updates those terms that have been excited. The procedure can be detailed in two steps:

2.3.1 Consequence Initialization

The initialization of the consequences can be done in two ways using the information given by a simple model with sufficient excitation or using expert knowledge. The use of expert knowledge demands the designer to initialize those rules with empirical knowledge. The initialization using a simple model with sufficient excitation operates as follows:

- The smallest fuzzy model $\hat{f}(x^t)$ is constructed by placing only two membership functions (triangular or polynomial) on each input with their modal values placed respectively in the maximum and minimum values of the universe of discourse and fixing the overlap value in $\frac{1}{2}$. This distribution of the membership functions will generate a fuzzy system with 2^N rules where N is the number of inputs. This fuzzy system has the property that any input presented will excite the whole set of rules. This property guarantees enough excitation such that the 2^N consequences can be estimated using the Least Squares solution given in the equation (2.36).
- If the constructed model uses triangular or polynomial membership functions with overlap $\frac{1}{2}$, the consequences of the rules can be initialized using the model $\hat{f}(x^t)$ as follows:

$$\bar{y}^{j_1 j_2 \cdots j_N} = \hat{f}(M^{j_1 j_2 \cdots j_N}) \tag{2.74}$$

with

$$M^{j_1 j_2 \cdots j_N} = \{m^1_{j_1}, m^2_{j_2}, \ldots, m^N_{j_N}\}^T$$

where $m^i_{j_i}$ are the modal values of the membership functions of the fuzzy model $f(x^t)$. If the model is not constructed as described above, the initial consequences can be estimated by using a data set generated from the model $\hat{f}(x^t)$, so that, the condition of sufficient excitation is guaranteed. This can be done just by generating input data regularly distributed and with "enough" density over the input space U.

This initialization method guarantees that the constructed fuzzy model will be at least as good as the best multilinear model, if the smaller model is constructed with triangular membership functions, or at least as good as the best third order multi polynomial model. These bounds guarantee the quality of the generalization even if the training data has no information about some of the regions described in the rule base.

2.3.2 Consequence Estimation

Once the consequences have been initialized the recursive least squares algorithm can be applied to improve the estimation. The algorithm is described as follows using the notation presented in section 2.2.1 :

$$\theta(k+1) = \theta(k) + \gamma(k)[y^t - W^t\theta(k)] \tag{2.75}$$

with $W^t = \{w^t_1, w^t_2, \ldots, w^t_t\}$, $\theta(k) = \{\bar{y}^1(k), \bar{y}^2(k), \ldots, \bar{y}^L(k)\}$ and:

$$\gamma(k) \quad = \quad P(k+1)W_{k+1} \tag{2.76}$$

$$= \frac{1}{W^t P(k) W^{t^T} + 1} P(k) W^t \quad (2.77)$$

$$P(k+1) = [I - \gamma(k) W^t] P(k) \quad (2.78)$$

with the initial value $P(0) = \alpha I$, where α is large scalar value. The procedure is repeated and each time the index k is incremented, also the index t is incremented until it reaches the value N, then the value of t is reset to $t = 1$. The initial values of $\theta(0)$ are the initialization values given by the procedure described before. To illustrate the way the present method to improve the generalization is applied the following example is presented.

2.3.3 Example

The objective is to approximate the function of two variables $f(x, y) = 6x + 4y + \cos(\pi x) + \cos(\pi y) + 50$ on the interval $(x, y) \in U$ $U = [-2, 2] \times [-2, 2]$. The function is plotted in figure 2.18.

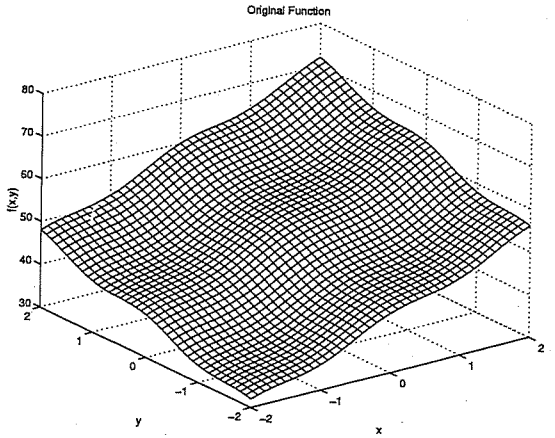


Figure 2.18: *Function to be approximated* $f(x, y) = 6x + 4y + \cos(\pi x) + \cos(\pi y) + 50$

The function is sampled at 153 points where obtained the sampling was done such that only one point falls in the interval $V = [-2, 0] \times [-2, 0]$, the data points are depicted in the figure 2.19.

The function will be approximated by a fuzzy system using five (5) triangular

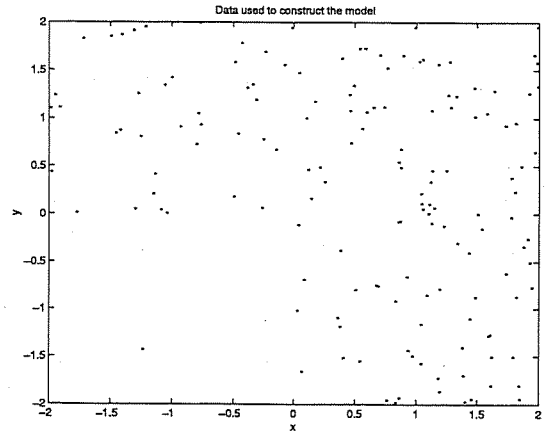
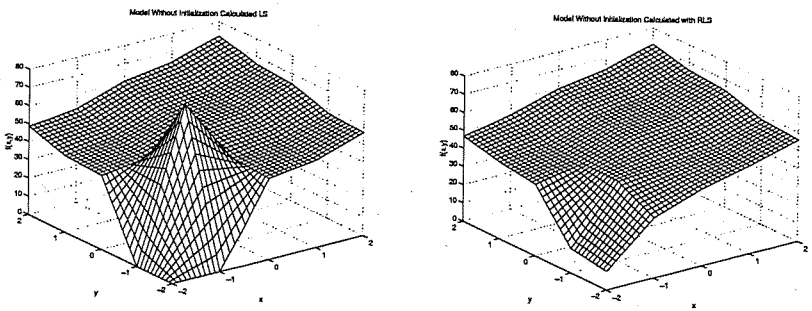


Figure 2.19: *Sampled Points to Approximate the function* $f(x, y) = 6x + 4y + \cos(\pi x) + \cos(\pi y) + 50$

membership functions equally distributed on each domain with overlap 0.5. A total of 25 rules is generated and the consequences will be estimated using three methods Least Squares (LS), Recursive Least Squares (RLS) and RLS with the consequence initialization method explained in section 2.3.1. Observe in figure 2.20 that the LS method and the "pure" RLS fail to approximate the function in the domain V and even the LS solution fails to make a good approximation in the region where the "training" points were selected.

The third method as was explained on section 2.3.1 first calculates the smallest fuzzy model $\hat{f}(x, y)$ with only two (2) membership functions with overlap 0.5 covering the whole domain on each input. The model has four rules that are excited by all the points such that the consequence estimation does not represent any numerical problem. The approximation generated by this model is shown in figure 2.21. Then the formula (2.74) is used to initialize the consequences of the rules of the model with 25 rules such that it generates an approximation perfectly equivalent to the approximation given by the model $\hat{f}(x, y)$.

Then the consequences are estimated using RLS. The results are shown in figure 2.22. Observe that the approximation is good in the whole domain U and the are no big changes in the region V where almost no data exists during the training. A final comparison was performed by generating 141 points in the domain U but



(a) (b)

Figure 2.20: *(a) Approximation obtained with the consequences calculated using LS (b)Approximation obtained with the consequences calculated using RLS*

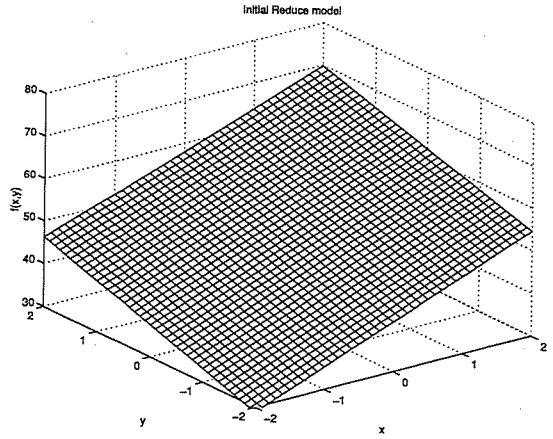


Figure 2.21: *Approximation generated by the "smallest" fuzzy model $\hat{f}(x,y)$ with only 4 rules*

excluding the region V (the same conditions used for the training) to observe the approximation error in the "well excited" region. The results are presented in the table 2.1, and the error index is calculated as $E = \frac{1}{141}\sum_{i=1}^{141} e^2$, where E is the difference between the real and the estimated value. From these results, it

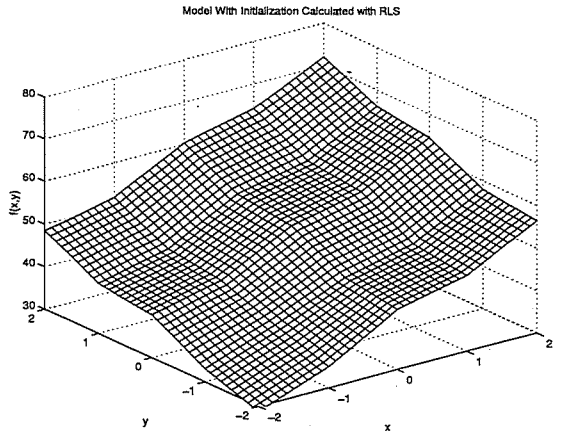


Figure 2.22: *Approximation generated by the fuzzy model initialized with $\hat{f}(x,y)$ and with the consequences calculated using RLS*

is clear that the LS method is the worse. The reason is that the inputs selected for the estimation did not excite some of the rules and in this specific case the $\text{rank}(W^TW) = 22$, therefore the LS solution is badly conditioned. The RLS solution is better because it updates only the excited rules but the badly excited rules are not updated making a bad generalization on "poorly" excited regions. Finally, the best performance is by far the one of the proposed method. The reason is that this method assumes the generalization given by a "well" excited model ($\hat{f}(x,y)$) and the tuning will only improve the approximation on these regions where there is enough excitation. $\Diamond$

Table 2.1: Example: Comparison between Methods for Consequences Calculation

Method	Approx. Error
Least Squares	0.3455
Recursive Least Squares	0.1659
RLS with initialization using $\hat{f}(x,y)$	0.0146

2.4 Fuzzy Modeling with Linguistic Integrity

This section presents one of the main contributions of this thesis [30][24]. The analysis and algorithms presented on this section are the main results obtained by the author in the area of fuzzy modeling.

2.4.1 Introduction

The use of models is an essential element of human behavior. When a human being predicts the impact of his actions is using a model. Causality is a paramount assumption that makes models useful. Causality is reflected in language as IF-THEN rules (IF **cause-happens** THEN **a consequence is foreseen**). A set of IF-THEN rules is a linguistic representation of a mental model created inside the brain. New instrumentation and data acquisition system have expanded the capacity of human beings beyond the five senses. This expanded sensorial capacity has been accompanied with an increase in the storage capacity but the capacity of the human brain to interact with this information remains limited. This situation motivates the development of computer techniques that can extract the "knowledge" and represent it in a linguistic way using IF-THEN rules. There is a trade-off between numerical accuracy and linguistic interpretability. This trade off is a consequence of a limitation of the human brain to represent a limited number of categories on a given domain. A consequence of this limitation is reflected in language. The number of linguistic labels that a human being can generate to represent categories on a given domain is limited to as much as nine and it will be typically seven.
On the other hand, the numerical accuracy is important in the implementation of policies and control actions based on the information given by model. This issue of accuracy is especially relevant when the models are used in a dynamic way where the predicted value is fed back.
In the previous section, some methods for model construction have been explained. Initially the algorithms based on gradient descent techniques, also known as neurofuzzy models [80] [40], have been oriented to minimize the numerical error. The use of this technique generates in some cases fuzzy sets with "too much" or "none" overlap making the interpretation of the model a difficult task.
To overcome the drawback of the initial selection of the fuzzy sets several methods have been proposed, some of them based on local error approximation [42] [74] and others based on clustering techniques [71] [84] [50]. These methods generate multidimensional fuzzy sets and project them into the input spaces. The projections also exhibit unsatisfactory overlap making the interpretation and the labeling of the fuzzy sets a difficult task.
This section presents the AFRELI algorithm (**A**utonomous **F**uzzy **R**ule **E**xtractor with **L**inguistic **I**ntegrity); the algorithm is able to fit input-output data while maintaining the semantic integrity of the rule base. Jang et. al mentioned the issue of interpretability at the end of chapter 12 in [41]. There they only mention some basic ideas about how to constraint the optimization of the ANFIS scheme to preserve the interpretability. In [77] Valente presents a formulation of some constraints to guarantee semantic integrity. For his formulation he assumes a number of sets on each input domain and then constrains the optimization of the membership functions to guarantee the integrity of the resulting fuzzy sets. The trade-off between precision and transparency is mentioned in [68] together with some examples where the consequences of the rules use two terms; one numeric and one linguistic. The AFRELI algorithm presented in this section uses clustering and projection techniques to find a good initial position for the fuzzy set in the input domains. A FuZion algorithm is introduced to reduce the complexity of the projected fuzzy sets. A rule base is constructed using the reduced representation of the fuzzy sets and the consequences are initialized and calculated with a method that improves generalization and avoids the lack of excitation in some rules. Finally, the consequences of the rules are represented by two fuzzy sets with different strength. The number of terms in the consequences of the fuzzy rules is reduced again using the FuZion algorithm.

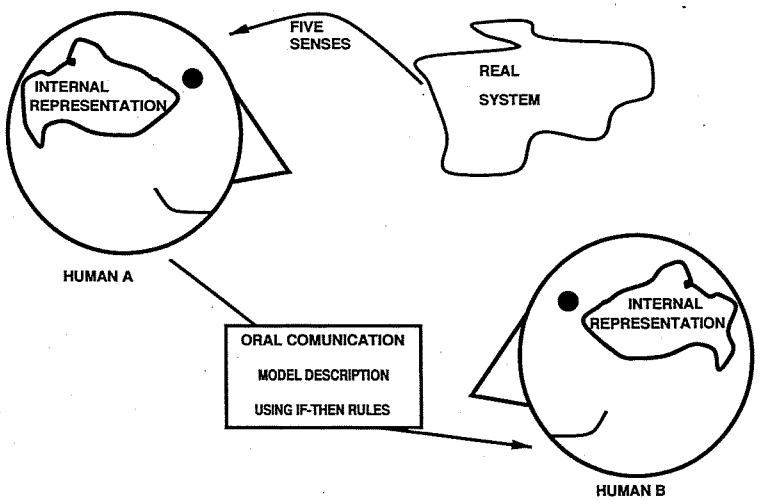


Figure 2.23: *Traditional Knowledge Acquisition*

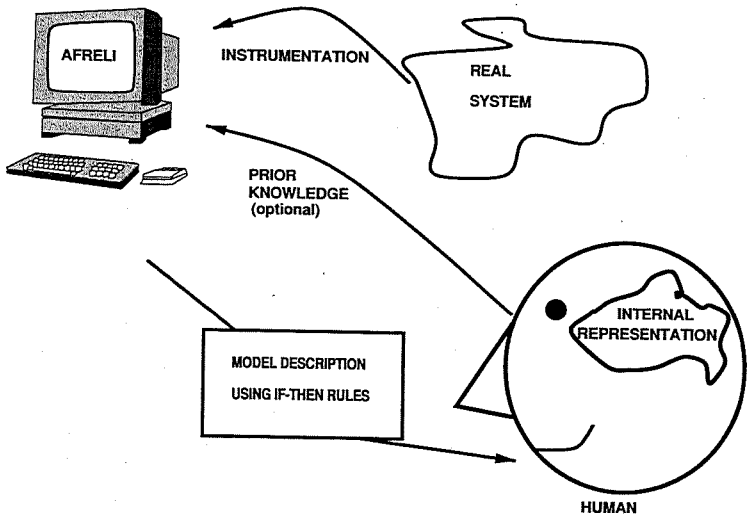


Figure 2.24: *Knowledge Acquisition using AFRELI*

2.4.2 Structure of the fuzzy model

The high number of degrees of freedom in a fuzzy inference system (shape and number of membership functions, T-norms, aggregation methods, etc) gives high flexibility to the fuzzy system but also demands systematic criteria to select these parameters. Some parameters can be fixed taking into account the following issues: optimal interface design [61] and semantic integrity [62].

- Optimal interface design
  - *Error-free Reconstruction*: In a fuzzy system a numerical value is converted into a linguistic value by means of fuzzification. A defuzzification method should guaranteed that this linguistic value can be reconstructed in the same numerical value

$$\forall x \in [a,b]: \qquad \mathcal{L}^{-1}[\mathcal{L}(x)] = x \tag{2.79}$$

where $[a,b]$ is the universe of discourse, $\mathcal{L}$ is the fuzzification process and $\mathcal{L}^{-1}$ is the defuzzification process. The use of triangular membership functions with overlap $\frac{1}{2}$ and centroid defuzzification will satisfy

this requirement (see proof: [61]). Polynomial membership functions with overlap $\frac{1}{2}$ also are used but a new defuzzification process must be designed to guarantee an error-free reconstruction.

- Semantic integrity: This property guarantees that the membership functions represent a linguistic concept. The conditions for semantic integrity are:
  - *Distinguishability:* Each linguistic label should have semantic meaning and the fuzzy set should clearly define a range in the universe of discourse. Therefore, the membership functions should be clearly different. The assumption of the overlap equal to $\frac{1}{2}$ makes sure that the support of each fuzzy set will be different. A minimum distance between the modal values of the membership functions makes sure that the membership functions can be distinguished. The defined modal value of a membership function is defined as the α-cut with $\alpha = 1$

$$m_i = \mu_{i(\alpha=1)}(x), \quad i = 1, \ldots, N \tag{2.80}$$

  - *Justifiable Number of Elements:* The number of sets on each domain should be compatible with the number of "quantifiers" that a human being can handle. This number should not exceed the limit of 7 ± 2 distinct terms [13]. The choice of the shape of the membership functions does not guarantee this property. To assure that this requirement is fulfilled the FuZion algorithm is presented further in this paper. This algorithm reduces the number of sets on each input or output domain.
  - *Coverage:* Any element from the universe of discourse should belong to at least one of the fuzzy sets. This concept is also mentioned in the literature as ϵ completeness [40].
  - *Normalization:* Due to the fact that each linguistic label has semantic meaning, at least one of the values in the universe of discourse should have a membership degree equal to one. In other words, all the fuzzy sets should be normal.

Based on these criteria the selected *membership functions* will be triangular and normal $(\mu_1(x), \mu_2(x), \ldots, \mu_n(x))$ with a specific overlap of $\frac{1}{2}$. It means that the height of the intersection of two successive fuzzy sets is

$$\mathrm{hgt}(\mu_i \cap \mu_{i\pm1}) = \frac{1}{2}. \tag{2.81}$$

The choice of the AND and the OR operation will be conditioned by the need of constructing a continuous and differentiable nonlinear map. This property is

important if optimization of the antecedent terms is needed in this case AND and OR operations using *product* and *probabilistic sum* will be preferred because their derivatives are continuous.

2.4.3 The AFRELI algorithm

The AFRELI (Automatic Fuzzy Rule Extractor with Linguistic Integrity) [30] [24] is an algorithm designed to obtain a good compromise between numerical approximation and linguistic meaning. This particular trade-off has been referenced for long time in science (for a compilation of remarks see: [1]). The main steps of this algorithm are:

- Clustering
- Projection
- Reduction of terms in the antecedents(FuZion see section 2.4.4)
- Consequence Calculation
- (optional step) Further antecedent optimization
- Reduction of terms in the consequences and rule modification (FuZion see section 2.4.4)

The AFRELI algorithm proceeds as follows:

1. Collect N points from the inputs ($X = \{x^1, \ldots, x^N\}$) and the output ($Y = \{y^1, \ldots, y^N\}$)

$$x^k = \begin{bmatrix} x_1^k \\ \vdots \\ x_p^k \end{bmatrix} \tag{2.82}$$

where $x^k \in \Re^p$ and $y^k \in \Re$ represent the inputs and the output on instant k and construct the *feature vectors*

$$U^k = \begin{bmatrix} x_1^k \\ \vdots \\ x_p^k \\ y^k \end{bmatrix} \tag{2.83}$$

$u^k \in \Re^{p+1}$.

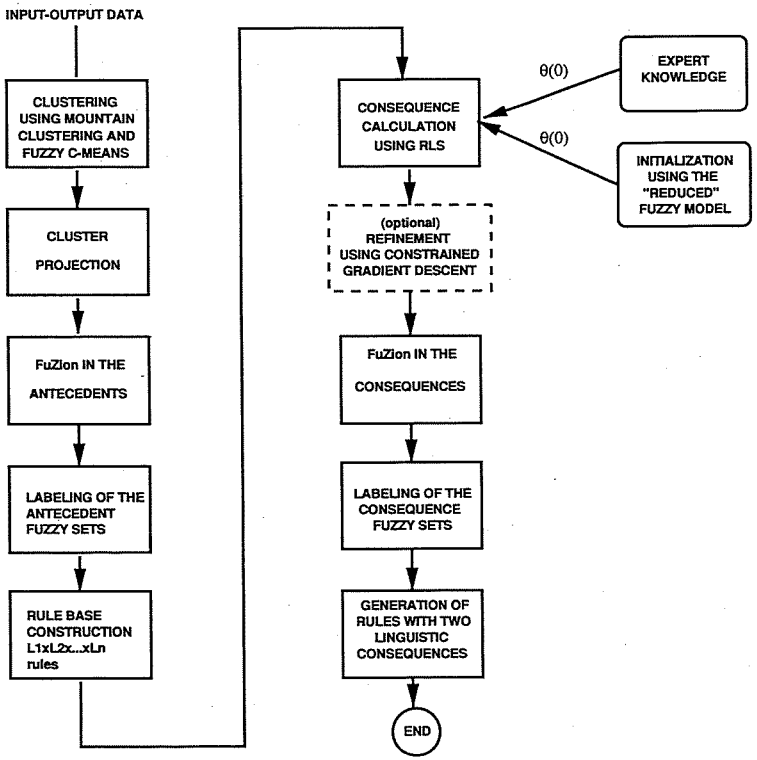


Figure 2.25: *Flow Diagram of AFRELI algorithm*

2. Using the N feature vectors find C clusters. Apply the mountain clustering method [84] [50] to initialize the centers and to obtain the number of clusters (C). Refine the clusters using the fuzzy C-means algorithm [8].

$$Uc = \begin{pmatrix} \tilde{x}_1^1 & \tilde{x}_1^2 & \ldots & \tilde{x}_1^C \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{x}_p^1 & \tilde{x}_p^2 & \ldots & \tilde{x}_p^C \\ \tilde{y}^1 & \tilde{y}^2 & \ldots & \tilde{y}^C \end{pmatrix} \tag{2.84}$$

with $Uc \in \Re^{p+1 \times C}$ and $\tilde{x}_i^j$ represents the i-th coordinate of the j-th cluster.

It is very important to remark that the use of *mountain clustering* will be limited to low dimensional problems (up to four or five dimensions). Its inherent advantage is that it can guide good initial points and number of clusters. For high dimensional problems the alternative is to use only fuzzy c-means and overestimated the number of clusters; the subsequent steps (FuZion) will reduce the number of terms.

3. Project the C prototypes of the clusters into the input spaces, by converting the projected value of each prototype into the modal value of a triangular membership function.

$$m_j^i = \tilde{x}_i^j \tag{2.85}$$

where $i = 1, \ldots, p$, $j = 1, \ldots, C$

4. Sort the modal values on each domain such that:

$$m_j^i \leq m_{j+1}^i \qquad \forall i = 1, \ldots, p \tag{2.86}$$

5. Add two more modal values to each input to guarantee full coverage of the input space.

$$m_0^i = \min_{k=1,\ldots,N} x_i^k \tag{2.87}$$

$$m_{C+1}^i = \max_{k=1,\ldots,N} x_i^k \tag{2.88}$$

6. Construct the triangular membership functions with overlap of $\frac{1}{2}$ as:

$$\mu_j^i(x_i^k) = \max\left[0, \min\left(\frac{x_i^k - m_{j-1}^i}{m_j^i - m_{j-1}^i}, \frac{x_i^k - m_{j+1}^i}{m_j^i - m_{j+1}^i}\right)\right] \tag{2.89}$$

where: $j = 1, \ldots, C$, and the trapezoidal membership functions at the extremes of each universe of discourse

$$\mu_0^i(x_i^k) = \max\left[0, \min\left(\frac{x_i^k - m_1^j}{m_0^j - m_1^j}, 1\right)\right] \tag{2.90}$$

$$\mu_{C+1}^j(x_i^k) = \max\left[0, \min\left(\frac{x_i^k - m_C^j}{m_{C+1}^j - m_C^j}, 1\right)\right] \tag{2.91}$$

7. Apply FuZion algorithm (see section 2.4.4) to reduce the number of membership functions on each input domain. This algorithm does a somewhat one-dimensional clustering among the modal values of the fuzzy sets.

8. Associate linguistic labels (p.e.BIG, MEDIUM, SMALL, etc.) to the resulting membership functions. This association will depend on the type of variable and the criteria of the designer. In fact the association of a fuzzy set with a label will be the result of the agreement between the fuzzy set proposed by the algorithm and the "sense" that this set creates in the mind of the user.

9. Construct the rule base with all possible antecedents (all possible permutations) using rules of the form:

$$\ldots$$
$$\text{IF } x_1^k \text{ is } \mu_l^1 \text{ AND } x_2^k \text{ is } \mu_l^2 \text{ AND } \ldots \text{AND } x_p^k \text{ is } \mu_l^p \text{ THEN } \hat{y}_k = \bar{y}_l$$
$$\ldots$$

Equivalently, the evaluation of the antecedents of each rule can be expressed in terms of the operators *min* and *product* . Using *min* operator:

$$\mu_l(x^k) = \min\{\mu_l^1(x_1^k), \mu_l^2(x_2^k), \ldots, \mu_l^n(x_p^k)\} \tag{2.92}$$

Using *product* operator:

$$\mu_l(x^k) = \mu_l^1(x_1^k) \cdot \mu_l^2(x_2^k) \cdot \ldots \cdot \mu_l^n(x_p^k) \tag{2.93}$$

Observe that if the number of fuzzy sets on the input i is L_i and there are n inputs the number of rules will be:

$$L_1 \times L_2 \times \ldots \times L_n$$

This structure guarantees a complete description of the system in the space interval U, because every possible condition will be represented in the rule base. Observe that the number of rules will grow very fast as the number of input increases. This fact is a limitation in the sense that the comprehension of a set of rules with a large number of antecedents is difficult. In addition, the storage problem generated by a large number of terms to be kept in the computer's memory. On the other hand, it does not represent a limitation in terms of execution time because the use of the described type of triangular membership functions will guarantee that at most 2^p rules will be evaluated during the inference process.

10. Propagate the N values of the inputs and calculate the consequences of the rules as singletons ($\bar{y}_l$). These singletons should be calculated with the

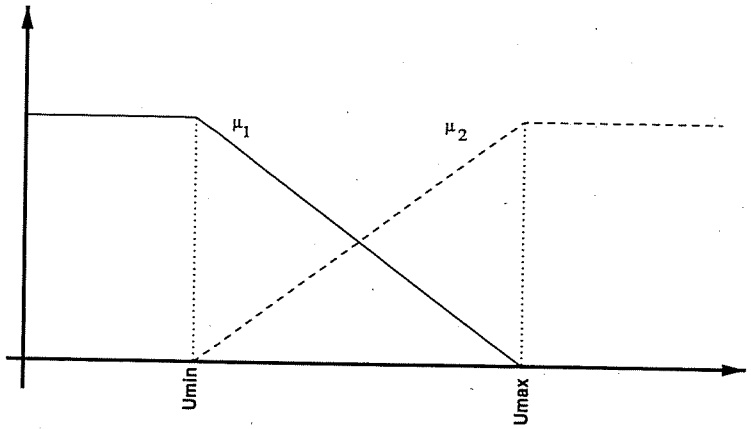


Figure 2.26: *Input membership functions for the smallest fuzzy model*

method explained in the section 2.3.It is important to remember that this method guarantees that the "full" model will be at least as good as the "reduced" model. Because the "reduced" model is the best multi linear model that can be build with the given data, there is a guarantee that the "full" fuzzy model will be at least as good as the best multi linear model.

11. **(Optional Step)** If further refinement is required to improve the approximation, constrained gradient descent methods using the expressions for the gradient calculated in section 2.2.2 can be applied to improve the location of the modal values of the antecedent membership functions. The main constraint applied in the optimization phase is the "distinguishability" that is represented as the minimum acceptable distance between consecutive modal values. The use of gradient descent methods will move the system parameters towards a "local minimum" close to the initial values. Because the improvement obtained by this step will not be very significant this step is considered *optional* and only will be recommended when the numeric performance of the model does not satisfy the requirements of the user.

12. Because the singletons in the consequences are crisp sets, the linguistic meaning of the rules will be lost. The next step is to convert the singletons of the consequences to triangular membership functions with overlap $\frac{1}{2}$ and

modal values equal to the position of the singleton $\bar{y}_l$. Consider the vector $\tilde{Y}$ whose entries are the L consequences of the rules but sorted in such a way that:

$$\tilde{y}_1 \leq \tilde{y}_2 \leq \ldots \leq \tilde{y}_L \tag{2.94}$$

The triangular membership function of the i-th consequence is:

$$\mu_i^{\tilde{y}}(y) = \max\left[0, \min\left(\frac{y-\tilde{y}_{i-1}}{\tilde{y}_i-\tilde{y}_{i-1}}, \frac{y-\tilde{y}_{i+1}}{\tilde{y}_i-\tilde{y}_{i+1}}\right)\right] \tag{2.95}$$

and the two membership functions of the extremes:

$$\mu_1^{\tilde{y}}(y) = \max\left[0, \min\left(\frac{y-2\tilde{y}_1+\tilde{y}_2}{-\tilde{y}_1+\tilde{y}_2}, \frac{y-\tilde{y}_2}{\tilde{y}_1-\tilde{y}_2}\right)\right] \tag{2.96}$$

$$\mu_L^{\tilde{y}}(y) = \max\left[0, \min\left(\frac{y-\tilde{y}_{L-1}}{\tilde{y}_L-\tilde{y}_{L-1}}, \frac{y-2\tilde{y}_L+\tilde{y}_{L-1}}{-\tilde{y}_L+\tilde{y}_{L-1}}\right)\right] \tag{2.97}$$

This description of the outer membership functions guarantees that their centers of gravity will be exactly on their modal values. This guarantees that the condition of error-free reconstruction for optimal interface will be achieved.

13. Apply FuZion algorithm (see section 2.4.4) to reduce the number of membership functions in the output universe. The FuZion process reduces groups of neighboring singletons to triangular membership functions whose modal values are representative for a group of singletons. It is optimal in a sense that the modal value of the "FuZioned" membership function is placed at the mean value of the neighboring singletons.

14. Associate linguistic labels to the resulting membership functions.

15. With the partition of the output universe, fuzzify the values of the singletons. Observe that each singleton will have a membership degree in at least one set and in as much as two.

16. Relate the fuzzified values with the corresponding rule. It means that each rule will have one consequence or two weighted consequences where the weights are the non-zero membership values of the fuzzified singleton. This description of the consequences of the rules using two linguistic fuzzy sets and two strength values improves the interpretability of the consequences compared when only one singleton describes the consequence. The advantage of this description is that interpretability is gained without a cost in numerical precision. This strategy was independently proposed previously in [57] and [25].

2.4.4 The FuZion algorithm

The FuZion algorithm is a routine that merges consecutive triangular membership functions when their values are "too close" to each other. This merging process is needed to preserve the *distinguishability* and the *justifiable number of elements* on each domain guaranteeing the semantic integrity. A fundamental parameter of this algorithm is the minimum acceptable distance between modal values and it is given by M. The FuZion algorithm goes as follows:

1. Take the triangular membership functions $\mu_1(x), \mu_2(x), \ldots, \mu_N(x)$ with $\frac{1}{2}$ overlap, and the modal values

$$m_i = \mu_{i(\alpha=1)}(x), \quad i = 1, \ldots, N \tag{2.98}$$

with:

$$m_1 \leq m_2 \leq \ldots \leq m_N \tag{2.99}$$

2. Define the minimum distance acceptable M between the modal values.
3. Calculate the difference between successive modal values as:

$$d_j = m_{j+1} - m_j, \quad j = 1, \ldots, N-1 \tag{2.100}$$

4. While $\exists d_j < M$ do
5. Find all the differences smaller than M.
6. Merge all the modal values corresponding to consecutive differences smaller than M using (2.101).

$$m_{new} = \frac{\sum_{i=a}^{b} m_i}{D} \tag{2.101}$$

$$D = b - a + 1 \tag{2.102}$$

where a and b are respectively the index of the first and the last modal value of the fusioned sequence and D is the number of merged membership functions.

7. Update N
8. Calculate the difference between the new successive modal values as:

$$d_j = m_{j+1} - m_j, \quad j = 1, \ldots, N-1 \tag{2.103}$$

9. end while
10. end

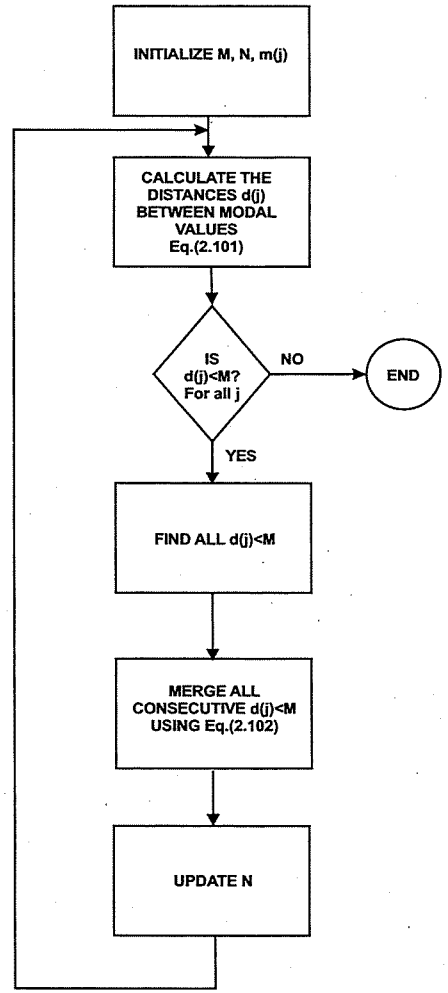


Figure 2.27: *Flow Diagram of FuZion algorithm*

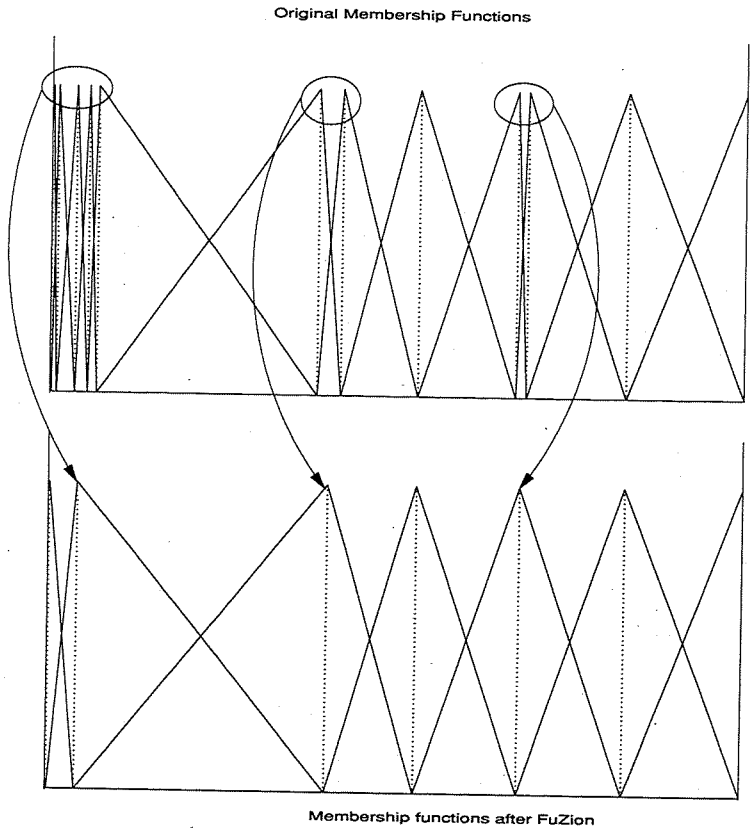


Figure 2.28: *Effect of the FuZion algorithm*

2.4.5 Examples

The present section shows three examples of applications of the AFRELI and FuZion algorithm, the first two examples are approximations of nonlinear static maps and the last is the prediction of a chaotic time series.

2.4.6 Example 1: Modeling a two input nonlinear function

In this example, we consider the function:

$$f(x,y) = \sin\left(\frac{\pi x}{10}\right)\sin\left(\frac{\pi y}{10}\right) \tag{2.104}$$

The steps applied to the current example will be numbered using the same numbering as the one used in the FuZion algorithm in the section 2.4.4.

- **Step 1** 441 points regularly distributed were selected from the interval $[-10,10]\times[-10,10]$. The graph of the function is shown in figure 2.29

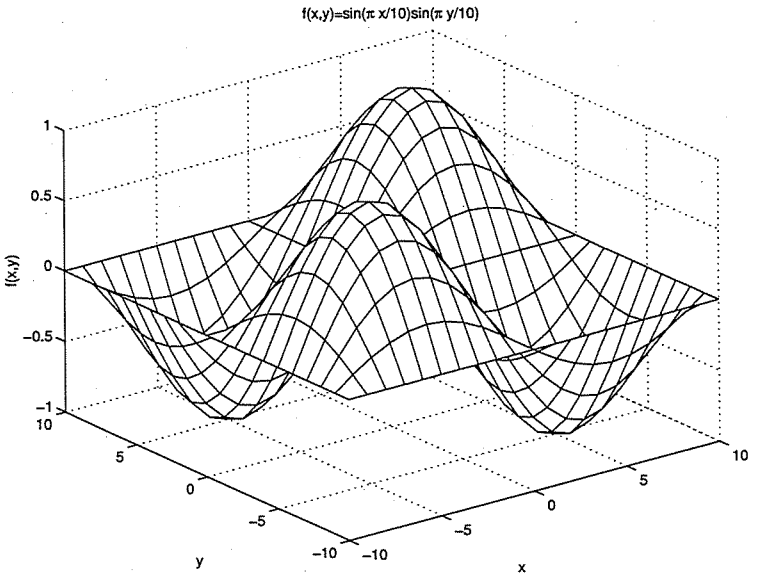


Figure 2.29: *Example 1:Function* $f(x,y) = \sin\left(\frac{\pi x}{10}\right)\sin\left(\frac{\pi y}{10}\right)$

- **Step 2** Using mountain clustering and fuzzy C-means algorithm 26 clusters were found and are shown in the figure 2.30 represented with 'x'.
- **Step 3** After the clusters were found their center values were projected into the input domains as is shown in figure 2.31.

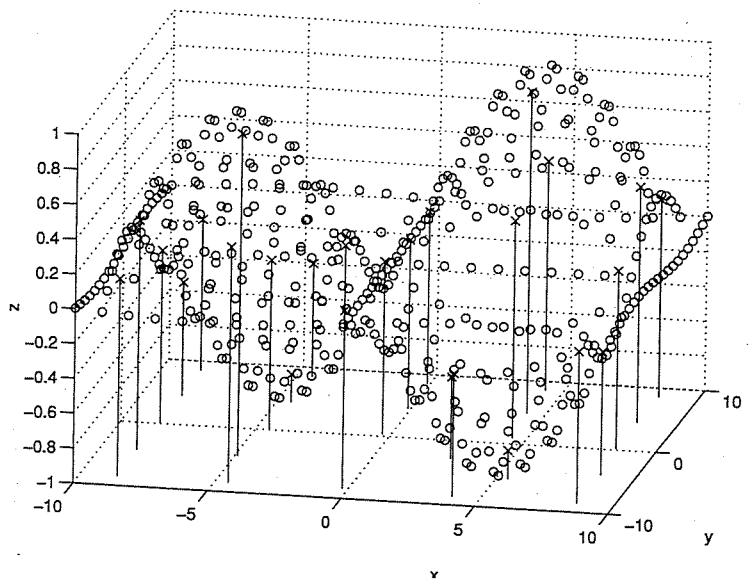


Figure 2.30: *Example 1:Extracted clusters ('x') from the data (o)*

- **Steps 4,5,6** The modal values were sorted and two more modal values were added to each input domain on -10 and 10. The triangular membership functions were constructed. Figure 2.32 shows the projected membership functions.

- **Step 7,8** The FuZion algorithm was applied with M equal to the 10 % of the universe of discourse on each domain; observe that with this value of M five (5) membership functions were generated as it is shown in figure 2.33. In figure 2.34 can be observed the obtained membership functions (seven (7)) when the M parameter in the FuZion algorithm is chosen equal to 7 % of the universe of discourse. It is clear that the smaller the value of M the larger the number of membership functions. Linguistic labels were associated to the membership functions as it is shown in figure 2.33.

- **Step 9** A rule base was generated by combining all the membership functions present on each domain $5 \times 5 = 25$ rules.

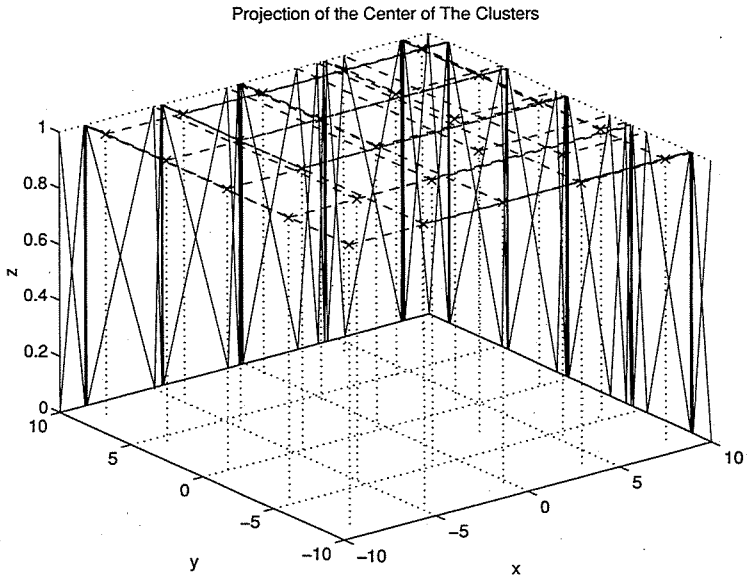


Figure 2.31: *Example 1:Projection of the centers of the clusters*

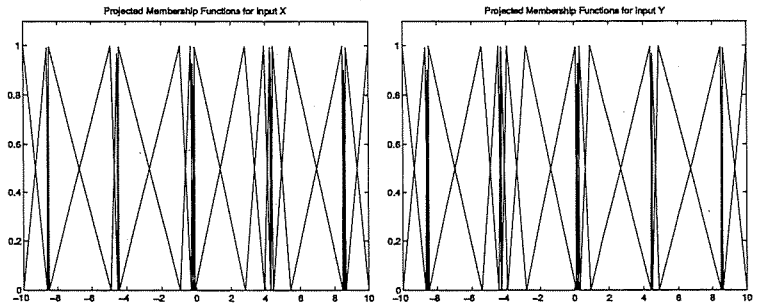


Figure 2.32: *Example 1:Projected membership functions*

- **Step 10** The 25 singletons of the consequences were calculated using RLS. The output membership functions are shown in figure 2.35(a).

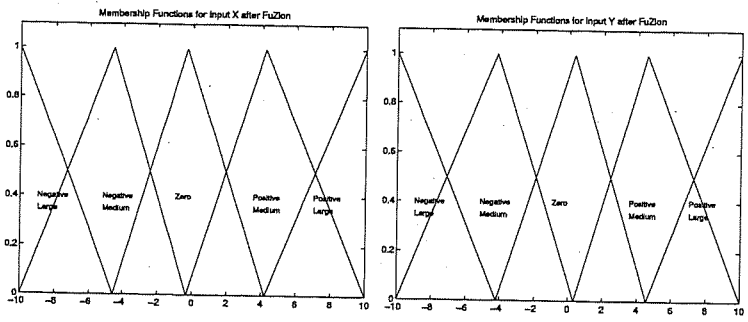


Figure 2.33: *Example 1:Membership functions after FuZion with M =10%*

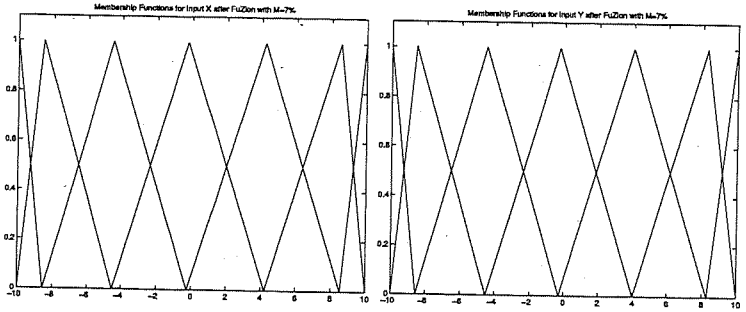


Figure 2.34: *Example 1:Membership functions after FuZion with M =7%*

- **Step 11** The optional step was not applied because the approximation was considered acceptable.
- **Step 12** The singletons were converted into 25 triangular membership functions.
- **Steps 13,14** The FuZion algorithm was applied to the 25 output triangular membership functions with M equal to the 10 % of the universe of discourse. Three membership functions were obtained and they received their linguistic values as it is shown in figure 2.35(b).
- **Steps 15,16** The 25 singletons were fuzzified using the three membership functions obtained in the previous step. All the non-zero membership

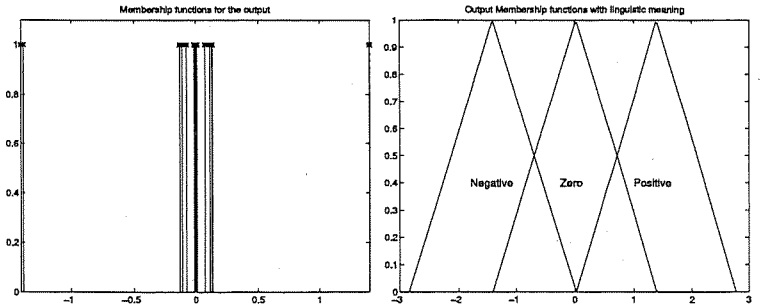


Figure 2.35: *Example 1:(a) Singletons (b) Membership functions with linguistic meaning*

values were associated to the consequences of the rules as it is shown in the following list.

1. **IF** x is Negative Large AND y is Negative Large **THEN** z is Negative with strength 0.01 AND Zero with strength 0.99
2. **IF** x is Negative Medium AND y is Negative Large **THEN** z is Zero with strength 0.92 AND Positive with strength 0.08
3. **IF** x is Zero AND y is Negative Large **THEN** z is Negative with strength 0.01 AND Zero with strength 0.99
4. **IF** x is Positive Medium AND y is Negative Large **THEN** z is Negative with strength 0.1 AND Zero with strength 0.9
5. **IF** x is Positive Large AND y is Negative Large **THEN** z is Negative with strength 0.03 AND Zero with strength 0.97
6. **IF** x is Negative Large AND y is Negative Medium **THEN** z is Zero with strength 0.96 AND Positive with strength 0.04
7. **IF** x is Negative Medium AND y is Negative Medium **THEN** z is Zero with strength 0.01 AND Positive with strength 0.99
8. **IF** x is Zero AND y is Negative Medium **THEN** z is Zero with strength 0.92 AND Positive with strength 0.08
9. **IF** x is Positive Medium AND y is Negative Medium **THEN** z is Negative with strength 0.99 AND Zero with strength 0.01
10. **IF** x is Positive Large AND y is Negative Medium **THEN** z is Negative with strength 0.1 AND Zero with strength 0.90
11. **IF** x is Negative Large AND y is Zero **THEN** z is Negative with strength 0.02 AND Zero with strength 0.98

12. **IF** x is Negative Medium AND y is Zero **THEN** z is Negative with strength 0.11 AND Zero with strength 0.89
13. **IF** x is Zero AND y is Zero **THEN** z is Negative with strength 0.03 AND Zero with strength 0.97
14. **IF** x is Positive Medium AND y is Zero **THEN** z is Zero with strength 0.92 AND Positive with strength 0.08
15. **IF** x is Positive Large AND y is Zero **THEN** z is Negative with strength 0.01 AND Zero with strength 0.99
16. **IF** x is Negative Large AND y is Positive Medium **THEN** z is Negative with strength 0.07 AND Zero with strength 0.93
17. **IF** x is Negative Medium AND y is Positive Medium **THEN** z is Negative with strength 1
18. **IF** x is Zero AND y is Positive Medium **THEN** z is Negative with strength 0.11 AND Zero with strength 0.89
19. **IF** x is Positive Medium AND y is Positive Medium **THEN** z is Positive with strength 1
20. **IF** x is Positive Large AND y is Positive Medium **THEN** z is Zero with strength 0.93 AND Positive 0.07
21. **IF** x is Negative Large AND y is Positive Large **THEN** z is Negative with strength 0.02 AND Zero with strength 0.98
22. **IF** x is Negative Medium AND y is Positive Large **THEN** z is Negative with strength 0.07 AND Zero with strength 0.93
23. **IF** x is Zero AND y is Positive Large **THEN** z is Negative with strength 0.02 AND Zero with strength 0.98
24. **IF** x is Positive Medium AND y is Positive Large **THEN** z is Zero with strength 0.96 AND Positive with strength 0.04
25. **IF** x is Positive Large AND y is Positive Large **THEN** z is Negative with strength 0.01 AND Zero with strength 0.99

Observe that the obtained rules exhibit a clear dominance of one of the consequences. When this happens it will be possible to eliminate the consequence with the small strength without a major impact on the numerical approximation. However, this step is a decision that must be left to the designer because it is case dependent. Figure 2.36 shows the identified surface. Observe that the main features of the function were captured by the fuzzy system. ◊

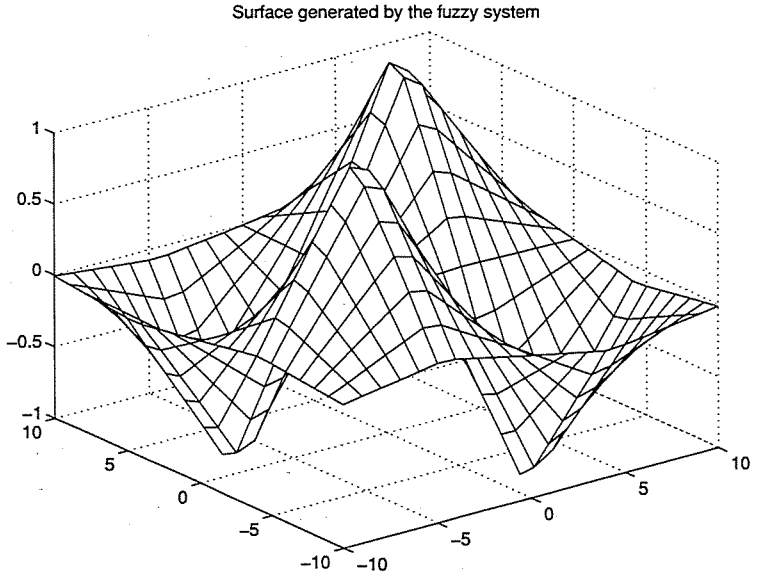


Figure 2.36: *Example 1:Surface Generated by the Fuzzy System*

2.4.7 Example 2:Modeling of a three input nonlinear function

For this example the data were generated using the function:

$$f(x,y,z) = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2 \tag{2.105}$$

For this example 216 random points from the input range $[1,6] \times [1,6] \times [1,6]$ were used as training set and 125 random points from the input range $[1.5,5.5] \times [1.5,5.5] \times [1.5,5.5]$ were used as validation set. As a performance index, we used the average percentage error (APE):

$$APE = \frac{1}{P}\sum_{i=1}^{P}\frac{|T(i) - O(i)|}{|T(i)|} \times 100\% \tag{2.106}$$

where $T(i)$ is the desired output and $O(i)$ is the predicted output. This performance index allows us to compare the present result with previous works. First a mountain clustering procedure was used and 11 clusters were found, further refinement was obtained by using fuzzy C-means clustering algorithm. In figure 2.37 the projected membership functions can be observed. After reduction using FuZion with a minimum distance factor of 15% of the size of the universe of discourse, the membership functions shown in figure 2.38 were obtained.

Figure 2.39 shows the singleton consequences and the consequences after FuZion. Table I shows the comparative results with previous work. From the results, it can be observed that the numeric performance is very similar to other proposed methods. It is important to remark the generalization capabilities of the system generated by the AFRELI method. Observe the small degradation of the performance when the training and the validation set are used, all the other techniques show significant degradation (i.e.ANFIS almost 2 orders of magnitude). This example shows that the method provides an acceptable numerical performance with the advantage that the interpretability is guaranteed. ◊

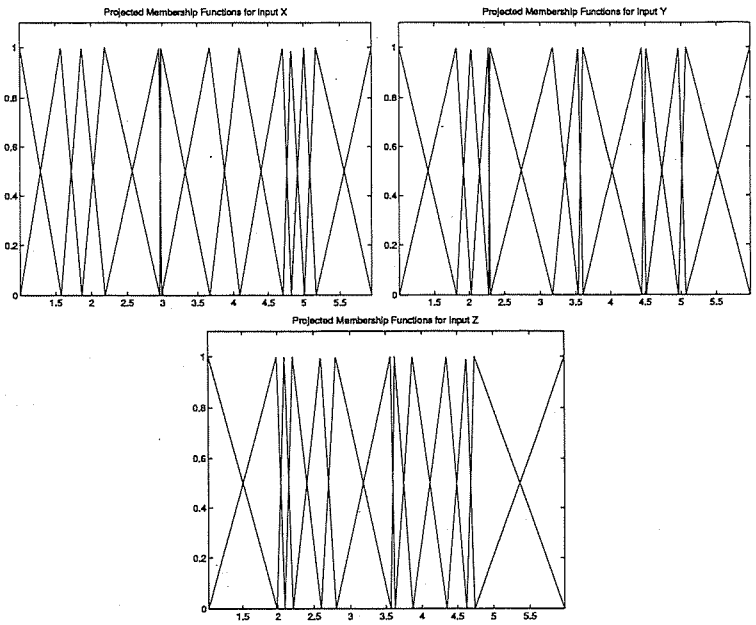


Figure 2.37: *Example 2:Projected membership functions*

Table 2.2: Example 2: Performance comparison with previous work. The results from previous works were taken from [40].

Model	APE_{TRN}	APE_{VAL}	Param. Num.	Size Train. Set	Size Valid. Set
AFRELI	1.002 %	1.091 %	80	216	125
ANFIS	0.043 %	1.066 %	50	216	125
GMDH	4.7 %	5.7 %	-	20	20
Fuzzy model 1	1.5 %	2.1 %	22	20	20
Fuzzy model 2	0.59 %	3.4 %	32	20	20

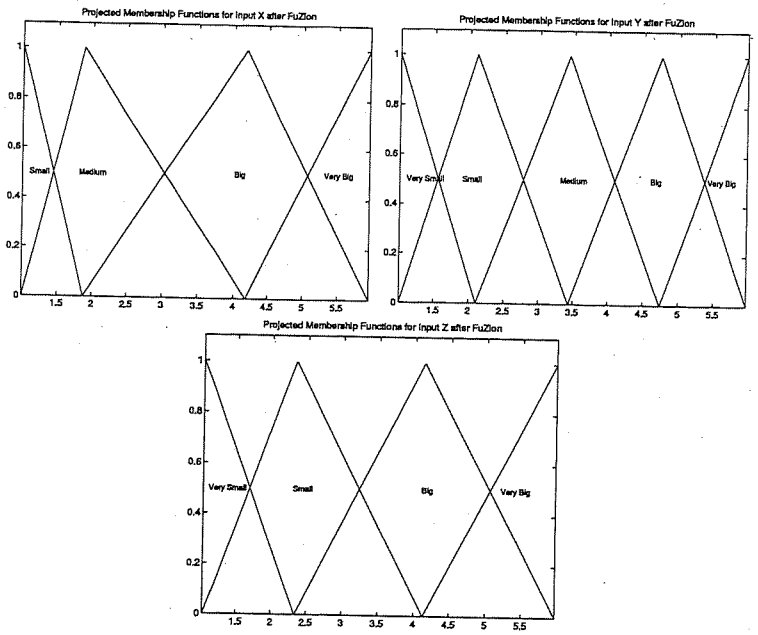


Figure 2.38: *Example 2: Membership functions after FuZion*

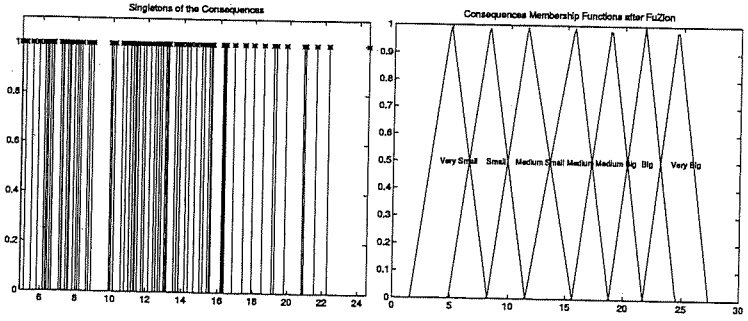


Figure 2.39: *Example 2:(a) Singletons (b) Membership functions with linguistic meaning*

2.4.8 Example 3:Predicting Chaotic Time Series

This example shows the capability of the algorithm to capture the dynamics governing the Mackey-Glass chaotic time series. These time series were generated using the following delay differential equation:

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{2.107}$$

where $\tau = 17$. The numerical solution of this differential equation was obtained using fourth order Runge-Kutta method, with a time step of 0.1 and initial condition $x(0) = 1.2$. The simulation was run for 2000 seconds and the samples were taken each second. To train and test the fuzzy system 1000 points were extracted $t = 118$ to 1117. The first 500 points were used as training set and the remaining as validation set. First, a six step ahead predictor is constructed using past outputs as inputs of the model:

$$[x(t-18)\, x(t-12)\, x(t-6)\, x(t)] \tag{2.108}$$

and the output will be $x(t+6)$.
After applying the mountain clustering method, 57 clusters were found. Some refinement on the position of the clusters was obtained by using Fuzzy C-Means clustering method. After projection and FuZion the membership functions shown in figure 2.40 were obtained.

To allow a comparison with previous works, the prediction error was evaluated using the so called Non-Dimensional Error Index (NDEI) defined as:

$$NDEI = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(T(i)-O(i))^2}}{\sigma(T)} \tag{2.109}$$

where $T(i)$ is the desired output, $O(i)$ is the predicted output and $\sigma(T)$ is the standard deviation of the target series.
The tables II and III show some comparative results. In this example, the impact of the use of the optional step of optimization can be observed. It is clear that the improvement of this optional step is small (reduction of about 30% on the NDEI) but of course on certain applications this value could be significant. Observe once more that the numeric performance is similar to other techniques but a significant value is added with the interpretability of the obtained rule base. $\Diamond$

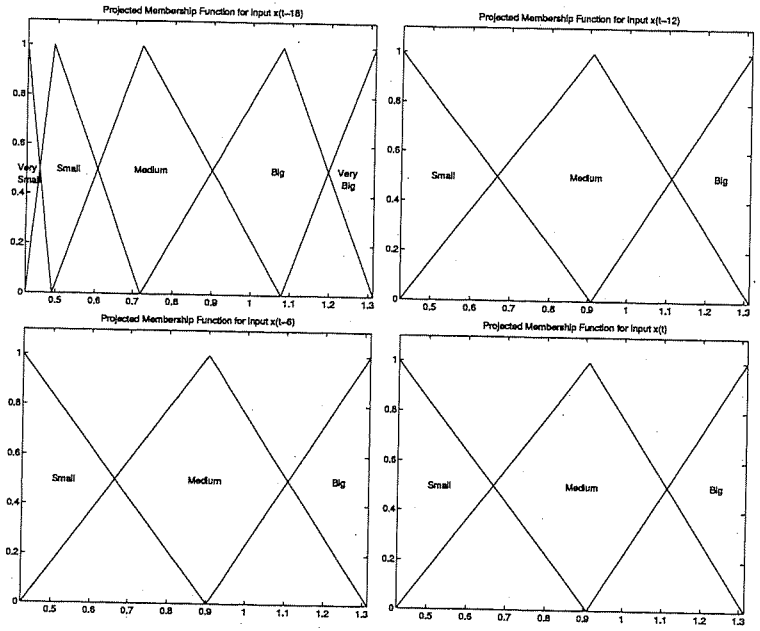


Figure 2.40: *Example 3:Membership functions after projection and FuZion*

Table 2.3: Example 3: Performance for prediction six steps ahead. The results from previous works were taken from [40].

Method	Training Cases	Non-dimensional error index
AFRELI	500	0.0493
AFRELI (with optional step)	500	0.0324
ANFIS	500	0.007
AR model	500	0.19
Cascaded-correlation NN	500	0.06
Backpropagation MLP	500	0.02
6th order polynomial	500	0.04
Linear predictive method	2000	0.55

Table 2.4: Example 3: Performance for prediction 84 steps ahead (the first seven rows) and 85 (the last four rows). Results for the first seven methods are obtained by simulation of the model obtained for prediction six steps ahead. Results for localized receptive fields (LRFs) and multiresolution hierarchies (MRHs) are for neurons trained to predict 85 steps ahead. The results from previous works were taken from [40].

Method	Training Cases	Non-dimensional error index
AFRELI	500	0.1544
AFRELI (with optional step)	500	0.1040
ANFIS	500	0.036
AR model	500	0.39
Cascaded-correlation NN	500	0.32
Backpropagation MLP	500	0.05
6th order polynomial	500	0.85
Linear predictive method	2000	0.60
LRF	500	0.10-0.25
LRF	10000	0.025-0.05
MRH	500	0.05
MRH	10000	0.02

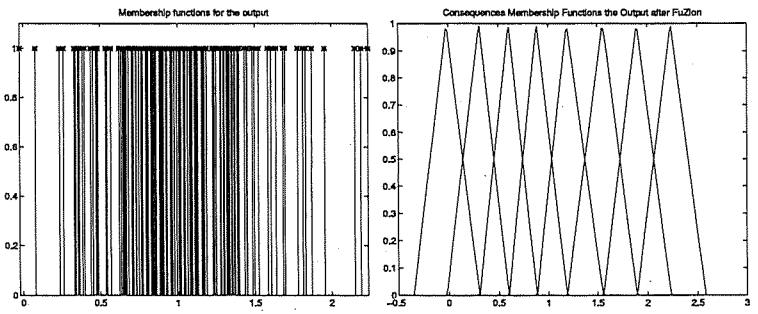


Figure 2.41: *Example 3:(a)Singletons (b) Membership functions with linguistic meaning*

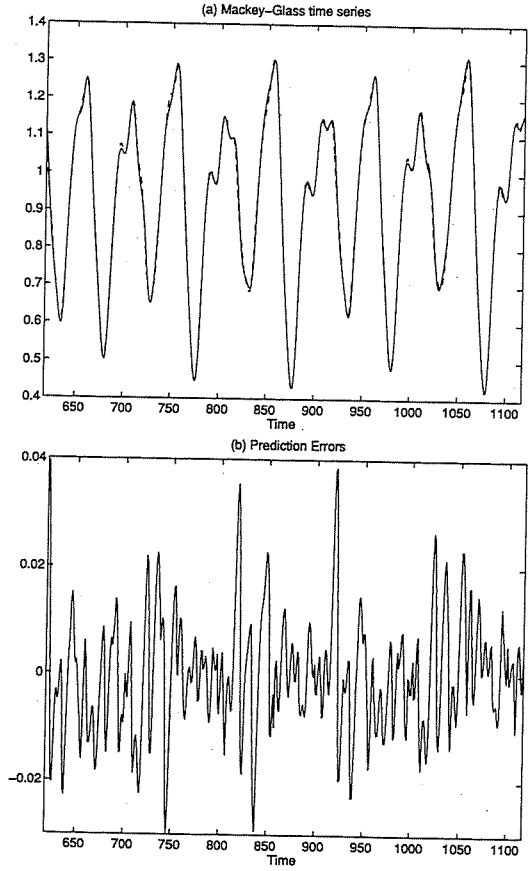


Figure 2.42: *Example 3:(a) Mackey-Glass time series (solid line) from* $t = 618$ *to 1117 and six steps ahead prediction (dashed line) (b) Prediction errors*

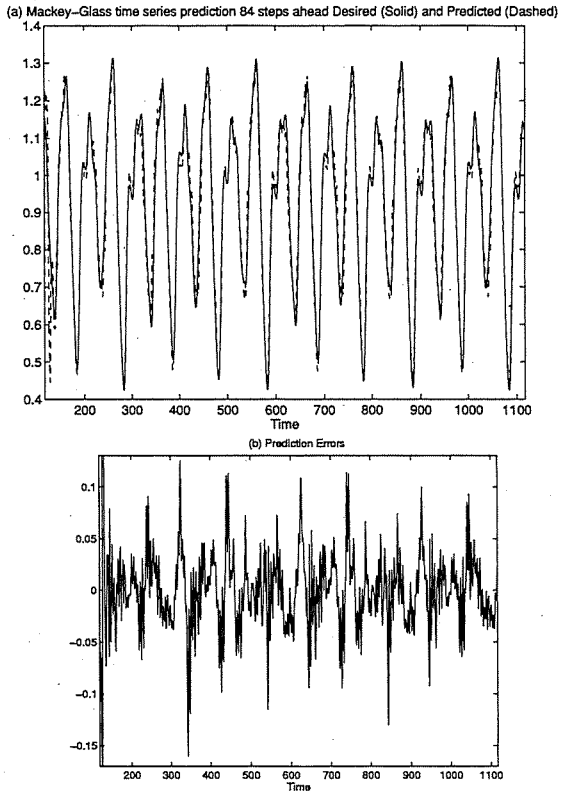


Figure 2.43: *Example 3:(a) Mackey-Glass time series (solid line) from* $t = 118$ *to 1117 and 84 steps ahead prediction (dashed line) (b) Prediction errors*

2.4.9 Remarks on the AFRELI algorithm

The AFRELI algorithm is an algorithm based on clustering methods to locate the rules. The clusters are projected into the input space and converted into fuzzy sets. The FuZion algorithm reduces the large number of projected sets on each input so that the linguistic integrity is preserved. With the reduce group of fuzzy sets a combinatorial number of IF-THEN rules is generated using only the AND operation in the antecedents. The rule base will cover every possible case in the compact set defined by the universes of discourse of the input domains. This is an important advantage because the system will be able to predict values of the approximated function even if there were no similar values in the data set used to construct the model. The generalization of this method for the case of rules badly excited will be as good as the generalization given by a multilinear model of dimension N where N is the number of inputs.
However, such large number of rules present some problems: i) a large set of rules with a large set of antecedents is difficult to understand and analyze; ii) a large set of rules demands large memory storage; for instance, a rule base created for a system with five inputs and six membership functions on each input will demand approximately 30 Kbytes of memory and a similar system with 10 inputs will demand 230 Mbytes of memory. On the other hand the evaluation time will not grow that fast, for the first system only $2^5 = 32$ rules need to be evaluated, for the second system $2^{10} = 1024$ rules.

2.5 Conclusions

The capabilities of fuzzy logic systems to approach non-linear functions have been shown. The constructing units have been detailed mathematically and graphically. Different methods to construct fuzzy models that approximate non-linear functions have been shown. The issue of lack of excitation and generalization has been analyzed and a method to guarantee good generalization has been proposed. This method guarantees a lower bound in the quality of the model (the fuzzy model will be at least as good as the best multi linear approximation).
Using the fact that the comparative advantage of fuzzy systems in comparison with other "universal approximators" is their linguistic interpretability the AFRELI algorithm has been proposed. The AFRELI algorithm in combination with the FuZion algorithm guarantees a good trade-off between numerical accuracy and interpretability. The method exploits some successful elements proposed in other methods to reduce the complexity of the model construction. The algorithm generates automatically the fuzzy sets and the interactive labeling process (with intervention of the designer) guarantees an agreement between

the fuzzy set and the assigned label.
The method generates a rule base covering all the possible cases, this guarantees the completeness of the rule base, but the associated drawback is the exponential growth of the rule base as the number of inputs increases. However, this is only a storage problem because the description of the fuzzy sets guarantees that only 2^N rules (N number of inputs) are activated on each inference. This makes the inference process fast because only a limited number of rules are evaluated.
The numerical accuracy of the algorithm is directly related with the choices of the M parameter governing the FuZion algorithm, and the choices in the clustering algorithm. When the number of inputs is large, the clustering will be limited to use Fuzzy C-means algorithm with an overestimated number of clusters. The poor performance of the Mountain Clustering method with large dimensions motivated this modification.
Some improvements of the numerical performance of the model can be obtained by making a "fine" tuning of the parameters of the antecedents by means of constrained gradient descent techniques.

Chapter 3

Nonlinear Identification Using Fuzzy Models

This chapter is oriented to the study of system identification using fuzzy models. The chapter shows the main issues involved in system identification using fuzzy models such as the structure of the fuzzy systems, experiment design for this type of models, regressors selection, structure selection, parameter calculation and validation. The main contribution of the author is the presentation of the possibilities, methods and issues related to the identification when using fuzzy models. A specific contribution is the analysis and derivation of the gradients for dynamic calculation of the parameters of dynamic fuzzy models. This result is condensed at the end of the thesis in the appendix C.
It is important to remark that fuzzy systems are one of several possibilities in the area of nonlinear system identification. Neural networks, Volterra series, wavelets and other universal approximators represent other possibilities in nonlinear system identification. The advantage of the use of fuzzy systems is their capacity to interact and to extract linguistic information from input-output data. It is a very valuable capacity of the fuzzy models that makes them different from other traditional black box techniques. The capacity to handle linguistic information adds an extra dimension to the identification and modeling because the validation process will be based not only on quantitative criterion but also on qualitative criterion such as whether or not the extracted rules "make sense". In this way, expert knowledge and empirical knowledge (which is normally not represented as quantitative information) can be exploited. Extra knowledge is gained when users of the models observe the extracted rules and realize what kind of reasoning process they apply on certain decision. This is typically the case when the objective is to make the identification of a group of human experts. Information is collected about their decision under given circumstances.

The human experts know how to take a decision, but they fail to explain the reasoning mechanism that they are using. Once the knowledge is extracted in the form of a fuzzy model with a rule base, they realize the reasoning mechanism that they are using. One of the main failures when human experts try to generate rules is to neglect the dynamic information, such as trends and other dynamic effects, which are tightly linked to the decision process. This capacity to extract knowledge does not come for free and the price paid for this property is a reduction in accuracy and the so-called curse of dimensionality. The curse of dimensionality arises because the number of rules grows exponentially when the number of inputs to the model is increased.
The modeling and function approximation using fuzzy models is a topic that has been studied extensively in recent years [73] [43] [81] [62] [39] [74], but very few authors had studied the whole problem in a formal way, most of them focus on training algorithms and function approximation. This chapter tries to fill this gap by studying most of the issues involved on system identification.

3.1 System Identification

System identification is a technique that permits to build mathematical models of dynamic systems based on input-output data. The output of the dynamic system at time t is $y(t)$ and the input $u(t)$. Then the "data set" will be described as:

$$Z^t = \{y(1), u(1), \ldots, y(t), u(t)\} \tag{3.1}$$

A model of the dynamic system can be constructed as a mapping from past data Z^{t-1} to the next output $y(t)$. This model is known as the predictor model and is represented by:

$$\hat{y}(t) = f(Z^{t-1}) \tag{3.2}$$

where $\hat{y}(t)$ represents the estimated output. The essence of identification using fuzzy systems is to try to represent the function f by means of a fuzzy model. It is important to see the fuzzy system as a parameterizable mapping,

$$\hat{y}(t|\theta) = f(Z^{t-1}|\theta) \tag{3.3}$$

where θ is the vector of parameters to be chosen (position and shape of the membership functions, consequences of the rules, etc.). The choice of these parameters is guided by the information embedded in the data. The structure of (3.2) is a very general structure and it has the drawback that the data set is continuously increasing. For this reason, it is better to use a vector $\varphi(t)$ of fixed dimension. So the general model is now formulated as:

$$\hat{y}(t|\theta) = f(\varphi(t)|\theta) \tag{3.4}$$

The vector φ is known as *regression vector* and its elements *regressors*.

$$\varphi(t) = [y(t-1), \ldots, y(t-n), u(t-1), \ldots, u(t-m)] \tag{3.5}$$

Using this parameterization, the problem can be decomposed in three subproblems:

1. How to choose the regressors in $\varphi(t)$ from past inputs and past outputs?
2. How to find the structure of the Fuzzy System $f(.,.)$?
3. How to find the parameters θ ?

The following sections will address most of these topics.

3.2 The basic structure of the Fuzzy System

The fuzzy systems are suitable for identification, from the mathematical point of view, because these structures are "universal approximators" as it was shown in the previous chapter. Using this way of reasoning there is a guarantee that the nonlinear system identification problem can be approached using fuzzy systems. As it was mentioned, there are many of these universal approximators (RBF,MLP, Wavelets, Fourier Series, Volterra Kernels, etc). The use of fuzzy systems for nonlinear identification is not motivated only by their approximation capabilities but mainly by their capacity to extract linguistic information in the form of IF-THEN rules.
As explained in the previous chapter the alternatives involved in the structure selection for fuzzy system are very wide: many alternatives in shapes for membership functions, *AND* and *OR* operations, implications, defuzzification methods, etc.
Due to the explicit need of parameter adjusting the operations product and the bounded sum have been selected as *AND* and *OR* operations because their derivatives are continuous. The shape of the membership functions of the consequences will be singletons or linear combinations of the inputs (Takagi Sugeno models). It is important to remember that the singletons can be converted into triangular membership functions and reduced using the rule description presented in the previous chapter in the AFRELI algorithm.
The choice of the type of consequence will determine the linguistic properties of the model: the use of singletons gives more linguistic meaning to the rules, but the use of T-S models can improve the approximation properties specially in the case of piecewise linear functions.

The shape of the membership functions for the fuzzification process will be preferred to be triangular or polynomial for their local and linguistic properties (see previous chapter), but Gaussian membership functions can also be used.
Since additional constraints are needed to preserve the linguistic meaning, the number of membership functions in every universe of discourse for the inputs should be limited to as much as 7 ± 2 and at least 2. Other additional constraints can be introduced to improve the linguistic interpretability of the model as they have been described in the previous chapter.
In this section the parameterization of the fuzzy system presented in the previous section will be used:

$$f(\mathbf{x}) = \frac{\sum_{l=1}^{L} \bar{y}^l \mu_l(\mathbf{x})}{\sum_{l=1}^{L} \mu_l(\mathbf{x})} \tag{3.6}$$

where $\mathbf{x}$ represents the input vector, in the present case the input vector will be the regressors $\varphi(t)$ and $\mu_l(\mathbf{x})$ represent the membership function of the rule l constructed during the inference process. Finally $\bar{y}^l$ is the singleton consequence of the l rule.
For the case of Takagi-Sugeno models the parameterization will be,

$$f(\mathbf{x}) = \frac{\sum_{l=1}^{L} A^l \mathbf{x} \mu_l(\mathbf{x})}{\sum_{l=1}^{L} \mu_l(\mathbf{x})} \tag{3.7}$$

where A^l is a row vector. In this case, the consequences are linear combinations of the inputs or other variables.
These are some initial choices. A final refinement of the structure of the fuzzy system is guided by the information provided by the data. The number of rules and the distribution of the input membership functions will be obtained from the data.
These structures as they are described will be used to identify Multiple Input Single Output (MISO) systems. Most of the comments expressed here are also valid for Multiple Input Multiple Output (MIMO) systems. However, the author considers that the problem of Nonlinear MIMO identification problems only makes sense, from the practical point of view, if the output variables are of the same type (p.e. all temperatures or all displacements) and similar order of magnitude. Otherwise, the formulation of a cost function that balances the approximation of different signals with different magnitude is very difficult. In addition, MIMO parameterizations where some outputs are weakly related with some inputs tend to generate overfitting and bad generalizations. In these cases is better to use multiple MISO models (one per output) where only strongly related inputs are used to construct the model. Because there is only one output no compromise among outputs should be taken into account to formulate the cost approximation function.

This motivation for choices of the parameters is considered an interesting contribution of this thesis to the subject.

3.3 Experiment Design

The objective of this stage is to extract as much information as possible from the dynamic system by means of a "good" input signal. A good input signal is a signal that exposes the most important properties of the plant. A good input signal for the identification of nonlinear systems is a signal that should be rich on amplitudes and frequency, limited in duration and, according to the application of the model, the signal must excite those characteristics which are relevant for the desired application.
The problem of optimal experiment design can be formulated as the minimization in the uncertainty of the estimated parameters in the model. This statement includes a fundamental assumption implying that the structure of the model is already known. This leads to the consequence that every *Optimal Experiment Design* for the identification of a nonlinear system should be an iterative process. In this iterative process, an initial signal is used to make a rough identification of the structure of the model (regressors and noise model). Once this structure is selected an initial description of the model is created and, with this description, an optimization can be performed.
In heuristic terms, a good excitation signal is a signal that will excite the complete rule base. The signal should expose all possible cases presented in the rule base and if the rules base is complete, (all possible combinations of antecedents) the excitation signal will cover all possible situations in a compact set. It is important to remark that the excitation of the rule implies not only the excitation of the consequence but also the excitation of the membership functions of the antecedents associated with the rule.
In general the probability distribution of the parameter estimates can be characterized by their bias $B(\hat{\theta})$ and covariance $\mathrm{cov}(\hat{\theta})$. If the estimator is unbiased, the bias will approach zero asymptotically as the number of training point grows and the covariance will be bounded [48].
Assuming the description of the fuzzy system given in equation (3.6) the fuzzy system can be described as:

$$\hat{y}(t) = \sum_{l=1}^{L} \bar{y}^l w_l(\varphi(t)) \tag{3.8}$$

with:

$$w_l(\varphi(t)) = \frac{\mu_l(\varphi(t))}{\sum_{l=1}^{L} \mu_l(\varphi(t))} \tag{3.9}$$

such that:

$$y(t) = \bar{Y}^T W(t) + e(t) \tag{3.10}$$

where $\bar{Y} = \{\bar{y}^1, \bar{y}^2, \ldots, \bar{y}^L\}^T$, $W(t) = \{w_1(\varphi(t)), w_2(\varphi(t)), \ldots, w_L(\varphi(t))\}^T$ and $e(t)$ is the prediction error which is assumed to have zero mean and variance σ^2. If the vector of the consequence values is estimated using Least Squares the covariance matrix of errors on the estimates of $\bar{Y}$ will be given by:

$$\mathrm{cov}\bar{Y} = \sigma^2 \left[\sum_{t=1}^{N} W(t) W(t)^T \right]^{-1} \tag{3.11}$$

Now the problem is to minimize (or maximize) some measure of the covariance matrix or its inverse the information matrix defined as $M_{\bar{Y}} = [\mathrm{cov}\bar{Y}]^{-1}$. Formally, the problem can be written as:

$$\min_{u(1),\ldots,u(N)} \phi(M_{\bar{Y}}) \tag{3.12}$$

According to the purpose of the model the cost function can be defined in different ways including even criteria such as *control relevance* of the inputs [16]. This optimization problem is nonlinear and non-convex so there is no guarantee that the obtained solution is the global minimum but, sometimes, a local minimum is an acceptable solution.
When no prior knowledge about the structure is present, using a combined sequence of random movements can create an excitation signal. The experience of the author shows that a good excitation can be obtained by constructing a signal composed by a "slow" random signal with some discrete values selected in the input range and a random wide band signal (fast signal) with an amplitude that covers the gap between the discrete values of the "slow signal". The reasoning behind the formulation of this kind of signal is that the slow signal will drive the plant to different operating points and the fast signal will guarantee enough excitation around the operating points. It is important to remark that the fast signal will be limited in amplitude and frequency such that saturation and slew rate limitations in the actuators will be respected. Figure 3.1 shows an example of such a signal. Combinations of this type of signals can be used for the identification of multi variable systems. Other excitation signals are the multi-sine signals with variable frequency and the swept sinus with random frequencies. These signals are frequently used in the identification of mechanical systems.

3.4 Choosing the Regressors

The problem of regressors selection for nonlinear modeling is a complex combinatorial problem that cannot be solved in polynomial time. The complexity of

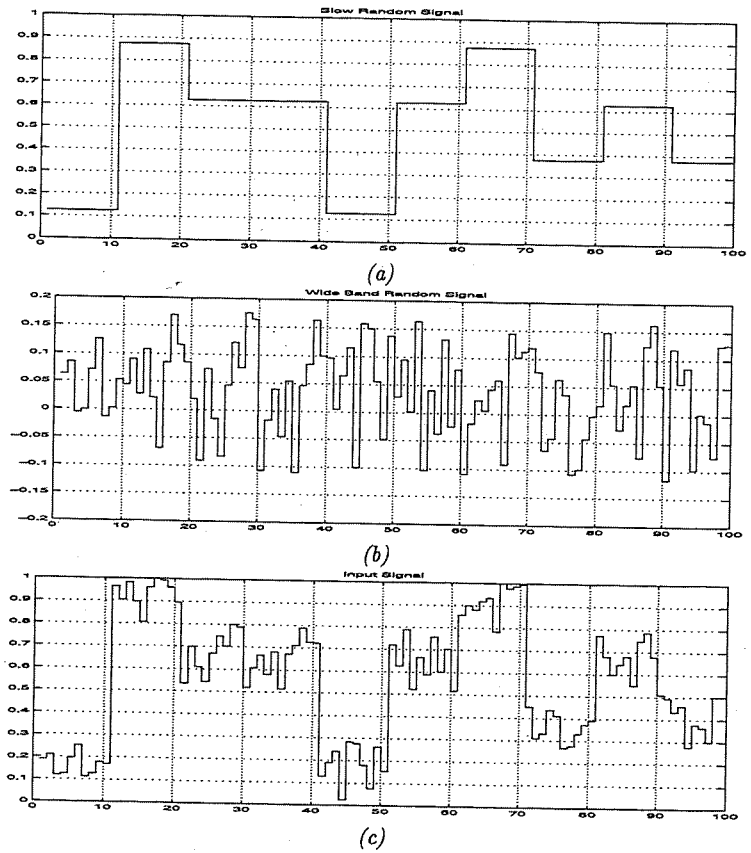


Figure 3.1: *Heuristically Designed Excitation Signal with four discrete levels between 0 and 1. (a) Slow random signal with 4 discrete levels (b) Wide band signal (c) Combined signals.*

the problem can be evaluated as follows: if there are N possible regressors, 2^N possibilities must be evaluated.
Figure 3.2 shows the sequence for regressors selection. First a selection algorithm to pick-up the regressors from a set of possible candidates is needed,

because the exhaustive search of the 2^N possibilities is not practical. Some shortcuts to guide the search have been proposed. Among these methods it is worth to mention the tree search methods and the genetic selection methods. The scheme shows that once a set of regressors is postulated the quality must be evaluated. To evaluate the quality of the regressors two types of strategies can be distinguished: Model based strategies, where a model should be built to evaluate the quality of the regressors, and model free strategies, where the quality of the regressors is evaluated using only the input-output data. Next sections will explore in detail these issues. Section 3.4.1 will show the search methods and the section 3.4.2 will show the evaluation methods.

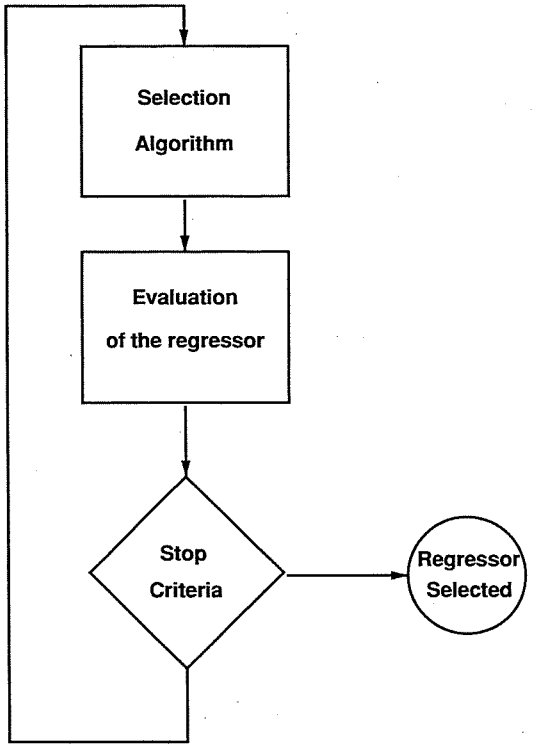


Figure 3.2: *Sequence for regressors selection*

3.4.1 Search methods

There are two classes of search methods, the heuristic search methods and the pseudo random methods.

Heuristic Search

In the class of heuristic search methods, a tree selection method is used. In this method, two approaches can be used: one approach constructs a model adding new inputs one by one. The other method makes a model with all possible inputs and regressors are dismissed according to the impact in the performance evaluation (those with small impact are removed first). For fuzzy systems, the first one is preferred because the size of the models made for the evaluation is smaller (in terms of number of inputs). The method works as follows: First a set of all possible models using one regressor is proposed (N models where N is the number of regressor candidates). The quality of the models is evaluated using any of the model based or model free methods shown in section 3.4.2. The regressor with the best performance index is selected and new set of models with two regressors is proposed. On of the two regressors will be always the one selected in the previous step, in this way $N-1$ possible models will be proposed and evaluated. Again the best regressors will be selected and the procedure will be repeated until a prescribed number of inputs is reached or no improvement in the performance index is observed. Figure 3.3 shows a graphical representation of an example of regressor selection using this method. At the worst case, this algorithm reduces the number of evaluated possibilities to:

$$\sum_{i=1}^{N} i \tag{3.13}$$

where N is the number of regressor candidates.

This tree search does not guarantee that the best regressors are selected. It ignores that some information can be carried only when two or more signals are present at the same time. To overcome this problem the search can evaluate the addition of two or more inputs at the same time.

Pseudo random methods

In this category, the genetic algorithm methods are included. The selection of the candidate regressors to be evaluated is given by the genetic algorithm. As explained in the previous chapter there is a high probability that after a given number of iterations the optimal solution is found. The advantage of using the genetic algorithm is that the codification of the problem is straightforward and

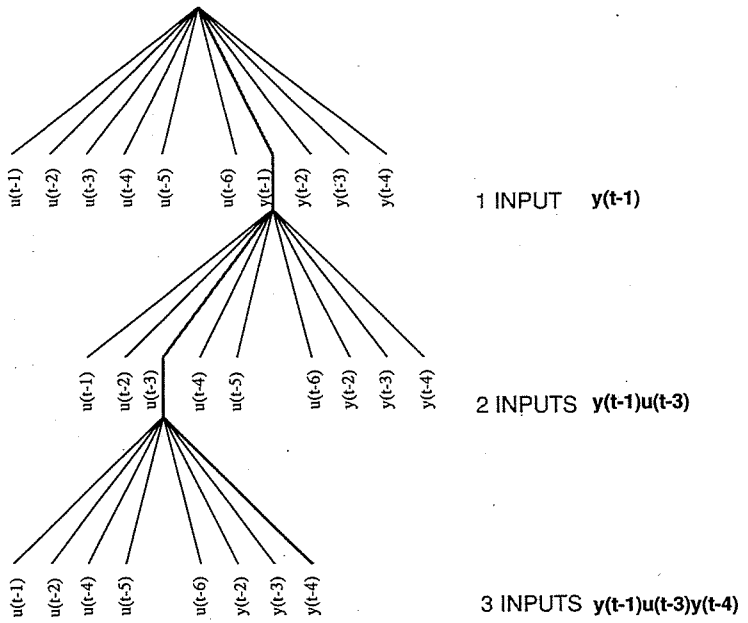


Figure 3.3: *Heuristic Search for regressors selection*

each bit of the string will be used to represent an active (bit equal "1") regressors or an inactive (bit equal "0") regressors. The figure 3.4 shows the codification for the selection of the regressors among 10 candidates and the effect of the crossover operator of the genetic algorithm. The number of evaluations for this method is at the worst:

$$\text{Size of the Population} \times \text{Number of generations} \tag{3.14}$$

3.4.2 Regressors Evaluation

Model based evaluation

The model-based evaluation evaluates the prediction error of the model constructed with the suggested regressors. Care must be taken during the construc-

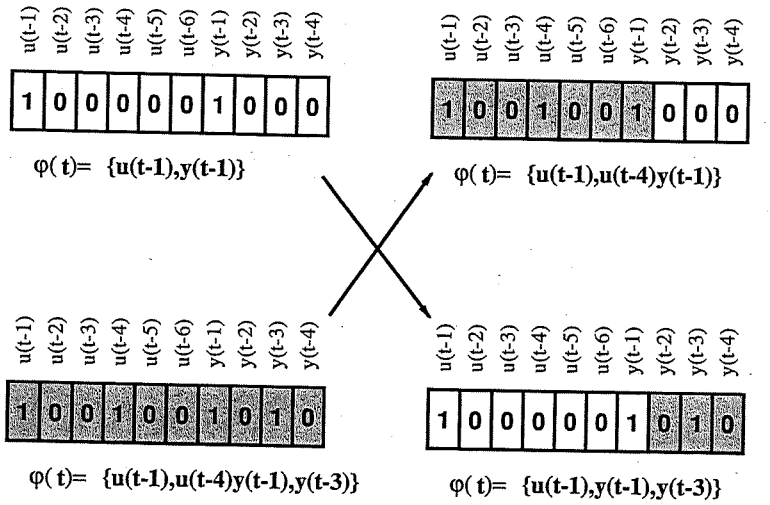


Figure 3.4: *Regressors selection using Genetic Algorithms. Effect of the crossover operator*

tion of the model to avoid problems such as lack of excitation, over-parameterization and large training time. An acceptable solution is the use of the *smallest fuzzy model* described in section 2.3.1. This model is evaluated very fast (only a Least Squares solution is needed), there is a guarantee of enough excitation and there is no chance of over-parameterization. In addition, the model is more complex than a linear model; in fact, the model is multilinear. The evaluation of a complete model with more than two membership functions per input is not recommended for two reasons:

- Computational cost could be very large and the regressor selection requires many model evaluations.
- Lack of excitation and over-parameterization could spoil the evaluation of the set of regressors.

Different cost functions have been presented in the literature to evaluate the quality of a model. Some of these cost functions penalize the validation error $\varepsilon = y(t) - \hat{y}(t|\theta)$ and the complexity of the model $\dim(\theta)$. The fact that these two measures have different units requires the use of weights to combine them in a cost function. These weights are in most of the cases arbitrary. Among these

types of criteria are the Akaike's Information Criteria (AIC) and the Rissanen's Minimum Description Length (MDL) [2]. The cost function for the Akaike's information criterion is given by

$$V_N(\theta, Z^N) = \left(1 + \frac{2\dim\theta}{N}\right)\frac{1}{N}\sum_{t=1}^{N}\varepsilon^2(t,\theta) \tag{3.15}$$

where

$$\varepsilon = y(t) - \hat{y}(t|\theta) \tag{3.16}$$

and $\dim\theta$ represents the length of the vector of the parameters, and N is the size of the validation set.
The Rissanen's Minimum Description Length criterion is very similar the AIC and its cost function is described as:

$$V_N(\theta, Z^N) = \left(1 + \frac{2\log N\dim\theta}{N}\right)\frac{1}{N}\sum_{t=1}^{N}\varepsilon^2(t,\theta) \tag{3.17}$$

The fact that the weighting between the objectives makes it very "subjective" is very difficult to say which cost function is more suitable.
Other criteria are practically motivated and they take into account only the validation error. Most of these criteria try to minimize the impact of the data set Z^N used for the evaluation. One example of this type of methods is the so-called **Regularity Criterion**. The application of this method to fuzzy modeling was suggested by Sugeno and Yasukawa [71]. The regularity criterion was first proposed in other modeling technique known as GMDH (group method of data handling). The cost function for this criterion is defined as follows.

$$RC = \left[\sum_{i=1}^{k_A}(y_i^A - \hat{y}_i^{AB})^2/k_A + \sum_{i=1}^{k_B}(y_i^B - \hat{y}_i^{BA})^2/k_B\right]/2 \tag{3.18}$$

where

- k_A and k_B Number of data of the groups A and B.
- y_i^A and y_i^B outputs of the data groups A and B.
- $\hat{y}^{AB}$ Model output for the group A input estimated using the model constructed with the data set B.
- $\hat{y}^{BA}$ Model output for the group B input estimated using the model constructed with the data set A.

This cost function evaluates the prediction error in a cross-validation scheme. Figure 3.5 shows a representation of the cross-validation scheme.

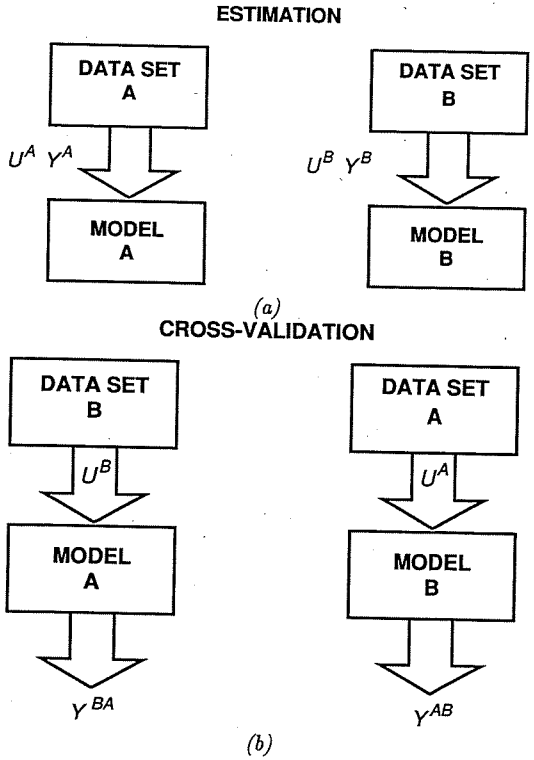


Figure 3.5: *Regularity Criterion. (a) Estimation of models A and B (b) Cross-validation of the models*

Model Free Evaluation

The main drawback of the model-based techniques is that the evaluation can be influenced by estimation errors or, in some cases, the calculation of the parameters of the model can be very time consuming. A good alternative is the model free test proposed by He and Asada in [37]. The method is based in the evaluation of the so-called *Lipschitz Quotients*. A Lipschitz quotient is defined in this framework: Given a nonlinear function $y = f(\mathbf{x})$ and N input output pairs $(y_i, (x)_i)$, the Lipschitz quotient q_{ij} is given by

$$q_{ij} = \frac{|y_i - y_j|}{|\mathbf{x}_i - \mathbf{x}_j|}, (i \neq j) \tag{3.19}$$

Using these quotients, the following index is formulated,

$$\bar{q}^{(n)} = (\prod_{k=1}^{p} \sqrt{n} q^{(n)}(k))^{1/p} \tag{3.20}$$

where $\bar{q}^{(n)}$ is called the Lipschitz number, $q^{(n)}(k)$ is the k-th largest Lipschitz quotient among all $q_{ij}^{(n)} (i \neq j; i, j = 1, 2, \ldots, N)$ where n is the number of input variables. The parameter p is selected as $0.01N < p < 0.02N$. From eq.(3.20) it is clear the $\bar{q}^{(n)}$ is the geometric mean of the sequence $q^{(n)}(1), q^{(n)}(2), \ldots, q^{(n)}(p)$. In practice, the criterion decreases as the number of regressors increases until a point where no further improvement is obtained. The index evaluates the smoothness of the mapping constructed with the data points in the regressor vector. This method works very well with noise free data; it can even find the regressors for a chaotic time series. The performance with noise is reduced but it still works with signals having a S/N ratio of around 8 dB.

3.5 Choosing the Structure

Following the nomenclature for linear models the following structures has been defined for non-linear systems [69] [15] as follows:

- NFIR models, in this case the vector of regressors is composed only with past inputs $\varphi = [u(t-k), \ldots, u(t-n)]$.
- NARX models, in this case the vector of regressors has past inputs and outputs $\varphi = [u(t-k), \ldots, u(t-n), y(t-k), \ldots, y(t-m)]$.
- NOE models, the vector of regressors has past inputs and past estimated outputs: $\varphi = [u(t-k), \ldots, u(t-n), \hat{y}(t-k|\theta), \ldots, \hat{y}(t-m|\theta)]$.
- NARMAX models, which use as regressors past inputs, past outputs and estimation errors $\varphi = [u(t-k), \ldots, u(t-n), y(t-k), \ldots, y(t-m), \varepsilon(t-k|\theta), \ldots, \varepsilon(t-l|\theta)]$.
- NBJ models, in this case the regressors are past inputs, estimation errors using past outputs and estimation errors using past estimated outputs $(\varepsilon_u(t-k|\theta))$: $\varphi = [u(t-k), \ldots, u(t-n), y(t-k), \ldots, y(t-m|\theta), \varepsilon(t-k|\theta), \ldots, \varepsilon(t-l|\theta), \varepsilon_u(t-k|\theta), \ldots, \varepsilon_u(t-p|\theta)]$.

- State Space models, for the deterministic case the model will be:

$$\hat{x}(t+1) = f(\hat{x}(t), u(t)) \quad (3.21)$$
$$\hat{y}(t) = g(\hat{x}(t), u(t)) \quad (3.22)$$

for the stochastic case the model will be:

$$\hat{x}(t+1) = f(\hat{x}(t), u(t), \varepsilon(t)) \quad (3.23)$$
$$\hat{y}(t) = g(\hat{x}(t), u(t)) \quad (3.24)$$

It is clear that there is a wide set of possible structures. The application, the information available and the complexity of the model condition the selection of one type of structure. Some remarks about the models and their applications are reviewed in the following lines.
The NFIR model is simple and has some interesting properties with respect to the stability, but its main drawback is that it needs a significant number of past inputs to capture simple dynamics. This large number of past inputs can make the rule base of the fuzzy model very big, thus making the model inefficient.
The NARX model is the most used structure because it is very easy to estimate given its non-recursive structure. For this reason, the NARX models are used during the phase of regressor selection where the computation of several models requires the use of simple models. It is interesting for applications where short term (one step ahead) forecasting is needed.
The NARMAX, NOE, NBJ and the State Space stochastic model are recurrent models where the estimated output is fed back. This characteristic make the estimation of the model more complicated. The NARMAX and the NBJ model can be used to construct longer term predictors. The advantage of the use of these models is that the information provided by the simulation model is corrected via a noise model.
NOE models are preferred for simulation purposes because no information from the "real system" is needed to operate a simulation, since past real outputs are not included in the model.

3.6 Calculating the Parameters

The parameter adjustment techniques for dynamic systems are very similar to the ones used for function approximation, as explained in the previous chapter. Clustering techniques and projection can be used to estimate initial distributions of the membership functions. This initial estimation must be done using the

NARX structure, because the construction of an NARX model is equivalent to the construction of a model for static function approximation. The initialization of the consequences must use the method explained in the previous chapter in section 2.3.
Once the structure of the model has been selected, the NARX model can be used to initialize the other structures; further improvement must be achieved using gradient descent techniques.
The classical cost function to be minimized is the quadratic cost function, defined as

$$V_N(\theta) = \frac{1}{2N}\sum_{t=1}^{N} |y(t) - \hat{y}(t|\theta)|^2 \quad (3.25)$$

where θ is a vector containing the parameters of the membership functions and the position of the singletons of the consequences or the parameters of the vectors $\mathbf{A}^l$ in the Takagi-Sugeno models. The iterative scheme to update the parameters is defined as

$$\hat{\theta}^{(i+1)} = \hat{\theta}^{(i)} - \mu_i R_i^{-1} \hat{g}_i \quad (3.26)$$

where μ_i is the step size, $\hat{g}_i$ is an estimate of the gradient $V_N'(\hat{\theta}^{(i)})$ and R_i is a matrix that modifies the search direction. *On-line* (*recursive*) and *Off-line* (*batch*) methods are available [35]. Depending on the R_i the methods will be:

- *Gradient direction* $R_i = I$
- *Gauss-Newton direction*

$$R_i = H_i = \frac{1}{N}\sum_{t=1}^{N} \psi(t, \hat{\theta}^{(i)})\psi^T(t, \hat{\theta}^{(i)})$$

  where

$$\psi(t, \theta) = \frac{\partial}{\partial\theta}\hat{y}(t|\theta)$$

- *Levenberg-Marquard direction*

$$R_i = H_i + \delta I$$

- *Conjugate gradient direction*

$$R_i = V_N''(\hat{\theta}^{(i)})$$

It is very important to remember that the gradient of some of the parameters involved in the model must be generated dynamically because of the recursive structure of the model. The following example can illustrate the complexity of the problem.

3.6.1 Example

Given the nonlinear dynamic system defined as $\hat{y}(t) = f(\hat{y}(t-k), \theta)$ where $\hat{y}(t)$ is the output of the model, $\hat{y}(t)$ is the output of the model delayed k units of time and $\theta \in \Re$ is a parameter of model. The gradient of $\hat{y}(t)$ with respect to θ is given by the expression:

$$\frac{\partial \hat{y}(t)}{\partial \theta} = \frac{\partial f(\hat{y}(t-k),\theta)}{\partial \theta} + \frac{\partial f(\hat{y}(t-k),\theta)}{\partial \hat{y}(t-k)} \frac{\partial \hat{y}(t-k)}{\partial \theta} \tag{3.27}$$

replacing $\frac{\partial \hat{y}(t)}{\partial \theta}$ by $g(t)$ the expression will look like:

$$g(t) = \frac{\partial f(\hat{y}(t-k),\theta)}{\partial \theta} + \frac{\partial f(\hat{y}(t-k),\theta)}{\partial \hat{y}(t-k)} g(t-k) \tag{3.28}$$

the expression shows clearly that the gradient $\frac{\partial \hat{y}(t)}{\partial \theta}$ is given by a dynamic system. $\Diamond$

As an important contribution to the subject studied in this chapter, the gradient expressions for different types of fuzzy systems are presented in the appendix C.

3.7 Validation

For nonlinear system, validation methods are restricted to criteria in time but not in frequency. The most typical validation test is the prediction error [48] $\varepsilon(t) = y(t) - \hat{y}(t|\theta)$. Once this test is acceptable, further tests can be introduced. The most important is the *residual validation test.* This test is based in correlation analysis of the residuals. The test allows to detect information that can be modeled and has not been captured by the model. The test also detects unmodeled dynamics or bias in the estimation.
Billings et. al. [9] shown that the residual validation test applied for linear systems,

$$\left.\begin{array}{rcll} \phi_{\varepsilon\varepsilon}(\tau) & = & \delta(\tau) & \\ \phi_{\varepsilon u}(\tau) & = & 0 & \forall\tau \end{array}\right\} \tag{3.29}$$

(ϕ stands for the normalized cross-correlation) was not enough to detect biases or unmodeled dynamics. The reason is that the test can only detect the that the residuals are uncorrelated with themselves and with the inputs. The test simply establishes that no more "linear" relations can be found in the current data.
An extension of this test to nonlinear systems includes the correlation test between powers of the inputs and the residuals [9], for instance:

$$\left.\begin{array}{rcll} \phi_{\varepsilon\varepsilon}(\tau) & = & \delta(\tau) & \\ \phi_{\varepsilon^2 u^2}(\tau) & = & 0 & \\ \phi_{\varepsilon u}(\tau) & = & 0 & \\ \phi_{\varepsilon u^2}(\tau) & = & 0 & \\ \phi_{\varepsilon\varepsilon u}(\tau) & = & 0 & \forall\tau \end{array}\right\} \tag{3.30}$$

These results can be used in a constructive way. For instance, if the residuals are correlated with an input to the second power, the model can be increased to include directly this new input ($u^2(t)$) or the number of membership functions used in the input $u(t)$ can be increased.

As a final remark, it is important to remember that fuzzy models can have additional validation by comparing the linguistic rules with the knowledge of an expert. In dynamic systems, this is not always a simple task because of the tendency of human beings to represent expert knowledge in a static form.

3.8 Example

In this section, some of the discussed techniques are applied to the data of the example given by Box and Jenkins [11]. The process is a gas furnace with single input $u(t)$ (gas flow) and single output $y(t)$ (CO_2 concentration). The data set contains 296 data points; here only the last 290 points are used. For the selection of the regressors the Tree Method explained in section 3.4.1 was applied. Two different regressor evaluation methods were used the Lipschitz Quotients and the Regularity Criterion. The set of initial candidates for regressors were $\{u(t-1), u(t-2), u(t-3), u(t-4), u(t-5), u(t-6), y(t-1), y(t-2), y(t-3), y(t-4)\}$. The Lipschitz quotients (with $p = 6$) method was applied as evaluation of the quality of a set of regressors. The results of the input selection using the Tree Method are shown in figure 3.6. The diagram shows that for one input and one output candidate the selected regressor was $y(t-1)$, for two inputs one output $y(t-1)$ and $y(t-2)$, finally the selection process was stopped with three inputs $y(t-1), y(t-2)$ and $u(t-3)$.

Another input selection using the Tree Method and evaluating the models by means of the Regularity Criterion generated the results shown in figure 3.7. The RC index obtained for a model with one regressors makes us select $y(t-1)$, for a model with two regressors $y(t-1), u(t-3)$ and for the model with three regressors $y(t-1), u(t-3), y(t-4)$. The constructed models where the "smallest" fuzzy models described in section 2.3.1.

Observe the fact that the two methods proposed different regressors. The selections using Lipschitz Quotients proposes $y(y-1), y(t-2), u(t-3)$ and the selection using the Regularity Criterion $y(t-1), u(t-3), y(t-4)$. The reason for

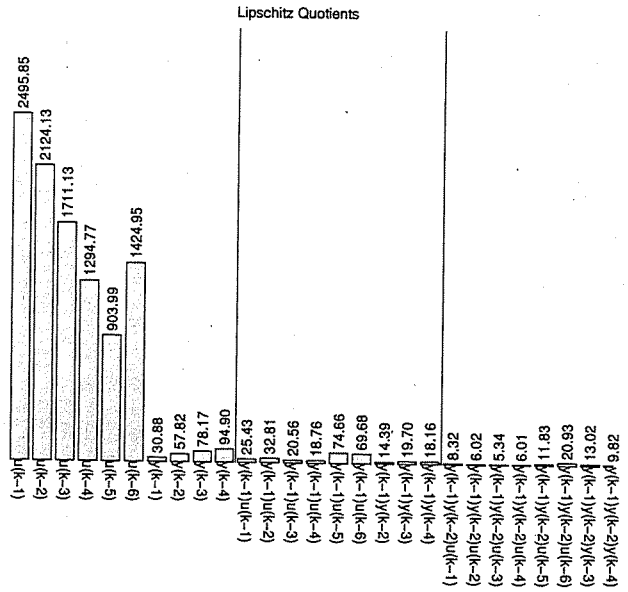


Figure 3.6: *Input Selection Using Lipschitz Quotients*

this discrepancy is that the quality of the regressors is measured with a complete different criterion. The Lipschitz Quotients method rewards the smoothness of the candidate function and the Regularity Criterion rewards the models with small prediction error.

Once the regressors are selected, the problem is to find the structure of the fuzzy system (number of rules, membership functions, consequences, etc). To extract the model structures clustering, projection and FuZion were used (see section 2.4.3). The clustering method applied was mountain clustering with $\alpha = 4$ and $\beta = 4$. For both sets of regressors 9 clusters were found and projected into the input space (see figures 3.8 and 3.9). The membership functions were reduced using the FuZion algorithm (see section 2.4.4) with $M = 10\%$ of the universe of discourse of the variable. The model using the regressors obtained using the Lipschitz Quotients $(y(t-1), y(t-2), u(t-3))$ used 4 triangular membership functions on each input as it is shown in figure 3.10. The total number of rules for this model is 64 ($4 \times 4 \times 4$). The model using the regressors obtained using the Regularity Criterion $(y(t-1), y(t-4), u(t-3))$ used 4 triangular membership

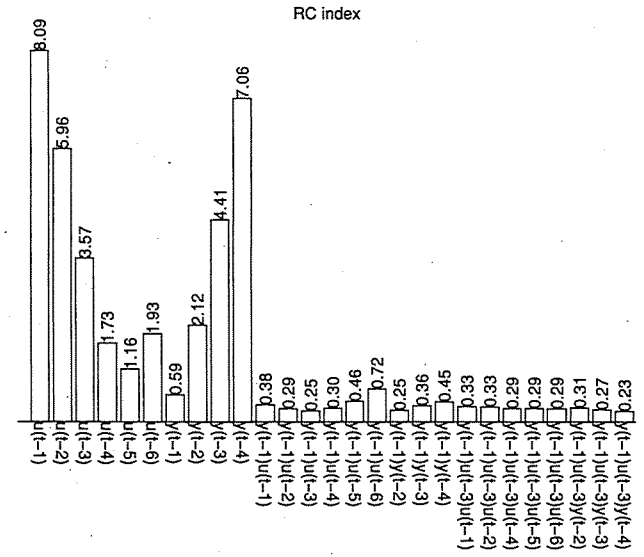


Figure 3.7: *RC index*

functions on the inputs $y(t-1)$ and $y(t-3)$ and 5 triangular membership functions on the input $y(t-4)$ as it is shown in figure 3.11. The total number of rules for this model is 80 ($4 \times 4 \times 5$).

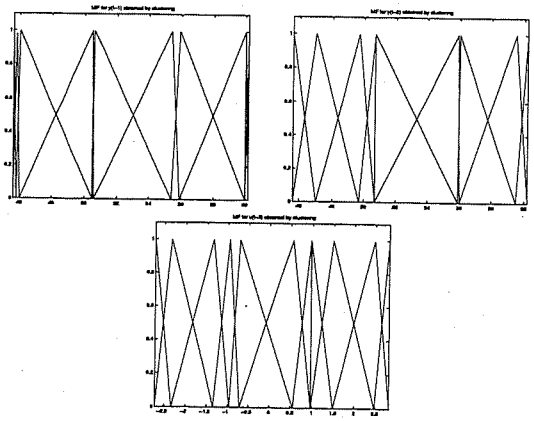


Figure 3.8: *Membership function after Clustering for the model using* $y(t-1), y(t-2), u(t-3)$.

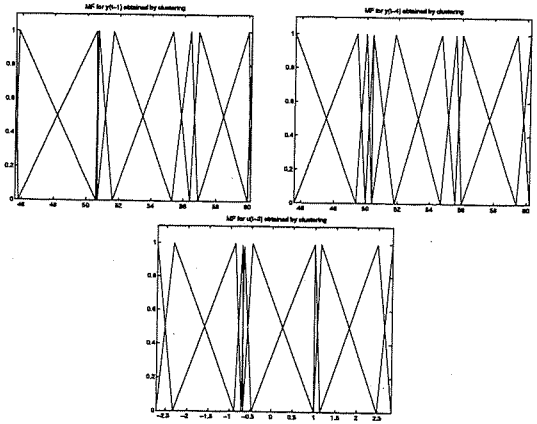


Figure 3.9: *Membership function after Clustering for the model using* $y(t-1), y(t-4), u(t-3)$.

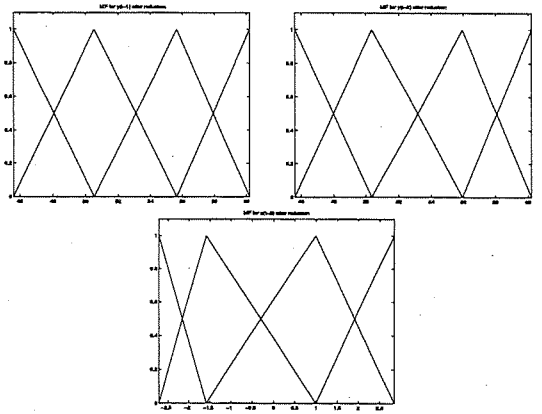


Figure 3.10: *Membership function after Clustering and FuZion for the model using* $y(t-1), y(t-2), u(t-3)$.

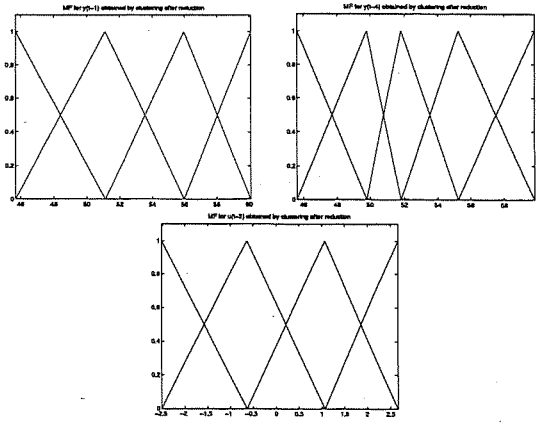


Figure 3.11: *Membership function after Clustering and FuZion for the model using* $y(t-1), y(t-4), u(t-3)$.

With each set of regressors two models were constructed: one NARX and one NOE. The consequences of the NARX model were calculated using the method described in section 2.3. The NOE model was calculated by initializing the system with the NARX solution and by running an optimization over the consequences, using the prediction error as a cost function. The results of the obtained models can be seen in the table 3.1 and in the figures 3.12, 3.13, 3.14 and 3.15.

Table 3.1: Example: Performance comparison between the models

Model Description	Number of Rules	Pred. Error whole data	Pred.Error validation data
NARX $[y(t-1), y(t-2), u(t-3)$	64	0.3189	0.5175
NOE $[y(t-1), y(t-2), u(t-3)$	64	0.8128	1.3812
NARX $[y(t-1), y(t-4), u(t-3)$	80	0.0648	0.1837
NOE $[y(t-1), y(t-4), u(t-3)$	80	0.0646	0.1741

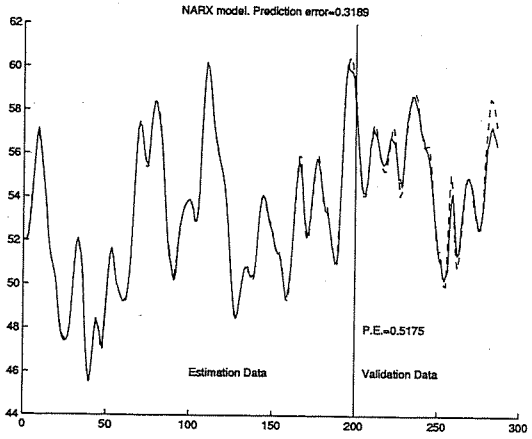


Figure 3.12: *Prediction error one step ahead for the NARX model with regressors* $y(t-1), y(t-2), u(t-3)$.

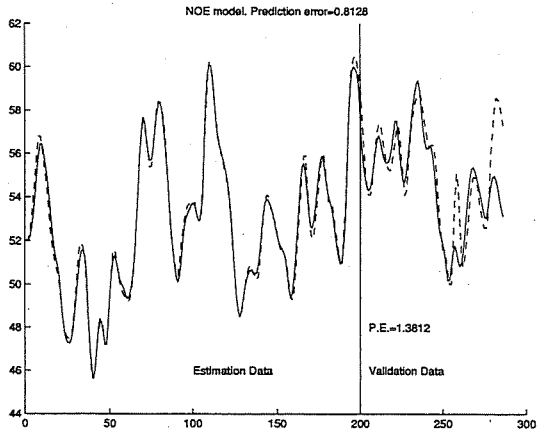


Figure 3.13: *Prediction error one step ahead for the NOE model with regressors* $y(t-1), y(t-2), u(t-3)$.

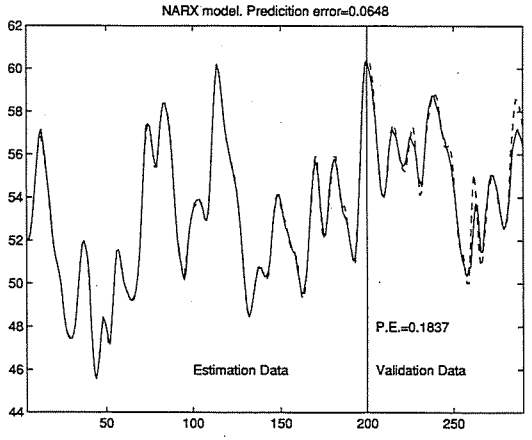


Figure 3.14: *Prediction error one step ahead for the NARX model with regressors* $y(t-1), y(t-4), u(t-3)$.

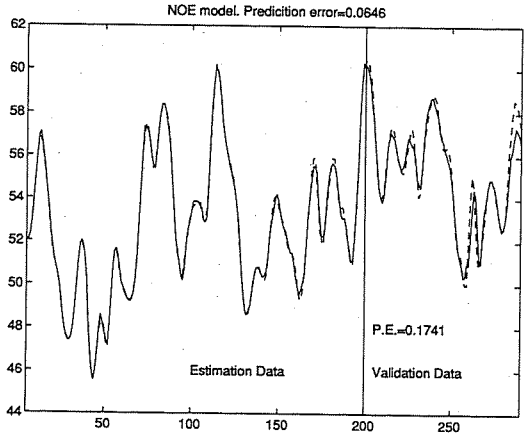


Figure 3.15: *Prediction error one step ahead for the NOE model with regressors* $y(t-1), y(t-4), u(t-3)$.

A final validation is obtained by using the correlation analysis. This is shown in the figures 3.16 and 3.17 which correspond to the NARX and the NOE models for the regressors $y(t-1), y(t-2), u(t-3)$ and figures 3.18 and 3.19 which correspond to the NARX and the NOE models for the regressors $y(t-1), y(t-4), u(t-3)$

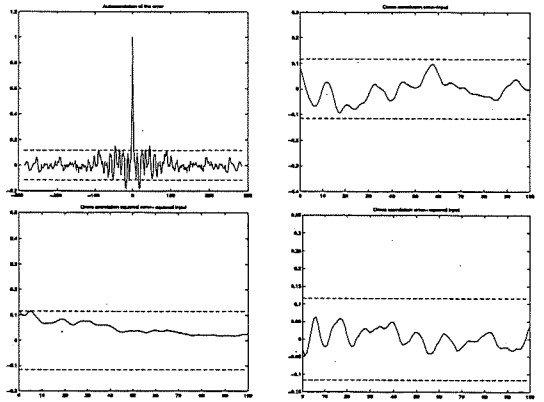


Figure 3.16: *Correlation analysis for the NARX model with regressors* $y(t-1), y(t-2), u(t-3)$.

From the pictures and the table it is clear that the models constructed with the regressors obtained using the Regularity Criterion have smaller prediction error. This is easily explained by the fact that the input selection based on the Regularity Criterion attempts to find the set of input which generates the smaller cross validation error. The regressors selected using the Lipschitz Quotients are chosen to improve the smoothness properties of the model but not the prediction error. The correlation analysis shows similar features for all the models, being acceptable according with this validation criterion.

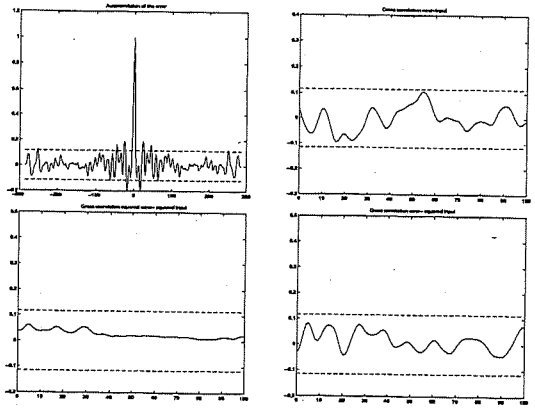


Figure 3.17: *Correlation analysis for the NOE model with regressors* $y(t-1), y(t-2), u(t-3)$.

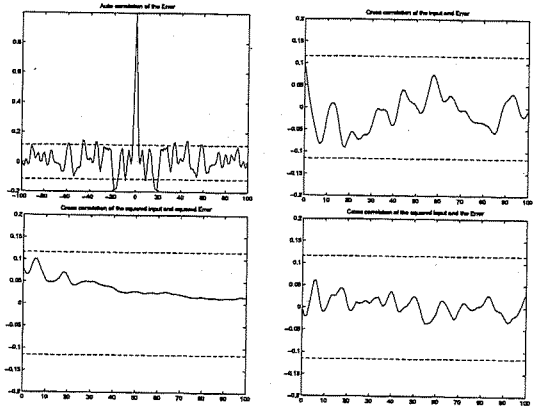


Figure 3.18: *Correlation analysis for the NARX model with regressors* $y(t-1), y(t-4), u(t-3)$.

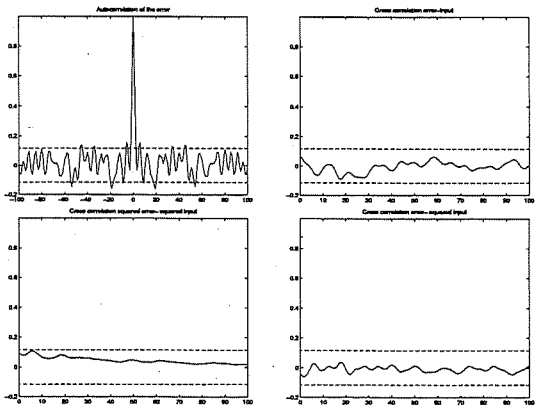


Figure 3.19: *Correlation analysis for the NOE model with regressors* $y(t-1), y(t-4), u(t-3)$.

3.9 Conclusions

This chapter has shown the use of fuzzy system for system identification purposes. The problems of regressors selection, experiment design, structure, parameter adjusting and validation have been studied.
The curse of dimensionality requires an extra effort for tasks such as regressor's selection. Efficient methods have been presented in this chapter to trade off the complexity of the calculation with the accuracy of the solution.
The optimization of the parameters is a task that involves the use of gradient descent techniques. The derivation of the dynamic systems that generates the gradients for the most common membership functions was presented.
In the validation methods issues about the use of the information provided by the correlation tests were discussed.
The main contribution of this chapter is the formalization of the use of fuzzy systems on nonlinear system identification. To achieve this goal the discussion has been oriented to analyze the advantages and disadvantages of the application of existing techniques to the class of fuzzy models. Some methods have been adapted and special attention was devoted to the derivation of formulas for parameter optimization using gradient descent techniques for dynamic fuzzy systems, which are included in the appendix C.

Chapter 4

Fuzzy Control

The use of fuzzy logic for control is a promising control strategy for its linguistic representation and its nonlinearity. On the other hand, a limitation arises from the complex description (from the mathematical point of view) of the nonlinearity.
The fuzzy controllers can be constructed in many different ways but it is possible to establish a classification between model based designed controllers and model-free. Model based controllers usually demand a "complete" description of the plant dynamics. The model free strategies are not completely model free, they are based on information extracted from simple experiments (relay experiments, etc.) or a heuristic model present in the designer's mind.
This chapter is included in this thesis as an overview of classical methods and it is by no means complete. This chapter fulfills two objectives. The first objective is to show to the reader some of the "classical" techniques to construct fuzzy controllers. The second objective is to report some minor contributions reflected in the publications [28] [31] [32] a in the appendixes D and E. The contribution in appendix D is a proof of the theorem which states that any linear controller with bounded states and inputs can be made into an exact equivalent fuzzy controller.

4.1 Model Free fuzzy control

4.1.1 Heuristic-Trial and Error Design

This methodology is probably the first technique used to design fuzzy controllers. The technique uses experience cumulated over years of manual control. The typical approach to construct these controllers has been the formulation of rule

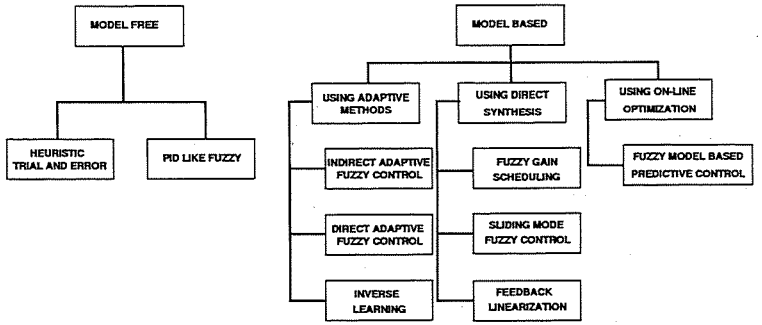


Figure 4.1: *Classification of the fuzzy controllers*

bases using the information provided by operator's manual. Most of the time, these controllers are used at high level as a kind of supervisory control where issues such as stability are not critical, since the stabilization of the plant is a task accomplished by the low level controllers. In many cases, the fuzzy controller is not used directly in the system in automatic mode but it is used as a support system for the operator. Successful applications of this technique have been reported in the areas of cement kiln control [58], boiler startup sequence, washing powder production [67], waste incineration and waste water treatment [79].

4.1.2 Design of PID-like fuzzy controllers

This PID-like fuzzy controller has been included in this model free class because a fuzzy controller can be designed using the same experiments designed to tune PID controllers in a model free basis or using simple models (step response models). The main idea behind this approach is that any PID with bounded input and output can be reproduced exactly by a fuzzy system (see theorem proof in appendix D). The design method proceeds as follows:

- Tune a PID controller using any of the traditional methods (Ziegler Nichols or Kappa-Tau form Aström and Hägglund. [4].
- Construct a Fuzzy Controller Equivalent to the tuned PID.
- Do further tuning of the Fuzzy Controller using heuristics.

Perhaps the most popular presentation of the low level (non-supervisory) fuzzy controller is the so called e, Δe controller. This is a fuzzy controller with two

inputs e=*error* and Δe=*change of the error*, and one output which is the control action u or Δu depending if the controller is PD or PI. One interesting characteristic of this description of the controller is the fact that a direct analogy can be established with the classical PD (Proportional Derivative) and PI(Proportional Integral) controllers. The fuzzy controller with a direct action will be analog to a PD controller and the fuzzy controller with incremental action will be analog to a PI controller (see figures 4.2 and 4.3).

Assuming five fuzzy sets (NL:Negative Large, NS:Negative Small, ZE:zero,

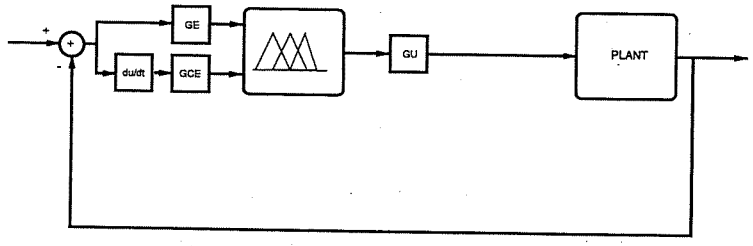


Figure 4.2: *Fuzzy PD controller*

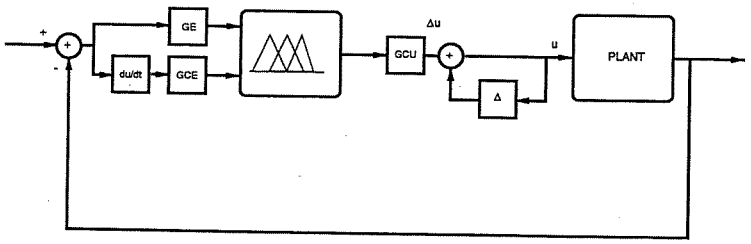


Figure 4.3: *Fuzzy PI controller*

PS:Positive Small and PL:Positive Large) on each input of the fuzzy controller and seven fuzzy sets in the output (the previous five plus PM:Positive Medium, NM:Negative medium, PVL: Positive Very Large and NVL: Negative Very Large) a typical rule base will look as follows:

e \ Δe	NL	NS	ZE	PS	PL
NL	PVL	PL	PM	PS	ZE
NS	PL	PM	PS	ZE	NS
ZE	PM	PS	ZE	NS	NM
PS	PS	ZE	NS	NM	NL
PL	ZE	NS	NM	NL	NVL

The distribution of the membership functions in the inputs can be very uniform as it is shown in the figure 4.4. Observe that the domain of the membership functions is distributed in the interval -100 to 100. Also the output membership functions have been scaled between -200 to 200 and are described using singletons. Using this rule base, triangular membership equally distributed and with 0.5 overlap, a fuzzy controller perfectly equivalent to a PI or PD controller can be build and the tuning parameters can be initially given by scaling factors at the input and the output of the controller. This will be a very safe method to tune a fuzzy controller by first replacing a PI or a PD controller by its equivalent fuzzy version and then a further tuning of the membership functions can improve the performance of the controller. The main guidelines of this tuning method were mentioned in [44].

The design of a PID controller can be achieved in different ways. One way is to build a fuzzy controller with three inputs the error (proportional action), the delta error (derivative action) and the sum of the error (integral action) (see figure 4.5). The inconvenience for such a controller is that the number of rules will grow and instead of 25 rules (for systems with five membership functions on each input), this controller will have 125 rules making the tuning task very difficult. A more efficient solution is to divide the controller in two controllers one which is the PD equivalent and another one that provides the integral action. It will reduce the number of rules to 30 (see figure 4.6).

The use of the "gain constants" can be avoided using a more complex initialization of the fuzzy system where the inputs are not scaled and the singletons of the output are calculated using its direct relation with the modal value of membership function in the antecedent, in a similar way as the rules are initialized in the training algorithm presented in section 2.3.1.

Further improvement will be done by trial and error, if there is no model of the plant, or via optimization, if there is a model available. Using this initialization, the controller will guarantee a performance at least as good as the linear PID controller.

Some of the typical actions in the trial and error are to increase the impact of the proportional action when the error is very large and reduce a little bit the integral action. These two actions tend to reduce the settling time and the overshoot.

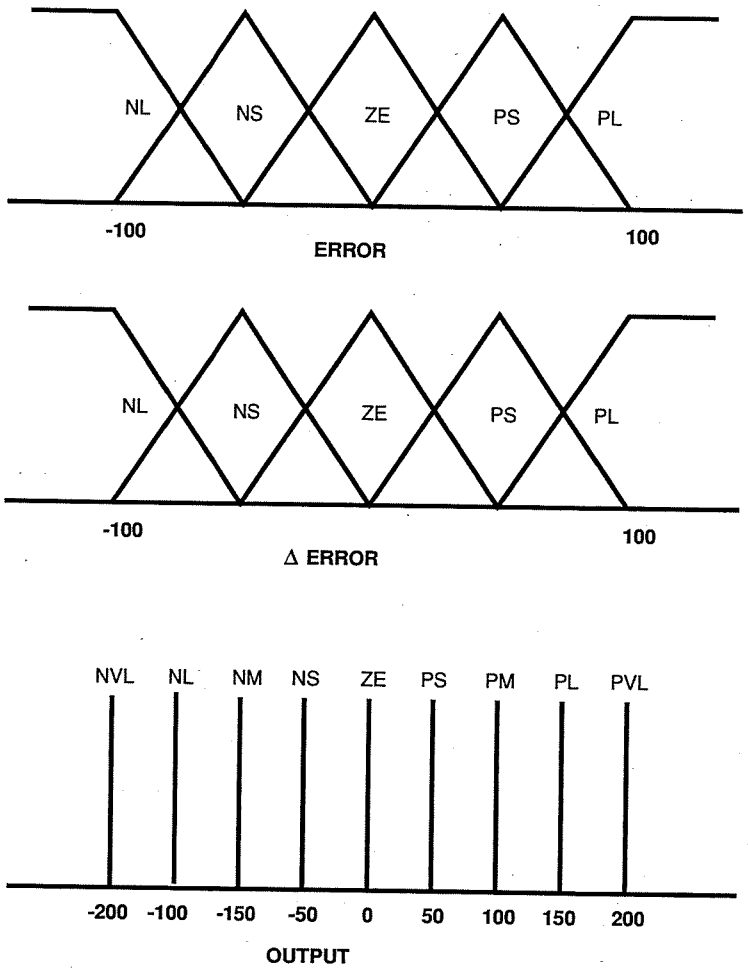


Figure 4.4: *Fuzzy System with linear relations*

4.2 Model Based Fuzzy Control

In this technique, three types of methods can be presented but most of the time these strategies appear mixed up to some level.

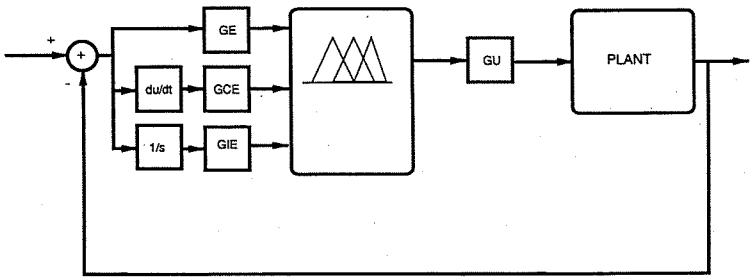


Figure 4.5: *Fuzzy PID controller*

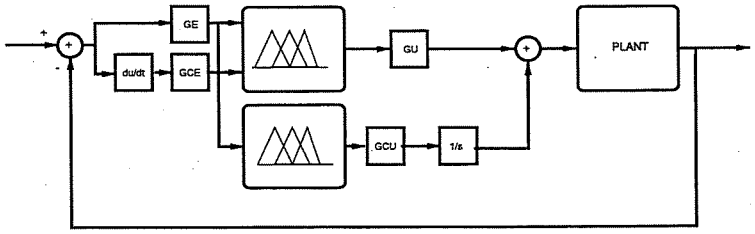


Figure 4.6: *Fuzzy PD+I controller*

- **Using Adaptive Methods** This design technique is an optimization based technique that uses the model of the plant and intensive simulations to optimize the parameters of the fuzzy controller. In most of the cases the optimizations are performed off-line, but in cases like the direct adaptive control the optimization can be done on-line.

- **Using Direct Synthesis** This design technique uses either the information given by the parameters of the model to construct the controller or the model itself.

- **Using On-Line Optimization** This technique uses the fuzzy model to predict the future behavior of the plant in a receding horizon with this information the future movements of the control actions are calculated via optimization.

4.2.1 Using Adaptive Methods

Inverse Learning

The fundament of this kind of controllers is the construction of an inverse model of the plant such that the controller generates an input to drive the state of the plant from the current state x_k to a desired state x^d_{k+n} [40]. For the application of this technique, it is assumed that the states of the plant are measurable. The dynamics of the plant are assumed to be discrete or at least sampled and represented by the function:

$$x_{k+1} = f(x_k, u_k) \tag{4.1}$$

where k represents the discrete time, x_k is the state and u_k is the input of the plant. The state of the plant for the time $k+N$ is given by:

$$x_{k+N} = \underbrace{f(f(f(x_k, u_k), u_{k+1}), \ldots, u_{k+N-1})}_{N\text{times}} \tag{4.2}$$

It is equivalent to say

$$x_{k+N} = F(x_k, U) \tag{4.3}$$

where F is a function representing the multiple composition of the function $f(.,.)$ and U is a vector with the input sequence $u_k, \ldots, u_{k+N-1}$. With this description, and assuming the invertibility of the function F an inverse map of the plant can be constructed as

$$U = G(x_k, x_{k+N}) \tag{4.4}$$

This function will generate the control sequence U to move the plant from the current state x_k to the state x_{k+N} in N steps. The existence of this inverse map is equivalent to the controllability condition for linear systems [41].
The existence of the map G does not guarantee the existence of an analytical closed form. Then a fuzzy system $\hat{G}$ will be used to approximate this map. The dimension of the map for a system of order n with one input will be $2n$ inputs and N outputs. It is clear that even for systems of low order the number of inputs will make the fuzzy system very big (in number of rules).
The use of the inverse model for control works as follows: the reference is given as the future state x^d_{k+N} and with the current state the inverse model $\hat{G}$ generates a vector U. The first entry of the vector is implemented and the function is evaluated once more at the next sampling time. If the reference is not known in advance the future reference will be replaced by the current reference generating a system which behaves as a pure delay system.
For practical purposes it is better to use an inverse model which generates only one value for the input sequence $N = 1$, the figure 4.7 shows the construction

of the controller.
This control technique is limited by the condition of invertibility of the plant and the fact that the minimization of the norm $||u_k - \hat{u}_k||$ does not guarantee the minimization of $||x_k - x^d_k||$. For practical purposes this method is very limited because it demands a full access to the states of the system. Moreover, the tuning demand that a large number of possible transitions from x_k to x_{k+1} are tested to guarantee a full coverage of the operating range of the controller. This problem is even worse when the order of the system is large. Already a fifth order model can generate some serious problems, because the controller will have $2 \times 5 = 10$ inputs and at least 1024 rules. In summary, this strategy can be applied only to low order (first or second order) invertible systems with full access to the state variables of the system.

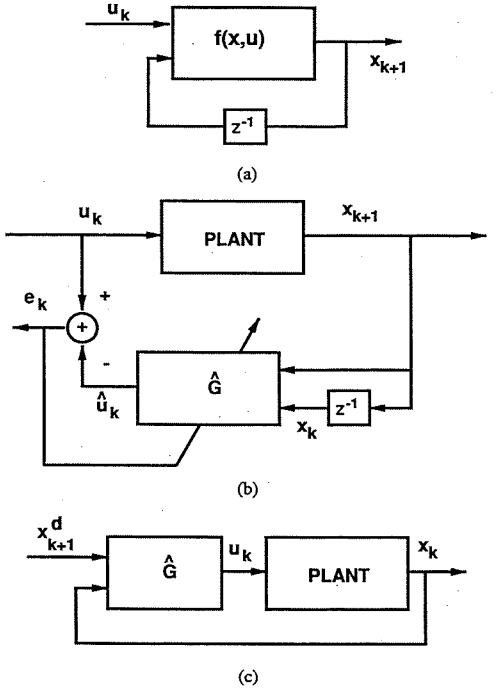


Figure 4.7: *Scheme for design of inverse controller (a) Plant (b)Learning Phase (c) Scheme of operation*

Specialized Learning or Direct Adaptive Fuzzy Control

This technique adjusts the parameters of the controller according to some performance measurement [72]. In this case, the controller is represented by a fuzzy system. Due to the multiple possible schemes the discussion will be restricted to two cases *state feedback* and *output feedback*.
The performance measure can be made against some performance specification (settling time, maximum overshoot, raise time) or a reference model (normally a linear one) which already includes the desired specifications. The general scheme is shown in figure 4.8.

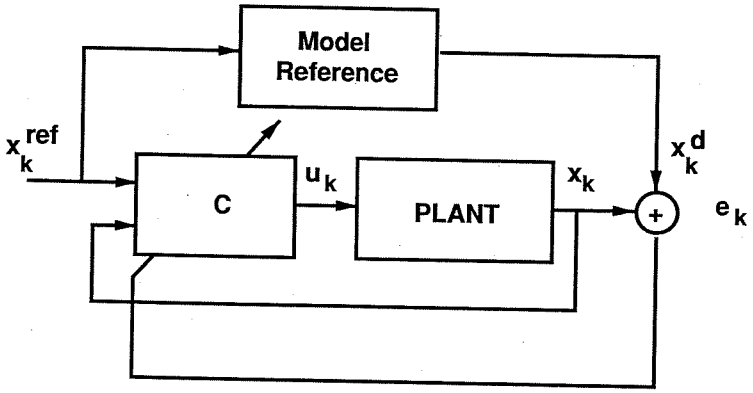


Figure 4.8: *Direct adaptive control*

The formulation for the *state feedback* is: Given a plant model

$$x_{k+1} = f(x_k, u_k) \tag{4.5}$$

the objective is to design a control system

$$u_k = g(x_k, x_k^{ref}, \theta) \tag{4.6}$$

this controller is a static nonlinear map with $2n$ inputs and a parameterization given by the vector θ. In this the system is a fuzzy system. The objective is that by changing the parameters θ the closed loop dynamics should mimic the dynamics of the system

$$x_{k+1}^d = \bar{f}(x_k^d, x_k^{ref}) \tag{4.7}$$

where x_k^d is the output of the reference model and x_k^{ref} is the reference. The

design can be formulated as:

$$\min_\theta J = \min_\theta \sum_{k=1}^{N} ||x_k^d - x_k|| \tag{4.8}$$

the closed loop expression for the system is:

$$x_{k+1} = f(x_k, g(x_k, x_k^{ref}, \theta)) \tag{4.9}$$

then the minimization can be written as:

$$\min_\theta \sum_{k=0}^{N-1} ||\bar{f}(x_k^d, x_k^{ref}) - f(x_k, g(x_k, x_k^{ref}, \theta))||^2 \tag{4.10}$$

if a gradient descent technique is going to be applied the adjustments will be directed by the gradient

$$\frac{\partial J}{\partial \theta} = 2 \sum_{k=0}^{N-1} -(x_{k+1}^d - x_{k+1}) \frac{\partial x_{k+1}}{\partial \theta} \tag{4.11}$$

where the gradient $\frac{\partial x_{k+1}}{\partial \theta}$ is generated by the dynamic system

$$\frac{\partial x_{k+1}}{\partial \theta} = \left[\frac{\partial f(x_k, u_k)}{\partial x_k} + \frac{\partial f(x_k, u_k)}{\partial u_k} \frac{\partial g(x_k, x_k^{ref})}{\partial x_k} \right] \frac{\partial x_k}{\partial \theta} + \frac{\partial f(x_k, u_k)}{\partial u_k} \frac{\partial g(x_k, x_k^{ref})}{\partial \theta} \tag{4.12}$$

It is clear that in order to apply this method a good model of the plant $f(.,.)$ is needed, to derive the expressions $\frac{\partial f(x_k, u_k)}{\partial x_k}$, $\frac{\partial f(x_k, u_k)}{\partial u_k}$ analytically or by means of numerical methods.
For the case of output feedback the plant model will be described by:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ y_k &= g(x_k) \end{aligned} \tag{4.13}$$

and the controller will be a dynamic fuzzy model represented by the equation:

$$\begin{aligned} z_{k+1} &= h(z_k, y_k, y_k^{ref} | \alpha) \\ u_k &= d(z_k, y_k, y_k^{ref} | \beta) \end{aligned} \tag{4.14}$$

where z_k are the states of the controller, y_k^{ref} is the reference input, α and β are the parameters of the functions $h(.,.,.)$ and $d(.,.,.)$. The closed loop system will be expressed as the dynamic system:

$$\begin{aligned} s_{k+1} &= m(s_k, y_k^{ref} | \theta) \\ u_k &= p(s_k, y_k^{ref} | \beta) \\ y_k &= q(s_k) \end{aligned} \tag{4.15}$$

where $s_k = \{x_k^T z_k^T\}^T$, $\theta = \{\alpha^T \beta^T\}^T$, $d(z_k, y_k, y_k^{ref}|\beta) = p(s_k, y_k^{ref}|\beta)$ and $q_{(s_k)} = g(x_k)$. Assuming a cost function

$$\min_{\theta} J = \min_{\theta} \sum_{k=1}^{N} ||y_k^{ref} - y_k|| + \lambda u_k^T u_k \tag{4.16}$$

the derivative will be given by:

$$\frac{\partial J}{\partial \theta} = 2 \sum_{k=0}^{N-1} -(y_k^{ref} - y_k) \frac{\partial y_k}{\partial \theta} + \lambda u_k \frac{\partial u_k}{\partial \theta} \tag{4.17}$$

with the terms $\frac{\partial y_k}{\partial \theta}$ and $\frac{\partial u_k}{\partial \theta}$ generated by the dynamic system:

$$\begin{aligned} \frac{\partial s_{k+1}}{\partial \theta} &= \frac{\partial m}{\partial s_k}\frac{\partial s_k}{\partial \theta} + \frac{\partial m}{\partial \theta} \\ \frac{\partial y_k}{\partial \theta} &= \frac{\partial q}{\partial s_k}\frac{\partial s_k}{\partial \theta} \\ \frac{\partial u_k}{\partial \theta} &= \frac{\partial p}{\partial s_k}\frac{\partial s_k}{\partial \theta} + \frac{\partial p}{\partial \theta} \end{aligned} \tag{4.18}$$

According to the parameters to be adjusted (membership functions, consequences or scaling values) the computation of the gradients is more complex. Figures 4.9 and 4.10 show the tuning of the values for a PID controller. The optimization procedure should evaluate a cost function that can include the optimization for disturbance rejection or for tracking a setpoint, or a compromise between the two.
Different parameters can be tuned using this method but according to the impact into the performance the priority will be to tune the scaling factors if there is any, then tune the consequences of the rules of the fuzzy system and finally if a very fine tuning is required, the parameters of the membership functions of the antecedents of the rules.
This method does not demand full access to the states, an output feedback strategy can be implemented if the system is controllable and observable..

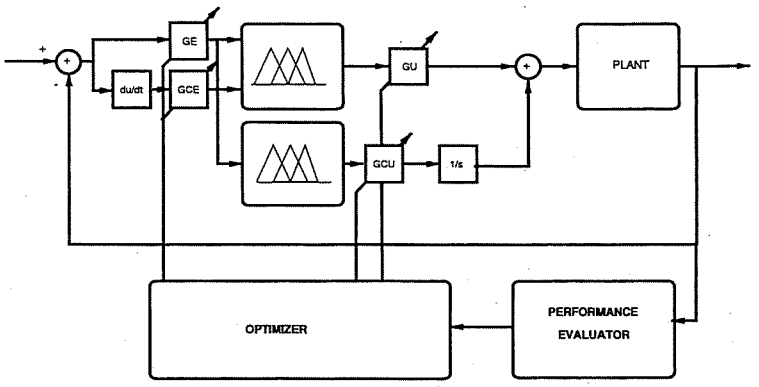


Figure 4.9: *Tuning of the gain constants using nonlinear optimization*

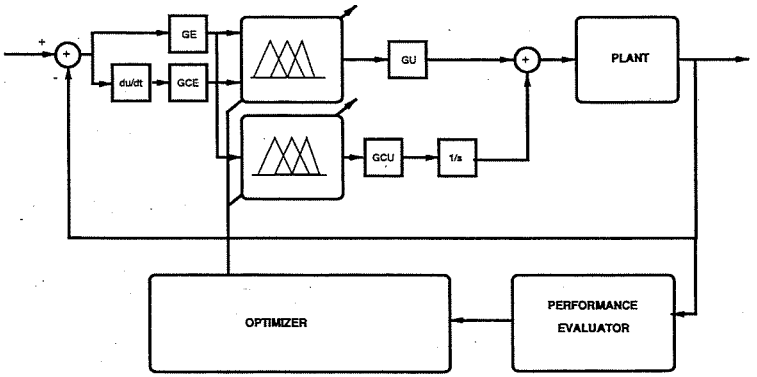


Figure 4.10: *Tuning of the internal parameter of the fuzzy controller (Membership functions, Consequences, etc)*

4.2.2 Using Direct Synthesis

Feedback Linearization

This methodology is applied to nonlinear systems of order n of the form:

$$\begin{aligned} x^{(n)} &= f(x,\dot{x},\ldots,x^{(n-1)}) + g(x,\dot{x},\ldots,x^{(n-1)})u \\ y &= x \end{aligned} \tag{4.19}$$

If the nonlinear functions $f(x)$ and $g(x)$ are known then a control signal u can be designed so that the nonlinearity is cancelled and the controller can be designed using linear techniques such as pole placement [82]. Using a control law,

$$u = \frac{1}{g(\mathbf{x})}(-f(\mathbf{x}) + y_{ref}^{(n)} + \mathbf{k}^T\mathbf{e}) \tag{4.20}$$

the control error is defined as $e = y_{ref} - y$, and the vector of state error is defined as $\mathbf{e} = \{e, \dot{e}, \ldots, e^{(n-1)}\}^T$. The vector $\mathbf{k} = \{k_n, \ldots, k_1\}^T$ defines the dynamics of the error and the dynamics of the closed loop. Introducing the control law to the plant will generate a dynamic system governed by:

$$\begin{aligned} x^{(n)} &= y_{ref}^{(n)} + \mathbf{k}^T\mathbf{e} \\ y &= x \end{aligned} \tag{4.21}$$

and the error dynamics will be:

$$e^{(n)} + k_1 e^{(n-1)} + \ldots + k_n e = 0 \tag{4.22}$$

From this equation it is clear that the closed loop dynamics will be governed by the dynamics determined by the components of the vector $\mathbf{k}$. A proper selection of the elements of $\mathbf{k}$ guarantees the stability and the convergence of y towards y_{ref}.

Fuzzy models can be used to represent the functions f and g, but there will be some modeling error. The function f will be approximated by the fuzzy inference system described by function

$$\hat{f}(\mathbf{x}|\theta_f) = \theta_f^T\sigma(\mathbf{x}) \tag{4.23}$$

where θ_f are the consequences of the rules and $\sigma(\mathbf{x})$ represents the inference process. Function g will be approximated by

$$\hat{g}(\mathbf{x}|\theta_g) = \theta_g^T\eta(\mathbf{x}) \tag{4.24}$$

where θ_g are the consequences of the rules and $\gamma(\mathbf{x})$ represents the inference process. If the mismatch between the function and its approximator is taken into account the error dynamics will be given by:

$$e^{(n)} = -\mathbf{k}^T\mathbf{e} + [\hat{f}(\mathbf{x}|\theta_f) - f(\mathbf{x})] + [\hat{g}(\mathbf{x}|\theta_g) - g(\mathbf{x})]u^* \tag{4.25}$$

where

$$u^* = \frac{1}{\hat{g}(\mathbf{x}|\theta_g)}(-\hat{f}(\mathbf{x}|\theta_f) + y_{ref}^{(n)} + \mathbf{k}^T\mathbf{e}) \tag{4.26}$$

The error dynamics can be written in vector form using:

$$\Lambda = \begin{bmatrix} 0 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1 \\ -k_n & -k_{n-1} & \ldots & \ldots & \ldots & \ldots & -k_1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{4.27}$$

as:

$$\dot{\mathbf{e}} = \Lambda\mathbf{e} + \mathbf{b}\{[\hat{f}(\mathbf{x}|\theta_f) - f(\mathbf{x})] + [\hat{g}(\mathbf{x}|\theta_g) - g(\mathbf{x})]u^*\} \tag{4.28}$$

the minimum approximation error will be defined as:

$$w = [\hat{f}(\mathbf{x}|\theta_f^*) - f(\mathbf{x})] + [\hat{g}(\mathbf{x}|\theta_g^*) - g(\mathbf{x})]u^* \tag{4.29}$$

where θ_f^* and θ_g^* represent the optimal approximation values of θ_f and θ_g. Using this representation the error dynamics can be written as:

$$\dot{\mathbf{e}} = \Lambda\mathbf{e} + \mathbf{b}\{[\hat{f}(\mathbf{x}|\theta_f) - \hat{f}(\mathbf{x}|\theta_f^*)] + [\hat{g}(\mathbf{x}|\theta_g) - \hat{g}(\mathbf{x}|\theta_g^*)]u^* + w\} \tag{4.30}$$

replacing the expressions for $\hat{f}(\mathbf{x}|\theta_f)$ and $\hat{g}(\mathbf{x}|\theta_g)$ the error dynamics become:

$$\dot{\mathbf{e}} = \Lambda\mathbf{e} + \mathbf{b}[(\theta_f - \theta_f^*)^T\sigma(\mathbf{x}) + (\theta_g - \theta_g^*)^T\eta(\mathbf{x})u^* + w] \tag{4.31}$$

The task is to define an adaptation law for θ_f and θ_g so that the tracking and the parameter errors are minimized. In [82] Wang presents the following procedure: Define the Lyapunov function:

$$V = \frac{1}{2}\mathbf{e}^TP\mathbf{e} + \frac{1}{2\gamma_1}(\theta_f - \theta_f^*)^T(\theta_f - \theta_f^*) + \frac{1}{2\gamma_2}(\theta_g - \theta_g^*)^T(\theta_g - \theta_g^*) \tag{4.32}$$

where γ_1 and γ_2 are positive constants and P is a positive definite matrix satisfying the Lyapunov equation:

$$\Lambda^TP + P\Lambda = -Q \tag{4.33}$$

where Q is an arbitrary positive definite matrix. The derivative of the Lyapunov function is given by:

$$\begin{aligned} \dot{V} &= -\frac{1}{2}\mathbf{e}^TP\mathbf{e} + \mathbf{e}^TP\mathbf{b}w + \frac{1}{\gamma_1}(\theta_f - \theta_f^*)^T[\dot{\theta}_f + \gamma_1\mathbf{e}^TP\mathbf{b}\sigma(\mathbf{x})] \\ &\quad + \frac{1}{\gamma_2}(\theta_g - \theta_g^*)^T[\dot{\theta}_g + \gamma_g\mathbf{e}^TP\mathbf{b}\eta(\mathbf{x})u^*] \end{aligned} \tag{4.34}$$

The minimization of the tracking error $\mathbf{e}$ and the parameter error is equivalent to the minimization of the Lyapunov function V. The adaptation law should guarantee that $\dot{V}$ is negative. The term $-\frac{1}{2}\mathbf{e}^T P\mathbf{e}$ is always negative. If the modeling task is well done the error w will be small and the last two terms will be made equal to zero, obtaining the following adaptation law:

$$\dot{\theta}_f = -\gamma_1 \mathbf{e}^T P \mathbf{b} \sigma(\mathbf{x}) \quad (4.35)$$

$$\dot{\theta}_g = -\gamma_2 \mathbf{e}^T P \mathbf{b} \eta(\mathbf{x}) u^* \quad (4.36)$$

Figure 4.11 shows an scheme of the proposed control system. It is important to remark that the sequence of design is:

1. Design an initial model of the plant $\hat{f}(\mathbf{x}|\theta_f), \hat{g}(\mathbf{x}|\theta_g)$ using the identification techniques depicted in previous section.
2. Design the vector $\mathbf{k}$ according to the desired behavior.
3. Connect the system in close loop and use the updating rule to update online the consequence values θ_f, θ_g of the rules.

The main advantage of this control strategy is that it makes possible to construct a controller directly using the model and the desired linear behavior. The main disadvantage is that it is only applicable to a limited set of continuous time nonlinear systems and since the controller is implemented in discrete time care must be taken during the implementation phase. Additionally, the scheme does not assume the presence of disturbances and relies on the adaptation of the parameters θ_f and θ_g to compensate the effect of the disturbances. The stability of the system is guaranteed but the tracking error will be proportional to the mismatch between the plant and the model.

Sliding Mode Fuzzy Control

This control strategy keeps some similarities with the feedback linearization technique; but its analysis include disturbances, making the problem more realistic. The objective is to "force" the system to behave like a linear system with dynamics described by the so-called "sliding surface"[76]. The control law is designed to steer the plant towards the "sliding surface". Once the system reaches the sliding surface, the controller guarantees that the closed loop system behaves according to the dynamics of the sliding surface. For this control system, the plant is assumed to have the form:

$$x^{(n)} = f(\mathbf{x}) + g(\mathbf{x})u + \tilde{d} \quad (4.37)$$

where $\mathbf{x}(t) = \{x, \dot{x}, \ldots, x^{(n-1)}\}^T$ is the state vector and $\tilde{d}(t)$ are disturbances with known upper bounds, and $f(\mathbf{x})$ and $g(\mathbf{x})$ are nonlinear functions describing

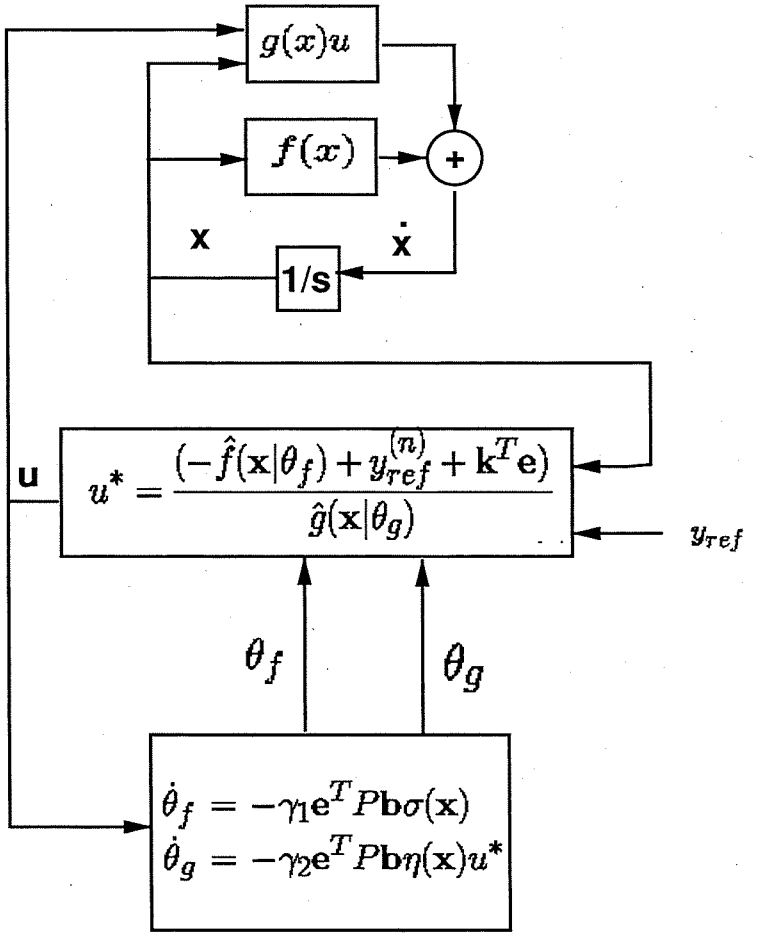


Figure 4.11: *Scheme for Adaptive Feedback Linearization*

the dynamics of the plant. Additionally the set ν_u is defined to represent the spectrum of unmodeled frequencies. The desired trajectory is given by $\mathbf{x}_{ref}(t)$

and the tracking error by

$$\mathbf{e}(t) = \mathbf{x}(t) - \mathbf{x}_{ref}(t) = \{e, \dot{e}, \ldots, e^{(n-1)}\}^T \tag{4.38}$$

The "sliding surface" is determined as:

$$s(\mathbf{x}, t) = (\frac{d}{dt} + \lambda)^{n-1} e = \sum_{k=0}^{n-1} \binom{n-1}{k} \lambda^k e^{(n-1-k)} = 0 \tag{4.39}$$

The stability of the "sliding surface" is guaranteed if $\lambda > 0$, which guarantees that the linear dynamics of the "sliding surface" is governed by poles placed in the negative real axis. The control problem requires the design of a control law to guarantee that the vector $\mathbf{e}(t)$ remains in the sliding surface $s(\mathbf{x}, t) = 0$ for $t \geq 0$. To construct this control law a Lyapunov function is proposed:

$$V = \frac{1}{2} s^2 \tag{4.40}$$

Observe that $V(0) = 0$ and $V > 0$ for $s > 0$. A sufficient condition for the stability of (4.37) is

$$\dot{V} = \frac{1}{2} \frac{d}{dt}(s^2) = s\dot{s} \leq -\eta |s| \tag{4.41}$$

with $\eta > 0$. From (4.41) the reaching condition can be derived,

$$\dot{s}\,\mathrm{sgn}(s) \leq -\eta \tag{4.42}$$

The first parameter that must be selected during the design phase is the parameter λ. Assuming $\nu_{u\min}$ as the lower bound of the unmodeled frequencies, then λ must be selected as:

$$\lambda << \nu_{u\min} \tag{4.43}$$

Calculating the time derivative of s we obtain:

$$\dot{s} = x^{(n)} - x_{ref}^{(n)} + \sum_{k=1}^{n-1} \binom{n-1}{k} \lambda^k e^{(n-k)} \tag{4.44}$$

replacing this expression and the plant model in the reaching condition (4.42)

$$\mathrm{sgn}(s) \left(f + g.u + \tilde{d} - x_{ref}^{(n)} + \sum_{k=1}^{n-1} \binom{n-1}{k} \lambda^k e^{(n-k)} \right) \leq -\eta \tag{4.45}$$

the sliding mode control law is defined as follows:

$$u = \hat{g}^{-1}(\tilde{u} - \hat{f}) \tag{4.46}$$

$$\tilde{u} = G.(\hat{u} - K(\mathbf{x}, t).\mathrm{sgn}(s)) \tag{4.47}$$

$$\hat{u} = x_{ref}^{(n)} - \sum_{k=1}^{n-1} \binom{n-1}{k} \lambda^k e^{(n-k)} \tag{4.48}$$

where $K(\mathbf{x}, t) > 0$ and $\hat{f}$ and $\hat{g}$ are estimates of the functions f and g. Observe that, so far, with the exceptions of the terms G and $K(\mathbf{x}, t)\mathrm{sgn}(s)$ the control strategy resembles the feedback linearization control strategy.
The term G is determined using the following bounds:

$$0 \leq \beta^{\min} \leq g\hat{g}^{-1} \leq \beta^{\max} \tag{4.49}$$

as:

$$G = (\beta^{\min}.\beta^{\max})^{-1/2} \tag{4.50}$$

Also a parameter β will be defined as:

$$\beta = \left(\frac{\beta^{\max}}{\beta^{\min}} \right)^{1/2} \tag{4.51}$$

The goal of the design now is to find a $K(\mathbf{x}, t)$ satisfying the reaching condition. Replacing the control law in the reaching condition,

$$\mathrm{sgn}(s)(\Delta f + (g\hat{g}^{-1}G - 1)\hat{u} + \tilde{d} - g\hat{g}^{-1}GK(\mathbf{x}, t)\mathrm{sgn}(s)) \leq -\eta \tag{4.52}$$

where $\Delta f = f - g\hat{g}^{-1}\hat{f}$. It is equivalent to

$$(\Delta f + (g\hat{g}^{-1}G - 1)\hat{u} + \tilde{d})\mathrm{sgn}(s) - g\hat{g}^{-1}GK(\mathbf{x}, t) \leq -\eta \tag{4.53}$$

This inequality is always true if

$$g\hat{g}^{-1}GK(\mathbf{x}, t) \geq |\Delta f + (g\hat{g}^{-1}G - 1)\hat{u} + \tilde{d}| + \eta \tag{4.54}$$

a stronger condition is given by:

$$g\hat{g}^{-1}GK(\mathbf{x}, t) \geq |\Delta f| + |(g\hat{g}^{-1}G - 1)||\hat{u}| + |\tilde{d}| + \eta \tag{4.55}$$

$g\hat{g}^{-1}$ can be substituted by its lower bound $\beta^{\min}$ and because $\beta^{-1} = \beta^{\min}G$ the condition for $K(\mathbf{x}, t)$ is given by:

$$K(\mathbf{x}, t) \geq \beta(|\Delta f| + (1 - \beta^{-1}).|\hat{u}| + |\tilde{d}| + \eta) \tag{4.56}$$

taking the upper bounds for

$$\begin{aligned} |\Delta f| &< \tilde{F} \\ |\hat{u}| &< \tilde{U} \\ |\tilde{d}| &< D \end{aligned}$$

then the condition for the sliding surface $s = 0$ to be a global domain of attraction is satisfied if:

$$K(\mathbf{x}, t) \geq \beta(\tilde{F} + (1 - \beta^{-1})\tilde{U} + D + \eta). \tag{4.57}$$

The steps to design the controller are:

1. Select λ from the lower bound of the unmodeled frequencies $\nu_{\min}$.
2. Derive $\hat{u}$ and its upper bound $\bar{U}$.
3. Find estimates $\hat{f}$ and $\hat{g}$ for f and g. To make this step Wang [82] suggested the use of fuzzy models for $\hat{f}$ and $\hat{g}$.
4. Derive G.
5. Determine η and the upper bounds $\bar{F}$ for $|\Delta f|$ and D for $|\bar{d}|$.
6. Compute $K(\mathbf{x}, t)$

An important advantage of this technique compared with the feedback linearization is that this technique takes into account disturbances and modeling mismatches. However, a significant disadvantage is that the control actions are very strong. Nevertheless the impact of this drawback can be reduced by the introduction of the so-called boundary layer (BL) near the sliding surface $s = 0$. The introduction of this boundary layer will affect the tracking accuracy. The width of the BL is described as 2Φ, $|s|$ is the distance of the state $\mathbf{e}$ to the sliding surface. A state $\mathbf{e}$ is located inside the BL if $|s| \leq \Phi$ and is outside if $|s| > \Phi$. The BL is introduced in the control law by replacing $\text{sgn}(s)$ by $\text{sat}(s/\Phi)$,

$$\begin{aligned} u &= \hat{g}^{-1}(\tilde{u} - \hat{f}) \\ \tilde{u} &= G[(\hat{u} - K(\mathbf{x}, t).\text{sat}(s/\Phi)] \end{aligned} \tag{4.58}$$

where the function $\text{sat}(z)$ is defined as

$$\text{sat}(z) = \begin{cases} z & \text{if } |z| < 1 \\ \text{sgn}(z) & \text{if } |z| \geq 1 \end{cases} \tag{4.59}$$

For practical purposes $K(\mathbf{x}, t)$ is reduced to a constant value. In [60] Palm et. al. proposed to replace the term $K(\mathbf{x}, t).\text{sat}(s/\Phi)$ with a fuzzy nonlinearity $K_{fuzz}(\mathbf{x}, t, s, \Phi)$ showing some improvements in the performance and reducing the chattering effect. In the same document some examples show the introduction of an integrator in this type of structure.
Figure 4.12(a) shows a fuzzy system with one input (s) representing the term $K(\mathbf{x}, t).\text{sat}(s/\Phi)$ when $K(\mathbf{x}, t)$ is a constant. Observe that the location of the membership functions will affect the shape of the function. In figure 4.12(b), it is clear that the separation of the membership functions from the center will cause the curve to flat in the middle. This will make the control action to react slowly for small errors or disturbances. The third plot shows a system where the slope is steep in the middle generating strong control actions for small errors. To sum up, in sliding mode control the fuzzy systems can be used to generate the estimates of the model ($\hat{f}$ and $\hat{g}$) and the control $K_{fuzz}(\mathbf{x}, t, s, \Phi)$ which forces the error vector $\mathbf{e}$ to remain in the sliding surface.

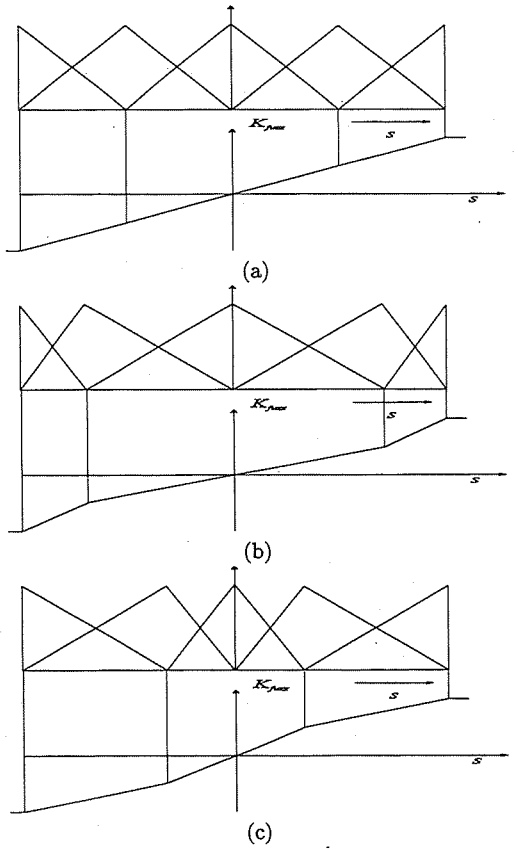


Figure 4.12: *Relation between the position of the membership functions and the shape of* $K_{fuzz}(\mathbf{x}, t, s, \Phi)$

Fuzzy Gain Scheduling

Gain scheduling has been used extensively on industrial applications including aircraft control. The advantage of this technique is the use of linear techniques to design controllers for non-linear systems by applying linearization around the different operating points. In this way several linear controllers are designed

and "connected" to the plant according to the current operating point. This design method is also known as the *the paradigm of parallel distributed compensation.* Fuzzy systems offer to this techniques the advantage of integrating in one mechanism the detection of the operating point and the interpolation among the different operating points, providing a smooth nonlinear control law. It has been proved the fuzzy controllers can be seen as smooth gain scheduling controllers [86].
To apply this compensation technique, the model of the plant is assumed to be given by a fuzzy system with rules of the form:

Rule i: IF $x_1(k)$ is μ_{i1} and ... and $x_n(k)$ is μ_{in}
THEN $x(k+1) = A_i x(k) + B_i u(k)$

where $x(k) = \{x_1(k), \ldots, x_n(k)\}^T$, $u(k) = \{u_1(k), \ldots, u_{Ni}(k)\}^T$ and $i = 1, \ldots, L$ with L is the number of rules. The dynamic system defined by the fuzzy system is given by:

$$x(k+1) = \frac{\sum_{i=1}^{L} \mu_i(x_k)[A_i x(k) + B_i u(k)]}{\sum_{i=1}^{L} \mu_i(x_k)} \tag{4.60}$$

A sufficient condition for stability of this system is given in [75] and is defined as: The equilibrium of the fuzzy system (4.60) with $u(k) = 0$ is asymptotically stable if there exists a common positive definite matrix P, so that

$$A_i^T P A_i - P < 0, \quad i = 1, 2, \ldots, L. \tag{4.61}$$

the search for such a P matrix is a complex task that cannot be solved efficiently by using analytic methods, but the use of numerical tools to solve this type Linear Matrix Inequalities [53][22] has simplified the problem. The search of P can be described in terms of a LMI as follows:

$$\begin{aligned} P &> 0 \\ A_i^T P A_i - P &< 0, \quad i = 1, 2, \ldots, L. \end{aligned}$$

This technique was applied to guarantee the stability of the system described in the appendix E.
A state feedback controller can also be designed using the paradigm of Parallel Distributed Compensation a state feedback controller can be described by a fuzzy system with rules of the form:

Rule i: IF $x_1(k)$ is μ_{i1} and ... and $x_n(k)$ is μ_{in}

THEN $u(k) = -K_i x(t)$

where $i = 1, \ldots, L$. The mathematical expression for the controller will be given by:

$$u(k) = \frac{-\sum_{i=1}^{L} \mu_i(x_k) K x(k)}{\sum_{i=1}^{L} \mu_i(x_k)} \tag{4.62}$$

the closed loop expression is given by:

$$x(k+1) = \frac{\sum_{i=1}^{L}\sum_{j=1}^{L} \mu_i(x_k)\mu_j(x_k)[A_i - B_i K_j]x(k)}{\sum_{i=1}^{L}\sum_{j=1}^{L} \mu_i(x_k)\mu_j(x_k)} \tag{4.63}$$

So the equilibrium of the closed loop system (4.63) is asymptotically stable if there exists a common positive definite P matrix such that:

$$[A_i - B_i K_j]^T P [A_i - B_i K_j] - P < 0 \tag{4.64}$$

for $\mu_i . \mu_j \neq 0$.
The state feedback matrices K_j can be calculated by solving the following LMIs (see proof:[88]).

$$\begin{pmatrix} Q & A_i Q + B_i Y_i \\ Q A_i^T + Y_i^T B_i^T & Q \end{pmatrix} > 0, \tag{4.65}$$

$$\begin{pmatrix} Q & \frac{A_i Q + B_i Y_j + A_j Q + B_j Y_i}{2} \\ \frac{Q A_i^T + Y_j^T B_i^T + Q A_j^T + Y_i^T B_j^T}{2} & Q \end{pmatrix} > 0, \; i < j \tag{4.66}$$

and the fuzzy state feedback matrices are constructed as $K_i = Y_i Q^{-1}$, $i = 1, 2, \ldots, L$. The design methods Using LMIs can be extended also to guarantee some decaying rate and even to account for uncertainties (further details see:[88]).

4.2.3 Conclusions and Future Perspectives

This chapter presented several control synthesis techniques for fuzzy controllers. The chapter was introduced as a reference to the "classical" ways to design fuzzy controllers. The description covered the widely used model free strategies characterized by their simplicity.

The heuristic methods are popular for first order systems and supervisory systems because they can provide an acceptable performance using the expert knowledge of the operators. The PID-like fuzzy controllers represent an interesting alternative to improve the performance of existing PID controllers with a small additional effort of tuning. Model based control techniques using adaptive methods have been described. These techniques require the use of a "good" plant model and the robustness of the controller obtained is not guaranteed. Also model based control techniques using direct synthesis have been presented. The feedback linearization technique is a technique limited to a certain class of nonlinear systems (affine nonlinear systems). This technique has been criticized for its lack of robustness. To overcome this drawback a stable adaptive control strategy has been presented: the sliding mode fuzzy control strategy. This strategy also demands a special structure of the nonlinear model. The strategy requires only a rough description of the model. This strategy is very robust and the performance can be improved by the use of fuzzy system in the design of the switching function. Finally, the fuzzy gain scheduling technique was presented. This design technique is quite "elegant" from the analytical point of view. The use of advanced algorithms for semidefinite programming facilitates the solution of the LMIs generated by the synthesis problem. More advanced synthesis techniques include the H^∞ criterion for disturbance rejection, robust synthesis and observer design by the same type of methods.
The stability analysis is still an open problem especially for the empirically designed controllers that are shown to be stable in practice (after trial and error) but no mathematical proof guarantees their stability.
The design of fuzzy controllers using LMIs has been considered conservative since a common P matrix has to be found. New alternatives have been proposed by Johansson et. al [45]. The relaxation of the problem is obtained by means of formulating a piecewise quadratic Lyapunov function.

Chapter 5

Predictive Control Based on Fuzzy Models

This chapter and the next one will be devoted to the presentation of a control technique that will exploit the capacity of fuzzy systems to represent nonlinear plants. The author considers the use of fuzzy models for predictive control a step further in the implementation of modern control techniques in the process industry. Predictive control based on linear models is a mature control technique with multiple applications in the process industry. The next natural step in this area is the development of predictive control based on nonlinear models. The use of controllers that take into account the nonlinearities of the plant implies an improvement in the performance of the plant by reducing the impact of the disturbances and by improving the tracking capabilities of the control system.
The inclusion of non-linear information demands the use of a parametric representation for the model of the plant. These chapters assume that the representation of the plant is given in terms of a fuzzy model. The use of fuzzy models together with the concept of predictive control is a promising technique because both techniques can be explained in simple terms to operators and commissioning engineers. Simplicity is paramount for the success of a control technique in the industrial environment.
Of course the use of nonlinear models increases the complexity of the problem and demands more information from the plant. One advantage of the use of fuzzy models is the fact that their complexity can be gradually increased as more information is gathered.
From the computational point of view, nonlinear predictive control can be quite demanding. The quest for shortcuts and simplification is considered one of the main research actions in this research subject. This chapter contains some ideas about ways to implement the simplification of the calculation. The chapter starts

presenting a solution for the case of single input-single output (SISO) systems without constraints. It is very important to remark the contributed method to estimate the step response used by the optimization algorithm. The method improves the estimation of the nonlinear information extracted from the model, improving the performance of the controller. In the second part of the chapter, the multivariable problem with constraints is studied. The method studied for SISO systems is extended for multivariable constrained systems, thereafter two new analysis are included where the structure of the Takagi-Sugeno fuzzy models is exploited to construct to types of predictive controllers. The included examples are based on realistic simulations of two industrial systems: a chemical reactor and a steam generator. The examples include analysis about the performance of the controllers.

5.1 The Predictive Control Strategy

The predictive control strategy is based on a receding horizon optimization, calculated on-line at each sampling time. The algorithm can be described as follows:

1. Sample the output of the plant
2. Use the model of the plant to predict its future behavior over a prediction horizon N_p when a control action is applied during a control horizon N_c.
3. Calculate the optimal control sequence $\{u(k), \ldots, u(k+Nc)\}$ that minimizes

$$\min_{u(k),\ldots,u(k+Nc)} J(u(k), y(k), w(k)) \tag{5.1}$$

subject:

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) \\ y(k) &= g(x(k), u(k)) \\ y_{min} &\leq y(k) \leq y_{max}, \forall k = 1, \ldots, Np \\ u_{min} &\leq u(k) \leq u_{max}, \forall k = 1, \ldots, Nu \\ \Delta y_{min} &\leq \Delta y(k) \leq \Delta y_{max}, \forall k = 1, \ldots, Np \\ & |\Delta u(k)| \leq \Delta u_{max}, \forall k = 1, \ldots, Nu \end{aligned}$$

where $J(.)$ is the "the cost function" and if it is quadratic it will be of the form:

$$J(u(k), y(k), r(k)) = \sum_{t=k}^{k+Np} (w(t) - y(t))^T Q (w(t) - y(t))$$

$$+ \sum_{t=k}^{k+Nc} (u(t)^T R u(t) + \delta u(t)^t S \delta u(t)) \quad (5.2)$$

where $x(k)$ represents the states of the system, $u(k)$ the inputs, $y(k)$ the outputs, the functions $f(.,.)$ and $g(.,.)$ represent the dynamic model of the plant, $w(k)$ is the reference signal, $\delta u(t) = u(t) - u(t-1)$, Q is positive definite matrix and R and S are positive semi-definite matrices.

4. Apply the input $u(k)$ and repeat the procedure at the next sampling time

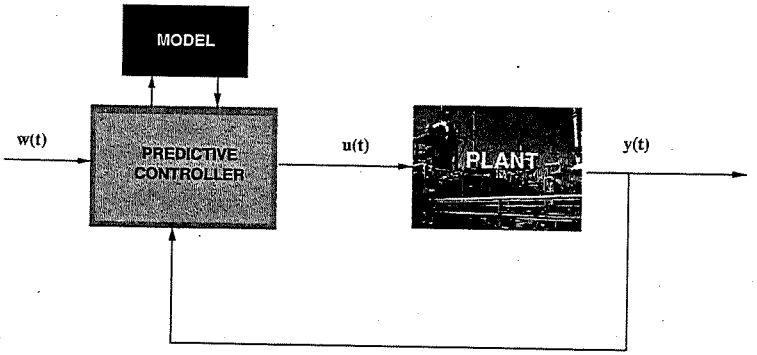


Figure 5.1: Predictive Controller

The problem presented in step 3 of the algorithm is a very complex optimization problem. This problem can be relaxed by assuming no constraints on the input nor on the outputs and a linear plant model. This problem has been extensively studied [65] [14] [33] [17]. The most important characteristic of this relaxation is the generation of a controller in a closed form, so that no optimization is solved on-line. A complete analysis of this strategy including stability and robustness issues has been presented in books such as [14] [70].
The complete problem without relaxation has been analyzed in previous studies [51] and [52] include some interesting studies. Probably the best overview published up today is [3] also interesting is [64]. The use of fuzzy techniques in predictive control was proposed for the first time by Yasunobu [85] on his

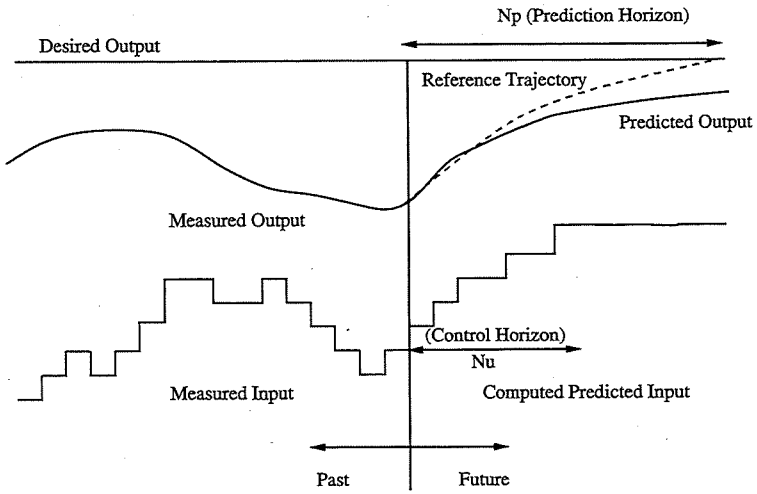


Figure 5.2: Prediction and Control Horizons

implementation of the control system for the Sendai Train. Other publications includes conference papers [66] and doctoral thesis [5] [18]. The analysis presented in this chapter assumes that the plant model included in the optimization (5.1) given by the non linear functions $f(.,.)$ and $g(.,.)$ will be represented by a fuzzy model. The control strategies presented in sections 5.2 and 5.3.2 can be applied to any type of fuzzy model. However, the control strategies presented in sections 5.3.3 and 5.3.4 demand the model to be a Takagi-Sugeno fuzzy model [29][23][26][27].

5.2 Unconstrained Nonlinear Predictive Control

In this section, the optimization problem of the unconstrained nonlinear predictive control with quadratic cost is analyzed. An approximate solution where the information given by a fuzzy model is used is presented in the following lines. The problem can be written as:

$$\min_{u(k),\ldots,u(k+Nc)} J(u(k), y(k), w(k)) = \sum_{t=k}^{k+Np} (w(t) - y(t))^T Q (w(t) - y(t)) + \delta u(t)^t R \delta u(t)) \quad (5.3)$$

subject:

$$\begin{aligned} x(k+1) &= f(x(k),u(k)) \\ y(k) &= g(x(k),u(k)) \end{aligned}$$

for the present case the plant model described by the functions $f(.,.)$ and $g(.,.)$ are parameterized by a fuzzy model. An alternative representation for the plant model is a NOE model represented as:

$$y(t) = \hat{y}(t) + n(t) = f(\hat{y}(t-1),\ldots,\hat{y}(t-m),u(t-1),\ldots,u(t-n)) + n(t) \quad (5.4)$$

where the function $f(.)$ corresponds to the fuzzy model and the noise model is given by

$$n(t) = \frac{C(z^{-1})}{D(z^{-1})}e(t) \quad (5.5)$$

where $e(t)$ is a *white noise* sequence. A typical choice of the noise model is:

$$n(t) = \frac{C(z^{-1})}{D(z^{-1})}e(t) = \frac{e(t)}{\Delta} \quad (5.6)$$

where $\Delta = 1 - z^{-1}$. This choice of the noise model will guarantee a zero steady state error for step disturbances and constant references, for details see [17] [33] [65].

An approximation of the predicted future output is given by:

$$y(t+k|t) = y_{\text{forced}}(t+k|t) + y_{\text{free}}(t+k|t) \quad (5.7)$$

where $y_{\text{forced}}(t+k|t)$ depends only on the future increments on the input and $y_{\text{free}}(t+k|t)$ depends only on the past inputs and outputs. In the present case the y_{free} sequence is given by:

$$y_{\text{free}}(t+k|t) = f(\hat{y}(t+k-1),\ldots,\hat{y}(t+k-m),u(t+k-1),\ldots,u(t+k-n)) + n(t+k|t) \quad (5.8)$$

with $u(t) = u(t+1) = \ldots = u(t+k-1) = u(t-1)$ which is equivalent to simulate the system assuming all the future inputs constant and equal to the last input value applied to the plant $u(t-1)$ and the sequence of predicted noise $n(t+k|t)$ is calculated assuming that $e(t+k|t) = 0$ for $k > 1$ (white noise assumption). The predicted y_{forced} will be given by

$$y_{\text{forced}}(t+k|t) = \sum_{i=0}^{k-1} g_i \delta u(t+k-i-1|t) \quad (5.9)$$

where g_i are the step response coefficients of the plant, calculated on the present operating point by simulating the step response on the model.
It is important to remark that the representation given by the equation (5.7) is not a linear representation because the term y_{free} is generated via nonlinear simulation and the coefficients g_i depend on the current operating point and the amplitude of the input signal. This is an important contribution to the analysis because it simplifies the analysis and solution of the problem without a significant degradation since most of the nonlinear information is preserved as it is shown by means of a simulation example in section 5.2.2
The optimization problem can be presented in matrix form as follows. First, the predictor is constructed as:

$$\mathbf{Y} = \mathbf{G}\Delta\mathbf{U} + \mathbf{Y}_{\text{free}} \quad (5.10)$$

where

$$\mathbf{Y} = \begin{bmatrix} y(t+1|t) \\ y(t+2|t) \\ \vdots \\ y(t+N|t) \end{bmatrix} \qquad \mathbf{G} = \begin{bmatrix} g_0 & 0 & \cdots & 0 \\ g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{N_u-1} & g_{N_u-2} & \cdots & g_0 \\ \vdots & \vdots & \vdots & \vdots \\ g_{N_p-1} & g_{N_p-2} & \cdots & g_{N_p-N_u-1} \end{bmatrix} \quad (5.11)$$

$$\Delta\mathbf{U} = \begin{bmatrix} \delta u(t|t) \\ \delta u(t+1|t) \\ \vdots \\ \delta u(t+N_u-1|t) \end{bmatrix}, \mathbf{Y}_{\text{free}} = \begin{bmatrix} y_{\text{free}}(t+1|t) \\ y_{\text{free}}(t+2|t) \\ \vdots \\ y_{\text{free}}(t+N_p|t) \end{bmatrix} \quad (5.12)$$

$$\begin{aligned} y_{\text{free}}(t+i|t) &= f(\hat{y}(t+i-1),\ldots,\hat{y}(t+i-m), \\ &\quad \hat{u}(t+i-1),\ldots,\hat{u}(t+i-n)) + n(t+i|t) \\ \text{with: } \hat{u}(k) &= \begin{cases} u(t-1) & \forall k > t-1 \\ u(k) & \text{otherwise} \end{cases} \end{aligned} \quad (5.13)$$

and with the reference vector described as:

$$\mathbf{W} = \begin{bmatrix} w(t) \\ w(t+1) \\ \vdots \\ w(t+N_p) \end{bmatrix} \quad (5.14)$$

the cost function can be written as:

$$J = \mathbf{E}^T Q \mathbf{E} + \Delta\mathbf{U}^T R \Delta\mathbf{U} \quad (5.15)$$

$$\begin{aligned} J &= (\mathbf{W}-\mathbf{Y})^T Q(\mathbf{W}-\mathbf{Y}) + \Delta\mathbf{U}^T R\Delta\mathbf{U} \\ &= (\mathbf{W}-\mathbf{G}\Delta\mathbf{U}-\mathbf{Y}_{\text{free}})^T Q(\mathbf{W}-\mathbf{G}\Delta\mathbf{U}-\mathbf{Y}_{\text{free}}) + \Delta\mathbf{U}^T R\Delta\mathbf{U} \end{aligned} \quad (5.16)$$

the minimization of the function J can be obtained by calculating the input sequence $\Delta\mathbf{U}$ so that $\frac{\partial J}{\partial \Delta\mathbf{U}} = 0$

$$\frac{\partial J}{\partial \Delta\mathbf{U}} = 2\mathbf{G}^T Q(\mathbf{Y}_{\text{free}} - \mathbf{W}) + 2(\mathbf{G}^T Q\mathbf{G} + R)\Delta\mathbf{U} = 0 \quad (5.17)$$

then the optimal sequence $\Delta\mathbf{U}$ is

$$\Delta\mathbf{U} = (\mathbf{G}^T Q\mathbf{G} + R)^{-1}\mathbf{G}^T Q(\mathbf{W} - \mathbf{Y}_{\text{free}}) \quad (5.18)$$

The input applied to the plant at time t is:

$$u(t) = u(t-1) + \delta u(t) \quad (5.19)$$

where $\delta u(t)$ is the first element of the vector $\Delta\mathbf{U}$.
Observe that the expression given by equation (5.18) is the same expression obtained for the Generalized Predictive Control [17]. However, in the GPC formulation the components involved in the calculation of the formula (5.18) come from linear model. In the present case the components introduced in the formula (5.18) are generated by the nonlinear model (a fuzzy model). A more rigorous formulation of (5.18) will be to represent the components as time variant matrices, as they are shown in the expression (5.20). At each sampling time the vectors $\mathbf{G}(t), \mathbf{Y}_{\text{free}}(t), \mathbf{W}(t)$ are reconstructed. The vector $\mathbf{Y}_{\text{free}}(t)$ is obtained by simulating the fuzzy model with the current input $u(t)$; the matrix $\mathbf{G}(t)$ is also reconstructed at each sampling time by using a method described in the next section.

$$\Delta\mathbf{U}(t) = (\mathbf{G}(t)^T Q\mathbf{G}(t) + R)^{-1}\mathbf{G}(t)^T Q(\mathbf{W}(t) - \mathbf{Y}_{\text{free}}(t)) \quad (5.20)$$

This time variant solution together with the estimation method for the matrix $\mathbf{G}(t)$ are considered the main constributions of this thesis to the solution of the problem of unconstrained nonlinear predictive control.

5.2.1 Estimation of the step response to construct $\mathbf{G}(t)$

The estimation of the step response is obtained by

$$g(k-1) = \frac{y_{\text{step}}(t+k|t) - y_{\text{free}}(t+k|t) - n(t+k|t)}{du} \quad (5.21)$$

where du represents the step size and

$$\begin{aligned} y_{\text{step}}(t+k|t) &= f(\hat{y}(t+k-1), \ldots, \hat{y}(t+k-m), \\ &\quad \hat{u}(t+k-1), \ldots, \hat{u}(t+k-n)) \\ \text{with: } \hat{u}(k) &= \begin{cases} u(k) & \forall k \le t-1 \\ u(t-1)+du & \forall k > t-1 \end{cases} \end{aligned} \quad (5.22)$$

The estimation of the step response is a very important element in the present algorithm, because the quality of the response of the system is determined by the accuracy of this estimation. The main difficulty to obtain an accurate estimate of the step response around some operating point is to select an appropriate value for the amplitude and the sign of the step du, because the "step response" of a nonlinear system is determined by the operating point, the size and the sign of the input step signal. A good value of du should satisfy the following requirements:

- It is very important that the value $u(t-1) + du$ should not saturate the "actuators" in the model. Even though the constraints are not taken into account for the optimization, the saturation constraints on the actuators must be taken into account. Failing to consider these saturation points can lead to an underestimated step response (with smaller gain) and this response will generate a response from the controller larger than needed thereby possibly generating unstable behavior.

- Also important is the fact that du should be "close" to the predicted $\Delta u(t)$. Since this value is only known after the optimization a good choice is to select $du(t) = \delta u(t|t-1)$ ($\delta u(t)$ obtained in the previous optimization step $t-1$), and it corresponds to the second element of the vector $\Delta\mathbf{U}(t-1)$. When the system reaches the steady state $\Delta\mathbf{U} \to 0$ it means that if we select du as stated before, the equation (5.22) will be badly conditioned. This imposes a minimum step size du_{min} in order to avoid this bad conditioning.

5.2.2 Predictive Control of a CSTR Using a Fuzzy Model

As example of the application of this control strategy, a plant representing a Continuous-Stirred Tank Reactor was chosen. The model of this plant was presented in [47] [38]. The model is described by the following differential equations:

$$\begin{aligned} \dot{C}_a(t) &= \tfrac{q}{v}(C_{a0} - C_a(t)) - k_0 C_a(t) e^{-\frac{E}{RT(t)}} \\ \dot{T}(t) &= \tfrac{q}{v}(T_0 - T(t)) + k_1 C_a(t) e^{-\frac{E}{RT(t)}} \\ &\quad + k_2 q_c(t)\left(1 - e^{-\frac{k_3}{q_c(t)}}\right)(T_{c0} - T(t)) \end{aligned} \quad (5.23)$$

The process describes the reaction where product A is converted into product B, the concentration $C_a(t)$ is the concentration of product A, $T(t)$ is the temperature of the mixture. The reaction is exothermic and it is controlled by a coolant flow whose rate is represented by $q_c(t)$. So, by changing the coolant flow, the temperature is controlled and by controlling the temperature, the concentration is controlled. C_{a0} is the inlet feed concentration, q is the process flow rate, T_0

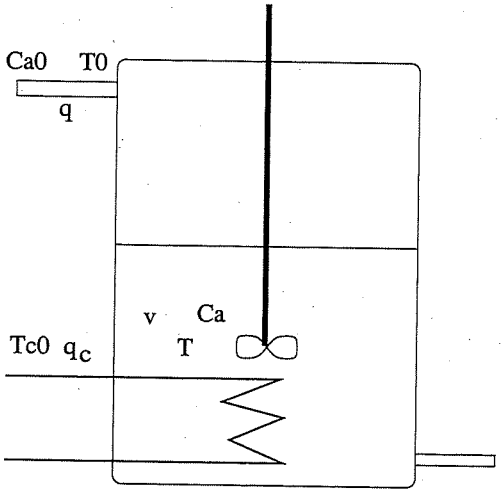


Figure 5.3: Continuous Stirred Tank Reactor

and T_{c0} are the inlet feed and coolant temperatures; all these values are assumed constant at nominal values. In the same way, k_0, E/R, v, k_1, k_2 and k_3 are thermodynamic and chemical constants. The numerical values of these parameters are given in table 5.1.

$$k_1 = -\frac{\Delta H k_0}{\rho C_p} \quad k_2 = \frac{\rho_c C_{pc}}{\rho C_p v} \quad k_3 = \frac{h_a}{\rho_c C_{pc}}$$

The nominal conditions for an product concentration $C_a = 0.1$ mol/l are:

$$T = 438.54\text{K} \qquad q_c = 103.411/\text{min}$$

5.2.3 Fuzzy identification of the CSTR

For the identification procedure a sampling time $Ts = 6$ sec. was chosen. A sequence of 7500 samples was generated; the first 5000 samples were used as training set and 2500 as validation set. Figure 5.4 shows the input and the output signal used for the identification (This set can be found in [19]).

The regressors were selected using the regularity criterion and the tree search

Parameter	Description	Nominal Value
q	process flow-rate	100 l/min
v	reactor volume	100 l
k_0	reaction rate constant	$7.2 \times 10^{10}\text{min}^{-1}$
E/R	activation energy	1×10^4K
T_0	feed temperature	350 K
T_{c0}	inlet coolant temp.	350 K
ΔH	heat of reaction	-2×10^5 cal/mol
C_p, C_{pc}	specific heats	1 cal/g/K
ρ, ρ_c	liquid densities	1×10^3 g/l
h_a	heat transfer coeff.	7×10^5 cal/min/K
C_{a0}	inlet feed concentration	1 mol/l

Table 5.1: The CSTR Parameters

method explained in chapter 3. The structure selected for the model is:

$$\hat{C}_a(k+1) = f(\hat{C}_a(k), \hat{C}_a(k-1), \hat{C}_a(k-2), q_c(k-1)) \qquad (5.24)$$

The identified model is a NOE model; the fuzzy model has three triangular membership functions distributed regularly on the universes of discourse. The number of rules extracted was $3\times3\times3\times3 = 81$ rules. The membership functions can be observed in figure 5.6 and the results of the validation in figure 5.5. The figure 5.5 shows the validation error on simulation. It can be observed that the quality of the model is very good and in fact the two signals appear overlapped; the error is plotted in the bottom part of the figure 5.5 in another scale to make it visible.

5.2.4 Implementation of the controller and performance results

The predictive controller was implemented using the following parameters: prediction horizon $N_p = 10$, control horizon $Nc = 2$, $Q = I^{N_p}$, $R = 10^{-4} I^{N_c}$, where I^n is an identity matrix such that $I^n \in \Re^{n\times n}$, $C(z^{-1}) = 1$ and $D(z^{-1}) = 1 - z^{-1}$, $du_{min} = 0.1(q_c^{max} - q_c^{min}) = 2$ l/min. Figure 5.8 shows the response of the system with a "stair" reference; each step has duration of 10 min and amplitude of 0.005 mol/l. The reference starts on 0.08 mol/l and it goes up to 0.12 mol/l. Perturbation steps of 0.005 mol/l are applied in the middle of each step. The reference and the perturbation signal are shown in the figure 5.7.

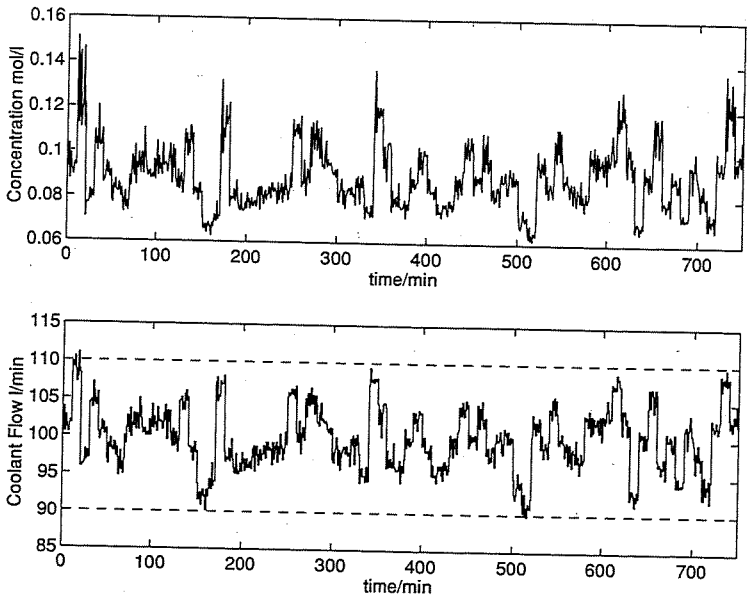


Figure 5.4: Identification Experiments

To compare the performance of the designed controller three other control strategies were implemented. A PID strategy optimized for the current reference. The parameters of the PID are:

$$K_p = 91.2 \quad T_i = 0.178\text{sec.} \quad T_d = 0.208\text{sec.}$$

The other control strategy was a Generalized Predictive Controller (GPC) designed with the same parameters as our predictive controller but using an Output Error linear model described as:

$$\hat{C}_a(k+1) = \frac{1.653 \times 10^{-4} z^{-1}}{1 - 2.43z^{-1} + 2.4z^{-2} - 1.189z^{-3} + 0.269z^{-4}} q_c(k) \qquad (5.25)$$

Finally, the "optimal" nonlinear control strategy is calculated using the Branch and Bound (B& B) optimization algorithm [46][56]. The Branch and Bound method is a discrete optimization algorithm. For this reason, the input range is discretized in 10 equidistant points. At each iteration the Branch and Bound algorithm will search in a space of 100 (10^{N_c}) possible solutions. The optimization

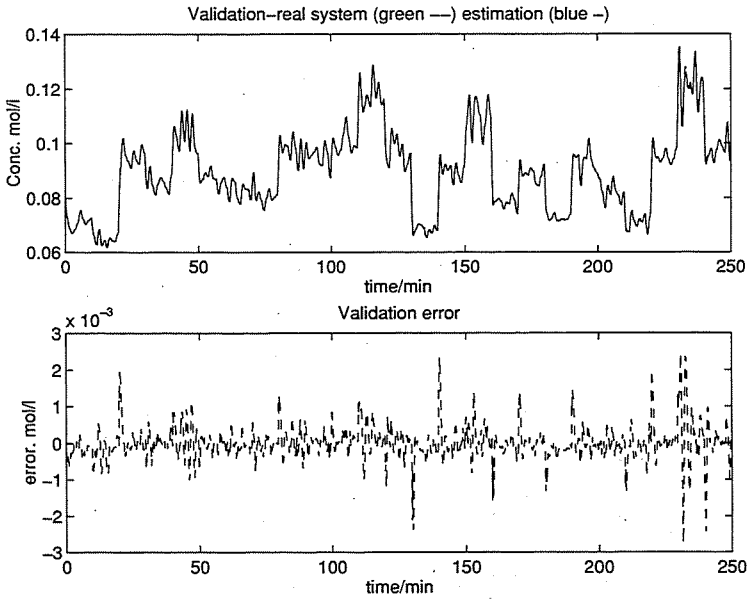


Figure 5.5: Validation Experiment

is refined with a local Newton optimization. The model used for the controller is the same model used to simulate the plant, so that there is no mismatch between the model and the plant. Figure 5.9 shows details of the performance comparison between the Predictive Controller based on the fuzzy model and the GPC and the PID. It is important to remark the degradation in the performance experimented by the systems with the linear GPC controller and the PID controller when the concentration is close to 0.12 mol/l. Observe that the PID controller is not able to stabilize the plant near this set point. In addition, the performance of the GPC is also degraded. The figures 5.10 and 5.11 show the comparison between the fuzzy model predictive controller and the "optimal" control strategy. Observe that, when the change in the operating point is big (between $t = 0, .., 2$), the two solutions are different, but when the changes are smaller the two solutions are almost the same, which indicates that the "suboptimal" solution is very close to the "optimal" solution. The execution time of

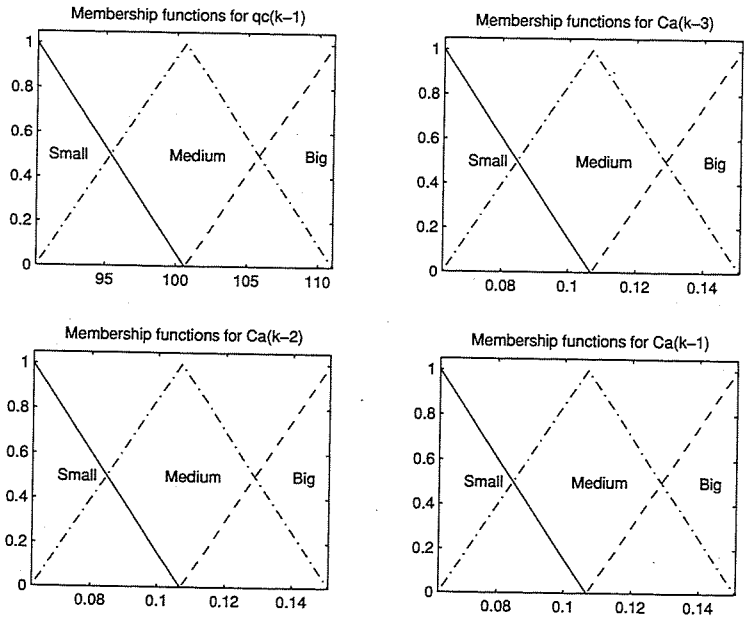


Figure 5.6: Membership function of the fuzzy model.

the algorithm is a very important parameter to evaluate its applicability. The execution time was evaluated using a workstation Sun Ultra-2 200Mhz, and the algorithm was implemented as a s-function in Simulink. Observe that this execution time gives the possibility of sampling frequencies of the order of 10 Hz.

Table 5.2: Execution time (msec.) vs. Horizons

Nu/Np	10	11	12	13	14	15
2	71.2	77.3	83.5	89.5	96.1	101.2
3	72	76.7	82.9	89.1	96.3	101.4
4	71.9	78.2	83.3	88.7	96	101.9
5	71.5	77.4	84	90.1	96.3	102.4

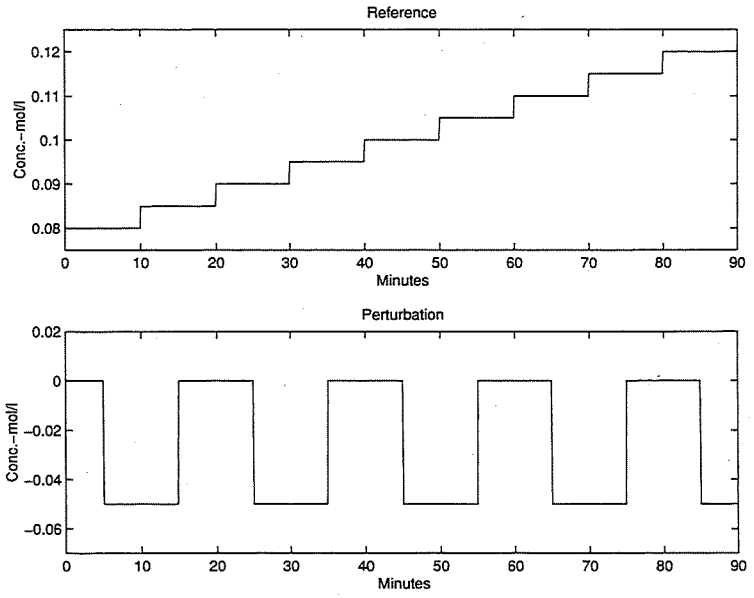


Figure 5.7: Reference and Perturbation

Other interesting comparisons with other Fuzzy Model Based Predictive control strategies using the present model has been presented in [23].

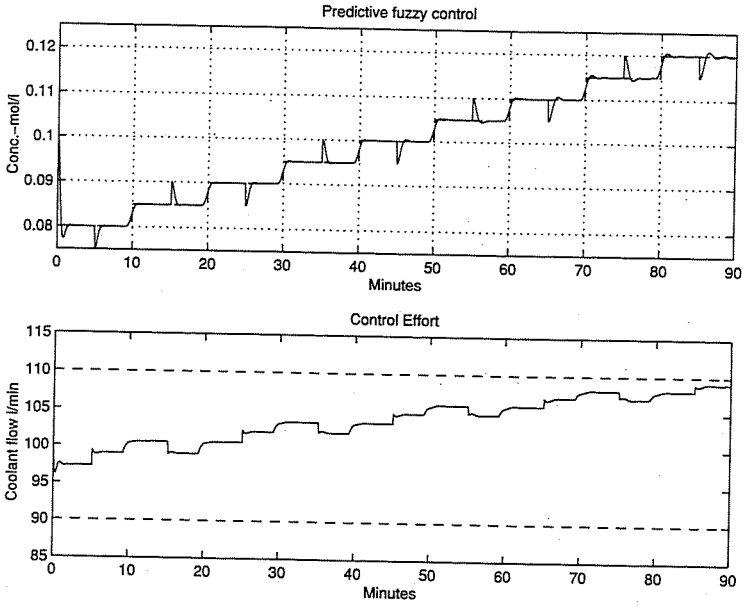


Figure 5.8: Response of the system

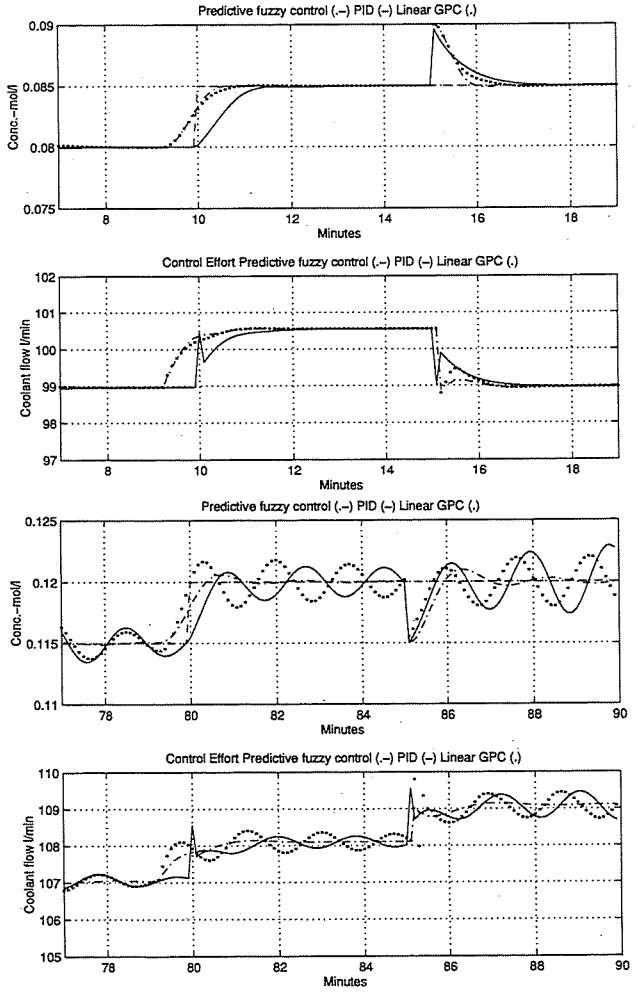


Figure 5.9: Detailed comparison for two different operating points

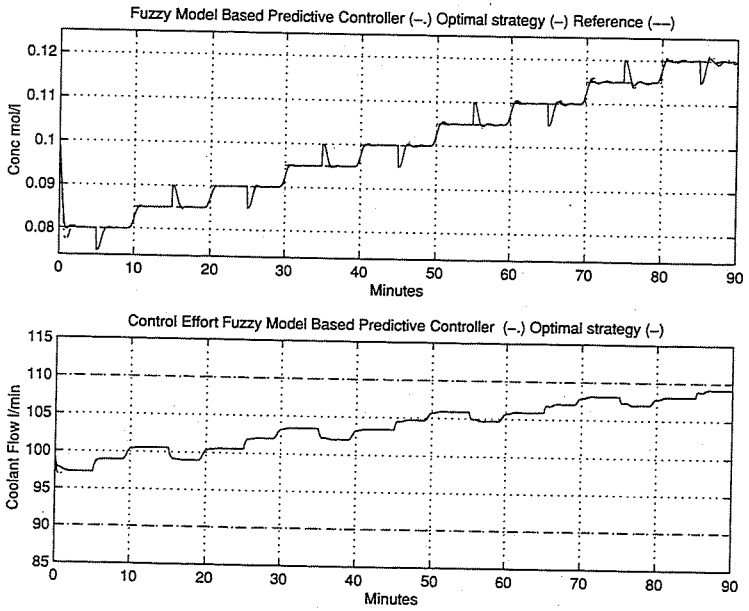


Figure 5.10: Comparison against the "optimal" strategy

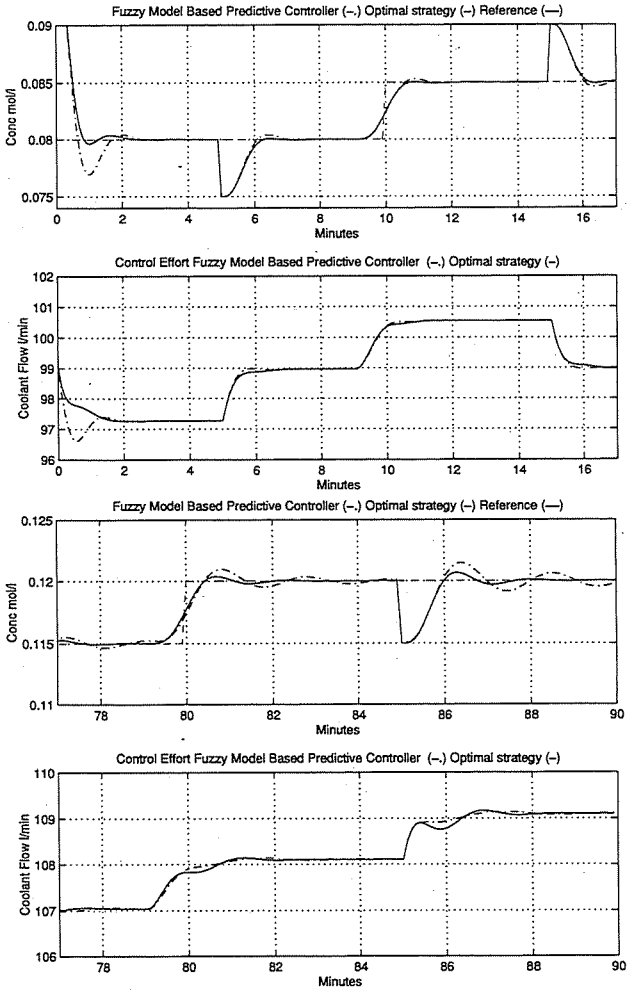


Figure 5.11: Detailed comparison against the "optimal" strategy for two different operating points

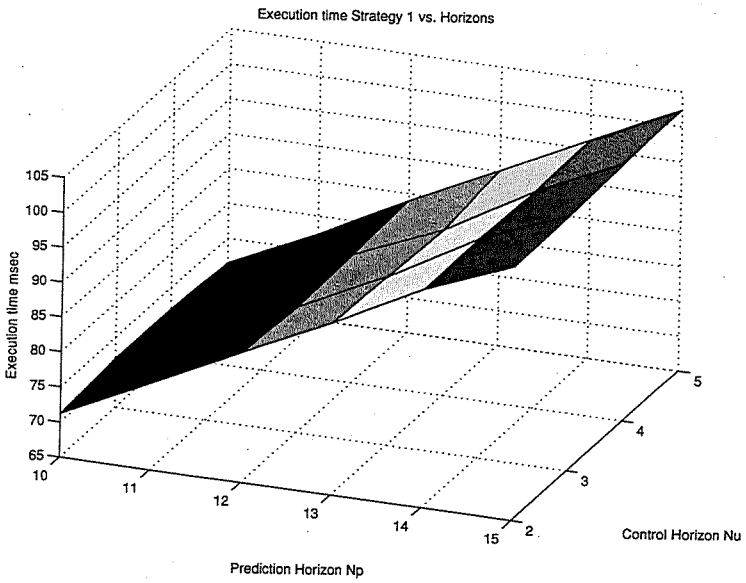


Figure 5.12: *Execution time (msec.) vs. Horizons*

5.3 Constrained Nonlinear Predictive Control

The introduction of constraints in the predictive control strategy is quite important. Constraints give more "realism" to the control actions by modeling saturation and slew rate constraints present in the actuators. It can also improve the formulation of a control system in terms of acceptable quality values (output constraints) and last but no least it guarantees maximal exploitation of the plant facility by driving the system closer to its maximum productivity without compromising the safety of the plant. In the next lines, three algorithms for Constrained Non-Linear Predictive Control based on fuzzy models are presented. Initially in section 5.3.1 the constrained non-linear predictive control problem is presented and a solution based on local linearization is derived. Sections 5.3.2, 5.3.3 and 5.3.4 present three algorithms to solve the constrained non-linear predictive control problem. The strategy presented in section 5.3.2 is an extension of the algorithm presented in previous section for the unconstrained case; this control strategy can be applied on any type of fuzzy models. The strategies presented in sections 5.3.3 and 5.3.4 exploit the structure of a Takagi-Sugeno fuzzy model to extract the information needed to solve the optimization problem. The strategy presented in section 5.3.4 converts the nonlinear problem into a linear time variant problem to simplify the optimization.

5.3.1 The Constrained Non-Linear Predictive Control Problem

The constrained Non-Linear Predictive Control Problem can be described as the problem of finding the "optimal" input sequence to move a dynamic system to a desired state, taking into account the constraints in the values of the inputs and the outputs. To achieve this objective an internal representation of the dynamical system is used in order to predict its future behavior (in this case a fuzzy model). The problem is solved at each sampling time and only the first movement of the calculated input is implemented. This guarantees disturbance rejection. Mathematically the problem can be written as:

$$\min_{u(k),\ldots,u(k+Nc)} J(u(k), x(k), r(k)) \tag{5.26}$$

subject:

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) \\ y(k) &= g(x(k), u(k)) \\ y_{min} &\leq y(k) \leq y_{max}, \forall k = 1, \ldots, Np \\ u_{min} &\leq u(k) \leq u_{max}, \forall k = 1, \ldots, Nc \end{aligned}$$

$$\begin{aligned} \Delta y_{min} \leq\ & \Delta y(k) \leq \Delta y_{max},\ \forall k = 1,\ldots,Np \\ & |\Delta u(k)| \leq \Delta u_{max},\ \forall k = 1,\ldots,Nc \end{aligned}$$

where $J(.)$ is the "the cost function" and it is of the form:

$$\begin{aligned} J(u(k),x(k),r(k)) &= \sum_{t=k}^{k+Np} (r(t)-x(t))^T Q(r(t)-x(t)) \\ &+ \sum_{t=k}^{k+Nc} (u(t)^T R u(t) + \delta u(t)^t S \delta u(t)) \end{aligned} \quad (5.27)$$

or

$$\begin{aligned} J(u(k),y(k),r(k)) &= \sum_{t=k}^{k+Np} (r(t)-y(t))^T Q(r(t)-y(t)) \\ &+ \sum_{t=k}^{k+Nc} (u(t)^T R u(t) + \delta u(t)^t S \delta u(t)) \end{aligned} \quad (5.28)$$

where $\delta u(t) = u(t) - u(t-1)$, Q is positive definite matrix and R and S are positive semi-definite matrices.

The solution to this problem is computationally very expensive because it involves the solution of a constrained nonlinear quadratic program. The exact solution to this problem ("finding the global minima") is very complicated and it cannot be done in real time. The next section shows two approaches to solve this problem in real time generating a suboptimal solution with a performance quite close to the "optimal solution".

The design of a predictive controller demands the construction of a predictor. In this section, the formulation is presented in state space form to generalize the problem in the multivariable form. The construction of the predictor in the state space representation can be conducted in the following form: Given the linearization of the system for the trajectory point $[x^*, u^*]$:

$$\begin{aligned} x(k+1) &= E + A(x(k)-x^*) + B(u(k)-u^*) \\ y(k) &= F + C(x(k)-x^*) + D(u(k)-u^*) \end{aligned} \quad (5.29)$$

where:

$$\begin{aligned} E &= f(x^*,u^*) \\ F &= g(x^*,u^*) \\ A &= \frac{\partial f(x,u)}{\partial x}|_{x^*,u^*} \end{aligned}$$

$$\begin{aligned} B &= \frac{\partial f(x,u)}{\partial u}|_{x^*,u^*} \\ C &= \frac{\partial g(x,u)}{\partial x}|_{x^*,u^*} \\ D &= \frac{\partial g(x,u)}{\partial u}|_{x^*,u^*} \end{aligned} \quad (5.30)$$

Using this simplification the prediction of the state and the output n steps ahead with a given input sequence U^n will be given by

$$\begin{aligned} x(k+n|k) &= \sum_{i=0}^{n-1} A^i L + A^n x(k) + [A^{n-1}BA^{n-2}B\ldots AB B] \begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+n|k) \end{bmatrix} \\ y(k+n|k) &= M + Cx(k+n|k) + Du(k+n|k) \end{aligned} \quad (5.31)$$

where:

$$\begin{aligned} L &= E - Ax^* - Bu^* \\ M &= F - Cx^* - Du^* \end{aligned}$$

In matrix representation the prediction can be seen as:

$$X^n = \underbrace{\Psi + \Phi x(k)}_{\text{Free Response}} + \underbrace{H\overline{U}^n}_{\text{Forced Response}} \quad (5.32)$$

$$Y^n = M^n + C^n X^n + D^n \underline{U}^n \quad (5.33)$$

$$Y^n = \underbrace{M^n + C^n[\Psi + \Phi x(k)]}_{\text{Free Response}} + \underbrace{C^n H\overline{U}^n + D^n\underline{U}^n}_{\text{Forced Response}}$$

$$Y^n = \underbrace{M^n + C^n[\Psi + \Phi x(k)]}_{\Gamma} + \underbrace{[C^n H\ 0] + [0\ D^n]}_{\Lambda} U^n \quad (5.34)$$

where

$$X = \begin{bmatrix} x(k+1|k) \\ x(k+2|k) \\ \vdots \\ x(k+n|k) \end{bmatrix}, \Psi = \begin{bmatrix} L \\ L+AL \\ \vdots \\ L+AL+\ldots+A^{n-1}L \end{bmatrix}, \Phi = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^n \end{bmatrix},$$

$$H = \begin{bmatrix} B & 0 & \ldots & 0 \\ AB & B & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{n-1}B & A^{n-2}B & \ldots & B \end{bmatrix}, M^n = \begin{bmatrix} M \\ M \\ \vdots \\ M \end{bmatrix}$$

$$U^n = \begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+n|k) \end{bmatrix}, \overline{U}^n = \begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+n-1|k) \end{bmatrix}, \underline{U}^n = \begin{bmatrix} u(k+1|k) \\ u(k+2|k) \\ \vdots \\ u(k+n|k) \end{bmatrix}$$

$$C^n = \begin{bmatrix} C & 0 & \dots & 0 \\ 0 & C & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C \end{bmatrix}, D^n = \begin{bmatrix} D & 0 & \dots & 0 \\ 0 & D & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & D \end{bmatrix}$$

when the predictor is constructed with an input sequence N_c and prediction horizon N_p the matrices of the predictor are converted to:

$$X = \begin{bmatrix} x(k+1|k) \\ x(k+2|k) \\ \vdots \\ x(k+N_p|k) \end{bmatrix}, \Psi = \begin{bmatrix} L \\ L+AL \\ \vdots \\ L+AL+\ldots+A^{N_p-1}L \end{bmatrix},$$

$$\Phi = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N_p} \end{bmatrix}, M^n = \begin{bmatrix} M \\ M \\ \vdots \\ M \end{bmatrix}$$

$$H = \begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-2}B & A^{N_p-3}B & \dots & \sum_{i=0}^{N_p-N_c-1} A^iB \\ A^{N_p-1}B & A^{N_p-2}B & \dots & \sum_{i=0}^{N_p-N_c} A^iB \end{bmatrix},$$

$$U^{N_u} = \begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+N_c|k) \end{bmatrix}, \overline{U}^n = \begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+N_c-1|k) \end{bmatrix}, \underline{U}^n = \begin{bmatrix} u(k+1|k) \\ u(k+2|k) \\ \vdots \\ u(k+N_c|k) \end{bmatrix}$$

$$C^{N_p} = \begin{bmatrix} C & 0 & \dots & 0 \\ 0 & C & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C \end{bmatrix}, \in R^{N_p n_o \times N_p n_s},$$

$$D^n = \begin{bmatrix} D & 0 & \dots & 0 \\ 0 & D & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & D \\ 0 & 0 & \dots & D \end{bmatrix}, \in R^{N_p n_o \times N_c n_i}$$

where n_s is the number of states, n_o the number of outputs and n_i the number of inputs.
Once this local linear representation has been obtained the optimization problem can be written as a quadratic program (QP).
Using the vector notation, the cost function will be written as:

$$J(U^n, Y^n) = (Y^n_{\text{ref}} - Y^n)^T Q (Y^n_{\text{ref}} - Y^n) + U^{nT} R U^n + (\Delta U^n - \bar{U}_{k-1})^T S (\Delta U^n - \bar{U}_{k-1}) \tag{5.35}$$

where

$$Y^n_{\text{ref}} = [y_{\text{ref}}(k+1), \ldots, y_{\text{ref}}(k+n)]^T$$

is the vector of the reference. Introducing (5.34) in (5.35):

$$J(U^n) = J_{\min} + 2[(\Gamma - Y^n_{\text{ref}})^T Q \Lambda - \bar{U}^T_{k-1} S \Delta] U^n + U^{nT} [\Lambda^T Q \Lambda + R + \Delta^T S \Delta] U^n \tag{5.36}$$

where

$$J_{\min} = Y^{nT}_{\text{ref}} Q Y^n_{\text{ref}} + \Gamma^T Q \Gamma - 2 Y^{nT}_{\text{ref}} Q \Gamma + \bar{U}^T_{k-1} S \bar{U}_{k-1}$$

is the minimum cost due to the reference and the free response and it cannot be modified by any control input. The constraints can be written as follows:

$$\begin{bmatrix} I^{(n_i N_c)} \\ -I^{(n_i N_c)} \\ I^{(n_i)} 0^{(n_i \times n_i N_c)} \\ -I^{(n_i)} 0^{(n_i \times n_i N_c)} \\ \Delta \\ -\Delta \\ \Lambda \\ -\Lambda \end{bmatrix} U^{N_c} \leq \begin{bmatrix} U^{N_c}_{max} \\ -U^{N_c}_{min} \\ \Delta U^{N_c}_{max} + \bar{U}_{k-1} \\ \Delta U^{N_c}_{max} - \bar{U}_{k-1} \\ \Delta U^{N_c}_{max} \\ \Delta U^{N_c}_{max} \\ Y^{N_p}_{max} - \Gamma \\ -Y^{N_p}_{min} + \Gamma \end{bmatrix} \tag{5.37}$$

where

$$I = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}, \Delta = \begin{bmatrix} I & 0 & 0 & \dots & 0 & 0 \\ -I & I & 0 & \dots & 0 & 0 \\ 0 & -I & I & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -I & I \end{bmatrix},$$

$$\bar{U}_{k-1}^{N_c} = \begin{bmatrix} u(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}, U_{max}^{N_c} = \begin{bmatrix} u_{max}(k) \\ u_{max}(k+1) \\ \vdots \\ u_{max}(k+N_c) \end{bmatrix}, U_{min}^{n} = \begin{bmatrix} u_{min}(k) \\ u_{min}(k+1) \\ \vdots \\ u_{min}(k+n) \end{bmatrix},$$

$$\Delta U_{max}^{N_c} = \begin{bmatrix} \Delta u_{max}(k) \\ \Delta u_{max}(k+1) \\ \vdots \\ \Delta u_{max}(k+N_c) \end{bmatrix}, Y_{max}^{N_p} = \begin{bmatrix} Y_{max}(k) \\ Y_{max}(k+1) \\ \vdots \\ Y_{max}(k+N_p) \end{bmatrix},$$

$$Y_{min}^{N_p} = \begin{bmatrix} Y_{min}(k) \\ Y_{min}(k+1) \\ \vdots \\ Y_{min}(k+N_p) \end{bmatrix}$$

where $I^{(n)} = I \in \Re^{n \times n}$ and $0^{(m \times n)} \in \Re^{m \times n}$. At each sampling time, this QP is solved with new parameters. The three algorithms presented in the following sections are based on this concept, but they differ in the way the parameters of the QP are obtained from the nonlinear model. It is important to remark that these methods generate a suboptimal solution to the original Nonlinear Quadratic Program (NLQP), because at each step a pseudo-linear approximation of the problem is used. This suboptimal solution will be very close to the exact solution of the NLQP if the nonlinearities of the plant are smooth such that along the prediction horizon the real trajectory is not very different from the trajectory generated by the proposed predictor. The algorithms also show some methods to improve the quality of the approximation.

5.3.2 Approach using estimated step response

This approach can be applied to any type of nonlinear model, whether it be a fuzzy model (Takagi-Sugeno or Mamdani), a Neural Network model, a Volterra series model, Wavelet model, gain scheduling, etc. The main idea is to solve also a QP by using an estimation of the step response as a linearized model. The formulation presented in this section is a generalization of the strategy presented for unconstrained predictive control. The advantage of this approach is that the linearization takes into account not only the current operating point (x^*, u^*) but also the direction of the control action in order to reach the next operating point $(x(k+1|k), u(k|k))$, by means of a similar reasoning as the one used in the equations (5.34) to (5.35). This comes from the fact that this method is based on the step response and the step response of a nonlinear system depends on the operating point, the amplitude of the step and the direction. In this method,

the estimation of the step response uses the "prior knowledge" gained from the previously calculated control actions.
The prediction of the state and the output n steps ahead can also be represented as:

$$\begin{aligned} x(k+n|k) &= \sum_{i=0}^{n-1} A^i L + A^n x(k) + \sum_{i=0}^{n-1} A^i B u_{k-1} + \\ &\quad + [\sum_{i=0}^{n-1} A^i B, \sum_{i=0}^{n-2} A^i B, \ldots, AB+B, B] \begin{bmatrix} \delta u(k|k) \\ \delta u(k+1|k) \\ \vdots \\ \delta u(k+n-1|k) \end{bmatrix} \\ y(k+n|k) &= M + Cx(k+n|k) + Du(k+n|k) \end{aligned} \tag{5.38}$$

where $\delta u(k|k) = u(k|k) - u(k-1)$, in matrix representation

$$X^n = \underbrace{\Psi + \Phi x(k) + G\bar{U}_{k-1}}_{\text{Free Response}} + \underbrace{G\overline{\Delta U}^{n-1}}_{\text{Forced Response}} \tag{5.39}$$

$$Y^n = M^n + C^n X^n + D^n \underline{U}^n \tag{5.40}$$

$$Y^n = \underbrace{M^n + C^n\Psi + C^n\Phi x(k) + ([C^nG\,0] + D_\Delta^n)\bar{U}_{k-1}}_{\text{Free Response}} + \underbrace{([C^nG\,0] + D_\Delta^n)\overline{\Delta U^n}}_{\text{Forced Response}}$$

$$Y^n = \underbrace{M^n + C^n\Psi + C^n\Phi x(k)}_{\Gamma} + \underbrace{([C^nG\,0] + D_\Delta^n)}_{\Xi}\bar{U}_{k-1} + \underbrace{([C^nG\,0] + D_\Delta^n)}_{\Xi}\overline{\Delta U^n}$$

$$Y^n = \underbrace{\Gamma + \Xi\bar{U}_{k-1}}_{\Sigma} + \Xi(\Delta U^n - \bar{U}_{k-1}) = \Sigma + \Xi(\Delta U^n - \bar{U}_{k-1}) \tag{5.41}$$

$$Y^n = \Gamma + \Xi\Delta U^n \tag{5.42}$$

where $M^n, X^n, \Phi, \Psi, C^n$ are equivalent to the ones previously described and the *free response* Γ is modified by the addition of the term $\Xi\bar{U}_{k-1}$,

$$G = \begin{bmatrix} B & 0 & \ldots & 0 \\ AB+B & B & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{n-1} A^i B & \sum_{i=0}^{n-2} A^i B & \ldots & B \end{bmatrix}$$

$$\overline{\Delta U^n} = \Delta U^n - \bar{U}_{k-1} = \begin{bmatrix} \delta u(k|k) \\ \delta u(k+1|k) \\ \vdots \\ \delta u(k+n|k) \end{bmatrix}, D_\Delta^n = \begin{bmatrix} D & D & 0 & \ldots & 0 \\ D & D & D & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ D & D & D & \ldots & D \end{bmatrix}$$

$$\Xi = \begin{bmatrix} CB+D & D & \dots & 0 & 0 \\ CAB+CB+D & CB+D & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum_{i=0}^{n-1} CA^iB+D & \sum_{i=0}^{n-2} CA^iB+D & \dots & CB+D & D \end{bmatrix}$$

Observe that the first column of the matrix Ξ corresponds to the *step response* of the system. So the response of the system can be described as the sum of the *free response* with the product of the Toeplitz matrix constructed with the "local" *step response* and the incremental input ΔU^n. The cost function (5.37) written with the response described by (5.41) is:

$$\begin{aligned} J(U^n) &= J_{\min} + 2[(\Sigma - \Xi\bar{U}_{k-1} - Y^n_{\text{ref}})^T Q\Xi\Delta - \bar{U}^T_{k-1} S\Delta]U^n + \dots \\ &\quad + U^{nT}[\Delta^T\Xi^T Q\Xi\Delta + R + \Delta^T S\Delta]U^n \end{aligned} \tag{5.43}$$

where

$$\begin{aligned} J_{\min} = Y^{nT}_{\text{ref}} QY^n_{\text{ref}} + (\Sigma - \Xi\bar{U}_{k-1})^T Q(\Sigma - \Xi\bar{U}_{k-1}) - \\ - 2Y^{nT}_{\text{ref}} Q(\Sigma - \Xi\bar{U}_{k-1}) + \bar{U}^T_{k-1} S\bar{U}_{k-1} \end{aligned}$$

is the minimum cost due to the reference and the free response. The QP program will be written using now the cost function (5.43) and the constraint definition (5.37).

In this procedure, the parameters of the QP are updated at each sampling time. The parameters that are updated at each sampling time are the matrices Σ and Ξ. The Σ matrix is the *free response* constructed by evolving the system form $k+1$ to $k+N_p$ as:

$$\begin{aligned} \bar{x}(k+1) &= f(\bar{x}(k), u^*) \\ \bar{y}(k) &= g(\bar{x}(k), u^*) \end{aligned} \tag{5.44}$$

where $f(.,.)$ and $g(.,.)$ is the description of the plant given by the fuzzy model, $u^* = u(k)$ is the current input of the system, $\bar{x}(k)$ and $\bar{y}(k)$ are respectively the "estimated" state and *free response*. Observe that this *free response* is different from the one calculated in previous version. The difference is that the *free response* of previous section is calculated by evolving the fuzzy system with $u^* = 0$. The Σ vector will look like:

$$\Sigma = \begin{bmatrix} \bar{y}(k+1|k) \\ \bar{y}(k+2|k) \\ \vdots \\ \bar{y}(k+N_p|k) \end{bmatrix} \tag{5.45}$$

The Ξ matrix is obtained by making step experiments. A step input is equivalent to $\overline{\Delta U}^n_{\text{step}} = [du, 0, \dots, 0]^T$, where du is a vector in $\Re^{n_i}$ where only one entry

(i) is different from zero. The $||du||$ is the size of the applied step. The *free response* of the system will be:

$$Y^n_{\text{free}} = \Gamma + \Xi\bar{U}_{k-1}$$

and the response with the step:

$$Y^n_{\text{step}} = \Gamma + \Xi\bar{U}_{k-1} + \Xi\overline{\Delta U}^n_{\text{step}}$$

operating:

$$Y^n_{\text{step}} - Y^n_{\text{free}} = \Xi\overline{\Delta U}^n_{\text{step}} = \begin{bmatrix} (CB+D)du \\ (CAB+CB+D)du \\ \vdots \\ (\sum_{j=0}^{n-1} CA^jB+D)du \end{bmatrix}$$

because only the i entry of du is different from zero, then the i column of Ξ will be obtained as,

$$\begin{bmatrix} (CB+D)_i \\ (CAB+CB+D)_i \\ \vdots \\ (\sum_{j=0}^{n-1} CA^jB+D)_i \end{bmatrix} = \frac{1}{du_i} \begin{bmatrix} (CB+D)_i du_i \\ (CAB+CB+D)_i du_i \\ \vdots \\ (\sum_{j=0}^{n-1} CA^jB+D)_i du_i \end{bmatrix} \tag{5.46}$$

Observe that only the first n_i columns (n_i is the number of inputs) are needed because Ξ is a block Toeplitz matrix.

An important remark is that the amplitude and the direction of the du_i element will determine the obtained step response. This is due to the fact that the step response of a nonlinear system is a function of the operating point, the amplitude and the direction of the step. Our estimation had already dealt with obtaining the step response at the operating point but the amplitude of the step and its direction have not been determined. A good method to determine the size and direction of du_i is to apply the most likely input that will be applied by the control action. This input will be obtained from the solution obtained in the previous optimization. So the most suitable value for du_i at time k will be:

$$du^k_i = \delta u(k|k-1)_i \tag{5.47}$$

observe that, when a steady state is reached, then $u(k|k-1)_i \approx u(k-1)_i$, making the value of $du_i \approx 0$ and the estimation of Ξ a badly conditioned operation. For this reason, a du_{min} vector should be defined with the minimum value of du, so that the estimation of Ξ is reliable. To sum up, the control algorithm will be:

Algorithm

At each sampling time

1. Read the current output of the system and update $y(k)$
2. With the input $u(k-1)$ calculate the *free response* Σ using the equation (5.44).
3. Calculate the entries of the du vector for the n_i step experiments using equation (5.47).
4. If $du_i < du^i_{min}$ make $du_i = du^i_{min}$.
5. Make the step experiments and calculate the Ξ matrix
6. With Ξ and Σ construct the matrices for the cost (eq.(5.43)) and the constrains (eq.(5.37)) of the QP problem.
7. Solve the QP problem.
8. Apply only the first control action $u(k|k)$.

The main contributions of this algorithm besides the formulation and the simulation of the *free response* using a fuzzy model are the derivation of the method to estimate the step response using the fuzzy model and the "prior knowledge" (amplitude and direction) given by the previous control step.

5.3.3 Approach using Takagi-Sugeno Fuzzy Models

This approach exploits the properties of Takagi-Sugeno fuzzy models with rules of the form:

$$\text{If } x(k) \text{ is } \mathcal{A}_i \text{ AND } u(k) \text{ is } \mathcal{U}_i \text{ THEN } \begin{array}{l} x(k+1) = E_i + A_i x(k) + B_i u(k) \\ y(k) = F_i + C_i x(k) + D_i u(k) \end{array}$$

The evaluation of this model will generate at each step not only the output $y(k)$, but also a linear models of the form:

$$\begin{aligned} x(k+1) &= E(k) + A(k)x(k) + B(k)u(k) \\ y(k) &= F(k) + C(k)x(k) + D(k)u(k) \end{aligned} \tag{5.48}$$

where

$$E(k) = \sum_{i=0}^{L} w_i E_i, \quad A(k) = \sum_{i=0}^{L} w_i A_i, \quad B(k) = \sum_{i=0}^{L} w_i B_i$$

$$F(k) = \sum_{i=0}^{L} w_i F_i, \quad C(k) = \sum_{i=0}^{L} w_i C_i, \quad D(k) = \sum_{i=0}^{L} w_i D_i$$

$$\text{such that } \sum_{i=0}^{L} w_i = 1 \tag{5.49}$$

L is the number of rules, the term w_i is the result of the inference process in the rule i and $E_i, A_i, B_i, F_i, C_i, D_i$ are the matrices of the dynamic system which are the consequences of the rule i.
If the membership functions are normal and the inference uses the product as AND operation, the result of the inference is a convex combination of the linear models of the "active" rules. Under this assumption, the result of each inference will be a local linearization of the nonlinear model.
The representation given by the eq.(5.48) is the representation of a Linear Time Variant System.
With this representation at each moment, we can obtain directly from the model a local linearization that can be used to construct the QP. The parameters $A(k), B(k), C(k), D(k)$ are used to construct the matrices C^n, H, D^n at each sampling time, and with this, to construct matrix Λ. The vector Γ corresponding to the *free response* is constructed by evolving the system (5.44) from $k+1$ to $k+N_p$ with input $u^* = 0$.

The control algorithm can be summarized in the following actions:

Algorithm

At each sampling time:

1. Read the current output of the system and update $y(k)$.
2. Make $u^* = 0$ and calculate the *free response* Γ (eq.(5.44)).
3. With the current states and inputs obtain from the fuzzy system the current $A(k), B(k), C(k), D(k)$ (eq.(5.49)).
4. With $A(k), B(k), C(k), D(k)$ compose the Λ matrix (eq.(5.34)).
5. With Λ and Γ construct the matrices for the cost (eq.(5.36)) and the constraints (eq.(5.37)) of the QP problem.
6. Solve the QP problem.
7. Apply only the first control action $u(k|k)$.

The contribution of this approach is the capacity to exploit the information given

directly by the Takagi-Sugeno fuzzy model. This approach is very attractive for systems of high order because no simulation is needed to obtain the parameters to solve the optimization; the matrices can be generated directly from the inference of the fuzzy system. The use of this approach is very attractive to the industry for practical reasons related with the capacity of this model structure to combine local models identified in experiments around the different operating points.

5.3.4 Approach using Takagi Sugeno Fuzzy models and multiple models in the predictor

This approach increases the nonlinear information included on the matrices operating in the QP. This goal is achieved by formulating the predictor using a time variant model over the prediction horizon. In the previous formulation, the model is assumed to change only at each iteration of the controller. The model is fixed along the prediction horizon in the predictor. The current approach assumes a change in the model of the predictor not only at each iteration of the controller, but also at each time step in the prediction horizon.
The formulation of the predictor will be similar to the formulation presented in the previous section but observe that the matrices used to build the predictor include the time variant information:

$$X^n = \underbrace{\Psi + \Phi x(k)}_{\text{Free Response}} + \underbrace{H\overline{U}^n}_{\text{Forced Response}} \tag{5.50}$$

$$Y^n = M^n + C^n X^n + D^n \underline{U}^n \tag{5.51}$$

$$Y^n = \underbrace{M^n + C^n[\Psi + \Phi x(k)]}_{\text{Free Response}} + \underbrace{C^n H \overline{U}^n + D^n \underline{U}^n}_{\text{Forced Response}}$$

$$Y^n = \underbrace{M^n + C^n[\Psi + \Phi x(k)]}_{\Gamma} + \underbrace{[C^n H\ 0] + [0\ D^n]}_{\Lambda} U^n \tag{5.52}$$

where

$$X = \begin{bmatrix} x(k+1|k) \\ x(k+2|k) \\ \vdots \\ x(k+N_p|k) \end{bmatrix}, \Phi = \begin{bmatrix} A(k) \\ A(k+1)A(k) \\ \vdots \\ \prod_{j=0}^{N_p-1} A(k+N_p-1-j) \end{bmatrix},$$

$$M^{N_p} = \begin{bmatrix} M(k+1) \\ M(k+2) \\ \vdots \\ M(k+N_p) \end{bmatrix},$$

$$\Psi = \begin{bmatrix} L(k) \\ L(k+1) + A(k+1)L(k) \\ L(k+2) + A(k+2)L(k+1) + A(k+2)A(k+1)L(k) \\ \vdots \\ L(k+N_p-1) + A(k+N_p-1)L(k+N_p-2) + \ldots \\ \ldots + \prod_{j=0}^{N_p-2} A(k+N_p-1-j)L(k) \end{bmatrix},$$

$$H = \begin{bmatrix} B(k) & 0 \\ A(k+1)B(k) & B(k+1) \\ A(k+2)A(k+1)B(k) & A(k+2)B(k+1) \\ \vdots & \vdots \\ \prod_{j=0}^{N_p-1} A(k+N_p-1-j)B(k) & \prod_{j=0}^{N_p-2} A(k+N_p-1-j)B(k+1) \end{bmatrix}$$

$$\begin{bmatrix} \ldots & 0 \\ \ldots & 0 \\ \ldots & 0 \\ \ddots & \vdots \\ \ldots & B(k+N_p-1) + \sum_{i=0}^{N_p-N_c-1} \prod_{j=0}^{i} A(k+N_p-1-j)B(k+N_p-i-2) \end{bmatrix},$$

$$U^{N_u} = \begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+N_c|k) \end{bmatrix}, \overline{U}^n = \begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+N_c-1|k) \end{bmatrix}, \underline{U}^n = \begin{bmatrix} u(k+1|k) \\ u(k+2|k) \\ \vdots \\ u(k+N_c|k) \end{bmatrix}$$

$$C^n = \begin{bmatrix} C(k+1) & 0 & \ldots & 0 \\ 0 & C(k+2) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & C(k+N_p) \end{bmatrix}, \in R^{N_p n_o \times N_p n_s},$$

$$D^n = \begin{bmatrix} D(k+1) & 0 & \ldots & 0 \\ 0 & D(k+1) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & D(k+N_p-1) \\ 0 & 0 & \ldots & D(k+N_p) \end{bmatrix}, \in R^{N_p n_o \times N_c n_i}$$

where n_s is the number of states, n_o the number of outputs and n_i the number of inputs.
These matrices then are replaced in the cost function (5.36) and in the constraints (5.37) and the QP is solved.
The algorithm work as follows:

Algorithm

1. Read the current output of the system and update $y(k)$.
2. Make $u^* = 0$ and calculate the *free response* Γ (eq.(5.44)).
3. With the current states and inputs obtain from the fuzzy system the current $A(k), B(k), C(k), D(k)$ (eq.(5.49)).
4. With $A(k), B(k), C(k), D(k)$ compose the Λ matrix (eq.(5.34)).
5. With Λ and Γ construct the matrices for the cost (eq.(5.36)) and the constraints (eq.(5.37)) of the QP problem.
6. Solve the QP problem.
7. Apply the generated input sequence to the fuzzy model and generate the set of $E(k), F(k), A(k), B(k), C(k), D(k)$ matrices over the prediction horizon .
8. Construct the Γ and the Λ matrices and using the cost (eq.(5.36)) and the constrains (eq.(5.37)) solve the QP problem.
9. If the time for calculation has not expire and the difference between two consecutive solutions is larger than a given value ϵ repeat iteration from step 7.
10. Apply only the first control action $u(k|k)$.

For the first iteration the constructed predictor assumes that the plant is linear time invariant in the term corresponding to the forced response. From this iteration, a suboptimal solution is obtained. This solution is equivalent to the solution presented in the section 5.3.3.
With the solution from the first iteration, it is possible to construct a predictor by using time variant linear models included in the term of the forced response. A new QP problem will be solved and a new input sequence will be generated. The process will be repeated until the sampling time is over or the difference among to consecutive input sequence is smaller than a given values ϵ. The contribution of this method is the improvement of the nonlinear information included in the optimization with the advantage that once the first iteration is done, a "good" suboptimal sequence is available and can be applied to the plant. This improvement will extend the applicability of these techniques to non-linear systems with "hard" nonlinearities.

5.3.5 Predictive Control of a Steam Generator Using a Fuzzy Model

This application is based on the simulation model presented by Pelegrinetti et al. [63]. There they obtain a model of the boiler at Abbott Power Plant in Champaign IL. The model is a multivariable model, which includes four inputs (Fuel, Air, Water flow, and Steam demand) and four outputs (Pressure, Oxygen, and Level in the drum, Steam flow). The obtained model includes perturbation

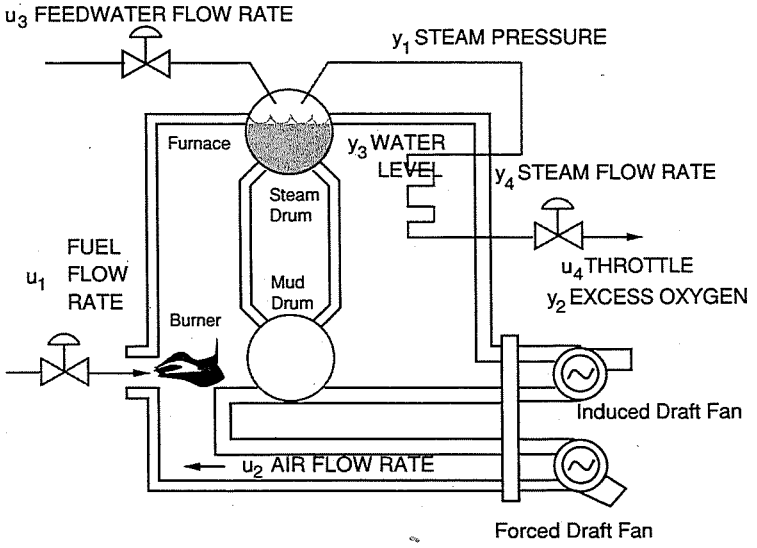


Figure 5.13: *Steam Generator*

and measurement noises. This model is used as the real plant for the simulations. The model of the steam generator is described by the following system of equations:

$$\dot{x}_1(t) = c_{11}x_4(t)x_1^{9/8}(t) + c_{12}u_1(t-\tau_1) - c_{13}u_3(t-\tau_3) + c_{14} \quad (5.53)$$

$$\dot{x}_2(t) = -c_{21}x_2 + \frac{c_{22}u_2(t-\tau_2) - c_{23}u_1(t-\tau_1 - c_{24}u_1(t-\tau_1 x_2(t)}{c_{25}u_2(t-\tau_2 + c_{26}u_1(t-\tau_1} \quad (5.54)$$

$$\dot{x}_3(t) = c_{31}x_1(t) - c_{32}x_4(t)x_1(t) + c_{33}u_3(t-\tau_3) \quad (5.55)$$

$$\dot{x}_4(t) = -c_{41}x_4(t) + c_{42}u_1(t-\tau_1) + c_{43} + u_4(t) + n_5 \quad (5.56)$$

$$y_1(t) = c_{51}x_1(t-\tau_4) + n_1(t) \quad (5.57)$$
$$y_2(t) = c_{61}x_2(t-\tau_5) + n_2(t) \quad (5.58)$$
$$y_3(t) = c_{70}x_1(t-\tau_6) + c_{71}x_3(t-\tau_6) + c_{72}x_4(t-\tau_6) + c_{73}u_3(t-\tau_3-\tau_6) + c_{74}u_1(t-\tau_1-\tau_6) + \frac{[c_{75}x_1(t-\tau_6)+c_{76}][1-c_{77}x_3(t-\tau_6)]}{x_3(t-\tau_6)[x_1(t-\tau_6)+c_{78}]} + c_{79} + n_3(t) \quad (5.59)$$
$$y_4(t) = [c_{81}x_4(t-\tau_7)+c_{82}]x_1(t-\tau_7) + n_4(7) \quad (5.60)$$

where x_1 is the drum pressure state (kgf/cm^2); y_1 is the measured drum pressure (PSI); y_2 and x_2 are the measured excess oxygen level and its state, respectively, (percent); x_3 is the system fluid's density (kg/m^3); y_3 is the drum water level (in); Y_4 is the steam flow rate (kg/s); u_1, u_2, u_3 are respectively the fuel, air, and feed water flow rate inputs which take values between 0 and 1; x_4 is the exogenous variable related to the steam demanded. The variables n_i are colored noise sequences generated by first order models driven by zero mean, unit variance white noise.

$$n_1 = \frac{0.75s+0.1}{s+0.001}w_1,\ n_2 = \frac{0.019s+0.001}{s+0.024}w_2,\ n_2 = \frac{0.105s+0.038}{s+0.010}w_3$$
$$n_4 = \frac{0.01s+0.0001}{s+0.001}w_4,\ n_5 = \frac{0.003s+0.003}{s+0.0075}w_5$$

where $n_i\, i = 1,\ldots,5$ are colored noise and w_i is the unit variance white noise. A linear model was obtained by linearizing around the operating point:

$$x^0 = [22.5\ 2.5\ 621.17\ 0.6941]^T$$
$$y^0 = [320\ 2.5\ 0.0\ 9.3053]^T$$
$$u^0 = [0.32270\ 0.39503\ 0.37404\ 0]^T$$

The linearized model is described by:

$$\dot{x} = A_p x + B_p u$$
$$y = C_p x + D_p u$$

where

$$A_p = \begin{bmatrix} -0.005509 & 0 & 0 & -0.1588 \\ 0 & -0.2062 & 0 & 0 \\ -0.01216 & 0 & 0 & -0.5672 \\ 0 & 0 & 0 & -0.040 \end{bmatrix}$$

$$B_p = \begin{bmatrix} 0.2800 & 0 & -0.01348 & 0 \\ -9.375 & 7.658 & 0 & 0 \\ 0 & 0 & 0.7317 & 0 \\ 0.02999 & 0 & 0 & 0.040 \end{bmatrix}$$

Table 5.3: Non linear Equation Coefficients

$c_{11} = -0.00478$	$c_{31} = 0.00533176$	$c_{70} = -0.1048569$
$c_{12} = 0.280$	$c_{32} = 0.0251950$	$c_{71} = 0.15479$
$c_{13} = 0.01348$	$c_{33} = 0.7317058$	$c_{72} = 0.4954961$
$c_{14} = 0.02493$	$c_{41} = 0.04$	$c_{73} = -0.20797$
$c_{21} = 0.1540357$	$c_{42} = 0.0299886$	$c_{74} = 1.2720$
$c_{22} = 103.5462$	$c_{43} = 0.018088$	$c_{75} = -324212.7805$
$c_{23} = 107.4835$	$c_{51} = 14.214$	$c_{76} = -99556.24778$
$c_{24} = 1.95150$	$c_{61} = 1.00$	$c_{77} = 0.0011850$
$c_{25} = 29.04$	$c_{81} = 0.85663$	$c_{78} = -1704.50476$
$c_{26} = 1.824$	$c_{82} = -0.18128$	$c_{79} = -103.7351$
$\tau_1 = 2, \tau_2 = 2, \tau_3 = 3, \tau_4 = 3, \tau_5 = 4, \tau_6 = 10, \tau_7 = 2.$		

$$C_p = \begin{bmatrix} 14.21 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0.3221 & 0 & 0.1434 & 11.16 \\ 0.4133 & 0 & 0 & 19.28 \end{bmatrix}$$

$$D_p = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1.272 & 0 & -0.2080 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Fuzzy identification of the Steam Generator

To start the identification of the model, the level control is first stabilized by means of a feed-forward control plus a PID control. The feed-forward control signal is proportional to the steam flow to compensate the mass of water in the drum; a PID controller performs further control. The representation of the system with this stabilizing control actions can be observed in the figure 5.14. With this modification the system will have the following inputs: Fuel rate (u_1), Air rate (u_2), Reference Level (Ref_1), Steam demand (u_4). A sampling time of $T_s = 3$ seconds was chosen and several experiments were performed. A total of five (5) Multiple Input- Single Output fuzzy models were extracted. The details of the extracted models are presented in the table 5.4. The data points used for this identification are available at [20]. The inputs for each model were selected by using the RC index. The five models obtained can be described as:

$$\hat{y_1}(k) = f_1(\hat{y_1}(k-2), \hat{x_4}, u_1(k-2), u_1(k-3)) \quad (5.61)$$

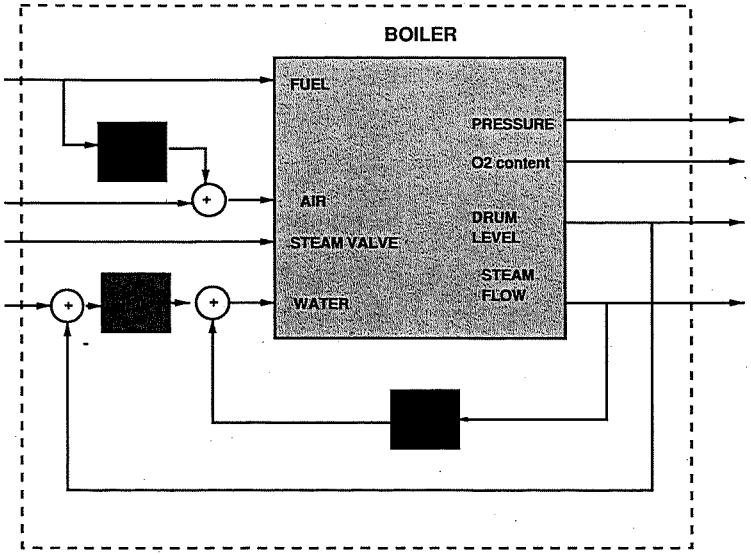


Figure 5.14: *Stabilization Scheme for the Steam Generator*

Table 5.4: *Characteristics of the Identified Fuzzy Model*

Model	No. Rules
$\hat{y_1}(k) = f_1(\hat{y_1}(k-2), \hat{x_4}, u_1(k-2), u_1(k-3))$	81
$\hat{y_2}(k) = f_2(\hat{y_2}(k-2), u_1(k-2), u_2(k-2))$	27
$\hat{y_3}(k) = f_3(Ref_1(k-8), \hat{y_4}(k-5), \hat{y_4}(k-6), u_1(k-4))$	81
$\hat{y_4}(k) = f_4(\hat{y_1}(k-1), \hat{x_4}(k-1))$	9
$\hat{x_4}(k) = f_5(\hat{x_4}(k-1), u_1(k-1), u_4(k-1))$	27
Total rules	225

$$\hat{y_2}(k) = f_2(\hat{y_2}(k-2), u_1(k-2), u_2(k-2)) \quad (5.62)$$
$$\hat{y_3}(k) = f_3(Ref_1(k-8), \hat{y_4}(k-5), \hat{y_4}(k-6), u_1(k-4)) \quad (5.63)$$
$$\hat{y_4}(k) = f_4(\hat{y_1}(k-1), \hat{x_4}(k-1)) \quad (5.64)$$
$$\hat{x_4}(k) = f_4(\hat{x_4}(k-1), u_1(k-1), u_4(k-1)) \quad (5.65)$$
$$(5.66)$$

where y_i represent the outputs (Pressure, Oxygen level, Level and Steam Flow respectively) and x_4 is an internal state variable. Figure 5.15 shows the validation results of the model. Another model (A Takagi-Sugeno Fuzzy Model) was

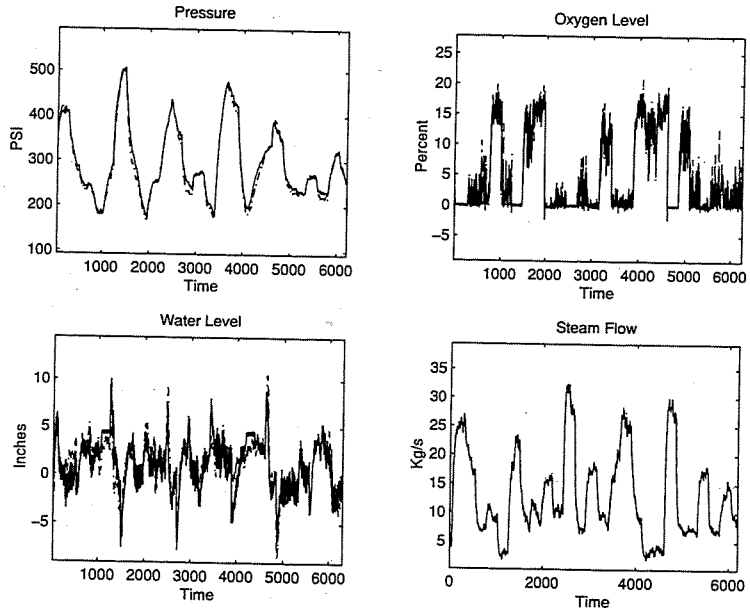


Figure 5.15: *Validation Experiment*

generated with only five rules. This second model only describes the dynamics around the nominal operating points of the Pressure, Level and Excess Oxygen, and uses five local models for different values of steam flow. The rules of this system look as follows:

$$\textbf{IF Steam Flow } (y_4(k)) \text{ is } \mathcal{F}_i \text{ THEN } \begin{array}{rcl} X(k+1) & = & E_i + A_i X(k) + B_i U(k) \\ Y(k+1) & = & F_i + C_i X(k) + D_i U(k) \end{array}$$

Implementation of the controller and performance results

Three predictive controllers were implemented. The first one corresponds to a linear MPC constructed using the linearized model presented in the previous

section (*MPC with linear model*), the second MPC corresponds to a MPC controller using the estimated step response method (*MPC with Fuzzy Model* see section: 5.3.2) and a third one corresponds to the strategy using Takagi Sugeno Fuzzy Models without time variant information ((*MPC with TS Fuzzy Model* see section: 5.3.3).
The following parameters were used in the construction of the controllers: prediction horizon $N = 5$, control horizon $Nc = 2$, $C_j(z^{-1}) = 1$ and $D_j(z^{-1}) = 1 - z^{-1}$ $\forall j = 1, \ldots, 4$, $\alpha = 0.9$ the Q matrix

$$Q = \begin{bmatrix} q_1 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 \\ 0 & 0 & q_3 & 0 \\ 0 & 0 & 0 & q_4 \end{bmatrix}$$

where q_i are $\Re^{N \times N}$ diagonal matrices with values [0.0014, 0.0001, 0.0002, 0.0503] and the R matrix is

$$R = \begin{bmatrix} r_1 & 0 & 0 & 0 \\ 0 & r_2 & 0 & 0 \\ 0 & 0 & r_3 & 0 \\ 0 & 0 & 0 & r_4 \end{bmatrix}$$

where r_i are $\Re^{Nc \times Nc}$ diagonal matrices with values [50 0.1 1 10^4]. The inputs corresponding to the Fuel and the Air valves are constrained to be in the interval [0 1] and with a $du_{max} = 0.1$; the reference of the level is constrained to be in the interval [-4 4] and with a $du_{max} = 1$ and the valve for the steam is constrained in the interval [-0.15 0.03] (the reason that this valve was not constrained in the interval [0 1] is that the valve was originally modeled as a disturbance variable) and with a $du_{max} = 1$. The solution with the linear MPC is not subject to constraints.
The performance of the controllers is evaluated by using an experiment where the energy generated by the plant is first reduced from the 39 % percent of the maximum power to the 27%; then it is raised to the 100% and finally reduced to the 27% . In order to achieve this, the pressure, the Oxygen Level and the Water level are kept constant with values 320 PSI, 2.5 % and 0 Inch, respectively and the Steam flow is changed from 9.3 Kg/s to 6.6 Kg/s, then it is raised to 22.096 Kg/s and finally reduced again to 6.6 Kg/s. The figure 5.16 and the table 5.5 show the comparative results of the experiments.

The comparison shows some interesting results. It is clear that the proposed control strategy outperforms the classical Linear MPC solution. Its capacity to reject disturbances can be clearly observed when the system is operated at de 100 % of the capacity the Linear MPC is not capable of keeping the level at the desired value. In addition, it is important to observe that the settling time has

Table 5.5: Comparative Results- Performance calculated for the Steam Flow

	MPC with Fuzzy Model	MPC with TSFuzzy Model	MPC with Linear Model
Variance on 27% of the capacity	0.0011	0.0055	0.0119
Variance on 100% of the capacity	4.42×10^{-4}	0.0043	0.0096
Overshoot on transition from 27% - 100 %	1.06 %	1.83 %	1.32 %
Overshoot on transition from 100% - 27 %	0.25%	1 %	2.58 %
Settling time on transition from 27% - 100 %	103 s.	150 s.	423 s.
Settling time on transition from 100% - 27 %	120 s.	190 s.	558 s.

been reduced to one quarter of the settling time for the system with the linear MPC without any increase in overshoot. The reduction of the variance once more compared with the linear MPC strategy is around one order of magnitude for the MPC strategy with estimated step response and half order of magnitude for the strategy base on the Takagi Sugeno model. The MPC based in the Takagi-Sugeno model has the best disturbance rejection capacity. It does make a better task controlling the drum pressure. The settling time of the MPC strategy with estimated step response is the shortest.

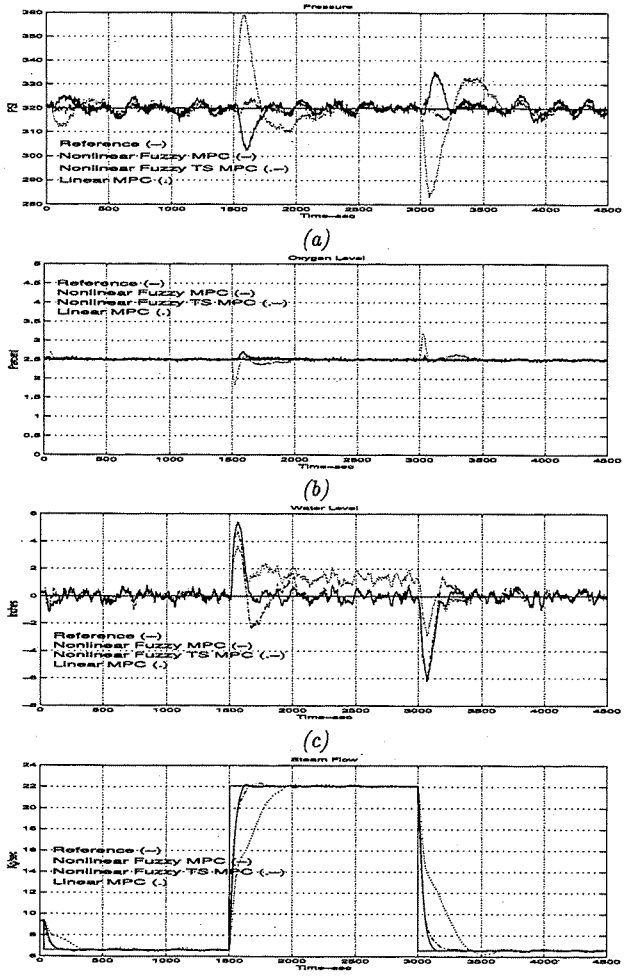


Figure 5.16: *Comparative result: (-) MPC with Fuzzy Model using step response, (.-) MPC using Takagi-Sugeno Fuzzy Model, (.) MPC with Linear Model-(a) Pressure (b) Oxygen Level (c) Water Level (d) Steam Flow*

5.4 Conclusions

This chapter has presented a novel approach to fuzzy model based control. The combination of fuzzy modeling and predictive control is the result of the maturation of the modeling and identification techniques using fuzzy structures. The goal was to design a simple control strategy. The strategy should be simple for the designer and the end user of the control systems.

The simplicity for the designer of the control system is reflected on the fact that the only task during the tuning process is the definition of the goal of the control systems in terms of a quadratic cost function and the constraints of the elements involved in the system. Other tuning parameters such as control and prediction horizons are closely related with the settling times and the model order. These parameters exhibit a monotonic tendency making the tuning of the controller a simple task. Of course a price must be paid to achieve this simplicity and this cost can be divided in two: i) the effort that must be put to build the model; and ii) the computational cost of calculating the control actions at each sampling time. The fact that the described algorithms are computationally intensive limits the sampling time with the current computational technology (SUN-Ultra-2 Creator 200 MHz) to 0.05 seconds for unconstrained system and around 3 seconds for constrained multivariable systems of low order. This will limit the applications where these strategies can be applied nowadays to applications to process control problems where sampling time is about 30 sec., but Moore's law guarantees that in the term of 3 or 4 years the system can be applied even to mechanic systems with fast dynamics.

This chapter includes four important contributions represented by the control algorithms with their respective derivations.

Two of the proposed algorithms are particularly interesting for their applicability to other types of nonlinear models and not only to fuzzy systems. These algorithms are the strategies based on the estimated step response.

It is also important to remark the advantages of the other two contributed strategies. The two strategies based on the use of Takagi-Sugeno fuzzy models exploit the structural and the simulation information provided by the model. Control engineers are particularly attracted by the fact that the model "look alike" gain scheduling models.

The key point of the strategies presented in the current chapter is the reduction of the original complex nonlinear program (with no possibility for on-line implementation) to a simple Quadratic Program (on-line implementable) by exploiting the structure of the problem and the structure of the models, and by relaxing the constraints imposed by the problem. The relaxation introduced by the methods presented in this chapter is a relaxation of the equality constraints, which are in fact the plant description. In some cases the solutions can be quite close to the "global optimal solution", especially for systems with monotonic nonlinearities.

The assumptions used to reduce the problem demands some smoothness in the nonlinearities, which are of course related with the prediction capability of the model when "strong" nonlinearities are involved.
The two examples presented are realistic simulation of industrial processes. The improvement in the performance is clear from the simulations. The systems used in the examples are characterized by strong changes in gain and dynamics at the different operating points.

Chapter 6

Robust Nonlinear Predictive Control using Fuzzy Models

6.1 Introduction

The increasing popularity of the fuzzy models for nonlinear system identification and modeling can be explained by the possibility to extract "local" information about the dynamics of the system. Different algorithms for identification of Takagi-Sugeno fuzzy models have been proposed in recent years [88] [5]. The representation of locally linearized models in the form of a Takagi-Sugeno fuzzy model has shown its advantages in simplifying the design of nonlinear controllers.
The previous chapter has shown the advantages of the use of model based predictive control techniques in the process control industry as a technique to design multi-variable controllers with direct performance specifications.
The complexity of the solution of predictive controllers for non-linear systems was discussed in the previous chapter. Relaxations to the problem and its reduction to a quadratic program were the strategies applied to obtain a solution. This solution is very close to the real "optimal" solution (see results of the example in section 5.2.2), but it demands a very accurate description of the nonlinear plant to guarantee such performance.
An accurate description of the plant is not always an achievable goal. The limitations are i) the impossibility of performing extensive experiments on the plant; and ii) the fact that the aging process of the plant generates a mismatch with respect to the model.
For stability reasons and for performance reasons it is very important to take into

account this mismatch during the optimization process. The solution obtained will guarantee a minimum performance that will be reflected in the quality of the product.
This chapter presents an algorithm for nonlinear predictive control using the description of the nonlinear plant provided by Takagi-Sugeno fuzzy models. The method solves the problem by formulating a Robust Quadratic Program. The solution of the Robust Quadratic Program is obtained by transforming the program into a Second Order Cone Program [49]. It is important to remark that the solution presented in [49] only takes into account uncertainties in the Hessian matrix. In this chapter, the uncertainties are extended to the linear term of the cost function and the problem is once more reformulated as a Second Order Cone Program. Among the advantages of the method, the robust performance is probably the most important because it makes the method more suitable for real applications. The chapter is organized as follows: section 6.2 presents the formulation of a Robust Quadratic Program as a Second Order Cone Program including the contribution where uncertainties in the linear term are included, section 6.3 describes the problem analytically, section 6.4 shows the nominal solution when the problem has no uncertainties (it is the same solution presented in the previous chapter but included here for completeness), section 6.5 presents an approximated solution to the problem of robust predictive control using robust quadratic programming, section 6.7 shows some possible ways to describe the uncertainty on the fuzzy model and section 6.8 concludes the chapter.

6.2 Robust Quadratic Programming

Robust programming has been a subject of study in recent years [59] [7] [6]. Boyd et. al. [12] have shown the practical applications of these techniques in control and filter design. Other applications include robust antenna array design and truss topologies. Robust programming is a class of optimization problems where the parameters such as coefficients of cost functions and/or constraints are uncertain and prescribed into a defined set.
Efficient interior point methods have been created for some of these problems. A very interesting property of these methods is that their computational complexity is polynomial with respect to the number of constraints. For instance, the computational complexity for the Second Order Cone Program solution is a value proportional to $\sqrt{l}$ where l is the number of constraints. This fact makes these algorithms very promising for large-scale optimization. This property opens wide possibilities in the application of these methods not only for robust optimization but also for plant wide optimizers. Some authors have already explored the applications of interior point methods to solve the nominal

problem of linear predictive control [83].
This section will focus its attention into the problem of Robust Quadratic Programming (RQP). The problem is formulated as follows:

$$\min_{x} \max_{P\in\mathcal{E},q\in\mathcal{F}} x^T P x + 2q^T x + r \tag{6.1}$$

the objective is to minimize the cost function (6.1). Where $x \in \Re^N$ is the optimization variable, $P \in \Re^{N\times N}$ is a symmetric positive definite matrix also known as the Hessian matrix, the vector $q \in \Re^N$ also known as the linear term of the cost function and the scalar value r. Observe that in this case the Hessian matrix P and the linear term q are considered uncertain and their uncertainties are described in by the sets $\mathcal{E}$ and $\mathcal{F}$. This set $\mathcal{E}$ describes the uncertainty in the following form for the P matrix:

$$\mathcal{E} = \left\{ P_0 + \sum_{i=1}^{m} P_i u_i \mid ||u|| \leq 1 \right\}$$

where P_0 and P_i are matrices in $\Re^{N\times N}$, u is a vector in $\Re^m$ and $||.||$ is the Euclidean Norm. The set $\mathcal{F}$ describes the uncertainty of the linear term q as follows:

$$\mathcal{F} = \left\{ q_0 + \sum_{i=1}^{n} q_i v_i \mid ||v|| \leq 1 \right\}$$

where q_0 and q_i are vectors in $\Re^N$, v is a vector in $\Re^n$ and $||.||$ is the Euclidean Norm.
In [49] Lobo et. al. show that a robust quadratic program can be written as a Second Order Cone Program (SOCP). However, in that formulation, the uncertainty was restricted to the Hessian matrix. This thesis extends the formulation presented in [49] by including the uncertainty in the linear term q, described by the set $\mathcal{F}$. Before this result is presented the next lines will describe the Second Order Cone Program.
A Second Order Cone Program is defined as follows:

$$\begin{aligned} \min_{x} \quad & f^T x \\ \text{subject to} \quad & ||A_i x + b_i|| \leq c_i^T x + d_i, \;\; i = 1, \ldots, N, \end{aligned} \tag{6.2}$$

where $x \in \Re^n$ is the optimization variable, $||.||$ is the Euclidean Norm and the problem parameters are $f \in \Re^n, A_i \in \Re^{(n_i-1)\times n}, b_i \in \Re^{n_i-1}, c_i \in \Re^n$ and $d_i \in \Re$.
As it was mentioned in previous lines there are very efficient methods to solve this optimization problem with complexity proportional to $\sqrt{l}$ where l is the number of constraints.
To transform the Robust Quadratic Program into a Second order cone Program

the description of the uncertainty sets $\mathcal{E}$ and $\mathcal{F}$ is introduced in the cost function (6.1),

$$\min \max_{\substack{||u||\leq 1 \\ ||v||\leq 1}} x^T P_0 x + \sum_{i=1}^{m} x^T P_i x u_i + 2q_0^T x + 2\sum_{j=1}^{n} q_j^T x v_j + r \tag{6.3}$$

applying the triangle inequality:

$$\begin{aligned} &\min[x^T P_0 x + 2q_0^T x + \max_{||u||\leq 1} ||x^T P_i x||||u|| + 2\max_{||v||\leq 1} ||q_j^T x||||v|| + r] \\ &\leq \\ &\min[x^T P_0 x + 2q_0^T x + \quad ||x^T P_i x|| \quad + \quad 2||q_j^T x|| \quad + r] \end{aligned} \tag{6.4}$$

Using this representation, the problem can be converted into a problem with linear cost with quadratic constraints,

$$\min f + 2q_0^T x + t + 2d + r \tag{6.5}$$

subject:

$$\begin{aligned} x^T P_0 x &\leq f \\ x^T P_i x &\leq w_i \\ ||q_j^T x|| &\leq d \\ ||w|| &\leq t \end{aligned}$$

This optimization can be formulated as a SOCP as follows:

$$\min f + 2q_0^T x + t + 2d + r \tag{6.6}$$

subject:

$$\begin{aligned} \left\| \begin{bmatrix} 2P_0^{\frac{1}{2}} x \\ f - 1 \end{bmatrix} \right\| &\leq f + 1 \\ \left\| \begin{bmatrix} 2P_i^{\frac{1}{2}} x \\ w_i - 1 \end{bmatrix} \right\| &\leq w_i + 1 \\ ||q_j^T x|| &\leq d \\ 0 &\leq f \\ 0 &\leq w_i \\ ||w|| &\leq t \end{aligned} \tag{6.7}$$

Observe that with this new formulation the search space is extended from the space of x to the spaces of f, w_i, t and d.

6.3 Problem Description

The formulation of the robust nonlinear predictive control problem is expressed in the following lines as:

$$\min_{u(k),\ldots,u(k+Nc)} \max_{\delta\in\Delta} J(Y_{\text{ref}}, y, u, \delta) \tag{6.8}$$

subject to:

$$\begin{aligned} x(k+1) &= f(x(k), u(k), \delta) \\ y(k) &= g(x(k), u(k), \delta) \\ & u_{min} \le u \le u_{max} \\ & |u(k) - u(k-1)| \le \Delta u_{max} \end{aligned}$$

where $J(.,.,.,.)$ is a cost function (typically quadratic) which penalizes the deviation of the output of the plant y with respect to the reference signal Y_{ref}, in a prescribed period (N_p samples-prediction horizon). The minimization searches for a sequence of inputs $(u(k+1), u(k+2), \ldots, u(k+N_c))$ subject to constraints such as the plant dynamics described by $f(.,.,.)$ and $g(.,.,.)$ and input constraints related with saturation and slew rate. The parameter $||\delta|| \le \rho$ represents a bounded uncertainty in the plant dynamics. For the present case the dynamics of the plant ($f(.,.,.)$ and $g(.,.,.)$) will be represented using dynamic fuzzy models, in a state space form or in an input-output form.
This optimization problem is a very complex problem of Robust Nonlinear Programming (RNLP) and with the actual computational resources it is impossible to guarantee that a solution is found in a prescribed number of steps.

6.4 Nominal Solution

The nominal problem (without uncertainty, $||\delta|| = 0$) is simpler than the robust one, but still it is a Nonlinear Program (NLP). In the previous chapter, some approximated solutions were found by reducing the problem to a Quadratic Program (QP). Other approaches for general classes of nonlinear system have also been proposed in the literature [51]. The approaches presented in the previous chapter explore the use of three different pseudo-linearization methods to convert the NLP into a QP.
The nominal problem assumes no uncertainty:

$$\min_{u(k),\ldots,u(k+Nc)} J(Y_{\text{ref}}, y, u) \tag{6.9}$$

subject to:

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) \\ y(k) &= g(x(k), u(k)) \\ & u_{min} \le u(k) \le u_{max} \\ & |u(k) - u(k-1)| \le \Delta u_{max} \end{aligned}$$

using a pseudo-linearization [1] given by:

$$Y = \underbrace{\Gamma}_{\text{Free Response}} + \underbrace{\Lambda u}_{\text{Forced Response}} \tag{6.10}$$

and a quadratic cost function given by:

$$\begin{aligned} J(Y, U^n) = (Y_{\text{ref}} - Y)^T Q (Y_{\text{ref}} - Y) + \\ + U^{nT} R U^n + (\Delta U^n - \bar{U}_{k-1})^T S (\Delta U^n - \bar{U}_{k-1}) \end{aligned} \tag{6.11}$$

the problem can be written as the following QP:

$$\begin{aligned} J(U^n) = J_{\min} + \\ + 2[(\Gamma - Y_{\text{ref}})^T Q \Lambda - \bar{U}_{k-1}^T S \Delta] U^n + \\ + U^{nT} [\Lambda^T Q \Lambda + R + \Delta^T S \Delta] U^n \end{aligned} \tag{6.12}$$

where

$$J_{\min} = Y_{\text{ref}}^t Q Y_{\text{ref}} + \Gamma^T Q \Gamma - 2 Y_{\text{ref}}^T Q \Gamma + \bar{U}_{k-1}^T S \bar{U}_{k-1}$$

is the minimum cost that cannot be modified by any control input. The constraints will be written as follows:

$$\begin{bmatrix} I^{(n_i N_c)} \\ -I^{(n_i N_c)} \\ I^{(n_i)} 0^{(n_i \times n_i N_c)} \\ -I^{(n_i)} 0^{(n_i \times n_i N_c)} \\ \Delta \\ -\Delta \end{bmatrix} U^{N_c} \le \begin{bmatrix} U_{max}^{N_c} \\ -U_{min}^{N_c} \\ \Delta U_{max}^{N_c} + \bar{U}_{k-1} \\ \Delta U_{max}^{N_c} - \bar{U}_{k-1} \\ \Delta U_{max}^{N_c} \\ \Delta U_{max}^{N_c} \end{bmatrix} \tag{6.13}$$

where I is an identity matrix.

$$U_{max}^n = \begin{bmatrix} u_{max}(k) \\ u_{max}(k+1) \\ \vdots \\ u_{max}(k+n) \end{bmatrix},$$

[1] The term "pseudo-linearization" comes from the fact that Γ is obtained by simulation of the *free response* in the nonlinear model and Λ is the result of a linearization (time invariant or variant).

$$U^n_{min} = \begin{bmatrix} u_{min}(k) \\ u_{min}(k+1) \\ \vdots \\ u_{min}(k+n) \end{bmatrix}, \Delta U_{max} = \begin{bmatrix} \Delta u_{max}(k) \\ \Delta u_{max}(k+1) \\ \vdots \\ \Delta u_{max}(k+n) \end{bmatrix}$$

where $I^{(n)} = I \in \Re^{n \times n}$ and $0^{(m \times n)} \in \Re^{m \times n}$.
This method works well when the nonlinearities are smooth, the model is a good representation of the plant and the control actions don't move the system far away from the region where the "pseudo-linearization" is valid.

6.5 Formulation of the Predictive Control problem as a Robust Quadratic Program

This section expands the nominal solution presented in section 6.4 by introducing uncertainty in the parameters of the cost function of the QP (see equation (6.12)).
For the current formulation the uncertainty will be restricted to the forced response term $(\Lambda + \delta\Lambda)U$. The problem described in equation (6.12) will be converted to:

$$\begin{aligned} J(U^n) = J_{\min} + \\ + 2[(\Gamma - Y_{\text{ref}})^T Q\Lambda - \bar{U}^T_{k-1} S\Delta + (\Gamma - Y_{\text{ref}})^T Q\delta\Lambda]U^n + \\ + U^{nT}[\Lambda^T Q\Lambda + R + \Delta^T S\Delta + \delta\Lambda^T Q\delta\Lambda + \Lambda^T Q\delta\Lambda + \delta\Lambda^T Q\Lambda +]U \end{aligned} \quad (6.14)$$

where $J_{\min}$ has the same description presented in equation (6.12).
Using this description the P matrix for the RQP will be given by:

$$P = P_0 + \sum_{i=1}^{m} P_i u_i$$

$$P_0 = \Lambda^T Q\Lambda + R + \Delta^T S\Delta \quad (6.15)$$

$$\sum_{i=1}^{m} P_i u_i = \delta\Lambda^T Q\delta\Lambda + \Lambda^T Q\delta\Lambda + \delta\Lambda^T Q\Lambda \quad (6.16)$$

where $\delta\Lambda$ is an n-dimensional ellipsoidal uncertainty in the matrix used to build the forced response. Observe that the n-dimensional ellipsoidal uncertainty in $\delta\Lambda$ is reflected as an m- dimensional ellipsoidal uncertainty in P with $m = n + \sum_{i=1}^{n} i$.
The q vector will be described as:

$$q = q_0 + \sum_{i=1}^{n} q_i v_i$$

$$q_0^T = (\Gamma - Y_{\text{ref}})^T Q\Lambda - \bar{u}^T_{k-1} S\Delta \quad (6.17)$$

$$\sum_{i=1}^{n} q_i^T v_i = (\Gamma - Y_{\text{ref}})^T Q\delta\Lambda \quad (6.18)$$

observe that no uncertainty is considered in the *free response* Γ. The main reason for this simplification is the complexity to obtain an analytical expression that can be introduced into the RQP. The addition of uncertain terms in the *free response* makes the solution computationally expensive and very conservative, destroying the advantages of the "pseudo-linearization". On the other hand, the uncertainty introduced by this term is only due to the mismatch between the plant and the model and there is no uncertainty due to the linearization process.

6.6 The Control Algorithm

The control algorithm can be described at each sampling time by the following steps:

Algorithm

1. Read the current output of the system and update $y(k)$
2. With the input $u(k-1)$ calculate the *free response* Γ using the equation (5.44).
3. Construct the Λ matrix.
4. With the Λ and the Γ matrices construct the matrix P_0 and the vector q_0.
5. Using the uncertainty description build the set of matrices P_i and the vectors q_i.
6. Solve the SOCP described in the equation (6.6).
7. Apply the first control action, which is the first entry of the solution of the SOCP.

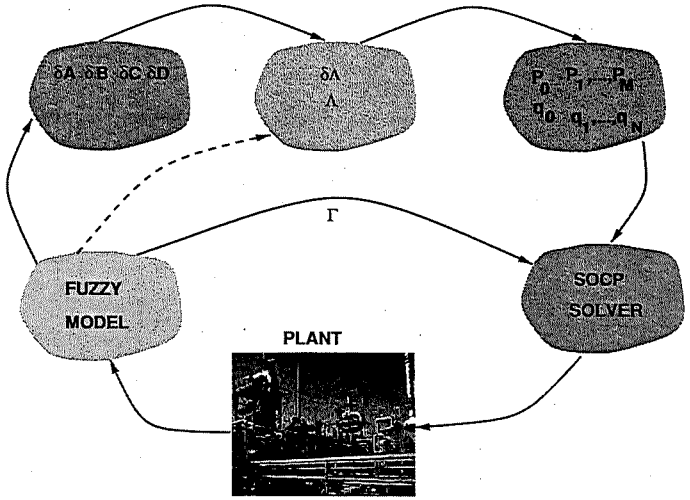


Figure 6.1: *Diagram with the stages of the control algorithm*

6.7 Uncertainty Description in Fuzzy Models

The following lines include some ideas about the way to represent the uncertainty by using the structure of the Takagi Sugeno fuzzy models.

6.7.1 Local uncertainty described on each rule

This is perhaps the simplest method to describe the uncertainty. The idea is to use a Takagi-Sugeno description of the plant where the consequences of the rules are linear dynamic systems with their respective uncertainties. In this case, the uncertainty and the linearization of the plant will be assumed to be the convex combination of the local uncertainties.

$$\begin{array}{c} \dots \\ \text{IF } x(k) \text{ is } \mathcal{A}_i \\ \text{THEN } x(k+1) = L_i + A_i x(k) + B_i u(k) \; |\{\delta A_i, \delta B_i\} \\ \dots \end{array}$$

Among the advantages of this method is the simplicity of the description and the possibility to describe locally the uncertainty based on the real knowledge

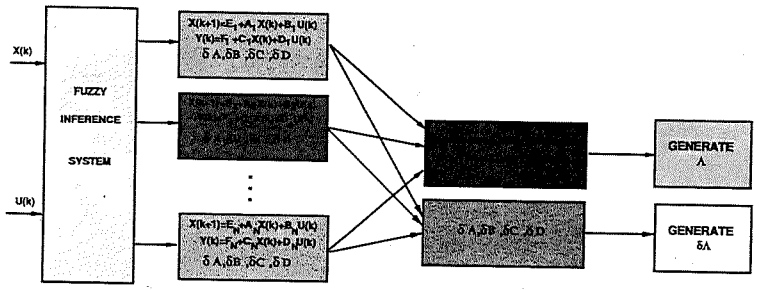


Figure 6.2: *Uncertainty description on each of the rules*

about the quality of each of the local models (for instance the covariance of the parameters during identification). It is important to remark that this description will be valid only if the fuzzy sets are complementary in the antecedents such that the local description will be equivalent to the local linearization.
It is probably the less conservative description, but also it demands more time during the design phase.

6.7.2 Using the Active Rules

This description assumes that the uncertainty is given by the polytope constructed with all the local descriptions of the active rules (rules with a firing value different from 0) at the present instant. This description is efficient but inaccurate specially when the coverage of the rules is small. As a result of the small coverage of the rules, it might happen that the system reaches rules which are beyond the set of active rules with in the prediction horizon. In this case, the uncertainty will be underestimated. For rules with wide coverage, it is a very interesting solution because the system will tend to remain within the set of active rules.

6.7.3 Using All the Rules

The polytope constructed with all the models present in the rules is a quite conservative approach, but it can also be constructed quickly because it can be pre-computed. The conservativeness can lead to in-feasibility in the solution. For rules with wide coverage it is a very interesting, but very conservative solution and it is recommended only for systems with gentle non-linearities.

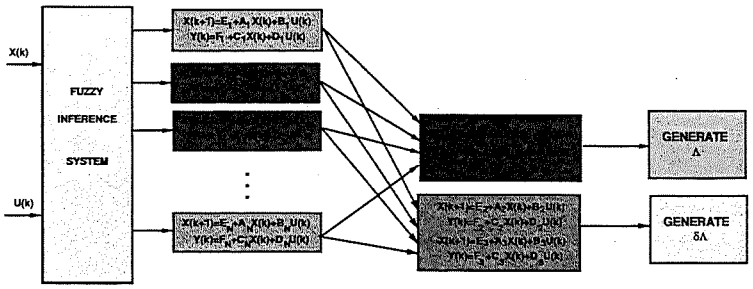


Figure 6.3: *Uncertainty description using the "polytope" of the active rules*

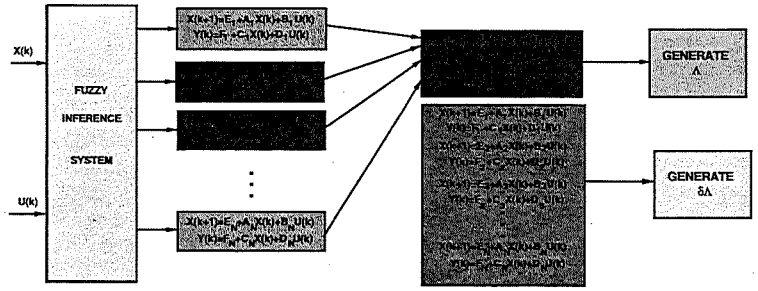


Figure 6.4: *Uncertainty description using the "polytope" of the complete set of rules*

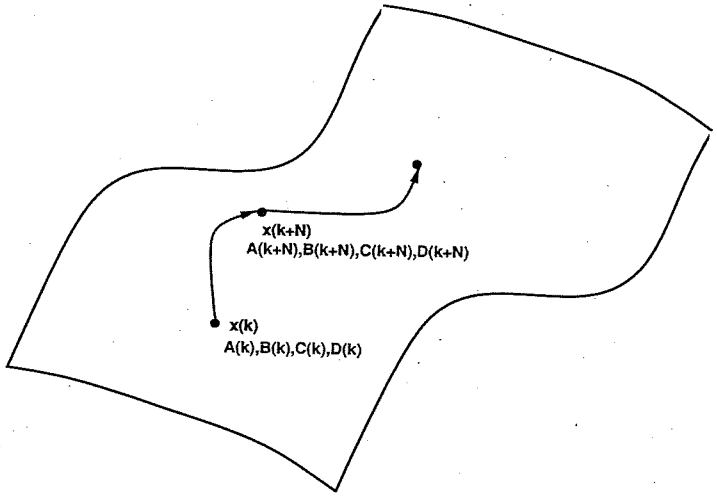


Figure 6.5: *Graphic representation of the reachable set*

6.7.4 Using the Reachable Set

In this method, the uncertainty is described also as a polytope, generated with all rules that can be reached in the control horizon. The regions of the state space that can be reached in Nc steps (control horizon) from the current state $x(k)$ are limited, because the input is constrained on its maximum value and increment. These regions compose the reachable set at time k. This description can be prepared in advance during the design phase and can be converted in the first description proposed to improve the performance. This method is a good compromise between the two previously described. Observe that the reachable set can be generated dynamically according not only to the maximum and minimum value of the inputs, but also using the information about the predictions of the input sequence. This input sequence will tell which rules will be activated in the prediction horizon and this information will be used to build uncertainty description, i.e. the polytope of all active rules in the prediction horizon.

6.8 Conclusions and Perspectives

This chapter presented a formulation for Robust Nonlinear Predictive Control based on Takagi-Sugeno fuzzy models. The structure of the Takagi-Sugeno fuzzy models has been exploited to generate the formulation of the problem. The formulation converts the Nonlinear min-max program into a robust quadratic program (RQP). The chapter shows the way to convert the (RQP) into a Second Order Cone Program. This type of problems can be solved in a very efficient way by using interior point methods.
The chapter shows that by means of using Takagi Sugeno fuzzy models it is possible to design a Robust Nonlinear Predictive Controllers with a certain degree of transparency given by the local linear representations of the system and its uncertainties.
Future work must be devoted to refine the uncertainty description such that it will be more compact. The stability of these control schemes can be guaranteed by using end constraints; however, there is no formal proof to this statement.
The main contributions included in this chapter are: the formulation of the problem of Robust Nonlinear Predictive Control in terms of a Robust quadratic program and the extension of the formulation of the robust quadratic program as a Second Order Program by extending the uncertainties to the linear term of the cost function. Another contribution is the formulation of possible ways to obtain and represent the uncertainty.

Chapter 7

Conclusions and Future Perspectives

7.1 Conclusions and Contributions

This thesis has presented several contributions to a multitude of different topics of fuzzy systems. The second chapter presented the capabilities of fuzzy logic systems to approach non-linear functions. A detailed mathematical description of the constructing units has been presented and graphical examples have been used to expand the comprehension of the subject. The main techniques to approximate nonlinear functions by using fuzzy models have been presented. The expression of the gradients applied during the optimization process to adjust the parameters of the models has been derived. A new technique to improve the generalization capabilities of the fuzzy models as well as to overcome the lack of excitation are counted among the contributions presented in the second chapter. This technique guarantees a lower bound in the quality of the model (the fuzzy model will be at least as good as the best multi linear approximation).
The fact that the comparative advantage of fuzzy systems in comparison with other "universal approximators" is its linguistic interpretability explains the formulation of the AFRELI algorithm. The AFRELI algorithm in combination with the FuZion algorithm has been designed to guarantee a good trade-off between numerical accuracy and interpretability. The method exploits some successful elements proposed in other methods to reduce the complexity of the model construction.
The algorithm generates automatically the fuzzy sets from the data and the interactive labeling process (with intervention of the designer) guarantees an agreement between the fuzzy set and its semantic meaning.

The algorithm generates a rule base covering all the possible cases; this guarantees the completeness of the rule base, but the associated drawback is the exponential growth of the rule base as the number of inputs increases. However this is only a storage problem because the description of the fuzzy sets guarantees that only 2^N rules (N number of inputs) are activated on each inference. This fact makes the inference process fast because only a limited number of rules are evaluated.
The numerical performance of the model can be improved by making a "fine" tuning of the parameters of the antecedents by means of constrained gradient descent techniques.
The third chapter included as main contribution the formulation of the problem of system identification in the framework of fuzzy systems. The chapter presented the main issues in this area such as regressors selection, experiment design, structure, parameter adjusting and validation.
Extra effort for tasks such as regressor's selection is needed in order to counter the curse of dimensionality that arises as the number of inputs of the model increases. Some efficient methods have been presented in this chapter to trade off the complexity of the calculation with the accuracy of the solution.
The optimization of the parameters of the models is a task that involves the use of gradient descent techniques. The recursive structure of the dynamic models demands the use of gradients generated dynamically. The derivation of the dynamic systems that generate the gradients for the most common membership functions is also an important contribution presented in the third chapter.
The fourth chapter opens the second part of the thesis, which has been devoted to the subject of control using fuzzy logic. The fourth chapter presented several control synthesis techniques. The chapter begins with the description of the controller designed using pure expert knowledge. The chapter also explains the "paradoxical" success of this type of controllers in process control applications. An interesting contribution is a method to tune fuzzy controllers using the parameters of a previously designed PID controller. This contribution is very important because it is possible to guarantee that the fuzzy controller will be able to achieve at least the same performance as obtained with the PID controller. Other linear controllers can be converted into an equivalent fuzzy controller making smoother the migration from linear control towards nonlinear control. A proof of this statement is another contribution, which can be found in the appendix D. Other techniques are also shown in the chapter. Among those, it is important to mention techniques such as inverse modeling and model referenced adaptive control. Techniques such as feedback linearization are also explained. This technique is limited to a certain class of nonlinear systems (affine nonlinear systems) and has been criticized for its lack of robustness. However, a more elaborated control technique based on the same principle has been presented. This technique is the sliding mode control. This technique deserves

special attention; its robustness makes it a very good candidate for applications where the dynamics of plant are not very well known. Only rough bounds on the gain are needed to achieve an acceptable performance. Finally, the fuzzy gain scheduling technique was presented. This design technique is quite "elegant" from the analytical point of view. The use of advanced algorithms for semidefinite programming facilitates the solution of the LMIs generated by the synthesis problem. More advanced synthesis techniques include H^∞ criterion for disturbance rejection, robust synthesis and observer design can be developed by the same type of methods.

The fifth chapter has presented as a contribution novel approach to fuzzy model based control. The strategies combine two ideas that have been accepted by the industry for their simplicity and good performance. The ideas combined in this strategy are predictive control and fuzzy modeling. The development of these strategies has been preceded by a maturation of the modeling and identification techniques using fuzzy structures. The aim of this research was to design a simple control strategy. Simple from the designer's point of view and simple to operate and maintain.

The simplicity during the design phase comes from the fact that the design task is focused in the formulation of the goal of the control systems in terms of a quadratic cost function and the physical or the safety constraints imposed to the elements involved in the system. Other tuning parameters such as control and prediction horizons are closely related with the settling time and model order and typically exhibit a monotonic tendency making the tuning of the controller a simple task. Of course, the simplicity does not come for free. The strategy concentrates the effort in two tasks. One task is the effort that must be put to build the model and the other task is the operation of the control system, which is computationally intensive, compared with more traditional controllers. For instance, the sampling time with the current computational technology (SUN-Ultra-2 Creator 200 MHz) has been limited to 0.05 seconds for unconstrained systems and around 3 seconds for constrained multivariable systems of low order. This will limit the applications where these strategies can be applied nowadays; applications to process control problems where sampling time is about 30 sec. can be implemented without problems. Despite this limitation, Moore's law guarantees that in the term of 3 to 4 years the control strategy can be applied even to mechanic systems with fast dynamics.

The contributions presented in this chapter include algorithms for unconstrained and constrained systems.

The algorithms presented in this chapter can be extended in some cases to other model structures using other universal approximators (Neural Networks, Volterra series, splines, etc.) or even physical models.

The central aspect of the strategies presented in the fifth chapter is the reduction of the original complex nonlinear program (with no possibility for on-line

implementation), to a simple Quadratic Program (on-line implementable) by exploiting the structure of the problem and the structure of the models. The relaxation introduced by the methods presented in this chapter can be explained as a relaxation of the equality constraints imposed by the plant description. In some cases, these approximated solutions can be quite close to the "global optimal solution" of the original nonlinear program. The assumptions used to reduce the problem demand some smoothness in the nonlinearities, which are directly related with the prediction capability of the model.

Further search was motivated by issues such as the error modeling as well as the approximate method to solve the problem on-line. This resulted in the text of the sixth chapter, where a formulation for Robust Nonlinear Predictive Control based on Takagi-Sugeno fuzzy models is presented. The structure of the Takagi-Sugeno fuzzy models has been exploited to generate a formulation of the problem as a robust optimization problem. The robust solution guarantees that under the uncertainty described the constraints will not be violated. This fact is very important to guarantee stability especially when it is enforced by means of end constraints. The formulation generates a big set of constraints, however the performance of the optimization procedure is only slightly degraded because the solution of the optimization problem is found by means of interior point optimization algorithms. The chapter also presented some possible ways to formulate the uncertainty in the problem.

7.2 Perspectives and Future Work

This work has presented some contributions to the problems of modeling and control using fuzzy systems. There are many open issues and future work must be oriented to enhance the knowledge about some of the strategies presented in this thesis and to overcome some of the drawbacks and limitations mentioned in the preceding chapters. In the area of modeling, the curse of dimensionality is an open problem that has limited the application of fuzzy systems to systems with a small number of inputs. The solution to this problem must keep in mind the importance of the linguistic relevance of the solution. A promising strategy is the formulation of hierarchies; however, there is no systematic method to define the priority of one variable with respect to the others. Heuristic methods are the only available tools. Some type of measurements should be defined to guide the selection and the position in the hierarchy of the variables.

Another important problem that deserves attention is the design of clustering techniques with variable shapes such that the number of rules obtained after projection is optimal.

The identification problem presents a series of open problems. Important prob-

lems are model order estimation and more constructive regressor selection methods. New validation methods and a better link between existing methods and the ways to improve the quality of the model are also very important.
When the identification problem includes Takagi-Sugeno models, the selection of the variables governing the scheduling of the local models is also an open problem. So far, only heuristic methods have been proposed.
Even if the models obtained via identification are consistent from the linguistic point of view, the use of delayed variables ($x(k-n)$) is not very intuitive. A more intuitive description is in terms of tendencies of the variables ($\Delta x(k)$). A method to convert a model described in terms of delayed variables into a model described with tendencies of the variables and viceversa will be very useful to extract more information from the model.
Experiment design is a very important issue that must be investigated. This issue has special relevance when the system to be identified is very complex; for experiments are very expensive or take such a long time that their number will be very limited. Identification along trajectories is a very interesting issue in the chemical process industry. Here good experiment design is paramount due to the high costs of production.
Even though one of the earliest development on the application of fuzzy systems has been the design of controllers using expert knowledge, still there is no test for stability for this kind of controllers. Probably the main difficulty comes from the fact that mathematical knowledge about the plant is very limited, and a formal test of stability demands some knowledge about the plant. The proposed technique to design fuzzy controllers from PID controllers has no formal proof for stability. However, if the plant is a linear system of second order, it is possible to design a synthesis method that guarantees the stability of the system. The author is more skeptical as to whether such a method can be constructed for a linear plant with any order.
The sliding mode control strategy using fuzzy systems is a very promising strategy; so far, the limitation is that such a technique is applicable only to affine systems. Further research must be oriented to extend this method to more general nonlinear plants. Finally, the design of fuzzy controllers using LMIs has been considered conservative for the difficulty to find a common P matrix. Johansson et al [45] has proposed new alternatives. The relaxation of the problem is obtained by means of formulating a piecewise quadratic Lyapunov function; the potential of this idea seems very wide.
The idea of predictive control using fuzzy models has been studied extensively in this thesis. However there is enough space for improvement and future research. One very important issue is to study the performance of the algorithms and their possible modification such that other cost functions involving direct economic cost can be taken into account. Such cost function can be in some cases quite nonlinear and even discontinuous. New research must be oriented

to find ways to introduce other types of inequality constraints. So far, only linear constraints have been treated. Linear inequality constraints cover a big spectrum of constraints such as saturation and slew rate constraints. The use of nonlinear constraints can make possible the introduction of energy related aspects.
The author has explored the use of active set methods and interior point methods to solve the quadratic optimization programs generated by the presented algorithms. Interior point methods are very promising for their capacity to handle a large number of constraints. However, some problems must be solved before these techniques can be applied exploiting their full potential. The matrices involved in the optimization exhibit a regular structure that can be exploited to improve the performance of the optimization algorithms.
The Robust Predictive Control strategy demands more research about the way to define the uncertainty and how to obtain such information from the plant and/or the model. Finally, the stability of this control schemes can be guaranteed by using end constraints, however, there is no formal proof to this statement.

Appendix A

Fuzzy Set Theory

A.1 Introduction

The purpose of this appendix is to present an introduction to the concepts of fuzzy sets. The concepts of fuzzy sets arise as an answer to the problems of paradoxes, uncertainties and absence of precision found in crisp sets. Fuzzy sets are more inherent to nature than crisp sets. Crisp sets only consider elements with very well defined characteristics, such that a clear set boundary can be established. Many authors try to relate fuzziness to probabilities. In some cases fuzziness and probabilities can be treated with similar rules, but it is very important to recall that fuzziness presents the degree of belonging of one element to a certain set, while probabilities describe the behavior of many elements that belong to a certain set.

A.2 Fuzzy Sets

Let X denote the universal set. A conventional (crisp) set A is defined by a characteristic (membership) function $\mu(x)\,(x \in X)$ which assigns the values 1 or 0 to each element $x \in X$, respectively if x belongs or does not belong to A. $\mu_A : X \rightarrow \{0,1\}$
A fuzzy set A is defined by a *membership function* $\mu_A : X \rightarrow [0,1]$ which describes the membership degree of the elements of A. Values of $\mu_A(x)$ closer to 1 denote higher degree of set membership.

A.2.1 Some examples of membership functions

Let $X = \Re$ for the statement 'x is around M'

- Delta membership:

$$\mu(x) = \begin{cases} 1 & x = M \\ 0 & \text{otherwise} \end{cases}$$

- Step membership:

$$\mu(x) = \begin{cases} 1 & M - M_1 \leq M \leq M + M_1 \\ 0 & \text{otherwise} \end{cases}$$

- Ramp or triangular membership:

$$\mu(x) = \begin{cases} 0 & x < M - M_1 \text{ or } x > M + M_1 \\ 1 + \frac{x-M}{M_1} & M - M_1 \leq x \leq M \\ 1 - \frac{x-M}{M_1} & M \leq x \leq M + M_1 \end{cases}$$

- Exponential membership:

$$\mu(x) = exp(-t|x - M|)$$

- Gaussian membership:

$$\mu(x) = exp(-t(x - M)^2)$$

A.3 Basic Definitions on Fuzzy Sets

A.3.1 Support

The support of a fuzzy set A over the universe X is defined as the crisp subset where the membership function $\mu_A(x)$ is larger than zero.

$$\text{Supp}(A) = \{x|\mu(x) > 0\}$$

A.3.2 Core

The core of a fuzzy set A over the universe X is defined as the crisp subset where the membership function $\mu_A(x)$ is equal to one.

$$\text{Core}(A) = \{x|\mu(x) = 1\}$$

A.3.3 Height

The height of a fuzzy set A is the supremum of its membership function:

$$\text{hgt}(A) = \sup_{x \in X} \mu_A(x)$$

A.3.4 Normal fuzzy set

The fuzzy set A is "normal" if exists at least one value of x in X such that $\mu_A(x) = 1$

A.3.5 α-cut

The α-cut of a fuzzy set A is defined as the crisp subset of X where $\mu_A(x) \leq \alpha$.

$$A_\alpha = \{x|\mu_A(x) \leq \alpha\}$$

A.3.6 Strict α-cut

The strict α-cut of a fuzzy set A is defined as the crisp subset of X where $\mu_A(x) < \alpha$.

A.3.7 Convexity

A fuzzy set A defined in $\Re^n$ is convex if each of its α-cuts is convex.

A.4 Operations on fuzzy sets

A.4.1 A is contained in B

A set A is contained in the set B or A is a subset of B, denoted by $A \subseteq B$, if

$$\mu_A(x) \leq \mu_B(x) \quad \forall x \in X$$

A subset is proper ($A \subset B$) if ($\mu_A(x) < \mu_B(x)$).

A.4.2 Complement, Negation

The membership function $\mu_{\bar{A}}(x)$ of the complement of A (denoted by $\bar{A}$) is defined by:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x), \quad \forall x \in X$$

The relative complement of the set A with respect to the set B is defined by:

$$\mu_{\bar{A}_B}(x) = \mu_B(x) - \mu_A(x), \; x \in X \text{ if } \mu_B(x) > \mu_A(x)$$

A.4.3 Intersection

The intersection of the set A with the set B is defined by:

$$A \cap B = \{x|x \in A \wedge x \in B\}$$

The most important operators for the intersection are:

Extreme operator: $\mu_{A \cap_1 B}(x) = \mu_A(x) \wedge \mu_B(x) = min\{\mu_A(x), \mu_B(x)\}$ $\forall x \in X$

Product operator: $\mu_{A \cap_2 B}(x) = \mu_A(x)\mu_B(x)$ $\forall x \in X$

For crisp sets:

$$\mu_{A \cap_1 B}(x) = \mu_{A \cap_2 B}(x)$$

For fuzzy sets:

$$\mu_{A \cap_1 B}(x) \geq \mu_{A \cap_2 B}(x)$$

Characteristics of the intersection operators:

- $\cap_1$ and $\cap_2$ are commutative.
- $\cap_1$ and $\cap_2$ are associative.
- Identity: $\mu_A \cap_i 1 = \mu_A$
- Absorption: $\mu_A \cap_i 0 = 0$
- $\cap_1$ is an idempotent operator. $\mu \cap_1 \mu = \mu$.
- $\cap_2$ is not an idempotent operator. $\mu \cap_2 \mu \neq \mu$, $\mu \cap_2 \mu \subset \mu$.
- For fuzzy sets the law of non-contradiction ($\mu \cap \bar{\mu} = 0$)does not hold. Example: Let $\mu = 0.5$, $\mu \cap_1 \bar{\mu} = 0.5$ or $\mu \cap_2 \bar{\mu} = 0.25$.

- Product intersection is a subset of minimum intersection: $(\mu_A \cap_1 \mu_B) \subset (\mu_A \cap_2 \mu_B)$
- For $\mu_A \subset \mu_B$, $\mu_A \cap_1 \mu_B = \mu_A$ and $\mu_A \cap_2 \mu_B \subset \mu_A$

A.4.4 Union

The union of the set A with the set B is defined by:

$$A \cup B = \{x | x \in A \vee x \in B\} \qquad \forall x \in X$$

The most important operators for the union operation are:

Extreme operator. $\mu_{A \cup_1 B}(x) = \mu_A(x) \vee \mu_B(x) = max\{\mu_A(x), \mu_B(x)\}$ $\forall x \in X$

Sum operator. $\mu_{A \cup_2 B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)$ $\forall x \in X$

For crisp sets:

$$\mu_{A \cup_1 B}(x) = \mu_{A \cup_2 B}(x)$$

For fuzzy sets:

$$\mu_{A \cup_1 B}(x) \leq \mu_{A \cup_2 B}(x)$$

Characteristics of the union operators:

- $\cup_1$ and $\cup_2$ are commutative.
- $\cup_1$ and $\cup_2$ are associative.
- Identity: $\mu_A \cup_i 0 = \mu_A$
- Absorption: $\mu_A \cup_i 1 = 1$
- $\cup_1$ is an idempotent operator. $\mu \cup_1 \mu = \mu$.
- $\cup_2$ is not an idempotent operator. $\mu \cup_2 \mu \neq \mu$, $\mu \cup_2 \mu \subset \mu$.
- For fuzzy sets the law of excluded middle $(\mu \cup \bar{\mu} = X)$ does not hold. Example: Let $\mu = 0.5$, $\mu \cup_1 \bar{\mu} = 0.5$ or $\mu \cup_2 \bar{\mu} = 0.75$.
- $(\mu_A \cup_1 \mu_B) \subset (\mu_A \cup_2 \mu_B)$
- For $\mu_A \subset \mu_B$, $\mu_A \cup_1 \mu_B = \mu_B$ and $\mu_A \cup_2 \mu_B \supset \mu_B$

A.5 Fuzzy relations

A relation represents the presence or absence of association, interaction or interconnection between the elements of two or more sets. A binary relation is any relation between two sets.
For example:

$$R(x,y) = (X \; and \; Y) \qquad x \in X, y \in Y$$

A fuzzy relation $R(x,y)$ is a fuzzy subset of $X \times Y$.
For membership function $\mu(x,y)$

$$R = \{\mu(x,y) : X \times Y \rightarrow [0,1]\}$$

or

$$R = \{(x,y), \mu_R(x,y)\} = \bigcup(x,y)\mu_R(x,y)$$

A fuzzy relation on sets $X_1, X_2, ..., X_n$ is a fuzzy subset of $X_1 \times X_2 \times ... \times X_n$

$$R_n = \{\mu(x_1, x_2, ..., x_n) : X_1 \times X_2 \times ... \times X_n \rightarrow [0,1]\}$$

or

$$R_n = \bigcup\{(x_1, x_2, ..., x_n), \mu_R(x_1, x_2, ..., x_n)\} : X_1 \times X_2 \times ... \times X_n \rightarrow [0,1]$$

A fuzzy relation can be represented by:

- Membership function.
- Matrix (table) if the number of elements is finite.
- Picture with gray shades corresponding to μ value.

A.5.1 Projection of fuzzy relations

In a binary fuzzy relation one can define two projections:

- First projection: $R^{(1)} = \{x, \max_y \mu(x,y)\} \; x,y \in X \times Y$
- Second projection: $R^{(2)} = \{y, \max_x \mu(x,y)\} \; x,y \in X \times Y$

The global projection (also known as the height of the relation) is given by:

$$R^g = \max_x \max_y \mu(x,y)$$

A fuzzy relation is called normal if the global projection is one.
The operations between fuzzy sets can be extended to relations, for instance intersection, union, etc.

A.5.2 Composition of relations

The combination of fuzzy sets and fuzzy relations is called *composition*
Given:

$$\begin{array}{ll} R(x,y) & (x,y) \in X \times Y \qquad R: X \times Y \to [0,1] \\ S(y,z) & (y,z) \in Y \times Z \qquad S: Y \times Z \to [0,1] \end{array}$$

Composition $C(x,z)$

- Max min composition:
  $\mu_c(x,z) = max\{min(\mu_R(x,y), \mu_S(y,z))\} \quad x \in X, y \in Y, z \in Z.$
- Max product composition:
  $\mu_c(x,z) = max\{\mu_R(x,y)\mu_S(y,z)\} \quad x \in X, y \in Y, z \in Z.$

A.6 Approximate Reasoning

A.6.1 Introduction

Approximate Reasoning is a well-known form of fuzzy logic and uses a group of inference rules. These inference rules use fuzzy propositions as premises. The output of the inference rules is a set of conclusions. In the case of crisp sets, these are either true or false, but when the inference rules use fuzzy premises the conclusions have a certain degree of truth.

A.6.2 Linguistic Variables

Lotfi Zadeh defines a linguistic variable as follows: "By a linguistic variable we mean a variable whose values are words or sentences in a natural or artificial language. For example, *Age* is a linguistic variable if its values are linguistic rather than numerical, i.e.*young, not young, very young, quite young, old, not very old and not very young*, etc, rather than 20, 21, 22,...", [87]
Driankov [21] represents a linguistic variable and the framework as:

$$\langle X, \pounds X, \mathfrak{X}, M_x \rangle$$

where X denotes the name of the symbolic variable, e.g., *age, height, speed, pressure, error, change of error*, etc. $\pounds X$ Is the set of linguistic values that X can take. In the case of the variable *age* A we have

$$\pounds A = \{very\ young,\ young,\ adult,\ mature,\ old,\ very\ old\}$$

In control systems, the linguistic variables are usually the *error* and the *change of error* and the set is usually represented by the set $\{NB, NM, NS, ZO, PS, PM, PB\}$

where NB means "Negative Big", NM "Negative Medium", ZO means "Zero" etc. $\pounds X$ is also called the term-set of X or the reference set of X. $\mathfrak{X}$ is the physical domain of the variables, for instance for the variable *age*, the domain could be $[0, 95]$. $\mathfrak{X}$ is also called U or the universe of discourse and can be continuous or discrete. M_x is a semantic function which gives an interpretation of the linguistic value in terms of the quantitative elements of $\mathfrak{X}$. It is the set of the membership function in the universe of discourse $\mathfrak{X}$ i.e.,

$$M_X : LX \to \widetilde{LX}$$

A.7 General Structure of a Fuzzy Inference System

The purpose of this section is to give an overview about the structure, characteristics and functioning of the fuzzy inference systems. The fuzzy inference system is an inference system based on linguistic rules sometimes generated by empirical knowledge. First the crisp values coming from quantitative measurements are converted to linguistic values ($\pounds X$). This process is called fuzzification. The fuzzification process uses the membership functions to make such a conversion. Figure A.1 shows the membership functions for the fuzzification of a crisp temperature.

A.7.1 Control rules as a knowledge representation

After the fuzzification, the control rules can be applied. These rules are presented in an *IF .. AND.. THEN ..OR ELSE* form and represent the set of decisions when the input variables belong to a certain fuzzy sets. The consequence of the rule is calculated as a max − min composition.
if $\pounds A_1$ and $\pounds B_1$ and .. and $\pounds N_1$ then $\pounds U_1$
or else
if $\pounds A_1$ and $\pounds B_1$ and .. and $\pounds N_2$ then $\pounds U_2$
or else
⋮
or else
if $\pounds A_N$ and $\pounds B_N$ and .. and $\pounds N_N$ then $\pounds U_N$

Represented in the form of a fuzzy relation:

$$R: R_1 \cup R_2 \cup \ldots \cup R_N = \bigcup_{i=1}^{N} (\pounds A_i \times \pounds B_i \times \ldots \times \pounds N_i \times \pounds U_i)$$

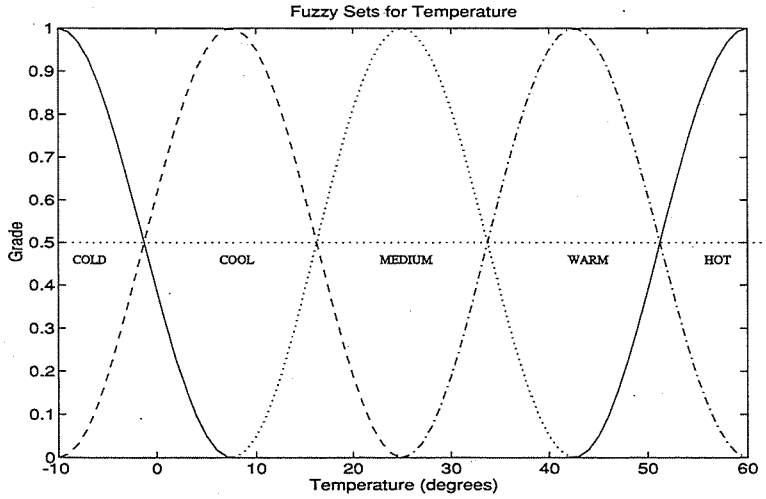


Figure A.1: *Fuzzy sets for temperature*

Because the AND operation can be represented as a min operation and the OR can be represented as the max operation a rule can be represented by:

$$R(\mu_A(l), \mu_B(j), \ldots, \mu_U(k)) = \max_{1 \leq i \leq N} \{\mathcal{L}A_l(\mu_A(l)) \wedge \mathcal{L}B_j(\mu_B(j)) \wedge \ldots \wedge \mathcal{L}U_i(\mu_U(k))\}$$

A.7.2 Defuzzification

After the inference process, the result of the process has to be converted to a crisp value. This procedure is called *Defuzzification*. Some defuzzification methods are:

- Center-of-Area/ Gravity
- Center-of-Sums
- Height

The influence of the mentioned defuzzification method on the controller performance is negligible. In the next lines are presented the different defuzzification methods.

Center-of-Area/ Gravity

For discrete systems:

$$x^* = \frac{\sum_{i=1}^{n} x_i \mu_U(x_i)}{\sum_{i=1}^{n} \mu_U(x_i)}$$

For continuous membership function:

$$x^* = \frac{\int_U x \mu_U(x) dx}{\int_U \mu_U(x) dx}$$

Center-of-Sums

For discrete systems:

$$x^* = \frac{\sum_{i=1}^{n} x_i \sum_{k=1}^{l} \mu_k(x_i)}{\sum_{i=1}^{n} \sum_{k=1}^{l} \mu_k(x_i)}$$

For continuous membership function:

$$x^* = \frac{\int_U x \sum_{k=1}^{l} \mu_k(x) dx}{\int_U \sum_{k=1}^{l} \mu_k(x) dx}$$

Height

This method takes the peak value of each consequence and makes a weighted sum of these peak values, where the weights are the degree of membership of the fired rule. The method is equal to the Center of Gravity when the consequence membership functions are singletons. Using a singleton as the membership functions of the consequences, the membership function of the i-th consequence is:

$$\mu_i(x) = 1 \quad \text{if} x = x_i$$
$$\mu_i(x) = 0 \quad \text{if} x \neq x_i$$

The defuzzification using this method will generate the following expression:

$$x^* = \frac{\sum_{i=1}^{L} x_i \mu_i(x)}{\sum_{i=1}^{L} \mu_i(x)}$$

This method is computationally fast and it generates continuous values making it very useful for function approximation.

Appendix B

Clustering Methods

Clustering methods are a set of techniques to reduce groups of information X represented as p- dimensional vectors into characteristic sets A_i characterized by feature vectors $v_i \in \Re^p$ and membership functions μ_A. The applications of these techniques include pattern recognition, classification and the applications presented on this thesis on modeling and identification. This appendix only presents the methods that are used in the thesis and it is by no means a complete survey of these techniques.

B.1 Fuzzy C- Means [8]

The fuzzy C-means clustering algorithm is based on the minimization of the cost function:

$$\min_{(U,V)} \left\{ J_m(U,V;X) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m ||x_k - v_i||_A^2 \right\} \tag{B.1}$$

where X is the set of vectors, $x_k \in \Re^p$ with the information, $V = [v_1, \ldots, v_c]$ is the set of feature vector, $||.||_A$ is the norm of the vector defined as $x^T A x$ where A is assumed to be the identity matrix and $U \in M_{fc}$ is the fuzzy partition matrix, defined as an element of the set:

$$M_{fc} = \left\{ U \in \Re^{c \times N} | \mu_{ik} \in [0,1], \forall i,k; \sum_{i=1} c\mu_{ik} = 1, \forall k; 0 < \sum_{k=1}^{N} \mu ik < N, \forall i \right\} \tag{B.2}$$

the i-th row of the fuzzy partition matrix contains the membership values of the vectors x to the A_i fuzzy set. The elements of U are calculated as:

$$u_{ik} = \left[\sum_{j=1}^{c} \left(\frac{||x_k - v_i||_A}{||x_k - v_j||_A} \right)^{\frac{2}{m-1}} \right]^{-1} \quad \forall i,k \tag{B.3}$$

the prototypes v_i are calculated as:

$$v_i = \frac{\sum_{k=1}^{n} (u_{ik})^m x_k}{\sum_{k=1}^{n} (u_{ik})^m} \quad \forall i \tag{B.4}$$

The fuzzy C means algorithm works as follows:

Algorithm Fuzzy C-Means

Given the data set X with N vectors, select the number of clusters $1 < c < N$, the exponent m, the termination tolerance $\epsilon > 0$ the matrix A to calculate the induced norm, and initialize the matrix U randomly such that $U^{(0)} \in M_{fc}$.

- **Repeat for** $j = 1, 2, \ldots$
- *Step 1:***Calculate the prototypes:**

$$v_i^{(j)} = \frac{\sum_{k=1}^{n} (u_{ik}^{(j-1)})^m x_k}{\sum_{k=1}^{n} (u_{ik})^m} \quad 1 \leq i \leq c \tag{B.5}$$

- *Step 2:***Calculate fuzzy partition matrix :**

$$u_{ik}^{(j)} = \left[\sum_{j=1}^{c} \left(\frac{||x_k - v_i^{(j)}||_A}{||x_k - v_j^{(j)}||_A} \right)^{\frac{2}{m-1}} \right]^{-1} \quad 1 \leq i \leq c, 1 \leq k \leq N \tag{B.6}$$

- **Until** $||U^{(j)} - U^{(j-1)}|| < \epsilon$

It is important to remark that when the vector x_k is equal to one of the prototypes v_i the expression B.5 becomes singular. For this case the membership value u_{ik} for this vector is equal to one and zero for all the other entries in the k-th row of U.
The parameter m is a very important parameter and as $m \to \infty$ the means of the clusters tend to the mean of the set X.

B.2 Mountain Clustering [84]

In this algorithm a super-set of the feature vectors V is proposed in advance, then some vectors are selected according to the value of the mountain function calculated for the given vector v_i. The mountain function is defined as:

$$M(v_i) = \sum_{k=1}^{N} e^{-(\alpha d(v_i, x_k))} \tag{B.7}$$

where α is a positive constant and $d(v_i, x_k)$ is a distance measure from v_i to x_k and it is typically but not necessarily:

$$d(v_i, x_k) = ||v_i - x_k||_2 \tag{B.8}$$

Two of the most popular elections of the feature vector candidates are:

- Take the set of feature candidates equal to the set of data points $V = X$. This is not very efficient for large data sets.

- Take a grid (arbitrary) defined in the interval where the points of X are defined. This is not very efficient for vectors defined on a large dimensional space.

Compared with other clustering methods the mountain clustering method has as main advantage the fact that the number of clusters does not need to be defined in advance. The algorithm works as follows:

Algorithm Mountain Clustering

Given the data set X with N vectors, select the parameters α and β , and the termination tolerance ϵ the set of feature vector candidates V with c elements.

- *Step 1:***Calculate the initial mountain function**

$$M^{(0)}(v_i) = \sum_{k=1}^{N} e^{-(\alpha d(v_i, x_k))} \; 1 \leq i \leq c \tag{B.9}$$

- **Repeat for** $j = 1, 2, \ldots$
- *Step 2:***Find the largest mountain value:**

$$M^{(j-1)*} = \max_{v_i} M^{(j-1)}(v_i) \tag{B.10}$$

- *Step 3:***Define the location of the maximum of the mountain value as the center of the** $j-1$ **cluster:**

$$v^{(j-1)*} = \arg\max_{v_i} M^{(j-1)}(v_i) \tag{B.11}$$

- *Step 4:***Calculate the revised mountain function** $M^(j)(v_i)$ **:**

$$M^{(j)}(v_i) = M^{(j-1)}(v_i) - M^{(j-1)*} \sum_{k=1}^{N} e^{-(\beta d(v^{(j-1)*}, x_k))} \; 1 \leq i \leq c \tag{B.12}$$

- **Until** $M^{(j-1)*} < \epsilon$

Appendix C

Gradient Expressions Used in Identification with Fuzzy Models

The gradients derived on this appendix are useful in the optimization of dynamic models. The cost function to be minimized is the quadratic cost function and it is defined as

$$V_N(\theta) = \frac{1}{2N}\sum_{t=1}^{N}|y(t) - \hat{y}(t|\theta)|^2 \tag{C.1}$$

where $y(t)$ is the output of the "real" system at time t,

$$\hat{y}(t|\theta) = f(\varphi(t), \theta) \tag{C.2}$$

$\hat{y}(t|\theta)$ is the output of the constructed model parameterized by the vector θ. The vector θ describes the membership functions and the position of the singletons in the consequences and

$$\varphi(t) = [y(t-1), \ldots, y(t-m), \hat{y}(t-1), \ldots, \hat{y}(t-n), u(t), \ldots, u(t-k), \varepsilon(t-1), \ldots, \varepsilon(t-l)]$$

is the set of regressors of the model. The derivations shown in this appendix can be applied to the structures NFIR, NARX, NOE, NARMAX and NBJ.
This appendix shows initially the derivation of the gradient for the consequences of the rules; in the second part it shows the derivation for the parameters of different types of membership functions.

C.1 Gradient for the singleton consequences

The expression for the gradient is given by:

$$\frac{\partial V_N(\theta)}{\partial \bar{y}^l} = \frac{1}{N}\sum_{t=1}^{N}(y(t) - \hat{y}(t|\theta))(-\frac{\partial \hat{y}(t|\theta)}{\partial \bar{y}^l}) \tag{C.3}$$

with

$$\frac{\partial \hat{y}(t|\theta)}{\partial \bar{y}^l} = \bar{y}^l\frac{\partial w_l(\varphi(t))}{\partial \bar{y}^l} + w_l(\varphi(t)) \tag{C.4}$$

observe that if the model is NARX or NFIR the term $\frac{\partial w_l(\varphi(t))}{\partial \bar{y}^l} = 0$ and the expression for the gradient will be the same used for static function approximation. The term $\frac{\partial w_l(\varphi(t))}{\partial \bar{y}^l}$ is dependent from previous gradient values and must be generated dynamically.

$$\frac{\partial w_l(\varphi(t))}{\partial \bar{y}^l} = \frac{\frac{\partial \mu_l(\varphi(t))}{\partial \bar{y}^l}\sum_{i=1}^{L}\mu_i(\varphi(t)) - \mu_l(\varphi(t))\frac{\partial \sum_{i=1}^{L}\mu_i(\varphi(t))}{\partial \bar{y}^l}}{\sum_{i=1}^{L}\mu_i(\varphi(t))} \tag{C.5}$$

where

$$\begin{aligned}\frac{\partial \mu_l(\varphi(t))}{\partial \bar{y}^l} &= \sum_{i\in\mathcal{Y}}\frac{\mu_l(\varphi(t))}{\mu_l^i(\hat{y}(t-k(i)))}\frac{\partial \mu_l^i(\hat{y}(t-k(i)))}{\partial \bar{y}^l} \\ &+ \sum_{j\in\mathcal{E}}\frac{\mu_l(\varphi(t))}{\mu_l^j(\varepsilon(t-m(j)))}\frac{\partial \mu_l^j(\varepsilon(t-m(j)))}{\partial \bar{y}^l}\end{aligned} \tag{C.6}$$

where $\mathcal{Y}$ represents the set of inputs related with the regressors $\hat{y}(.)$ with delay $k(i)$ and $\mathcal{E}$ represent the set of inputs related with the regressors $\varepsilon(.)$ with delay $m(j)$, and

$$\begin{aligned}\frac{\sum_{i=1}^{L}\mu_i(\varphi(t))}{\partial \bar{y}^l} &= \sum_{l=1}^{L}\left\{\sum_{i\in\mathcal{Y}}\frac{\mu_l(\varphi(t))}{\mu_l^i(\hat{y}(t-k(i)))}\frac{\partial \mu_l^i(\hat{y}(t-k(i)))}{\partial \bar{y}^l}\right. \\ &\left. + \sum_{j\in\mathcal{E}}\frac{\mu_l(\varphi(t))}{\mu_l^j(\varepsilon(t-m(j)))}\frac{\partial \mu_l^j(\varepsilon(t-m(j)))}{\partial \bar{y}^l}\right\}\end{aligned} \tag{C.7}$$

finally according to the type of membership functions used the terms $\frac{\partial \mu_l^j(\varepsilon(t-m(j)))}{\partial \bar{y}^l}$ and $\frac{\partial \mu_l^i(\hat{y}(t-k(i)))}{\partial \bar{y}^l}$ will have the following expressions:

C.1.1 with Trapezoidal Membership Functions

For *trapezoidal membership functions* using the parameterization given in the equation (2.42):

$$\frac{\partial \mu_l^i(\hat{y}(t-k(i)))}{\partial \bar{y}^l} = \begin{cases} 0 & \text{if } \hat{y}(t-k(i)) < a_l^i \\ \frac{1}{b_l^i - a_l^i}\frac{\partial \hat{y}(t-k(i))}{\partial \bar{y}^l} & \text{if } a_l^i < \hat{y}(t-k(i)) < b_l^i \\ 0 & \text{if } b_l^i < \hat{y}(t-k(i)) < c_l^i \\ \frac{-1}{d_l^i - c_l^i}\frac{\partial \hat{y}(t-k(i))}{\partial \bar{y}^l} & \text{if } c_l^i < \hat{y}(t-k(i)) < d_l^i \\ 0 & \text{if } \hat{y}(t-k(i)) > d_l^i \end{cases} \tag{C.8}$$

$$\frac{\partial \mu_l^j(\varepsilon(t-m(j)))}{\partial \bar{y}^l} = \begin{cases} 0 & \text{if } \varepsilon(t-m(j)) < a_l^j \\ \frac{-1}{b_l^j - a_l^j}\frac{\partial \hat{y}(t-m(j))}{\partial \bar{y}^l} & \text{if } a_l^j < \varepsilon(t-m(j)) < b_l^j \\ 0 & \text{if } b_l^j < \varepsilon(t-m(j)) < c_l^j \\ \frac{1}{d_l^j - c_l^j}\frac{\partial \hat{y}(t-m(j))}{\partial \bar{y}^l} & \text{if } c_l^j < \varepsilon(t-m(j)) < d_l^j \\ 0 & \text{if } \varepsilon(t-m(j)) > d_l^j \end{cases} \tag{C.9}$$

C.2 with Polynomial Membership Functions

For *polynomial membership functions* using the parameterization given in the equation (2.56)

$$\frac{\partial \mu_l^i(\hat{y}(t-k(i)))}{\partial \bar{y}^l} =$$

$$\begin{cases} 0 & \text{if } \hat{y}(t-k(i)) < a_l^i \\ \frac{6[\hat{y}(t-k(i))^2 - (a_l^i + b_l^i)\hat{y}(t-k(i))] + {a_l^i}^3 - 3b{a_l^i}^2}{(a_l^i - b_l^i)^3}\frac{\partial \hat{y}(t-k(i))}{\partial \bar{y}^l} & \text{if } a_l^i < \hat{y}(t-k(i)) < b_l^i \\ 0 & \text{if } b_l^i < \hat{y}(t-k(i)) < c_l^i \\ \frac{6[\hat{y}(t-k(i))^2 - (d_l^i + c_l^i)\hat{y}(t-k(i))] + {d_l^i}^3 - 3c{d_l^i}^2}{(d_l^i - c_l^i)^3}\frac{\partial \hat{y}(t-k(i))}{\partial \bar{y}^l} & \text{if } c_l^i < \hat{y}(t-k(i)) < d_l^i \\ 0 & \text{if } \hat{y}(t-k(i)) > d_l^i \end{cases} \tag{C.10}$$

$$\frac{\partial \mu_l^j(\varepsilon(t-m(j)))}{\partial \bar{y}^l} =$$

$$\begin{cases} 0 & \text{if } \varepsilon(t-m(j)) < a_l^j \\ \frac{6[(a_l^j + b_l^j)\varepsilon(t-m(j)) - \varepsilon(t-m(j))^2] - {a_l^j}^3 + 3b{a_l^j}^2}{(a_l^j - b_l^j)^3}\frac{\partial \hat{y}(t-m(j))}{\partial \bar{y}^l} & \text{if } a_l^j < \varepsilon(t-m(j)) < b_l^j \\ 0 & \text{if } b_l^j < \varepsilon(t-m(j)) < c_l^j \\ \frac{6[(d_l^j + c_l^j)\varepsilon(t-m(j)) - \varepsilon(t-m(j))^2] - {d_l^j}^3 + 3c{d_l^j}^2}{(d_l^j - c_l^j)^3}\frac{\partial \hat{y}(t-m(j))}{\partial \bar{y}^l} & \text{if } c_l^j < \varepsilon(t-m(j)) < d_l^j \\ 0 & \text{if } \varepsilon(t-m(j)) > d_l^j \end{cases}$$

(C.11)

C.2.1 with Gaussian Membership Functions

For *Gaussian membership functions* using the parameterization presented in (2.70)

$$\frac{\partial \mu_l^i(\hat{y}(t-k(i)))}{\partial \bar{y}^l} = -2\frac{(\hat{y}(t-k(i)) - \bar{x}_l^i)}{{\sigma_l^i}^2}\mu_l^i(\hat{y}(t-k(i)))\frac{\partial \hat{y}(t-k(i))}{\partial \bar{y}^l} \tag{C.12}$$

$$\frac{\partial \mu_l^j(\varepsilon(t-m(j)))}{\partial \bar{y}^l} = 2\frac{(\varepsilon(t-m(j)) - \bar{x}_l^j)}{{\sigma_l^j}^2}\mu_l^j(\varepsilon(t-m(j)))\frac{\partial \hat{y}(t-m(j))}{\partial \bar{y}^l} \tag{C.13}$$

C.3 Gradient for the parameters of the membership functions

The expression for the gradient is given by:

$$\frac{\partial V_N(\theta)}{\partial \bar{y}^l} = \frac{1}{N}\sum_{t=1}^{N}(y(t) - \hat{y}(t|\theta))(-\frac{\partial \hat{y}(t|\theta)}{\partial \alpha}) \tag{C.14}$$

where $\alpha \subset \theta$ represents all the adjustable parameter in the model excluding the consequences, and

$$\frac{\partial \hat{y}(t|\theta)}{\partial \alpha_n} = \frac{1}{\sum_{l=1}^{L}\mu_l(\varphi(t))}\sum_{l\in\mathcal{U}}(\bar{y}^l - \hat{y}(t|\theta))\frac{\mu_l(\varphi(t))}{\mu_j^i(\hat{y}(t-k(l)))}\frac{\partial \mu_j^i(\hat{y}(t-k(l)))}{\partial \alpha_n} \tag{C.15}$$

where α_n is a parameter of the membership function $\mu_j^i(.)$ and $\mathcal{U}$ is the set of rules that include in the antecedents the membership function $\mu_j^i(.)$.
According to the type of membership functions and the regressors the term $\frac{\partial \mu_j^i(\hat{y}(t-k(l)))}{\partial \alpha_n}$ will have expressions which are presented in the following lines.

C.3.1 with Trapezoidal Membership Functions

For *trapezoidal membership functions* α_n could be $a_i^j, b_i^j, c_i^j, d_i^j$ and the gradients will be

$$\frac{\partial \mu_j^i(\hat{y}(t-k(l)))}{\partial a_i^j} = \begin{cases} 0 & \hat{y}(t-k(l)) < a_i^j \\ \frac{\frac{\partial \hat{y}(t-k(l))}{\partial a_i^j}(b_i^j - a_i^j) + (\hat{y}(t-k(l)) - b_i^j)}{(b_i^j - a_i^j)^2} & a_i^j < \hat{y}(t-k(l)) < b_i^j \\ 0 & \hat{y}(t-k(l)) > b_i^j \end{cases} \tag{C.16}$$

$$\frac{\partial \mu_j^i(\varepsilon(t-m(l)))}{\partial a_i^j} = \begin{cases} 0 & \varepsilon(t-m(l)) < a_i^j \\ \frac{-\frac{\partial \hat{y}(t-m(l))}{\partial a_i^j}(b_i^j - a_i^j) + (\varepsilon(t-m(l)) - b_i^j)}{(b_i^j - a_i^j)^2} & a_i^j < \varepsilon(t-m(l)) < b_i^j \\ 0 & \varepsilon(t-m(l)) > b_i^j \end{cases} \tag{C.17}$$

$$\frac{\partial \mu_j^i(\hat{y}(t-k(l)))}{\partial b_i^j} = \begin{cases} 0 & \hat{y}(t-k(l)) < a_i^j \\ \frac{\frac{\partial \hat{y}(t-k(l))}{\partial b_i^j}(b_i^j - a_i^j) - (\hat{y}(t-k(l)) - a_i^j)}{(b_i^j - a_i^j)^2} & a_i^j < \hat{y}(t-k(l)) < b_i^j \\ 0 & \hat{y}(t-k(l)) > b_i^j \end{cases} \tag{C.18}$$

$$\frac{\partial \mu_j^i(\varepsilon(t-m(l)))}{\partial b_i^j} = \begin{cases} 0 & \varepsilon(t-m(l)) < a_i^j \\ \frac{-\frac{\partial \hat{y}(t-m(l))}{\partial b_i^j}(b_i^j - a_i^j) - (\varepsilon(t-m(l)) - a_i^j)}{(b_i^j - a_i^j)^2} & a_i^j < \varepsilon(t-m(l)) < b_i^j \\ 0 & \varepsilon(t-m(l)) > b_i^j \end{cases} \tag{C.19}$$

$$\frac{\partial \mu_j^i(\hat{y}(t-k(l)))}{\partial c_i^j} = \begin{cases} 0 & \hat{y}(t-k(l)) < c_i^j \\ \frac{-\frac{\partial \hat{y}(t-k(l))}{\partial c_i^j}(d_i^j - c_i^j) + (d_i^j - \hat{y}(t-k(l)))}{(d_i^j - c_i^j)^2} & c_i^j < \hat{y}(t-k(l)) < d_i^j \\ 0 & \hat{y}(t-k(l)) > d_i^j \end{cases} \tag{C.20}$$

$$\frac{\partial \mu_j^i(\varepsilon(t-m(l)))}{\partial c_i^j} = \begin{cases} 0 & \varepsilon(t-m(l)) < c_i^j \\ \frac{\frac{\partial \hat{y}(t-m(l))}{\partial c_i^j}(d_i^j - c_i^j) + (d_i^j - \varepsilon(t-m(l)))}{(d_i^j - c_i^j)^2} & c_i^j < \varepsilon(t-m(l)) < d_i^j \\ 0 & \varepsilon(t-m(l)) > d_i^j \end{cases} \tag{C.21}$$

$$\frac{\partial \mu_j^i(\hat{y}(t-k(l)))}{\partial d_i^j} = \begin{cases} 0 & \hat{y}(t-k(l)) < c_i^j \\ \frac{\frac{\partial \hat{y}(t-k(l))}{\partial d_i^j}(d_i^j - c_i^j) - (d_i^j - \hat{y}(t-k(l)))}{(d_i^j - c_i^j)^2} & c_i^j < \hat{y}(t-k(l)) < d_i^j \\ 0 & \hat{y}(t-k(l)) > d_i^j \end{cases} \tag{C.22}$$

$$\frac{\partial \mu_j^i(\varepsilon(t-m(l)))}{\partial d_i^j} = \begin{cases} 0 & \varepsilon(t-m(l)) < c_i^j \\ -\frac{\frac{\partial \hat{y}(t-m(l))}{\partial d_i^j}(d_i^j - c_i^j) - (d_i^j - \varepsilon(t-m(l)))}{(d_i^j - c_i^j)^2} & c_i^j < \varepsilon(t-m(l)) < d_i^j \\ 0 & \varepsilon(t-m(l)) > d_i^j \end{cases} \tag{C.23}$$

C.3.2 with Polynomial Membership Functions

For *polynomial membership functions* α_n could be $a_i^j, b_i^j, c_i^j, d_i^j$ and the gradients will be

$$\frac{\partial \mu_j^i(\hat{y}(t-k(l)))}{\partial a_i^j} = \begin{cases} 0 & \text{if } \hat{y}(t-k(l)) < a_i^j \\ 6\frac{(\hat{y}(t-k(l))-b_i^j)(\hat{y}(t-k(l))-a_i^j)}{(a_i^j-b_i^j)^3}\left[\frac{\partial \hat{y}(t-k(l))}{\partial a_i^j} - \frac{(\hat{y}(t-k(l))-b_i^j)}{a_i^j-b_i^j}\right] & \text{if } a_i^j < \hat{y}(t-k(l)) < b_i^j \\ 0 & \text{if } \hat{y}(t-k(l)) > b_i^j \end{cases} \tag{C.24}$$

$$\frac{\partial \mu_j^i(\varepsilon(t-m(l)))}{\partial a_i^j} = \begin{cases} 0 & \text{if } \varepsilon(t-m(l)) < a_i^j \\ 6\frac{(\varepsilon(t-m(l))-b_i^j)(\varepsilon(t-m(l))-a_i^j)}{(a_i^j-b_i^j)^3}\left[-\frac{\partial \hat{y}(t-m(l))}{\partial a_i^j} - \frac{(\hat{y}(t-k(l))-b_i^j)}{a_i^j-b_i^j}\right] & \text{if } a_i^j < \varepsilon(t-m(l)) < b_i^j \\ 0 & \text{if } \varepsilon(t-m(l)) > b_i^j \end{cases} \tag{C.25}$$

$$\frac{\partial \mu_j^i(\hat{y}(t-k(l)))}{\partial b_i^j} = \begin{cases} 0 & \text{if } \hat{y}(t-k(l)) < a_i^j \\ 6\frac{(\hat{y}(t-k(l))-b_i^j)(\hat{y}(t-k(l))-a_i^j)}{(a_i^j-b_i^j)^3}\left[\frac{\partial \hat{y}(t-k(l))}{\partial b_i^j} + \frac{(\hat{y}(t-k(l))-a_i^j)}{a_i^j-b_i^j}\right] & \text{if } a_i^j < \hat{y}(t-k(l)) < b_i^j \\ 0 & \text{if } \hat{y}(t-k(l)) > b_i^j \end{cases} \tag{C.26}$$

$$\frac{\partial \mu_j^i(\varepsilon(t-m(l)))}{\partial b_i^j} = \begin{cases} 0 & \text{if } \varepsilon(t-m(l)) < a_i^j \\ 6\frac{(\varepsilon(t-m(l))-b_i^j)(\varepsilon(t-m(l))-a_i^j)}{(a_i^j-b_i^j)^3}\left[-\frac{\partial \hat{y}(t-m(l))}{\partial b_i^j} + \frac{(\hat{y}(t-k(l))-a_i^j)}{a_i^j-b_i^j}\right] & \text{if } a_i^j < \varepsilon(t-m(l)) < b_i^j \\ 0 & \text{if } \varepsilon(t-m(l)) > b_i^j \end{cases} \tag{C.27}$$

$$\frac{\partial \mu_j^i(\hat{y}(t-k(l)))}{\partial c_i^j} =$$

$$\begin{cases} 0 & \text{if } \hat{y}(t-k(l)) < c_i^j \\ 6\frac{(\hat{y}(t-k(l))-d_i^j)(\hat{y}(t-k(l))-c_i^j)}{(c_i^j-d_i^j)^3}\left[-\frac{\partial\hat{y}(t-k(l))}{\partial c_i^j}+\frac{(\hat{y}(t-k(l))-d_i^j)}{c_i^j-d_i^j}\right] & \text{if } c_i^j < \hat{y}(t-k(l)) < d_i^j \\ 0 & \text{if } \hat{y}(t-k(l)) > d_i^j \end{cases} \tag{C.28}$$

$$\frac{\partial\mu_j^i(\varepsilon(t-m(l)))}{\partial c_i^j} = \begin{cases} 0 & \text{if } \varepsilon(t-m(l)) < c_i^j \\ 6\frac{(\varepsilon(t-m(l))-d_i^j)(\varepsilon(t-m(l))-c_i^j)}{(c_i^j-d_i^j)^3}\left[\frac{\partial\hat{y}(t-m(l))}{\partial c_i^j}+\frac{(\hat{y}(t-k(l))-d_i^j)}{c_i^j-d_i^j}\right] & \text{if } c_i^j < \varepsilon(t-m(l)) < d_i^j \\ 0 & \text{if } \varepsilon(t-m(l)) > d_i^j \end{cases} \tag{C.29}$$

$$\frac{\partial\mu_j^i(\hat{y}(t-k(l)))}{\partial d_i^j} = \begin{cases} 0 & \text{if } \hat{y}(t-k(l)) < c_i^j \\ -6\frac{(\hat{y}(t-k(l))-d_i^j)(\hat{y}(t-k(l))-c_i^j)}{(c_i^j-d_i^j)^3}\left[\frac{\partial\hat{y}(t-k(l))}{\partial d_i^j}+\frac{(\hat{y}(t-k(l))-c_i^j)}{c_i^j-d_i^j}\right] & \text{if } c_i^j < \hat{y}(t-k(l)) < d_i^j \\ 0 & \text{if } \hat{y}(t-k(l)) > d_i^j \end{cases} \tag{C.30}$$

$$\frac{\partial\mu_j^i(\varepsilon(t-m(l)))}{\partial d_i^j} = \begin{cases} 0 & \text{if } \varepsilon(t-m(l)) < c_i^j \\ -6\frac{(\varepsilon(t-m(l))-d_i^j)(\varepsilon(t-m(l))-c_i^j)}{(c_i^j-d_i^j)^3}\left[-\frac{\partial\hat{y}(t-m(l))}{\partial d_i^j}+\frac{(\hat{y}(t-k(l))-c_i^j)}{c_i^j-d_i^j}\right] & \text{if } c_i^j < \varepsilon(t-m(l)) < d_i^j \\ 0 & \text{if } \varepsilon(t-m(l)) > d_i^j \end{cases} \tag{C.31}$$

C.3.3 with Gaussian Membership Functions

With *Gaussian Membership Functions* α_n could be $\bar{x}_j^i, \sigma_j^i$ and the gradients will be:

$$\frac{\partial\mu_j^i(\hat{y}(t-k(l)))}{\partial\bar{x}_j^i} = -2\frac{(\hat{y}(t-k(l))-\bar{x}_j^i)}{\sigma_j^{i\,2}}\mu_j^i(\hat{y}(t-k(l)))\left[\frac{\partial\hat{y}(t-k(l))}{\partial\bar{x}_j^i}-1\right] \tag{C.32}$$

$$\frac{\partial\mu_j^i(\varepsilon(t-m(l)))}{\partial\bar{x}_j^i} = -2\frac{(\varepsilon(t-m(l)))-\bar{x}_j^i)}{\sigma_j^{i\,2}}\mu_j^i(\varepsilon(t-m(l)))\left[-\frac{\partial\hat{y}(t-m(l))}{\partial\bar{x}_j^i}-1\right] \tag{C.33}$$

$$\frac{\partial\mu_j^i(\hat{y}(t-k(l)))}{\partial\sigma_j^i} = -2\frac{(\hat{y}(t-k(l))-\bar{x}_j^i)}{\sigma_j^{i\,3}}\mu_j^i(\hat{y}(t-k(l)))\left[\sigma_j^i\frac{\partial\hat{y}(t-k(l))}{\partial\sigma_j^i}-(\hat{y}(t-k(l))-\bar{x}_j^i)\right] \tag{C.34}$$

$$\frac{\partial\mu_j^i(\varepsilon(t-m(l)))}{\partial\sigma_j^i} = -2\frac{(\varepsilon(t-m(l)))-\bar{x}_j^i)}{\sigma_j^{i\,3}}\mu_j^i(\varepsilon(t-m(l)))\left[-\sigma_j^i\frac{\partial\hat{y}(t-m(l))}{\partial\sigma_j^i}-(\varepsilon(t-m(l))-\bar{x}_j^i)\right] \tag{C.35}$$

C.3.4 with Triangular Membership Functions with 0.5 overlap

For *triangular membership functions with 0.5 overlap* the α_n parameters could be m_{j-1}, m_j, m_{j+1} and the derivatives will be given by:

$$\frac{\partial\mu_j^i(\hat{y}(t-k(l)))}{\partial m_j^i} = \begin{cases} 0 & \hat{y}(t-k(l)) < m_{j-1}^i \\ \frac{\frac{\partial\hat{y}(t-k(l))}{\partial m_j^i}(m_j^i-m_{j-1}^i)-(\hat{y}(t-k(l))-m_{j-1}^i)}{(m_j^i-m_{j-1}^i)^2} & m_{j-1}^i < \hat{y}(t-k(l)) < m_j^i \\ \frac{\frac{\partial\hat{y}(t-k(l))}{\partial m_j^i}(m_{j+1}^i-m_j^i)-(\hat{y}(t-k(l))-m_{j+1}^i)}{(m_{j+1}^i-m_j^i)^2} & m_j^i < \hat{y}(t-k(l)) < m_{j+1}^i \\ 0 & \hat{y}(t-k(l)) > m_{j+1}^i \end{cases} \tag{C.36}$$

$$\frac{\partial\mu_j^i(\varepsilon(t-m(l)))}{\partial m_j^i} = \begin{cases} 0 & \varepsilon(t-m(l)) < m_{j-1}^i \\ \frac{-\frac{\partial\hat{y}(t-m(l))}{\partial m_j^i}(m_j^i-m_{j-1}^i)-(\varepsilon(t-m(l))-m_{j-1}^i)}{(m_j^i-m_{j-1}^i)^2} & m_{j-1}^i < \varepsilon(t-m(l)) < m_j^i \\ \frac{-\frac{\partial\hat{y}(t-m(l))}{\partial m_j^i}(m_{j+1}^i-m_j^i)-(\varepsilon(t-m(l))-m_{j+1}^i)}{(m_{j+1}^i-m_j^i)^2} & m_j^i < \varepsilon(t-m(l)) < m_{j+1}^i \\ 0 & \varepsilon(t-m(l)) > m_{j+1}^i \end{cases} \tag{C.37}$$

$$\frac{\partial \mu^i_{j+1}(\hat{y}(t-k(l)))}{\partial m^i_j} = \begin{cases} 0 & \hat{y}(t-k(l)) < m^i_j \\ \frac{\frac{\partial \hat{y}(t-k(l))}{\partial m^i_j}(m^i_{j+1}-m^i_j)+(\hat{y}(t-k(l))-m^i_{j+1})}{(m^i_{j+1}-m^i_j)^2} & m^i_j < \hat{y}(t-k(l)) < m^i_{j+1} \\ 0 & \hat{y}(t-k(l)) > m^i_{j+1} \end{cases} \tag{C.38}$$

$$\frac{\partial \mu^i_{j+1}(\varepsilon(t-m(l)))}{\partial m^i_j} = \begin{cases} 0 & \varepsilon(t-m(l)) < m^i_j \\ \frac{-\frac{\partial \hat{y}(t-m(l))}{\partial m^i_j}(m^i_{j+1}-m^i_j)+(\varepsilon(t-m(l))-m^i_{j+1})}{(m^i_{j+1}-m^i_j)^2} & m^i_j < \varepsilon(t-m(l)) < m^i_{j+1} \\ 0 & \varepsilon(t-m(l)) > m^i_{j+1} \end{cases} \tag{C.39}$$

$$\frac{\partial \mu^i_{j-1}(\hat{y}(t-k(l)))}{\partial m^i_j} = \begin{cases} 0 & \hat{y}(t-k(l)) < m^i_{j-1} \\ \frac{\frac{\partial \hat{y}(t-k(l))}{\partial m^i_j}(m^i_j-m^i_{j-1})+(\hat{y}(t-k(l))-m^i_{j-1})}{(m^i_j-m^i_{j-1})^2} & m^i_{j-1} < \hat{y}(t-k(l)) < m^i_j \\ 0 & \hat{y}(t-k(l)) > m^i_j \end{cases} \tag{C.40}$$

$$\frac{\partial \mu^i_{j-1}(\varepsilon(t-m(l)))}{\partial m^i_j} = \begin{cases} 0 & \varepsilon(t-m(l)) < m^i_{j-1} \\ \frac{-\frac{\partial \hat{y}(t-m(l))}{\partial m^i_j}(m^i_j-m^i_{j-1})+(\varepsilon(t-m(l))-m^i_{j-1})}{(m^i_j-m^i_{j-1})^2} & m^i_{j-1} < \varepsilon(t-m(l)) < m^i_j \\ 0 & \varepsilon(t-m(l)) > m^i_j \end{cases} \tag{C.41}$$

C.3.5 with Polynomial Membership Functions with 0.5 overlap

For *polynomial membership functions with 0.5 overlap* the α_n parameters could be m_{j-1}, m_j, m_{j+1} and the derivatives will be given by:

$$\frac{\partial \mu^i_j(\hat{y}(t-k(l)))}{\partial m^i_j} =$$

$$\begin{cases} 0 & \text{if } \hat{y}(t-k(l)) < m^i_{j-1} \\ 6\frac{(\hat{y}(t-k(l))-m^i_{j-1})(\hat{y}(t-k(l))-m^i_j)}{(m^i_{j-1}-m^i_j)^3}\left[\frac{\partial \hat{y}(t-k(l))}{\partial m^i_j}+\frac{(\hat{y}(t-k(l))-m^i_{j-1})}{m^i_{j-1}-m^i_j}\right] & \text{if } m^i_{j-1} < \hat{y}(t-k(l)) < m^i_j \\ 6\frac{(\hat{y}(t-k(l))-m^i_{j+1})(\hat{y}(t-k(l))-m^i_j)}{(m^i_j-m^i_{j+1})^3}\left[-\frac{\partial \hat{y}(t-k(l))}{\partial m^i_j}+\frac{(\hat{y}(t-k(l))-m^i_{j+1})}{m^i_j-m^i_{j+1}}\right] & \text{if } m^i_j < \hat{y}(t-k(l)) < m^i_{j+1} \\ 0 & \text{if } \hat{y}(t-k(l)) > m^i_{j+1} \end{cases} \tag{C.42}$$

$$\frac{\partial \mu^i_j(\varepsilon(t-m(l)))}{\partial m^i_j} =$$

$$\begin{cases} 0 & \text{if } \varepsilon(t-m(l)) < m^i_{j-1} \\ 6\frac{(\varepsilon(t-m(l))-m^i_{j-1})(\varepsilon(t-m(l))-m^i_j)}{(m^i_{j-1}-m^i_j)^3}\left[-\frac{\partial \hat{y}(t-m(l))}{\partial m^i_j}+\frac{(\hat{y}(t-k(l))-m^i_{j-1})}{m^i_{j-1}-m^i_j}\right] & \text{if } m^i_{j-1} < \varepsilon(t-m(l)) < m^j_j \\ 6\frac{(\varepsilon(t-m(l))-m^i_{j+1})(\varepsilon(t-m(l))-m^i_j)}{(m^i_j-m^i_{j+1})^3}\left[\frac{\partial \hat{y}(t-m(l))}{\partial m^i_j}+\frac{(\hat{y}(t-k(l))-m^i_{j+1})}{m^i_j-m^i_{j+1}}\right] & \text{if } m^i_j < \varepsilon(t-m(l)) < m^j_{j+1} \\ 0 & \text{if } \varepsilon(t-m(l)) > m^i_{j+1} \end{cases} \tag{C.43}$$

$$\frac{\partial \mu^i_{j+1}(\hat{y}(t-k(l)))}{\partial m^i_j} =$$

$$\begin{cases} 0 & \text{if } \hat{y}(t-k(l)) < m^i_j \\ 6\frac{(\hat{y}(t-k(l))-m^i_{j+1})(\hat{y}(t-k(l))-m^i_j)}{(m^i_j-m^i_{j+1})^3}\left[\frac{\partial \hat{y}(t-k(l))}{\partial m^i_j}-\frac{(\hat{y}(t-k(l))-m^i_{j+1})}{m^i_j-m^i_{j+1}}\right] & \text{if } m^i_j < \hat{y}(t-k(l)) < m^i_{j+1} \\ 0 & \text{if } \hat{y}(t-k(l)) > m^i_{j+1} \end{cases} \tag{C.44}$$

$$\frac{\partial \mu^i_{j+1}(\varepsilon(t-m(l)))}{\partial m^i_j} =$$

$$\begin{cases} 0 & \text{if } \varepsilon(t-m(l)) < m^i_j \\ 6\frac{(\varepsilon(t-m(l))-m^i_{j+1})(\varepsilon(t-m(l))-m^i_j)}{(m^i_j-m^i_{j+1})^3}\left[-\frac{\partial \hat{y}(t-m(l))}{\partial m^i_j}-\frac{(\hat{y}(t-k(l))-m^i_{j+1})}{m^i_j-m^i_{j+1}}\right] & \text{if } m^i_j < \varepsilon(t-m(l)) < m^j_{j+1} \\ 0 & \text{if } \varepsilon(t-m(l)) > m^i_{j+1} \end{cases} \tag{C.45}$$

$$\frac{\partial \mu_{j-1}^{i}(\hat{y}(t-k(l)))}{\partial m_j^i} =$$

$$\begin{cases} 0 & \text{if } \hat{y}(t-k(l)) < m_{j-1}^i \\ -6\frac{(\hat{y}(t-k(l))-m_{j-1}^i)(\hat{y}(t-k(l))-m_j^i)}{(m_{j-1}^i-m_j^i)^3}\left[\frac{\partial \hat{y}(t-k(l))}{\partial m_j^i}+\right. & \\ \left.\frac{(\hat{y}(t-k(l))-m_{j-1}^i)}{m_{j-1}^i-m_j^i}\right] & \text{if } m_{j-1}^i < \hat{y}(t-k(l)) < m_j^i \\ 0 & \text{if } \hat{y}(t-k(l)) > m_j^i \end{cases} \tag{C.46}$$

$$\frac{\partial \mu_{j-1}^{i}(\varepsilon(t-m(l)))}{\partial m_j^i} =$$

$$\begin{cases} 0 & \text{if } \varepsilon(t-m(l)) < m_{j-1}^i \\ -6\frac{(\varepsilon(t-m(l))-m_{j-1}^i)(\varepsilon(t-m(l))-m_j^i)}{(m_{j-1}^i-m_j^i)^3}\left[-\frac{\partial \hat{y}(t-m(l))}{\partial m_j^i}+\right. & \\ \left.\frac{(\hat{y}(t-k(l))-m_{j-1}^i)}{m_{j-1}^i-m_j^i}\right] & \text{if } m_{j-1}^i < \varepsilon(t-m(l)) < m_j^j \\ 0 & \text{if } \varepsilon(t-m(l)) > m_j^i \end{cases} \tag{C.47}$$

Observe once more that when there is no feedback in the model $\frac{\partial \hat{y}(t-m(l))}{\partial \alpha_n} = 0$ the gradient expressions are reduced to the ones presented in the previous chapter for static maps.

The presence of the terms $\frac{\partial \hat{y}(t-m(l))}{\partial \alpha_n}$ in the gradients, makes the gradient computation into a calculation of a numerical solution of a discrete dynamic system which must be updated each time a new data is presented to the model.

Appendix D

Discrete Linear Dynamical System Approximation Theorem

Theorem 2 *Any stable single-input single-output discrete linear system with transfer function $f(z)$ and bounded input can be represented by a fuzzy system with normal triangular membership functions with overlap $\frac{1}{2}$.*

Proof: Suppose the transfer function $f(z)$ is given by:

$$f(z) = \frac{y(z)}{u(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}} \tag{D.1}$$

By applying the inverse Z transform to the transfer function the system can be converted into the difference equation:

$$y(k) = -a_1 y(k-1) - \ldots - a_n y(k-n) + b_0 uk + \ldots + b_n u(k-n) \tag{D.2}$$

this formula can be represented in vector form as:

$$y(k) = T^T X(k) \tag{D.3}$$

where $T = [-a_1, \ldots, -a_n, b_0, \ldots, b_n]^T$, $X(k) = [y(k-1), \ldots, y(k-n), u(k), \ldots, u(k-n)]^T$ and $X(k), T \in \Re^N$ with $N = 2n+1$. A fuzzy system to map the function $y(k) = \mathcal{F}(X(k))$ representing the linear dynamic system can be constructed by placing an arbitrary number of normal, triangular membership functions with overlap $\frac{1}{2}$ covering the universe of discourse of each of the N inputs. A rule base is constructed using only AND operations and covering all possible combinations of antecedents and each of the rules is initialized as follows:

IF x_1 IS $\mathcal{A}^1_{j_1}$ AND ... AND x_N IS $\mathcal{A}^N_{j_N}$ THEN $y^j = t_1 m^1_{j_1} + \ldots + t_N m^N_{j_N}$

where x_i is the i-th entry of the vector X, $\mathcal{A}^i_{j_i}$ is the fuzzy set defined at the i-th input used as antecedent of the j-th rule this fuzzy set is described by the membership function:

$$\mu^i_{j_i}(x_i) = \begin{cases} \frac{x_i - m^i_{j_i-1}}{m^i_{j_i} - m^i_{j_i-1}} & \text{if } x_i \in [m^i_{j_i-1}, m^i_{j_i}] \\ \frac{m^i_{j_i+1} - x_i}{m^i_{j_i+1} - m^i_{j_i}} & \text{if } x_i \in [m^i_{j_i}, m^i_{j_i+1}] \\ 0 & \text{Otherwise} \end{cases} \tag{D.4}$$

$m^i_{j_i}$ is the modal value of the membership function $\mu^i_{j_i}(.)$ and t_i is the i-th entry of the vector T.
With this description of the fuzzy model, the evaluation of an arbitrary vector $X(k) \in [m^1_{j_1}, m^1_{j_1+1}] \times \ldots \times [m^N_{j_N}, m^N_{j_N+1}]$ will activate 2^N rules. To analyze the result of this evaluation assume without lost of generality $N = 2$. The point described by the vector $X(k)$ will be in the interval $[m^1_{j_1}, m^1_{j_1+1}] \times [m^2_{j_2}, m^2_{j_2+1}]$ and the set of active rules will be describe as:

1. IF x_1 IS $\mathcal{A}^1_{j_1}$ AND x_2 IS $\mathcal{A}^2_{j_2}$ THEN $y^j = t_1 m^1_{j_1} + t_2 m^2_{j_2}$
2. IF x_1 IS $\mathcal{A}^1_{j_1+1}$ AND x_2 IS $\mathcal{A}^2_{j_2}$ THEN $y^{j+1} = t_1 m^1_{j_1+1} + t_2 m^2_{j_2}$
3. IF x_1 IS $\mathcal{A}^1_{j_1}$ AND x_2 IS $\mathcal{A}^2_{j_2+1}$ THEN $y^{j+2} = t_1 m^1_{j_1} + t_2 m^2_{j_2+1}$
4. IF x_1 IS $\mathcal{A}^1_{j_1+1}$ AND x_2 IS $\mathcal{A}^2_{j_2+1}$ THEN $y^{j+3} = t_1 m^1_{j_1+1} + t_2 m^2_{j_2+1}$

Then, the value of fuzzy system evaluated at the point $X(k)$ is given by the expression:

$$\begin{aligned} \mathcal{F}(X(k)) = & \\ & \frac{y^j(\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2)) + y^{j+1}(\mu^1_{j_1+1}(x_1)\mu^2_{j_2}(x_2))}{\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2)) + \mu^1_{j_1+1}(x_1)\mu^2_{j_2}(x_2)) + \mu^1_{j_1}(x_1)\mu^2_{j_2+1}(x_2)) + \mu^1_{j_1+1}(x_1)\mu^2_{j_2+1}(x_2))} \\ + & \frac{y^{j+2}(\mu^1_{j_1}(x_1)\mu^2_{j_2+1}(x_2)) + y^{j+3}(\mu^1_{j_1+1}(x_1)\mu^2_{j_2+1}(x_2))}{\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2)) + \mu^1_{j_1+1}(x_1)\mu^2_{j_2}(x_2)) + \mu^1_{j_1}(x_1)\mu^2_{j_2+1}(x_2)) + \mu^1_{j_1+1}(x_1)\mu^2_{j_2+1}(x_2))} \\ = & \; y^j(\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2)) + y^{j+1}(\mu^1_{j_1+1}(x_1)\mu^2_{j_2}(x_2)) + \\ & \; y^{j+2}(\mu^1_{j_1}(x_1)\mu^2_{j_2+1}(x_2)) + y^{j+3}(\mu^1_{j_1+1}(x_1)\mu^2_{j_2+1}(x_2)) \end{aligned}$$

at the interval $[m^i_j, m^i_{j+1}]$, $\mu^i_{j+1}(x) = 1 - \mu^i_j(x)$. Then, the expression (D.5) becomes:

$$\mathcal{F}(X(k)) = y^j \mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2) + y^{j+1}(1 - \mu^1_{j_1}(x_1))\mu^2_{j_2}(x_2)$$

$$
\begin{aligned}
& +y^{j+2}\mu^1_{j_1}(x_1)(1-\mu^2_{j_2}(x_2)) + y^{j+3}(1-\mu^1_{j_1}(x_1))(1-\mu^2_{j_2}(x_2)) \\
= \; & y^j\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2) + y^j\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2) + y^{j+1}\mu^2_{j_2}(x_2) \\
& -y^{j+1}\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2) + y^{j+2}\mu^1_{j_1}(x_1) - y^{j+2}\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2) \\
& -y^{j+3}\mu^2_{j_2}(x_2) + y^{j+3}\mu^1_{j_1}(x_1)\mu^2_{j_2}(x_2) - y^{j+3}\mu^1_{j_1}(x_1) + y^{j+3} \qquad \text{(D.5)}
\end{aligned}
$$

replacing the values of the consequences in the expression (D.5):

$$
\begin{aligned}
\mathcal{F}(X(k)) &= t_1 m^1_{j_1}\mu^1_{j_1}(x_1) + t_2 m^2_{j_2}\mu^2_{j_2}(x_2) - t_1 m^1_{j_1+1}\mu^1_{j_1}(x_1) - t_2 m^2_{j_2+1}\mu^2_{j_2}(x_2) \\
&= t_1 m^1_{j_1+1} + t_2 m^2_{j_2+1} \qquad \text{(D.6)}
\end{aligned}
$$

Finally, replacing (D.4) in (D.6) the expression becomes

$$\mathcal{F}(X(k)) = t_1 x_1 + t_2 x_2 = T^T X(k) \qquad \text{(D.7)}$$

Being $\mathcal{F}(X(k)) = T^T X(k)$ then the constructed fuzzy system will be equal to the linear discrete dynamical system of eq.(D.1)□

Appendix E

Fuzzy Control for a Continuously Variable Transmission

This appendix presents the design of a nonlinear PI-like fuzzy controller. The task of the fuzzy system is to generate the parameters (K_p and T_i) for a proportional-integral (PI) controller integrated in the control system to govern a continuously variable transmission (CVT). Nonlinear compensations were also designed to improve the performance.

E.1 Introduction and process description

The continuously variable transmission (CVT) is a type of transmission used in cars with combustion engine. It is different from classical gearbox transmission where the rotation ratio between the engine and the wheels changes in discrete steps (first, second, third speed and so on) instead CVT can give a continuous speed ratio between the motor and the wheels. This specification improves the exploitation of the power given by the motor, such that the system can be commanded to reduce the fuel consumption and pollution (Economic Mode) or maximize the power transmitted to the wheels (Sport Mode).
The system is constructed using two conic pulleys connected to each other with a belt. The radius of the pulleys is variable and in fact increases if the two cones of the pulley approximate to each other and reduces if they split apart. The distance between the cones is controlled by means of a hydraulic cylinder connected to a hydraulic circuit, which feeds not only the pulleys but also the clutch. The

schematic representation presented in Figure E.1 explains the working principle. The pressure applied to the pulleys must be accurately controlled in such a way that the belt works keeps an optimal tension. If the tension is too high the belt will not move and it can break apart because of the friction and the stress. If the tension is too low the motor doesn't transmit the movement to the wheels and the belt can also slip off the pulleys destroying the complete system (the rotation speed of the motor is about 1000-4000 r.p.m.).
The pressures for the cylinders are generated by a valve body, which behaves as a hydraulic amplifier. There is a third pressure that drives the wet-plate clutch. Therefore, the system has three control inputs. The pressures are controlled using a servo pressure and the servo pressures are regulated by a servo Pulse Width Modulated (PWM) valve driven by electric signals. In this way, the signals generated by the controller of CVT system are only electrical signals.

The three reference values for the pressures of the three valves are generated by master control system. This master control system generates the set points of the pressures depending according to the conditions of the road, desired speeds and throttles position. (See figure E.2)

The control of CVTs is challenging due to the complexity generated by large number of variables involved (speed, temperature, oil viscosity, valve construction, etc.) and their interactions. The system is highly non-linear and difficult to control by means of classical linear control techniques.

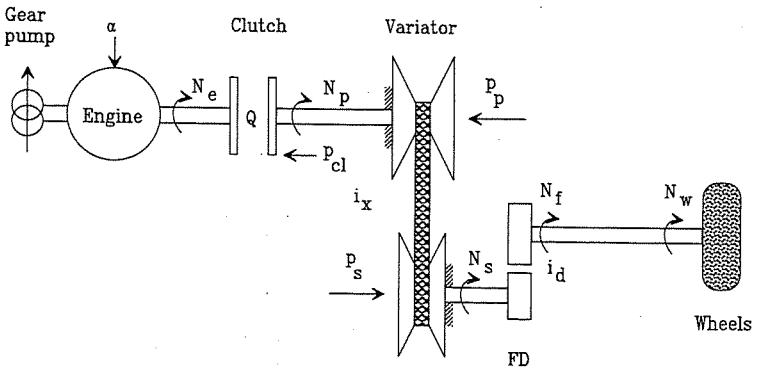


Figure E.1: *Power train of a car with a CVT and wet-plate clutch. FD=Final reduction.*

The appendix presents the design of the control system that regulates the pressure applied to the pulleys (P_p and P_s in figure E.1) is presented. This control system receives the set-point signals from the master control system and guarantees the exact positioning of the pulleys despite the disturbances.

E.2 Performance Specifications

The main objective of the control system is to follow the set-point pressures generated by the master control system, with a minimum overshoot and with a settling time of 60 to 70ms. The condition of minimum overshoot is a consequence of the fact that overshoots can generate a release of the tension in the belt generating a slip off of the pulleys and the damage of the system. Hence, some overshoot is allowed when the pressures are increased (pulleys closing) but it has to be small when the pressures are decreased (pulleys opening).

E.3 A Physical Model for the CVT

A physical model of the system was obtained by Minten and Vanvuchelen [55]. This model is a full physical model oriented to functional simulation. The model

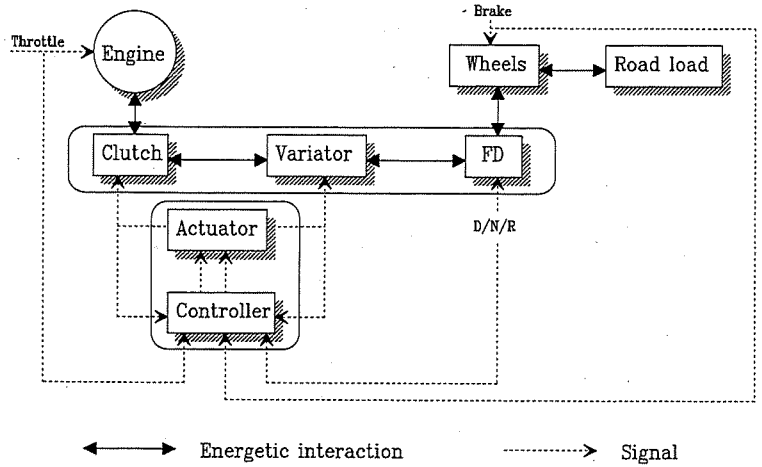


Figure E.2: *Diagram of the system including the high level controller.*

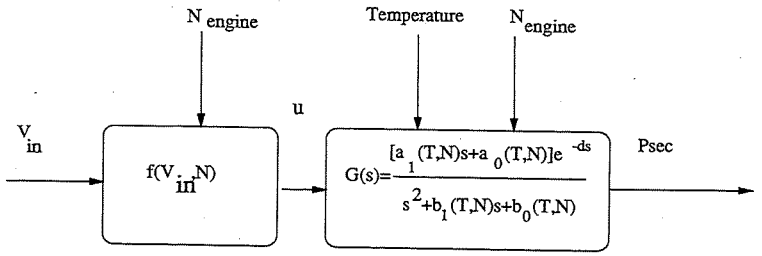


Figure E.3: *Representation of the process by means of a static nonlinearity $f(V_{in}, N_{engine})$ and a dynamic linear system $G(s)$ with variable parameters.*

is non-linear and describes the full operating range of the CVT. This model is suitable for simulation of physical properties but it is too complex for control design because of its long simulation time. Therefore, a simpler model for control was obtained. The system has one manipulated input the voltage applied to the PWM servo valve V_{in} and two measured disturbances temperature T, and engine speed N_{engine}. The only output considered in this study was the pressure applied to the pulley.
A simplified representation of the plant, for a given temperature T and on the engine speed N_{engine} is a nonlinearity in series with a second order linear plant (a Hammerstein model). In other words the system can be represented by a Hammerstein model where the coefficients are functions of the temperature T and the engine speed N_{engine}. Figure E.3 shows that the nonlinearity f depends mainly on the speed of the engine N_{engine} and that the dynamics of the second order system $G(s)$ depends on the temperature T and on the engine speed N_{engine}. Figure E.4 shows the clear dependence of f on the engine speed. Using this assumption a set of nonlinearities and linear plant models are extracted for different operating points in temperature and engine speed.

E.4 Design of the Controller

The first step in the controller design is to compensate the nonlinearity $f(V_{in}, N_{engine})$ by means of $\hat{f}^{-1}(V_{in}, N_{engine})$. This function $\hat{f}^{-1}(.,.)$ is generated from measurement data using power series to approximate the function. Observe that this procedure can be applied due to the monotonicity of the function $f(.,.)$. Once the "local" linear models for different values of T and N_{engine} and the $\hat{f}^{-1}(.,.)$ function are obtained, an optimization procedure (see [78]) is applied to calculate a suboptimal proportional-integral (PI) controller for the operating

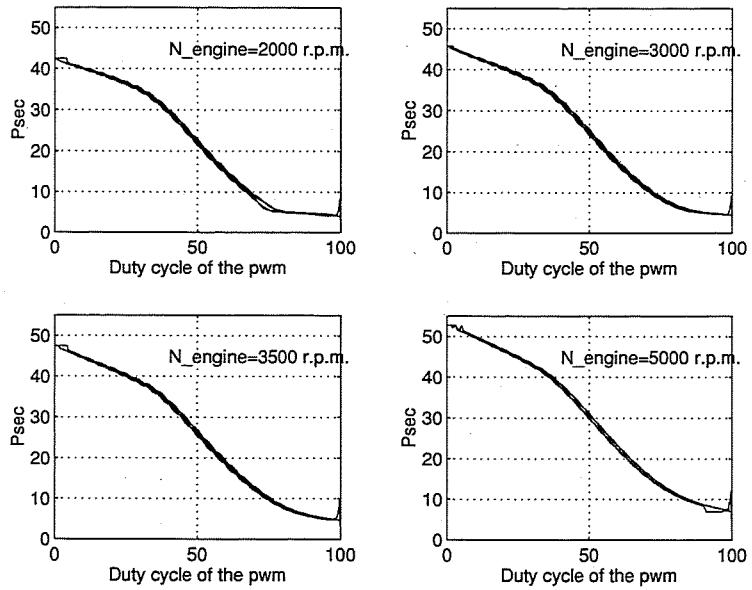


Figure E.4: *Static nonlinearity f for different values of engine speed N_{engine} and constant temperature.*

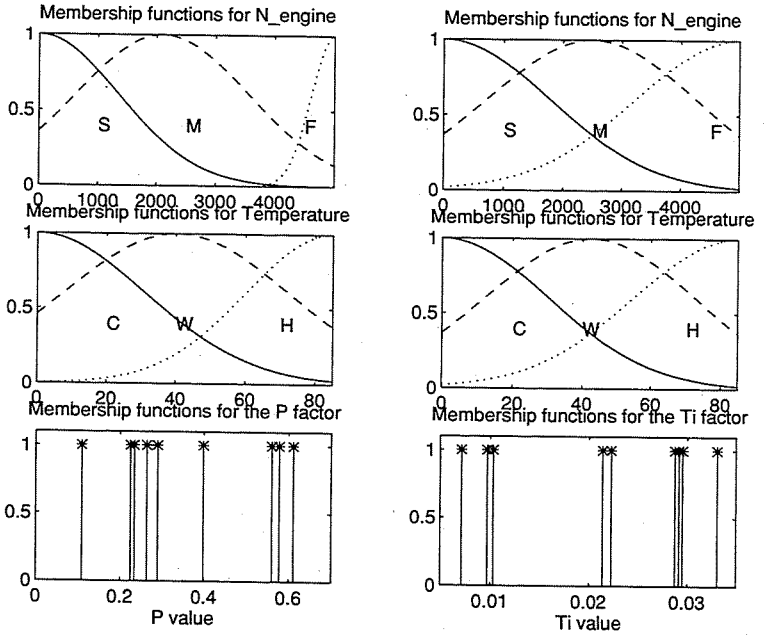


Figure E.5: *Membership functions of the Fuzzy Inference Systems that generates the parameters P and T_i for the PI controller.*

point. The cost function of the optimization is defined as:

$$J(K_p, T_i) = \lambda o.s. + (1-\lambda)\int_0^\infty t(e^2(t)dt)^{\frac{1}{2}}$$

where K_p and T_i are the proportional gain and the integral time of the PI controller, $\lambda = 0.5$ is a weight that defines the importance of the overshoot $o.s.$ in the cost function, $e(t) = P_{ref}(t) - P_{sec}(t)$ is the tracking error, t is the time and $o.s.$ is the overshoot for a unitary step response of the closed loop system and is defined as:

$$o.s. = \max\{\max(y(t)-1), 0\}$$

The starting point for the optimization is obtained using the relay method for PI tuning. The quasi-Newton method for unconstrained multivariable optimization was applied (see [36]). This procedure generates a set of PI controllers that depend on the temperature and the engine speed. The PI controller including the set of local PI controllers can be represented as:

$$C(s) = K_p(T, N_{engine})\left(1 + \frac{1}{T_i(T, N_{engine})s}\right)$$

for continuous time or

$$C(z) = K_p(T, N_{engine})\left(1 + \frac{1}{T_i(T, N_{engine})(1-z^{-1})}\right)$$

for discrete time.
The optimization of the controller was done using the discrete description of the controller. In this way, all quantization effects are present and the controller will be more accurate. The values of $K_p(T, N_{engine})$, $T_i(T, N_{engine})$ and $\hat{f}^{-1}(V_{in}, N_{engine})$ are only defined for some operating points. An interpolation method is needed to make the transitions between the different operating points an interpolation method is needed. The overall performance of the controller is related with the number of operating points evaluated; this implies that a large number of parameters should be stored in a look-up table. We found that a good solution will be to approximate this look-up table with a fuzzy inference system (FIS). The main characteristics of this FIS are Gaussian membership functions, product-sum composition and defuzzification using center of gravity. The membership functions were uniformly distributed in the domain of the temperature and the engine speed, and tuned by using a backpropagation algorithm. A picture of the membership functions can be seen in figure E.5. The function $\hat{f}^{-1}(.,.)$ is also scheduled depending on the speed. The controller is shown in figure E.6.

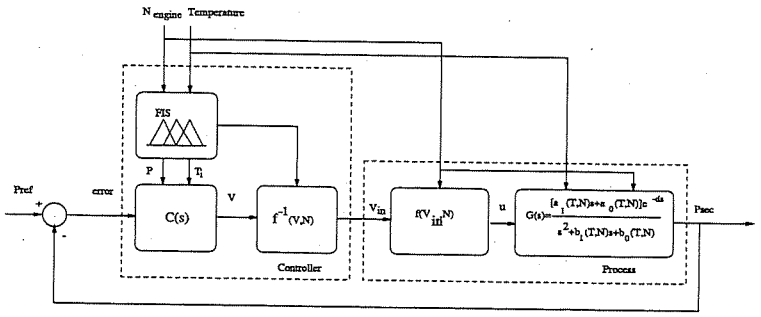


Figure E.6: *Block diagram of the process with the nonlinear controller.*

E.5 Stability Analysis

The closed loop system can be described by means of a Takagi -Sugeno fuzzy model (assuming perfect cancellation of the nonlinearity $f(.,.)$) with rules like:

$$\text{Rule}_i \text{ IF } Temperature \text{ is } \alpha_i \text{ AND } N_{engine} \text{ is } \beta_i \text{ THEN } \dot{x} = \tilde{A}_i x + \tilde{B}_i u$$

The condition for stability is the existence of a common P matrix such that:

$$P > 0 \quad \text{(E.1)}$$
$$\tilde{A}_i^T P + P\tilde{A}_i < 0 \quad \text{(E.2)}$$
$$\forall i \quad i \in \{1, \ldots, N_{rules}\} \quad \text{(E.3)}$$

The search for this P matrix can be conducted by means of solving the feasibility LMI problem in chapter 4. A feasible solution was found for this problem so that stability is guaranteed.
Figure E.7 shows the improvements obtained with the fuzzy controller compared with the linear scheduled controller. It is important to observe that there is not only improvement in the performance, but also that any modification of the design will be easier to implement, due to the extra information provided by the rule base description. The rules of the controller have the following form:

IF T is A_j AND N_{engine} is B_j THEN P is O_j AND T_i is Q_j
IF N_{engine} is C_j THEN f^{-1} is R_j

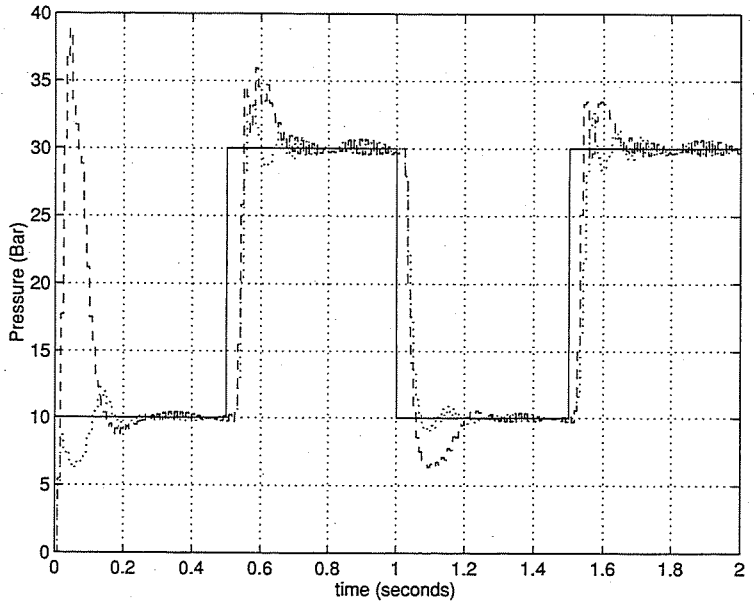


Figure E.7: *Comparative response of the systems continuous line (_) reference, dashed line () linear controller with feedforward action and dotted line (..) fuzzy controller. Observe that the overshoot and the settling time of the fuzzy with the fuzzy controller is significantly reduced.*

E.6 Conclusions

This appendix has shown the design procedures of a control systems for a CVT. The control system is based on a fuzzy inference system that "schedules" the parameters of the controller according to some measured disturbances. In the procedure, several linear controllers were calculated by means of optimization for different operating points. In addition, different nonlinear compensators were obtained for different values of N_{engine}. Stability of the closed loop system is guaranteed by means of the solution of a LMI. Improvements in the performance were observed when the new control system was compared with the scheduled linear version. Another advantage of the designed controller is the linguistic description of the scheduling action, which simplifies the re-tuning of the system in the industrial framework.

Bibliography

[1] *Fuzzy Logic Toolbox, User's Guide, V-2.0.* Mathworks, USA, 1998.

[2] H. Akaike. A new look at the statistical model identification. *IEEE Trans. on Automatic Control*, 19:716–723, 1974.

[3] F. Allgöwer, T. Badgwell, J. Qin, J. Rawlings, and S. Wright. *Nonlinear Predictive Control and Moving Horizon Estimation-An Introductory Overview*, pages 391–449. Advances in Control-Highlights of ECC'99. Springer Verlag, 1999.

[4] K. Astrom and T. Hagglund. *PID Controllers : Theory, Design and Tuning.* ISA, 1995.

[5] R. Babuska. *Fuzzy Modeling and Identification.* PhD thesis, Delft University of Technology, Nederlands, 1997.

[6] A. Ben-Tal and A. Nemirovski. Robust convex programming. Technical report, Minerva Optimization Center, Haifa-Israel, 1996.

[7] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs via convex programming. Technical report, Minerva Optimization Center, Haifa-Israel, 1997.

[8] J. Bezdek. A physical interpretation of fuzzy isodata. *IEEE Trans. Syst.,Man,Cybern.*, pages 387–389, 1976.

[9] S. Billings and W. Voon. Correlation based model validity tests for nonlinear models. *Int J. Control*, 44:235–244, 1986.

[10] U. Bodenhofer. Tuning of fuzzy systems using genetic algorithms. Master's thesis, Institut für Mathematik Johannes Kepler Universität, Linz, Austria, 1996.

[11] G. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control.* San Francisco: Holden Day, 1970.

[12] S. Boyd, L. Vandenberghe, and M. Grant. Efficient convex optimization for engineering design. In *IFAC Symposium on Robust Control Design*, Rio de Janeiro- Brazil, September 1994.

[13] D. Broadbent. *The Magic Number Seven After Fifteen Years*, page 3. Studies in Long Term Memory. Addison Wesley, 1975.

[14] E. Camacho and C. Bordons. *Model Predictive Control in the Process Industry.* Springer Verlag, London, 1995.

[15] S. Chen and S. Billings. Neural networks for nonlinear dynamic system modelling and identification. *Int.J.Control*, 56:319–346, 1992.

[16] Y. Chikkula and J. Lee. Input sequence design for parametric identification of nonlinear systems. In *Proceedings of the American Control Conference*, pages 3037–3041, Albuquerque - USA, June 1997.

[17] D. Clarke, C. Mohtadi, and P. Tuffs. Generalized predictive control-parts i - ii. *Automatica*, 23:137–160, 1987.

[18] J. Da Costa Sousa. *A Fuzzy Approach to Model-Based Control.* PhD thesis, Delft University of Technology, Nederlands, 1998.

[19] B. De Moor. Daisy:database for the identification of systems. *Department of Electrical Engineering-ESAT-K.U.Leuven,Belgium.http://www.esat.kuleuven.ac.be/sista/daisy/, Used dataset:Continuous Stirred Tank Reactor, Section: Process Industry Systems, code: 98-002.*

[20] B. De Moor. Daisy:database for the identification of systems. *Department of Electrical Engineering-ESAT-K.U.Leuven,Belgium.http://www.esat.kuleuven.ac.be/sista/daisy/, Used dataset:Model of a steam generator at Abbott Powe Plant in Champaign IL., Section: Process Industry Systems, code: 98-003.*

[21] D. Driankov, H. Hellendoorn, and M. Reinfrank. *An Introduction to Fuzzy Control.* Springer Verlag, 1993.

[22] Ecole Nationale Supérieure de Techniques Avancées (ENSTA),Optimization and Control Group. *LMITOOL-2.0 package*, 1998.

[23] J. Espinosa, M. Hadjili, V. Wertz, and J. Vandewalle. Predictive control using fuzzy models-comparative study. In *Proceeding of the European Control Conference-99*, Karlsruhe, 1999.

[24] J. Espinosa and J. Vandewalle. Fuzzy modeling and identification, using afreli and fuzion algorithms. In *Proceedings of the 5th. International Conference on Soft Computing IIZUKA-98*, pages 535–540, Iizuka-Japan, October 1998.

[25] J. Espinosa and J. Vandewalle. Fuzzy modeling with linguistic integrity. In *Proceedings of the International Workshop on Advanced Black Box Techniques for Nonlinear Modeling*, pages 197–202, Leuven - Belgium, July 1998.

[26] J. Espinosa and J. Vandewalle. Predictive control using fuzzy models applied to a steam generating unit. In D. Ruan, H. A. Abderrahim, P. D'hondt, and E. Kerre, editors, *Fuzzy Logic and Intelligent Technologies for Nuclear Science and Industry*, Singapore, 1998. World Scientific.

[27] J. Espinosa and J. Vandewalle. Constrained predictive control using fuzzy models. In *Proc. of the Eight International Fuzzy Systems Association World Congress (IFSA-99)*, pages 649–654, Taipei-Taiwan, 1999.

[28] J. Espinosa and J. Vandewalle. Fuzzy control and pid. In *Lecture Notes of the Studienamiddag + Tentoonstelling PID-regelaars : Moderne afstellingsmethodes*, Antwerp, Belgium, March 1999.

[29] J. Espinosa and J. Vandewalle. Predictive control using fuzzy models. In R. Roy, T. Furuhashi, and Chawdhry, editors, *Advances in Soft Computing Engineering Design and Manufacturing*, London, 1999. Springer Verlag.

[30] J. Espinosa and J. Vandewalle. Constructing fuzzy models with linguistic integrity-afreli algorithm. *IEEE Trans. on Fuzzy Systems*, 8(5):591–600, October 2000.

[31] J. Espinosa, P. Vanvuchelen, and J. Vandewalle. Fuzzy scheduling to control a hydraulic servo system. *Journal A*, 36(3):51–55, October 1995.

[32] J. Espinosa, P. Vanvuchelen, and J. Vandewalle. Fuzzy control for a continuously variable transmission. In *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing, EUFIT-97*, Aachen, Germany, 1997.

[33] C. Garcia, D. Prett, and M. M. Model predictive control: Theory and practice-a survey. *Automatica*, 25:335, August 1989.

[34] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, USA, 1989.

[35] G. Goodwin and K. Sin. *Adaptive Filtering Prediction and Control.* Prentice Hall, 1984.

[36] A. Grace. *Optimization Toolbox - For use with MATLAB.* Mathworks, Inc., 1992.

[37] X. He and H. Asada. A new method for identifying orders of input-output models for nonlinear dynamic systems. In *Proceedings of the American Control Conference*, pages 2520–2523, San Francisco California, 1993.

[38] M. Henson and D. Seborg. Input-output linearization of general nonlinear processes. *AIChE Journal*, 36(11):1753–1757, November 1990.

[39] R. Jager. *Fuzzy Logic in Control.* PhD thesis, T.U.Delft, Nederlands, 1995.

[40] J.-S. Jang. *Neuro-Fuzzy Modeling: Architectures, Analyses and Applications.* PhD thesis, University of Berkeley-California, 1992.

[41] J.-S. Jang, C.-T. Sun, and E. Mizutani. *Neuro Fuzzy and Soft Computing.* Prentice Hall International, USA, 1997.

[42] J.-S. R. Jang. Structure determination in fuzzy modeling: A fuzzy cart approach. In *Proc. of IEEE international conference on fuzzy systems*, 1994.

[43] J.-S. R. Jang and C.-T. Sun. Neuro-fuzzy modeling and control. *The Proceedings of the IEEE*, 83(3):378–406, 1995.

[44] J. Jantzen. *Fuzzy Control.* Technical University of Denmark, 1991.

[45] M. Johansson and A. Rantzer. Computation of piecewise quadratic lyapunov functions for hybrid system. In *Proceedings of the European Control Conference*, Brussels-Belgium, 1997.

[46] E. Lawler and E. Wood. Branch-and-bound methods: A survey. *Journal of Operations Research*, 14:699–719, 1966.

[47] G. Lightbody and G. Irwin. Nonlinear control structures based on embedded neural system models. *IEEE Trans. on Neural Networks*, 8:553–567, August 1997.

[48] L. Ljung. *System Identification Theory for the User.* Prentice Hall, 1987.

[49] M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and Applications*, 284:193–228, Nov. 1998.

[50] N. Lori and P. Costa Branco. Autonomous mountain-clustering method applied to fuzzy systems modeling. In C. Dagli, M. Akay, C. Philip, Chen, B. Fernández, and J. Ghosh, editors, *Intelligent Engineering Systems*

Through Artificial Neural Networks, Smart Engineering Systems: Fuzzy Logic and Evolutionary Programming, volume 5, pages 311–316. ASME Press, New York,U.S.A., 1995.

[51] S. Loureiro de Oliveira. *Model Predictive Control for Constrained Nonlinear Systems.* PhD thesis, California Institute of Technology, USA, 1996.

[52] L. Magni. *Nonlinear Receding Horizon Control:Theory and Application.* PhD thesis, Università degli Studi di Pavia, Italy, 1998.

[53] Mathworks, Inc. *LMI Control Toolbox Version 1.0.4*, 1997.

[54] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer Verlag, Berlin, 1996.

[55] W. Minten and P. Vanvuchelen. Modeling and simulation of an ellectronically controlled variable transmission. Master's thesis, K.U.LEUVEN, Belgium, 1991.

[56] L. Mitten. Branch-and-bound methods: General formulation and properties. *Journal of Operations Research*, 18:24–34, 1970.

[57] K. Nozaki, H. Ishibuchi, and H. Tanaka. A simple but powerful heuristic method for generating fuzzy rules from numerical data. *Fuzzy Sets and Systems*, (86):251–270, 1997.

[58] J.-J. Østergaard. *High Level Process Control in the Cement Industry by Fuzzy Logic*, pages 163–172. Fuzzy Logic in Control-Advances and Applications. World Scientific, 1999.

[59] F. Oustry, L. El Ghaoui, and H. Lebret. Robust solutions to uncertain semidefinite programs. Accepted for publication in SIAM J. Optimization.

[60] R. Palm, D. Driankov, and H. Hellendoorn. *Model Based Fuzzy Control.* Springer Verlag, 1997.

[61] W. Pedrycz. Why triangular membership functions? *Fuzzy Sets and Systems*, 64:21–30, 1994.

[62] W. Pedrycz and J. Valente de Oliveira. Optimization of fuzzy models. *IEEE Trans. Syst.,Man,Cybern. Part B.*, 26:627–636, 1996.

[63] G. Pellegrinetti and J. Bentsman. Nonlinear control oriented boiler modeling-a benchmark problem for controller design. *IEEE Trans. on Control Sys. Tech.*, 4(1):57–64, January 1996.

[64] J. Rawlings. Tutorial: Model predictive control technology. In *Proceedings of the ACC*, San Diego,Cal. US, June 1999.

[65] J. Richalet. Industrial applications of model based predictive control. *Automatica*, 29:1251–1274, 1993.

[66] J. A. Roubos, R. Babuska, P. Bruijn, and H. Verbruggen. Predictive control by local linearization of a takagi-sugeno fuzzy model. *Proceedings of FUZZ-IEEE-98*, pages 37–42, May 1998.

[67] M. Setnes. *Implementing Fuzzy Control in the Manufacturing of Washing Powders*, pages 193–203. Fuzzy Logic in Control-Advances and Applications. World Scientific, 1999.

[68] M. Setnes, R. Babuska, and H. Verbruggen. Rule-based modeling: Precision and transparency. *IEEE Trans. Systems Man and Cybernetics-Part C*, 28(1), 1998.

[69] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P. Glorennec, J. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31:1691–1724, 1995.

[70] R. Soeterboek. *Predictive Control, A unified approach.* PhD thesis, Technische Universiteit Delft, Nederlands, 1990.

[71] M. Sugeno and T. Yasukawa. A fuzzy logic based approach to qualitative modeling. *IEEE Trans. on Fuzzy Systems*, 1:7–31, 1993.

[72] J. Suykens. *Artificial Neural Networks for Modeling and Control of Nonlinear Systems.* PhD thesis, Faculty of Engineering, K.U.Leuven, Leuven, Belgium, 1995.

[73] T. Takagi and M. Sugeno. Fuzzy identification of systems and its application to modeling and control. *IEEE Trans. Syst.,Man,Cybern.*, 15:116–132, 1985.

[74] S. Tan and J. Vandewalle. An on-line structural and parametric scheme for fuzzy modelling. In *Proc. of the 6th International Fuzzy Systems Association World Congress IFSA-95*, pages 189–192, 1995.

[75] K. Tanaka and M. Sugeno. Stability analysis and design of fuzzy control systems. *Fuzzy Sets and Systems*, 45(2):135–156, 1992.

[76] V. Utkin. Variable structure systems: A survey. *IEEE Transactions on Automatic Control*, 22(2):212–222, 1977.

[77] J. Valente de Oliveira. Semantic constraints for membership function optimization. *IEEE Trans. Systems Man and Cybernetics-Part A*, 29(1), 1999.

[78] P. Vanvuchelen and B. DeMoor. A multi-objective optimization approach for parameter setting in system and control design. In *Extended abstracts of the IFIP Conference on System Modelling and Optimization*, Compiegne, France, 1993.

[79] C. von Altrock. *Fuzzy Logic Applications in Europe*, pages 277–310. Industrial Applications of Fuzzy Logic and Intelligent System. IEEE-Press, 1995.

[80] L. Wang. *Adaptive Fuzzy Systems and Control.* Prentice Hall, 1994.

[81] L. Wang. Analysis and design of fuzzy identifiers of nonlinear dynamic systems. *IEEE Trans. on Automatic Control*, 40:11–23, 1995.

[82] L. Wang. *A Course in Fuzzy Systems and Control.* Prentice Hall, 1997.

[83] S. Wright. Applying new optimization algorithms to model predictive control. *Chemical Process Control-V, CACHE, AIChe, Symposium Series*, 93(316):147–155, 1997.

[84] R. Yager and D. Filev. *Essentials of fuzzy modeling and control.* John Wiley & Sons, 1994.

[85] S. Yasunobu and S. Miyamoto. A predictive fuzzy control for automatic train operation. *in Japanese Systems and Control*, 28(10):605–613, 1984.

[86] H. Ying. The takagi-sugeno fuzzy controllers using the simplified linear control rules are nonlinear variable gain controller. *Automatica*, 34(2), 1998.

[87] L. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. Syst.,Man,Cybern. Part B.*, 3(1):28–44, 1973.

[88] J. Zhao. *System Modeling, Identification and Control Using Fuzzy Logic.* PhD thesis, Université Catolique de Louvain, Belgium, 1995.