AN IMPLEMENTATION OF FUZZY CONTROL ALGORITHMS FOR CONTROLLING THE CRANE ARM OF MULTI-BARRELED ROCKET LAUNCHER AMMUNITION TRANSPORT VEHICLES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ATİLA BOSTAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF COMPUTER ENGINEERING

DECEMBER 1997

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Tayfur ÖZTÜRK

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Fatoş YARMAN-VURAL

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Adnan YAZICI

Supervisor

Examining Committee Members :

Prof. Dr. Ziya AKTAŞ (Chairman)

Assoc. Prof. Dr. Adnan YAZICI

Asst. Prof. Dr. Faruk TOKDEMİR

Asst. Prof. Dr. İ. Hakkı TOROSLU

Asst. Prof. Dr. Murat ÜÇÜNCÜ

ABSTRACT

AN IMPLEMENTATION OF FUZZY CONTROL ALGORITHMS FOR CONTROLLING THE CRANE ARM OF MULTI-BARRELED ROCKET LAUNCHER AMMUNITION TRANSPORT VEHICLES

BOSTAN, Atila

M.S., Department of Computer Engineering

Supervisor : Assoc. Prof. Dr. Adnan YAZICI

December 1997, 136 pages

In this thesis, the control of security limits for the crane arm of the multi-barreled rocket launcher ammunition trucks which have recently begun to be used in the Turkish Army, by fuzzy control algorithms was aimed.

For this purpose, some models were developed and simulations were directed for Mamdani and Takagi-Sugeno type fuzzy control algorithms. The result of these simulations revealed that the Takagi-Sugeno approach for this problem is more preferable.

At the end of this study a fuzzy control model of the crane arm which meets the requirements of the Logistics Department of the Turkish Army has been configured.

Key words : Fuzzy control, Mamdani and Takagi-Sugeno fuzzy control methods.

ÖZ

ÇOK NAMLULU ROKETATAR MÜHİMMAT TAŞIYICI ARAÇLARININ VİNÇ KOLLARINI KONTROL İÇİN BULANIK DENETİM ALGORİTMALARININ UYGULANMASI

BOSTAN, Atila
Yüksek Lisans Tezi, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Adnan YAZICI
Aralık 1997, 136 sayfa

Bu yüksek lisans tezinde Türk Kara Kuvvetleri Komutanlığı bünyesinde çok namlulu roketatar mühimmatı taşınmak amacı ile kullanımına başlanan araçlardaki kaldırma vincinin güvenlik kontrollerinin bulanık denetim algoritmaları ile gerçekleştirilmesi amaçlanmıştır.

Bu amaç ile, Mamdani ve Takagi-Sugeno bulanık denetim algoritmaları kullanılarak modeller geliştirilmiş ve simülasyonlar yapılmıştır. Yapılan bu çalışma sonucunda Takagi-Sugeno yaklaşımının bu problem için en uygun çözümü verdiği görülmüştür.

Bu uygulama sonucunda Türk Kara Kuvvetleri Komutanlığı Lojistik Başkanlığı tarafından duyulan ihtiyacı karşılayabilecek bir kontrol modeli elde edilmiştir.

Anahtar kelimeler : Fuzzy kontrol, Mamdani ve Takagi-Sugeno fuzzy kontrol metodları.

To The Memory of My Father

ACKNOWLEDGMENTS

I would like to express my deepest and sincere thanks to Assoc.Prof.Dr. Adnan YAZICI, my supervisor, who gave his valuable support, encouragement and guidance to me throughout in the preparation of this thesis.

I started out with anxiety but came to the conclusion with this thesis with the close and sincere help of METU academic environment happily.

I would like to thank to Lt.Col.Asst. Prof.Dr. Murat ÜÇÜNCÜ and Capt. Cemal GEMCİ for giving their valuable times for discussions and for being helpful and encouraging during my study.

I am grateful to all of the people that I can't name them here who have been around me in the last year. I can't imagine what my study would be without their firm support.

Also my heartful thanks go to my beloved mother. To my wife, Zeliha, I offer sincere thanks for her unshakable faith in me and her willingness to endure with me. To my children, Seda and Hakan, I thank them for understanding me of my frequent absences and my struggle among the papers instead of playing with them.

TABLE OF CONTENTS

ABSTRACT iii

ÖZ iv

DEDICATION v

ACKNOWLEDGMENTS vi

TABLE OF CONTENTS vii

LIST OF TABLES xi

LIST OF FIGURES xii

CHAPTER

1. INTRODUCTION 1

2. BACKGROUND 5

2.1. The History of Computerized Control Theory 6

2.2. The Philosophy of Classical Control 7

2.3. The Philosophy of Modern Control 10

2.4. Computer-Control Theory 15

2.4.1. Is There a Need for a Theory for Computer-Controlled Systems? 16

2.4.2. Inherently Sampled Systems 22

2.4.3. Discrete-Time Systems as Models for Computer Algorithms 22
2.4.4. Sampling Due to the Measurement System. 24
2.4.5. Sampling Due to Pulsed Operation. 25
2.4.6. How Theory Developed. 27
2.4.7. The Sampling Theorem 27
2.4.8. Difference Equations 28
2.4.9. Transform Methods 28
2.4.10. State-Space Theory. 31
2.4.11. Optimal and Stochastic Control. 31
2.4.12. Algebraic System Theory. 32
2.4.13. System Identification. 32
2.4.14. Adaptive Control 33
2.4.15. Automatic Tuning 34

3. GENERAL CONCEPT OF CONVENTIONAL CONTROL-SYSTEMS 35

3.1. Open Loop Control Systems 35
3.2. Closed Loop Control Systems 37
3.3. Summary of Traditional Control Systems 40

4. FUZZY CONTROL SYSTEMS 41

4.1. Background 41
4.2. Fuzzy Relations 47
4.2.1. Identification of Fuzzy Relations 48
4.2.2. Fuzzy Matrices and Fuzzy Graphs 49

4.2.3. Basic Properties of Fuzzy Relations 51
4.2.4. Fuzzy Relations and Fuzzy Reasoning 51
4.2.5. Fuzzy Relational Equations 59
4.2.6. Various Types of Fuzzy Relations 62
4.3. Fuzzy Control 65
4.3.1. Classification of Fuzzy Logic Systems 66
4.3.1.1. Pure Fuzzy Logic Systems 67
4.3.1.2. Takagi & Sugeno's Fuzzy System 69
4.3.1.3. Fuzzy Logic Systems With Fuzzifier and Defuzzifier (Mamdani Type) 71
4.3.2. Fuzzy Rule Base 72
4.3.3. Fuzzy Inference Engine 77
4.3.3.1. Interpretations of a Fuzzy IF-THEN Rule.. 78
4.3.3.2. Overall Mapping of The Fuzzy Inference Engine 80
4.3.4. Fuzzifier 81
4.3.5. Defuzzifier 81
4.3.6. Membership Function Selection 83
4.3.7. Methodology 85
4.3.7.1. Mamdani Type 85
4.3.7.2. Takagi and Sugeno Type 93
4.3.7.2.1. Consequence Parameters Identification 95

4.3.7.2.2. Premise Parameters Identification 99

4.3.7.2.3.Choice of Premise Variables..... 101

5. APPLICATION 105

5.1. Problem Definition 105

5.2. Implementation 110

5.2.1. Mamdani Type 110

5.2.2. Takagi & Sugeno Type 119

5.2.2.1. Premise Parameter Identification 120

5.2.2.2. Consequence Parameter Identification..... 120

5.2.3. Simulation Procedure 122

5.3. Results 123

6. CONCLUSIONS AND FUTURE WORK 131

REFERENCES 133

LIST OF TABLES

TABLE

4.1. Basic Properties of Fuzzy Relations. 51

4.2. Various Implication Formulae. 54

4.3. Results of Reasoning. 57

5.1. Length-Weight Relation for Interpolation. 106

5.2. Results of Experiments. 109

5.3. The Data Pairs of Which the Error Part is PL. 112

5.4. The Data Pairs of Which the Error Part is PM. 113

5.5. The Data Pairs of Which the Error Part is PS. 113

5.6. The Data Pairs of Which the Error Part is ZZ. 114

5.7. The Data Pairs of Which the Error Part is NS. 114

5.8. The Data Pairs of Which the Error Part is NM. 115

5.9. The Data Pairs of Which the Error Part is NL. 115

5.10. Combined Rule base. 117

5.11. Beginning Positions for Simulations. 123

LIST OF FIGURES

FIGURES

2.1. Computer Controlled Systems. 6

2.2. a) Block Diagram of Digital Filter. 17

b) Step Responses y_s, of a Digital Computer Implementation of a first-order Lag for Different Delays in the Input Step Compared with the First Sampling Instant. 18

2.3. Sinusoidal Excitation of Sampled System in Example 1. 19

2.4. Step Response of a Double-Integrator Plant with State Feedback, Continuous-Time Solution y_c and Sampled Approximation y_s, when the Sampling Period is 0.2 s. 20

2.5. Computer Control of the Double-Integrator Plant with a Deadbeat Strategy. The Sampling Period is 1s. 22

2.6. Two Graphic Illustrations of the Iterative Scheme in Example 5.. 23

2.7. Scheduling of a Computer Program. 26

2.8. Thyristor Control Circuit. 28

3.1. Schematic Presentation of an Open-Loop Control. 36

3.2. Schematic Presentation of a Closed-Loop Control. 37

3.3. Temperature Variation. 37

3.4. Proportional Controller. 38

3.5. Temperature Fluctuation. 38

4.1. Fuzzy Matrix for Fuzzy Relation R. 50

4.2. Fuzzy Matrix and Graph for Equation (4.4)..... 50

4.3. Fuzzy System with Fuzzy Input and Output. 60

4.4. Various Fuzzy Relations. 63

4.5. Basic Configuration of Fuzzy Logic System with Fuzzifier and Defuzzifier. 67

4.6. Basic Configuration of Pure Fuzzy Logic System. 69

4.7. Basic Configuration of Takagi and Sugeno's Fuzzy System..... 71

4.8. Whole Overlap Parameter Evaluation..... 84

4.9. Divisions of the Input and Output Spaces Into Fuzzy Regions and the Corresponding Membership Functions $\mu(x_1)$, $\mu(x_2)$, $\mu(y)$. 87

4.10. The Form of Fuzzy Rule Base. 90

4.11. Outline of The Algorithm. 93

4.12. Results of Identification. 98

4.13. Input-Output Data. 98

4.14. Consequences and Noised Data. 101

4.15. Choice of Premise Variables..... 102

5.1. Telescopic Arm of the Crane. 106

5.2. General Configuration of the Controller. 110

5.3. Membership Function for Input and Output Variables. 113

5.4. a) Input-Output Relation. 119

b) Partition of Input-Output Relation. 119

5.5. Distribution of Error Probabilities (proportional to length) 122

5.6. Step-Position for Center of Gravity Weighted by Matching

In : 800 cm. , we : 7000 kg 124

5.7 Time-Position for Center of Gravity Weighted by Matching

In : 800 cm. , we : 7000 kg 124

5.8. Step-Position for Center of Gravity with Largest Matching

In : 800 cm. , we : 7000 kg 125

5.9. Time-Position for Center of Gravity with Largest Matching

In : 800 cm. , we : 7000 kg 125

5.10. Step-Position for Maximum Value Weighted by Matching

In : 800 cm. , we : 7000 kg 125

5.11. Time-Position for Maximum Value Weighted by Matching

In : 800 cm. , we : 7000 kg 126

5.12. Step-Position for Maximum Value with Largest Matching

In : 800 cm. , we : 7000 kg 126

5.13. Time-Position for Maximum Value with Largest Matching

In : 800 cm. , we : 7000 kg 126

5.14.. Step-Position for Center of Gravity Weighted by Matching

In : 800 cm. , we : 10250 kg 127

5.15. Time-Position for Center of Gravity Weighted by Matching

In : 800 cm. , we : 10250 kg 127

5.16. Step-Position for Center of Gravity with Largest Matching
In : 800 cm. , we : 10250 kg..... 127
5.17. Time-Position for Center of Gravity with Largest Matching
In : 800 cm. , we : 10250 kg..... 128
5.18. Step-Position for Maximum Value Weighted by Matching
In : 800 cm. , we : 10250 kg..... 128
5.19. Time-Position for Maximum Value Weighted by Matching
In : 800 cm. , we : 10250 kg..... 128
5.20. Step-Position for Maximum Value with Largest Matching
In : 800 cm. , we : 10250 kg..... 129
5.21. Time-Position for Maximum Value with Largest Matching
In : 800 cm. , we : 10250 kg..... 129
5.22. Step-Position for Takagi and Sugeno Type
In : 800 cm. , we : 7000 kg..... 129
5.23. Time-Position for Takagi and Sugeno Type
In : 800 cm. , we : 7000 kg..... 130
5.24. Step-Position for Takagi and Sugeno Type
In : 800 cm. , we : 10250 kg..... 130
5.25. Time-Position for Takagi and Sugeno Type
In : 800 cm. , we : 10250 kg..... 130

CHAPTER 1

INTRODUCTION

The past ten years have witnessed a rapid growth in the use of fuzzy logic in a wide variety of consumer products and industrial systems. Prominent examples of such use are electronically stabilized camcorders, autofocus cameras, washing machines, air conditioners, automobile transmissions, subway trains, and cement_ kilns.

Despite the visible successes of fuzzy logic [5], there is still a substantial misunderstanding of what fuzzy logic is, how it compares with other system design methodologies, and what its strengths and limitations are. In part, the misunderstanding stems from a duality of the meaning of fuzzy logic. In its narrow sense, fuzzy logic is a logic of approximate reasoning which may be viewed as a generalization and extension of multivalued logic. But in a broader and much more significant sense, fuzzy logic is coextensive with the theory of fuzzy sets, that is, classes of objects in which the transition from membership to nonmembership is gradual rather than abrupt [9]. In its wider sense-which is becoming predominant in the literature, fuzzy logic has many branches ranging from fuzzy arithmetic and fuzzy automata to fuzzy pattern recognition, fuzzy languages and fuzzy expert

systems. In fact, any field X can be fuzzified and called fuzzy X by replacing the concept of a set in X by the concept of a fuzzy set.

What is important to recognize is that fuzzy logic in its narrow sense plays a relatively small role in fuzzy control. Indeed what is essential to the understanding of fuzzy control is the theory of fuzzy relations [7] and, in particular, the calculus of fuzzy if-then rules or, more simply, the calculus of fuzzy rules (CFR). Typically, a fuzzy rule may be expressed as if X is A then Y is B, where X and Y are variables and A and B are their linguistic values, for example, small and large, which are interpreted as labels of fuzzy sets in their respective universes of discourse.

Basically, the calculus of fuzzy rules provides an effective methodology for dealing with imprecise dependencies in systems analysis. The methodology of CFR differs very substantially-both in spirit and in substance- from the conventional approaches which are employed in control theory and systems design. An important feature of CFR is that it is close to human intuition. In fact, most of our experiences are stored in our memory in the form of fuzzy rules and almost all of human reasoning involves a manipulation of such rules on both conscious and subconscious levels. In this perspective, the role model for the calculus of fuzzy rules is the human mind.

A good engineering approach should be capable of making use of all the available information effectively. For many practical problems, an important portion of information comes from human experts. Usually, the expert information is not precise and is represented by fuzzy terms like *small, large, not very large* and so on. There are many reasons why expert information is usually expressed in,

fuzzy terms, such as for convenience or lack of more precise knowledge, ease of communication, and so forth. In order to make use of the expert information in a systematic manner, the so-called intelligent approaches (intelligent control, intelligent signal processing, and others [3]) have been emerging in the engineering community. However, most such intelligent approaches are ad hoc in nature in the sense that there are no analytic tools for general design procedures to guarantee basic performance criteria. Usually, these intelligent approaches combine expert systems with conventional engineering systems in an ad hoc manner, and then simulations are performed to show the validity of the approaches to the specific problems. There are serious limitations to using expert information in this way because it is inefficient, is not generally applicable, and has no guarantee of performance.

An important research topic is to develop general approaches to incorporate expert information systematically for which theoretical analyses can be performed to study the performance of the resulting systems, for example, the stability of an intelligent control system. However, this is not an easy task. In addition to the expert information, another important portion of information is numerical information, which is collected from various sensors or obtained according to physical laws. Numerical information and expert linguistic information have many fundamental differences. For example, numerical information obeys physical laws and mathematical axioms, whereas there are no such laws and axioms for linguistic information. In other words, the laws governing linguistic information are fundamentally different from the laws governing numerical information. There are two worlds-the physical world and the human world-and as man-machine

interaction increases, more and more engineering systems belong to the mixture of these two different worlds. In order to analyze the systems in this mixed world, it is essential that we find a common framework to represent key elements in these two worlds. Fuzzy control systems provide such a common framework.

CHAPTER 2

BACKGROUND

The desire of people to control nature's forces successfully has been the catalyst for progress throughout history. The goal has been to control these forces in order to help perform physical tasks which were beyond human's capabilities. During the dynamic and highly motivated 20th century, the control-system engineer has transformed many of our hopes and dreams into reality.

Control-system engineer have made very important contributions to our advancements in the 20th century, and they are building the foundation for greater advancements as we approach the 21st century. As we look back, control-system engineers have made contributions to robotics, space-vehicle systems, including the successful accomplishment of the lunar soft landing, aircraft autopilots and controls, control systems for ships and submarines, guidance systems for intercontinental missiles, and automatic control systems for hydrofoils, surface-effect ships, high-speed rail systems, and most recently, control systems for the magnetic-levitation rail systems. The future lies in our imagination, creativity, and ability to transform ideas into working automatic control systems that we can build

to work reliably and accurately, and which can be manufactured at a profit and on schedule.

The control of systems is an interdisciplinary subject and cuts across all specialized fields. The versatile subject of automatic control ranks today as one of the most promising fields, and its growth potential appears unlimited.

2.1. The History Of Computerized Control Theory

A computer-controlled system can be schematically described as in Fig 2.1. The output from the process y(t) is a continuous -time signal. The output is converted into digital form by the analog-to-digital (A-D) converter. The A-D converter can be included in the computer or regarded as a separate unit, according to one's preference. The conversion is done at the sampling times, t_k. The computer interprets the converted signal, $\{y(t_k)\}$, as a sequence of numbers, $\{u(t_k)\}$. This sequence is converted to an analog signal by a digital-to-analog (D-A) converter. Notice that the system runs open loop in the interval between the A-D and the D-A conversion.

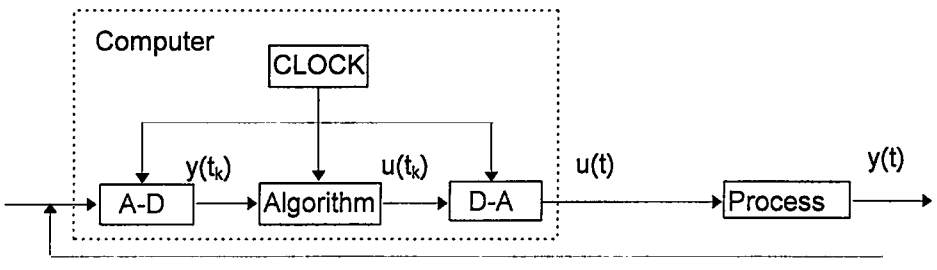


Fig. 2.1. Computer Controlled System

The idea of using digital computers as components in control systems emerged around 1950. Applications in missile and aircraft control were investigated first. Studies showed that there was no potential for using the general-purpose digital computers that were available at that time. The computers were too big, they consumed too much power, and they were not sufficiently reliable. For this reason special-purpose computers-digital differential analyzers (DDA)- were developed for the early aerospace applications [10]. The major developments in computer control occurred in the process industries.

The major tasks of the computer were to find the optimal operation conditions, to perform scheduling and production planning, and to give reports about production and raw-material consumption. The problem of finding the best operating conditions was viewed as a static optimization problem. Mathematical models of the processes were necessary in order to perform the optimization [11]. The modules used-which were quite complicated- were derived from physical models and from regression analysis of process data. Attempts were also made to carry out on-line optimization.

2.2. The Philosophy Of Classical Control

Having some understanding of the history of automatic control theory, we may now briefly discuss the philosophies of classical and modern control theory.

Developing as it was done for feedback amplifier design, classical control theory was naturally couched in the frequency domain and the s-plane [6].

Relaying on transform methods, it is primarily applicable for linear time invariant systems, though some extensions to nonlinear systems were made using, for instance, the describing function.

The system description needed for controls design using the methods of Nyquist and Bode is the magnitude and phase of the frequency response [20]. This is advantageous since the frequency response can be experimentally measured. The transfer function can then be computed. The block diagram is heavily used to determine transfer functions of composite systems. An exact description of the internal system dynamics is not needed for classical design; that is, only the input/output behavior of the system is of importance.

The design may be carried out manually using graphical techniques. These methods impart a great deal of intuition and afford the controls designer with a range of design possibilities, so that the resulting control systems are not unique. The design process is an engineering art.

A real system has disturbances and measurement noise, and may not be described exactly by the mathematical model the engineer is using for design. Classical theory is natural for designing control systems that are robust to such disorders, yielding good closed-loop performance in spite of them. Robust design is carried out by using notions like the gain and phase margin.

Simple compensators like proportional-integral-derivative (PID), lead-lag, or wash-out circuits are generally used in the control structure. The effects of such circuits on the Nyquist, Bode, and root locus plots are easy to understand, so that a suitable compensator structure can be selected. Once designed, the compensator can be easily tuned on line.

A fundamental concept in classical control is the ability to describe closed-loop properties in terms of open-loop properties, which are known or easy to measure. For instance, the Nyquist, Bode, and root locus plots are in terms of open-loop transfer function[20]. Again, the closed-loop disturbance rejection properties and steady-state error can be described in terms of the return difference and sensitivity.

Classical control is difficult to apply in multi-input/multi-output (MIMO) systems. Due to the interaction of the control loops in a multi variable system, each single-input/single-output (SISO) transfer function can have acceptable properties in terms of step response and robustness, but the coordinated control of the system can fail to be acceptable.

Thus classical MIMO or multiloop design requires painstaking effort using the approach of closing one loop at a time by graphical techniques. A root locus, for instance, should be plotted for each gain element, taking into account the gains previously selected. This is a trial-and-error procedure that may require multiple iterations, and it does not quarantee good results, or even closed-loop stability.

The multivariable frequency-domain approaches developed by the British school during the 1970s, as well as quantitative feedback theory, overcome many of these limitations, providing an effective approach for the design of many MIMO systems.

2.3. The Philosophy of Modern Control

Modern controls design is fundamentally a time-domain technique [12]. A state space model of the system to be controlled, or plant, is required. The linear version is a first order differential equation of the form

$$\dot{x} = Ax(t) + Bu(t) \qquad (2.1)$$

$$y(t) = Cx(t)$$

where x(t) is a vector of internal variables or system states, u(t) vector of control inputs, and y(t) is a vector of measured outputs. It is possible to add noise terms to represent process and measurement noises. Note that the plant is described in the time domain.

The power of modern control has its roots in the fact that the state- space model can as well represent a MIMO system as a SISO system. That is, u(t) and y(t) are generally vectors whose entries are the individual scalar inputs and outputs. Thus A, B, C are matrices whose elements describe the system dynamical interconnections.

Exactly as in classical case, some fundamental questions on the performance of closed-loop system can be attacked by investigating open-loop properties. For instance open-loop properties of controllability and observability of (2.1) give insight on what is possible to achieve using feedback control. The difference is that, to deal with the state-space model, a good knowledge of matrices and linear algebra is required.

To achieve suitable closed-loop properties, a feedback control of the form

$$u = -Kx \tag{2.2}$$

may be used. The feedback gain K is a matrix whose elements are individual control gains in system. Since all the states are used for feedback, this is called state-variable feedback. Note that multiple feedback gains and large systems are easily handled in this framework. Thus, if there are n state components (where n can be very large in aerospace or power distribution system) and m scalar controls, so that u(t) is an m-vector, then K is an m x n matrix with mn entries, corresponding to mn control loops.

In the standart linear quadratic regulator (LQR), the feedback gain K is chosen to minimize a quadratic time-domain performance index (PI) like

$$J = \int_0^\infty \left(x^T Q x + u^T R u\right) dt. \tag{2.3}$$

The minimum is sought over all state trajectories. This is an extension to MIMO systems of the sorts of PIs that were used in classical control. Q and R are weighting matrices that serve as design parameters. Their elements can be selected to provide sutable performance.

The key to LQR design is the fact that, if the feedback gain matrix K can be successfully chosen to make J finite, then integral (2.3) involving the norms of u(t) and x(t) is bounded. If Q and R are correctly chosen, well-known mathematical

principles then ensure that x(t) and u(t) go to zero with time. This quaranties closed-loop stability as well as bounded control signals in the closed-loop system.

It can be shown that the value of K that minimize the PI is given by

$$K=R^{-1}B^{T}S \tag{2.4}$$

where S is an n x n matrix satisfying the Ricatti equation

$$0 = A^{T}S+SA-SBR^{-1}B^{T}S+Q \tag{2.5}$$

Whitin this LQ framework, several points can be made. First as long as the system (2.1) is controllable and Q and R are suitably chosen, the K given by these equations guarantiees the stability of the closed-loop system

$$\dot{x}(t) = (A - BK)x(t) + Bu. \tag{2.6}$$

Second this technique is easy to apply even for multiple-input plants, since u(t) can be a vector having many components.

Third, the LQR solution relies on the solution of the matrix design equation (2.5), and so is unsuited to hand calculations. Fortunately, many design packages are by now available on digital computers for solving the Ricatti design equation for S, and hence for obtaining K[12]. Thus, computer aided design is an essential feature of modern controls.

The LQR solution is a formal one that gives a unique answer to the feedback control problem once the design parameter Q and R have been selected.

In fact, the engineering art in modern design lies in the selection of the PI weighting matrices Q and R. A body of theory on this selection process has evolved. Once Q and R are properly selected, the matrix design equation is formally solved for the unique K that guarantees stability.

Observe that K is computed in terms of the open-loop quantities A, B, Q, R, so that modern and classical design have this feature of determining closed-loop properties in terms of open-loop quantities in common. However, in modern control, all the entries of K are determined at the same time by using the matrix design equations. This corresponds to closing all the feedback control loops simultaneously, which is in complete contrast to the one-loop-at-a-time procedure of classical controls design.

Unfortunately, formal LQR design gives very little intuition on the nature or properties of the closed-loop system. In recent years, this deficiency has been addressed from a variety of standpoints.

Although LQR design using state feedback guarantees closed-loop stability, all the state components are seldom available for feedback purposes in a practical design problem. Therefore output feedback of the form

$$u = -Ky \tag{2.7}$$

is more useful. LQR design equations for output feedback are more complicated than (2.5), but are easily derived.

Modern output-feedback design allows one to design controllers for complicated systems with multiple inputs and outputs and many control loops by formally solving matrix design equations on a digital computer.

Another important factor is the following. Although the state feedback (2.2) involves feedback from all states to all inputs, offering no structure in the control system, the output feedback control law (2.7) can be used to design a compensator with a desired dynamical structure, regaining much of the intuition of classical control design.

Feedback laws like (2.2) and (2.7) are called static, since the control gains are constants, or at most time-varying. An alternative to static feedback is to use a dynamic compensator of the form

$$z' = Fz + Gy + Eu \qquad (2.8)$$

$$u = Hz + Dy.$$

The inputs of this compensator are the system inputs and outputs. This yields a closed loop and is called dynamic output feedback. The design problem is to select the matrices F, G, E, H, D for good closed-loop performance. An important result of modern control is that closed-loop stability can be guaranteed by selecting F= A-LC for some matrix L which is computed using a Riccati design equation similar to (2.5). The other matrices in (2.8) are then easily determined. This design is based on the vital ***seperation principle*** [6].

A disadvantage with design using F=A-LC is that then the dynamic compansator has the same number of internal states as the plant. In complicated

modern aerospace and power plant applications, this dimensions can be very large. Thus, various techniques for controller reduction and reduce-order design have been developed.

In standard modern control, the system is assumed to be exactly described by the mathematical model (2.1). In practice, however, this model may be only an approximate description of the real plant. Moreover, in practice there can be disturbances acting on the plant, as well as measurement noise in determining y(t).

With the work on robust modern design, much of the intuition of classical controls techniques can now be incorporated in modern multivariable design.

With modern developments in digital control theory and discrete-time systems, modern control is very suitable for the design of control systems that can be implemented on micro processors. This allows the implementation of controller dynamics that are more complicated as well as more affective than the simple PID and lead-lag structure of classical control.

With recent work in matrix-fraction descriptions and polynomial equation design, a MIMO plant can be described not in state-space form, but in input/output form. This is a direct extension of the classical transfer function description and, for some applications, is more suitable than the internal description (2.1).

2.4. Computer-Control Theory

The schematic diagram of a computer-controlled system (shown in Fig. 2.1) contains essentially five parts: the process, the A-D and D-A converters, the control algorithm, and the clock. Its operation is controlled by the clock. The times

when the measured signals are converted to digital form are called the sampling instants; the time between successive samplings is called the sampling period and is denoted by h. Periodic sampling is normally used but there are, of course, many other possibilities. For example, it is possible to sample when the output signals have changed by a certain amount. It is also possible to use different sampling periods for different loops in a system. This is called multirate sampling [6].

The only difference between a computer-controlled system and an ordinary analog-feedback system is that the control law is implemented using a digital computer, so the class of control laws that can be used conveniently is greatly increased. For example, it is easy to use nonlinear calculations, to incorporate logic, and to perform substantial calculations in the controller. Tables can be used to store data in order to accumulate knowledge about the properties of the system.

2.4.1. Is There a Need for a Theory for Computer-Controlled Systems?

A good theory should make it possible to understand how a system like the one in Fig. 2.1 works and how it should be designed. It seems clear that a sampled system would behave as a continuous-time system if the sampling period were sufficiently small. This is certainly true under very reasonable assumptions. Is there then any need for a special theory for computer-controlled systems?

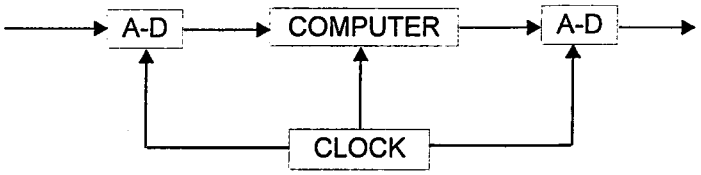
Some examples will be used to show that the system in Fig. 2.1 cannot be fully understood in terms of the theory of time-invariant, linear systems even if the process to be controlled is a linear, time-invariant, continuous-time system.

Example 1-Time dependence

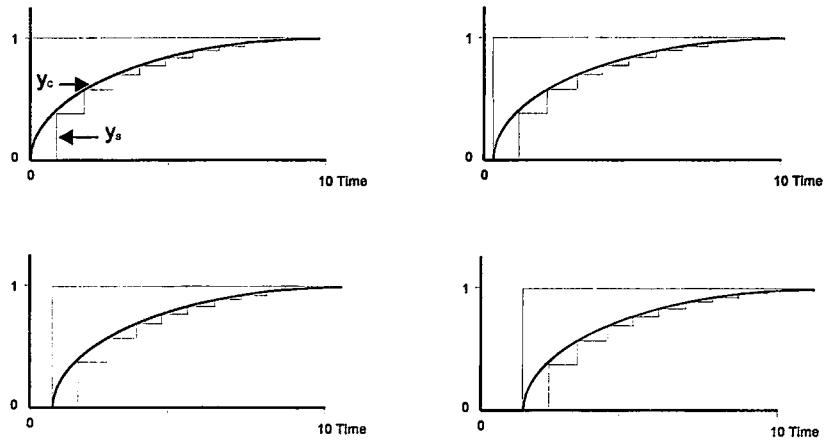
Suppose that we want to implement a compensator that is simply a first-order lag. Such a compensator can be implemented using A-D conversion, a digital computer, and D-A conversion. The first-order differential equation is approximated by a first-order difference equation. The step response of such a system is shown in Fig. 2.2 The figure clearly shows that the sampled system is not time invariant because the response depends on the time when the step occurs. If the input is delayed, then the output is delayed by the same amount only if the delay is a multiple of the sampling period.

The phenomenon illustrated in Fig. 2.2 depends on the fact that the system is controlled by a clock (compare with Fig. 2.1). The response of the system to an external stimulus will then depend on how the external event is synchronized with the internal clock of the computer system.

A computer-controlled system with periodic sampling is a periodic system. The effects of the periodicity can be made arbitrarily small by choosing a sufficiently high sampling rate; however, it is necessary to consider the periodic nature of sampled systems to understand fully how they work. This is further illuminated by another example.



a)



b)

Fig. 2.2

a. Block diagram of a digital filter.

b. Step responses, y_s, of a digital computer implementation of a first-order lag for different delays in the input step compared with the first sampling instant. For comparison the response of the corresponding continuous-time system, y_c, is also shown.

Example 2-Higher-order harmonics

It is well known that a sinusoidal input signal applied to a stable, linear, time-invariant, continuous-time system will-after a transient-give signals in the system that are sinusoidal with the same frequency as that of the input. Fig.2.3 shows what can happen when a computer-controlled system is subjected to a periodic excitation. A sinusoidal signal of frequency 4.9 Hz is applied to the system in Example 1. It is clear that the phenomena shown in Fig. 2.3 cannot be explained in terms of linear, time-invariant systems. The beats in the output of the computer-controlled system are due to

interference between the input frequency and a higher frequency generated through the sampling process.

There are many aspects of sampled systems that can indeed be understood by linear, time-invariant theory. The examples given indicate, however, that the sampled systems cannot be fully understood within that framework. It is thus useful to have other tools for analysis.

In solving the problem of controller design, it is certainly possible to use ordinary continuous-time control theory to design a control law and then to use a reasonable discrete-time approximation. An example follows.

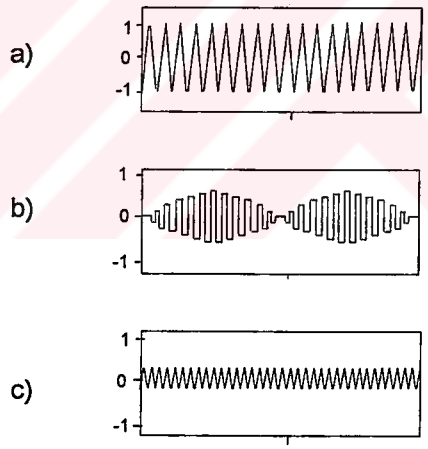


Fig.2.3 Sinusoidal excitation of the sampled system in Example 1.

a. Input sinusoidal with frequency 4.9 Hz.

b. Sampled-system output. The sampling period is 0.1 s.

c. Output of the corresponding continuous-time system.

Example 3-Discrete-time approximation

A double-integrator plant can easily be controlled by state feedback. A straight forward way to do this is simply to calculate the feedback gains using continuous-time theory and then to implement the state feedback using computer control. If the sampling period is sufficiently small, digital control can be expected to have the same properties, for all practical purposes, as continuous-time control. Fig.2.4 illustrates that this is indeed the case. The agreement with the continuous-time regulator can be made even better by reducing the sampling period.

Based on the results of Example 3 it would be tempting to conclude that we do not need a theory for sampled systems. This is, however, not correct, because computer-controlled systems can indeed perform better than their continuous-time equivalents. This is the main reason why a theory for sampled systems is useful! An example illustrates this further .

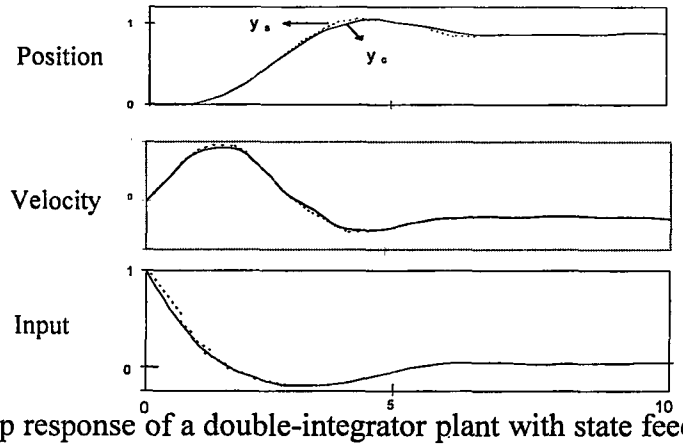


Fig.2.4 Step response of a double-integrator plant with state feedback, continuous-time solution y_c and sampled approximation y_s, when the sampling period is 0.2 s.

Example 4-Deadbeat control

Consider the double-integrator plant of Example 3. The result of computer-controlled state feedback with a special linear-feedback strategy is shown in Fig 2.5 The strategy is of the same form as the control strategy used in Example 3, i.e. a linear feedback from the sampled values of the states. The feedback coefficients and the sampling period are, however, different. The particular control strategy is called deadbeat control.

A comparison of Fig. 2.4 and 2.5 shows that the system in Fig. 2.5 settles faster than the continuous-time system, even if the largest magnitude of the control signals is the same in both cases. The magnitude of the velocity is, however larger in Fig. 2.5 The sampled system does not have any overshoot. Observe also that the signals settle on constant values after a finite time. This cannot happen with-continuous-time systems because the solutions to such systems are sums of functions that are products of polynomials and exponential functions. Notice also that the sampling period used in Fig.2.5 is five times longer than the sampling period used in the approximation of the continuous-time control system in Fig. 2.4.

The example demonstrates clearly that even in the linear case something better than just an approximation of a continuous-time regulator is possible.

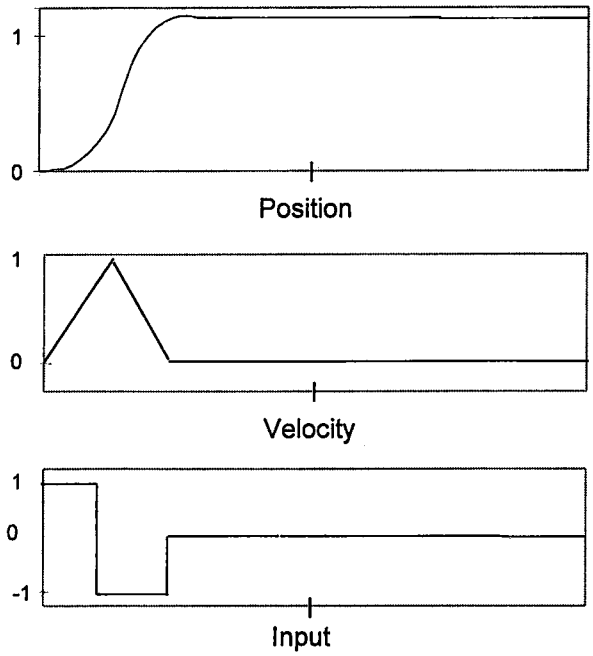


Fig.2.5 Computer control of the double-integrator plant with a deadbeat strategy. The sampling period is 1 s.

2.4.2. Inherently Sampled Systems

Sampled models are natural descriptions for many phenomena. The theory of sampled-data systems, therefore, has many applications outside the field of computer control. A few examples follow.

2.4.3. Discrete-Time Systems as Models for Computer Algorithms

Algorithms in computers can be described as discrete-time systems. This will be illustrated with an iterative algorithm and a real-time application.

Example 5-Iterative solution

Iterative algorithms are examples of inherently sampled systems. Assume that the solution to an equation of the form

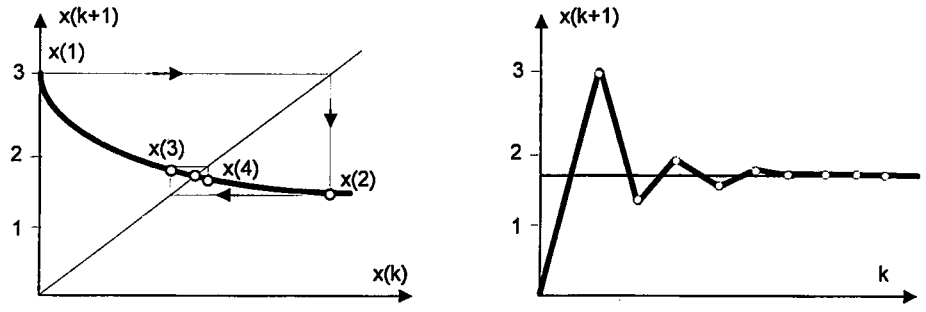
$$x - f(x) = 0$$

is desired. One way to find the solution is to guess an initial value and then use Picard's algorithm, i.e., to use the iterative scheme

$$x(k + 1) = f[x(k)]$$

where x(k) is the kth iteration. The numerical algorithm can thus be interpreted as a discrete-time system in which the time represents the number of iterations.

Specifically, assume that $f(x) = 3 - \sqrt{x}$. In this case it is easy to show that the solution is $x = (7 - \sqrt{13})/2 = 1.697$. The sequence of numbers shown in Fig. 2.6 is obtained if one starts with the initial guess x(0) = 0.



a) b)

Fig.2.6 Two graphic illustrations of the iterative scheme in Example 5.

Example 6-Control algorithm

A simple computer algorithm for a proportional and integral (PI) controller follows:

uc:=adin(in1) {read reference value}

y:=adin(in2) {read process value}

e:=uc - y

u:=k*(e + i)

dout(u) {output control signal}

i:=i + h*e/ti

The program is executed every sampling period by a scheduling program, as illustrated in Fig. 2.7. The computer code is equivalent to the following difference equations:

$$e(k) = uc(k) - y(k)$$

$$u(k) = k[e(k) + i(k - 1)]$$

$$i(k) = i(k - 1) + he(k)/ti$$

2.4.4. Sampling Due to the Measurement System.

In many cases, sampling will occur naturally in connection with the measurement procedure. A few examples follow.

•

Example 7-Radar

When a radar antenna rotates, information about range and direction is naturally obtained once per revolution of the antenna. A sampled model is thus the natural way to describe a radar system. Attempts to describe radar systems were, in fact, one of the starting points of the theory of sampled systems.

Example 8-Analytical Instruments

In process-control systems, there are many variables that cannot be measured on-line, so a sample of the product is analyzed off-line in an analytical instrument such as a mass spectrograph or a chromatograph.

Example 9-Economic systems

Accounting procedures in economic systems are often tied to the calendar. Although transactions may occur at any time, information about important variables is accumulated only at certain times-e.g., daily, weekly, monthly, quarterly, or yearly,

2.4.5. Sampling Due to Pulsed Operation.

Many systems are inherently sampled because information is transmitted using pulsed information. Electronic circuits are a prototype example. They were also one source of inspiration for the development of sampled-data theory. Other examples follow.

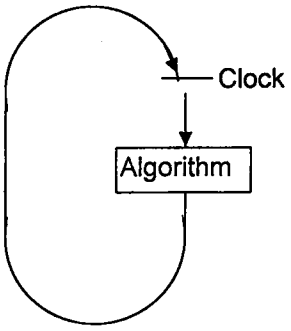


Fig.2.7 Scheduling of a computer program.

Example 10-Thyristor control

Power electronics using thyristors are sampled systems. Consider the circuit in Fig. 2.8 The current can be switched on only when the voltage is positive. When the current is switched on, it remains on until the current has a zero crossing. The current is thus synchronized to the periodicity of the main's supply.

Example 11-Biological systems

Biological systems are fundamentally sampled because the signal transmission in the nervous system is in the form of pulses.

Example 12-Internal combustion engines

An internal combustion engine is a sampled system. The ignition can be viewed as a clock that synchronizes the operation of the engine. A torque pulse is generated at each ignition.

The systems in these examples are periodic because of their pulsed operation. Periodic systems are quite difficult to handle, but they can be considerably simplified by studying the systems at instants synchronized with the pulses-that is, by using sampled-data models. The processes can then be described as time-invariant, discrete-time systems at the sampling instants. Examples 10 and 12 are of this type.

2.4.6. How Theory Developed.

Although the major applications of the theory of sampled systems are currently in computer control, many of the problems were encountered earlier. In this section some of the main ideas in the development of the theory are discussed.

2.4.7. The Sampling Theorem.

Since all computer-controlled systems operate on values of the process variables at discrete times only, it is very important to know the conditions under which a signal can be recovered from its values in discrete points only. The key issue was explored by Nyquist [20], who showed that to recover a sinusoidal signal from its samples, it is necessary to sample at least twice per period. A complete solution was given in an important work by Shannon [43].

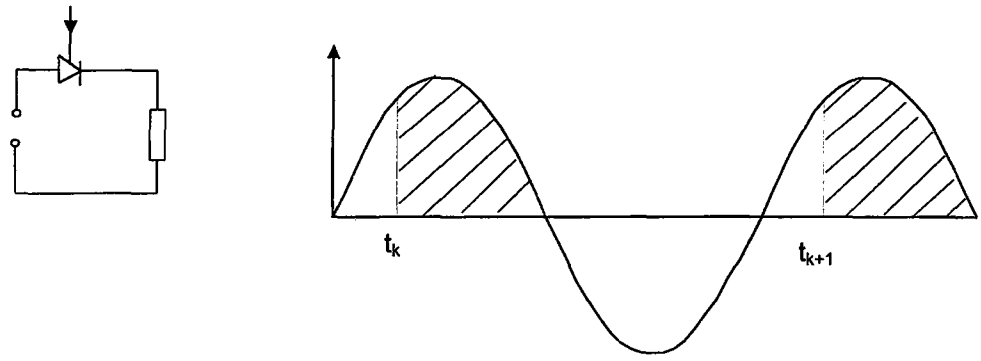


Fig.2.8 Thyristor control circuit.

2.4.8. Difference Equations.

The first germs of a theory for sampled systems appeared in connection with analyses of specific control systems. The behavior of the chopper-bar galvanometer, investigated in Oldenburg and Sartorius [44], was one of the earliest contributions to the theory. It was shown that many properties could be understood by analyzing a linear time-invariant difference equation. The difference equation replaced the differential equations in continuous-time theory. For example, stability could be investigated by the Schur-Cohn method, which is equivalent to the Routh-Hurwitz [6] criterion.

2.4.9. Transform Methods.

During and after World War II, a lot of activity was devoted to the analysis of radar systems. These systems are naturally sampled because a position measurement is

obtained once per antenna revolution. Since transform theory had been so useful for continuous-time systems, it was natural to try to develop a similar theory for sampled systems. The first steps in this direction were taken by Hurewicz) [45].He introduced the transform of a sequence {f(kh)}, defined by

$$z\{f(kh)\} = \sum_{k=0}^{\infty} z^{-k} f(kh)$$

This transform is similar to the generating function, which had been used so successfully in many branches of applied mathematics. The transform was later defined as the z-transform by Ragazzini and Zadeh [38]. Transform theory was developed independently in the Soviet Union, in the United States, and in Great Britain. Tsypkin [42] called the transform the discrete Laplace transform and developed a systematic theory for pulse-controlled systems based on the transform. The transform method was also independently developed by Barker [39] in England

In the United States the transform was further developed in a Ph.D. dissertation by Jury at Columbia University [40]. Jury developed tools both for analysis and design. He also showed that sampled systems could be better than their continuous-time equivalents. Jury also emphasized that it was possible to obtain a closed-loop system that exactly achieved steady state in finite time. In later works he also showed that sampling can cause cancellation of poles and zeros. A closer investigation of this property later gave rise to the notions of observability and reachability.

The z-transform theory leads to comparatively simple results. A limitation of the theory, however, is that it tells what happens to the system only at the sampling instants. The behavior between the sampling instants is not just an academic question, because it was found that systems could exhibit hidden oscillations. These oscillations are zero at the sampling instants, but very noticeable in between.

Another approach to the theory of sampled system was taken by Linvill [41]. Following ideas due to MacColl [46], he viewed the sampling as an amplitude modulation. Using a describing-function approach Linvill effectively described intersample behavior. Yet another approach to the analysis of the problem was the delayed z-transform, which was developed by Tsypkin [42], Barker [39], and Jury [40]. It is also known as the modified z-transform.

Much of the development of the theory was done by a group at Columbia University in the United States, led by Ragazzini. Jury, Kalman, Bertram, Zadeh, Franklin, Friedland, Kranc, Freeman, Sarachik, and Sklansky all did their Ph.D. work for Ragazzini.

Toward the end of the fifties, the z-transform approach to sampled systems had matured, and several textbooks appeared almost simultaneously: Ragazzini and Franklin [33], Jury [34], Tsypkin [35], and Tou [32]. This theory, which was patterned after the theory of linear, time-invariant, continuous-time systems, gave good tools for analysis and synthesis of sampled systems. A few modifications had to be made because of the time-varying nature of sampled systems. For example, all operations in a block-diagram representation do not commute!

2.4.10. State-Space Theory.

A very important event in the late fifties was the development of state-space theory. The major inspiration came from mathematics and the theory of ordinary differential equations and from mathematicians such as Lefschetz, Pontryagin, and Bellman. Kalman deserves major credit for the state-space approach to control theory. He formulated many of the basic concepts and solved many of the important problems.

Several of the fundamental concepts grew out of an analysis of the problem of whether it would be possible to get systems in which the variables achieved steady state in finite time. The analysis of this problem led to the notions of reachability and observability. Kalman's work also led to a much simpler formulation of the analysis of sampled systems: The basic equations could be derived simply by starting with the differential equations and integrating them under the assumption that the control signal is constant over the sampling period.

2.4.11 Optimal and Stochastic Control.

There were also several other important developments in the late fifties. Bellman) [37] and Pontryagin and others [30] showed that many design problems could be formulated as optimization problems. For nonlinear systems this led to non-classical calculus of variations. An explicit solution was given for linear systems with quadratic loss functions by Bellman and others [36]. Kalman [31] showed in a celebrated paper that the linear quadratic problem could be reduced to a solution of

a Riccati equation. Kalman also showed that the classical Wiener filtering problem could be reformulated in the state-space framework. This permitted a "solution" in terms of recursive equations, which were very well suited to computer calculation.

In the beginning of the sixties, a stochastic variational problem was formulated by assuming that disturbances were random processes. The optimal control problem for linear systems could be formulated and solved for the case of quadratic loss functions. This led to the development of stochastic control theory. The work resulted in the so-called Linear Quadratic Gaussian (LQG) theory. This is now a major design tool for multivariable linear systems.

2.4.12. Algebraic System Theory.

The fundamental problems of linear system theory were reconsidered at the end of the sixties and the beginning of the seventies. The algebraic character of the problems was reestablished which resulted in a better understanding of the foundations of linear system theory. Techniques to solve specific problems using polynomial methods were another result [see Kalman and others [28], Rosenbroek [27], Wonham [24], Blomberg and Ylinen [16], and Kucera [21]].

2.4.13. System Identification.

All techniques for analysis and design of control system are based on the availability of appropriate models for the process dynamics. The success of classical control theory that almost exclusively builds on Laplace transforms was

largely due to the fact that the transfer function of a process can be determined experimentally using frequency response. The development of digital control was accompanied by a similar development of system identification methods. These allow experimental determination of the pulse transfer function or the difference , equations that are the starting point of analysis and design of digital control systems. Good sources of information on these techniques are [26], [22].

2.4.14. Adaptive Control.

When digital computers are used to implement a controller it is possible to implement more complicated control algorithms. A natural step is to include both parameter estimation methods and control design algorithms. In this way it is possible to obtain adaptive control algorithms that determine the mathematical models and perform control system design on-line. Research on adaptive control began in the mid-fifties. Significant progress was made in the seventies when feasibility was demonstrated in industrial applications. The advent of the microprocessor made the algorithms cost-effective, and commercial adaptive regulators appeared in the early eighties. This has stimulated vigorous research on theoretical issues and significant product development. See, for instance, [25], [17].

2.4.15. Automatic Tuning.

Controller parameters are often tuned manually. Experience has shown that it is difficult to adjust more than two parameters manually. From the user point of view it is therefore helpful to have tuning tools built into the controllers. Such systems are similar to adaptive controllers. They are, however, easier to design and use. With computer based controllers it is easy to incorporate tuning tools. Such systems also started to appear industrially in the mid-eighties. See Aström and Hagglund [19].

CHAPTER 3

GENERAL CONCEPT OF CONVENTIONAL CONTROL-SYSTEMS

Control systems can be defined as devices which regulate the flow of energy, matter, or other resources. Their arrangement, complexity, and appearance vary with their purpose and function. In general, control systems can be categorized as being either open loop or closed loop. The distinguishing feature between those two types of control systems is the use of feedback comparison for closed-loop operation.

3.1. Open-Loop Control Systems

Open-loop control systems represent the simplest, form of controlling devices [10]. In this control system only the desired signal is input to the system, no feedback from the system to the controller is present. So the controller remains insensitive to the unpredicted changes and to some disturbances. In the Fig.3.1 a schematic presentation of an open-loop control for controlling the temperature in a room is shown.

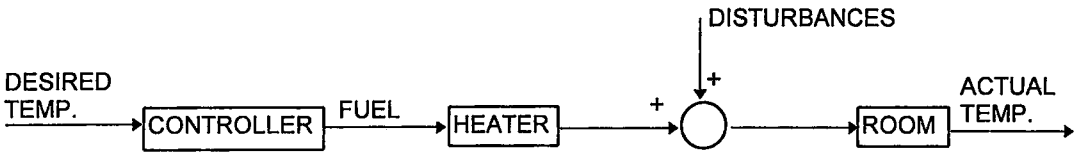


Fig. 3.1 Schematic presentation of an open-loop control.

Open-loop control systems are unable to cope with;

- The effects of disturbances,
- Unexpected variations of outside temperature,
- Changes in the wind speed,
- Opening and closing of doors and windows,
- Fluctuations in the number of people in the room,

.

.

This control system does not have any feedback comparison, and the term *open loop* is used to describe this absence.

The effect of disturbance torques, or other secondary inputs, is detrimental to the accurate functioning of an open-loop control system. It has no way of automatically correcting its output, because there is no feedback comparison. We must resort to changing the input manually in order to compensate for secondary inputs.

3.2. Closed-Loop Control Systems

To achieve a good control on the output of the controlled system closed-loop (control with feedback) is to be used [10]. In this control system the actual output of the system is used with the desired input to calculate the next command. In the Fig.3.2 the previous example Fig 3.1 is shown with closed-loop control-design and a switching device is used with ±δ sensivity to control the flow of fuel.

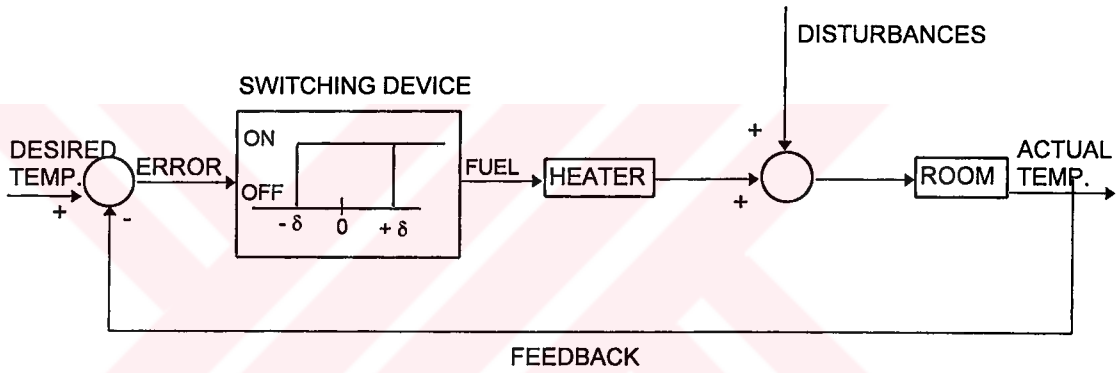


Fig. 3.2 Schematic presentation of a closed-loop control

In this case the stability of the system fluctuates between the ±δ sensivity of the switching device which is δ in this example. In the graph in Fig 3.3 the variation of the temperature is shown.

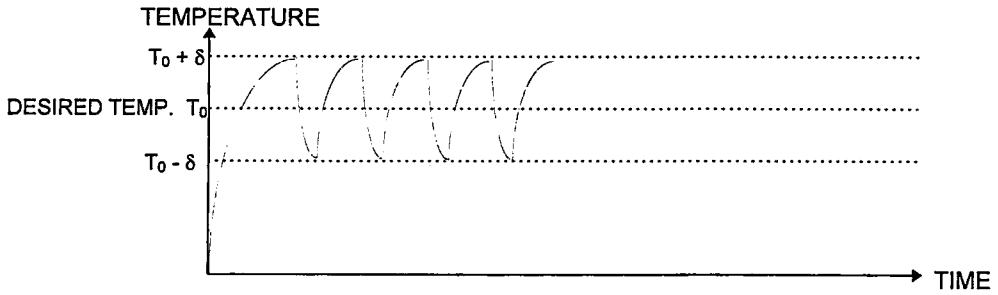


Fig. 3.3 Temperature variation.

To overcome these variations caused by the sensivity of switching devices the proportional controllers are used. In this kind of controllers an adjustable device (such as an adjustable valve for fuel) is used instead of a switching one. Proportional controller computes the difference between the actual output and the desired level and calculates an error. Control action is taken, proportional to the level of this error. Consider the system of proportional controller shown in Fig. 3.4.

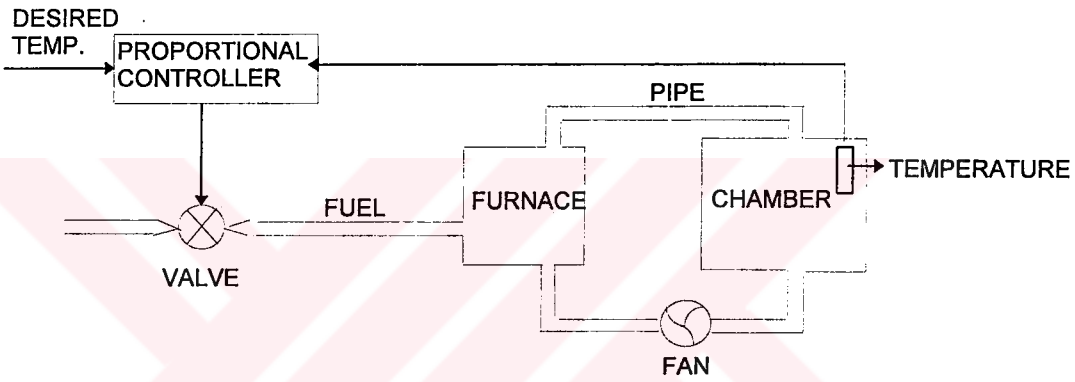


Fig. 3.4 Proportional controller.

Owing to the time delay for heat to propagate between the furnace and the chamber the control signal fluctuates for some period before reaching the stable condition.(see Fig 3.5)

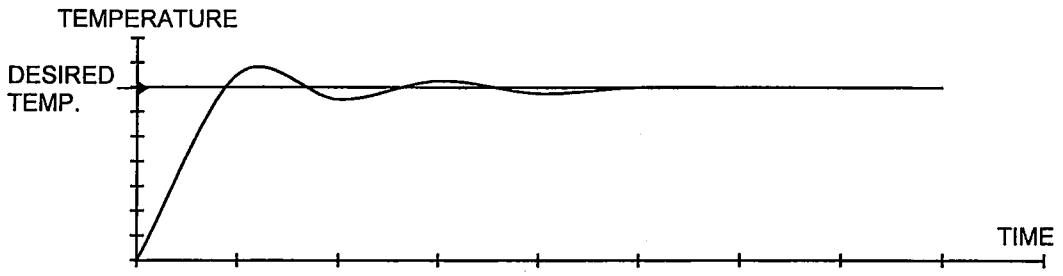


Fig. 3.5 Temperature fluctuation.

Closed-loop control systems derive their valuable accurate reproduction, of the input from feedback comparison. An error detector derives a signal proportional to the differences between the input and output. The closed-loop control system drives the output until it equals to the input and the error is zero. Any differences between the actual and desired output will be automatically corrected by closed-loop control system. Through proper design, the system can be made relatively independent of secondary inputs and changes in component characteristics.

Feedback control systems used to control position, velocity, and acceleration are very common in industrial and military applications. They have been given the special name of servomechanisms. With all their many advantages, feedback systems have a very serious disadvantage, because the closed-loop system may inadvertently act as an oscillator. Through proper design, however, all the advantages of feedback can be utilized without having an unstable system.

Industrial fields where automated control is very common. [10]

Robotics

NASA Space Programs

Rail Transportation

Air Transportation

Military Systems

Surface-Effect Ships and Hydrofoils

Biomedical Control Systems

Modeling Systems Unrelated to Control

3.3. Summary Of Traditional Control Systems

The classical and modern control philosophies utilize the mathematical model of the process. This mathematical model can be in matrix or polynomial equation form [11]. Thus in both of these forms the precious definition of the mathematical model is to be defined in order to implement any kind of control. To give an exact definition of a system in mathematical model is not easy, especially if the system includes many variables to be controlled (you have to define a control input for each of the possible output of the process). Furthermore in many of the complex processes our knowledge is not precious, but general, for example in a chemical process if the temperature is high then the reaction is intense. This kind of knowledge in some cases may be the only information source.

As can be concluded from the preceding titles, the control-systems could be classified as open and closed loop. In open loop there exist no feedback from the process to the controller, the controller signals only the desired commands to the process and assumes that the devices and the procedures which take part in the process, give accurate and exact response to the commands. But in the closed-loop systems the response given to the commands of the controller (which can be the output of the process) taken into consideration to calculate the difference between the desired state and the actual state of the process, and this calculated difference is used in the generation of next command given to the process.

CHAPTER 4

FUZZY CONTROL SYSTEMS

4.1. Background

Fuzzy sets are a mathematical concept proposed by Prof. L. A. Zadeh in 1965 [29], but in the motivations one can see a concealed wish to improve the relation-ship between humanity and the computer. If there are points of similarity between computers, which are logical machines, and the thinking of people, with their emotions and intuition, there are also differences. If the capabilities of humans and computers could be put together, a remarkable system would be possible. Whether or not fuzzy sets can serve as a go-between depends on the extent of their applications.

Putting aside the purely theoretical study of fuzzy sets, it is always necessary to make comparisons with other methodologies when we think about applications. If there are no results that cannot be obtained from another method, even if those similar results are obtained with simpler processing, it is difficult for a new methodology to survive. Above all, the outstanding feature of fuzzy sets is the ability to express the amount of ambiguity in human thinking and subjectivity

(including natural language) in a comparatively undistorted manner. Therefore, the fields of application involve problems connected to the very heart of humankind.

There are also other mathematical models that do not use physical laws. Data from actual phenomena are collected, and modeling is done through statistical processes. Although this is the method most commonly used, human and social interpretation of a model generated in this way is not always easy, and there is no guarantee of accuracy. Therefore, even though this type of model is expressed in quantitative terms, it should be understood qualitatively. Also, a large amount of uniform data is needed for statistical models. With a complex, ambiguous object, the word uniform has no clear definition, and it is not easy to gather large amounts of data. Even if some data that show the essence of the object are collected, they will be buried by the large amount of worthless data, when statistical operations are performed. But since the essence is still unknown, no data can be removed. In the world of science, decisions about truth cannot be made by majority.

Recently, logical models, which take a different form from mathematical ones, have come into use. One of them is the structural model, wherein the principle factors that make up the problem are determined and the presence or absence of mutual relationships among these factors investigated and with complex, ambiguous problems because the structural characteristics appeal to our visual sense. Also, even though it is a model, human intuition and insight are necessary for interpretation, as with a written description. However, since it is a graphic representation, there is greater freedom for interpretation, and since the structure itself is objective, it works to bring together subjectivity and objectivity. The problems with structural models are that the definition of the word

relationship for the principle factors is not clear and that it uses a two-valued, "yes" or "no" system to represent relationships.

Predicate logic offers a different logical model. Used not only in artificial intelligence but also in cognitive psychology, it forms the basis of knowledge engineering [13]. In predicate logic, regardless of subjectivity or objectivity, the knowledge that humans have is expressed by short sentences called propositions. The propositions are a written model, so they, can be used for ambiguous objects. To use these propositions for predicate logic, the definitions and meanings must be strictly determined. The problems here are as would be expected: the strictness and the two-valued logic. This is because the subjects of the propositions are divorced from reality and cannot be called models. Also, there is room for examination of the reasoning and coordination among the propositions themselves.

As a result, one can say that none of the modeling approaches used up to now are very suitable for use with problems with complex ambiguities. At this point we want something that satisfies our requirements for such contradictions as logic and illogic, subjectivity and objectivity, macro and micro, qualitative and quantitative, ambiguity and precision. The way in which humans think incorporates these kinds of contradictions, so it would be ideal if they could be expressed.

Fuzzy set theory meets these requirements to a certain extent [14]. It is especially suitable for expressing the ambiguity of meaning found in natural language, since it loosens up the overly inhuman logicality of predicate logic. However, if the problem taken as an object is not fully understood when it is used, there will be no results. As with other models, there are both good points and

shortcomings to fuzzy models. If they are used without considering their shortcomings or when another model would be more appropriate, the result will be that one goes to all the trouble of making the calculations and building the model but is unable to interpret it and will not understand the results.

When we make use of fuzzy sets, we must make the following points clear:

(1) What part of the problem do we fuzzify, and for what purpose ?

(2) What kind of fuzzy model will we use ?

To understand the features of fuzzy sets as they apply to modeling, it is better to divide our thinking into two categories, "set models" and "fuzziness." Let us first take a look at the problem of modeling with ordinary sets. Since the sets are groups made from the constants, variables, and functions of the object system, the expression is macroscopic and more ambiguous than models that are not based on groupings, just as showing the range in which a numerical value exists is not as strict as just stating a value. If the state and constraints of a system or its input and output and evaluation are expressed as an ordinary set, that in itself introduces one kind of ambiguity. In this case, “the laws" of the system (input/output relationships, constraints, etc.) are the "mappings" that express the relationships between sets. When functions are used for mapping the model comes close to being mathematical, but logical operations are also common, and in this case we get a logic-type model. Other than predicate logic, set models are not very widely known, so their characteristics, meanings, and features must be fully understood. For example, the set elements, the range of their definition, the expression of relationships between sets, what that expression means, etc., must be clear. Even

with predicate logic, if the first definitions and understanding are ambiguous and expressed by unsatisfactory sentences, there is a good chance that the results will be misunderstood, because of the flexibility of the sentences.

Next comes the conversion to fuzzy sets. This is done by gradation of the boundaries of the states, relationships, constraints, and goals of the system model made with ordinary sets. Care has to be taken that even though the boundaries are made vague, the logical structure itself is not. Therefore, the effectiveness of the conversion to fuzzy sets must be that the range of definition becomes ambiguous and that gives rise to a haziness in the relationships that go along with them. In this way, the unnaturalness of dividing the meanings of natural language and the truth of propositions into two values is canceled. Making the boundaries of sets vague has a completely different meaning from statistical variation. The former can freely delineate an ambiguous situation using things like individual subjectivity, experiences, and common sense. The ambiguity of language and emotion are of this type. On the other hand, as stated before, the latter expresses what proportion of the total of a large number of data is occupied by a certain type of data. Therefore, when there is little data, when the total number of the data has no meaning, or when the occurrence of events is not clear, probability cannot express the amount of ambiguity. Next we move on to using these fuzzy sets to make a goal of model for which the goals must first be clear. We may choose from among the three fields of application-machine, human and human-machine systems- for which the aims are as follows:

(1) Express human experiences, common sense, etc., in a form that machines can use.

(2) Make models of human feelings or language.

(3) Imitate human pattern recognition, overall judgment, or general understanding.

(4) Convert information into a form that people can easily understand.

(5) Compress large amounts of information.

(6) Make models of human psychology or behavior.

(7) Make models of societal systems.

Once the goal has been decided, the problem arises of what part of the system and what form the conversion to fuzzy sets will take. There are many forms of system models, but most of them include the following: state variables, independent variables, decision variables, disturbance, laws of cause and effect (transition), their truth values, goals, constraints, evaluation functions, various types of constants.

All of the above are ambiguous. However, if all of them are expressed in terms of fuzzy models, the results will not serve as a model. That is because a model gets its meaning from being a concise expression of the essence of a real problem. Also, the various amounts of ambiguity mentioned above are mutually related. For example, if the state variables are ambiguous, there will naturally be ambiguity in an evaluation based on the state, even if the evaluation functions are definitive.

There are two methods for developing fuzzy models: using the laws of cause and effect and using those of transition. The first makes use of the laws for operations with sets, so the rules for composition and reasoning express the

relationships among the variables expressed in the sets. The problem of which rules are best used is determined by the ease of operation and the quality of phenomenological expression. The other method uses ordinary equations to express cause-and-effect relationships, and fuzzy sets are used for the variables. In this case the meaning of the model itself is clear, so the results of conversion to fuzzy sets are easy to explain.

The procedure for developing a fuzzy model ideally follows the two-stage order. Sets and logical relationships are set up first, and afterwards the conversion to fuzzy sets takes place. This approach makes clear the whereabouts of problems, the results of conversion to fuzzy sets, and the interpretation of the model.

As we said before, the application of fuzzy sets is firmly tied to human thinking and behavior, and is what could be called human simulation. Therefore, the study of humans themselves is very important for getting good results. The greatest goal for the use of fuzzy logic is to take in the good points of human beings and compensate for their shortcomings.

4.2. Fuzzy Relations

In this chapter we will explain fuzzy relations [9], which can be discussed as generalizations of ordinary relations. We will begin with a definition of fuzzy relations, express fuzzy relations in terms of matrices and graphs, and then define the various operations. Then we will give a somewhat detailed explanation of how fuzzy reasoning, which plays an important role in fuzzy control and fuzzy

diagnosis, is explained in terms of compositions of fuzzy relations. In addition, we will show fuzzy relational equations and introduce their solutions.

4.2.1. Identification of Fuzzy Relations

Ambiguous relationships such as " x and y are almost equal," " x and y look very similar," and " x is much more beautiful than y" are often topics of everyday conversation, but expressing these kinds of ambiguous relationships in terms of ordinary relations is very difficult. Fuzzy relations are what makes it possible to express these frequently used ambiguous relationship.

Fuzzy relation can be explained as extensions of ordinary relations, and their range of application is very wide. For example, they are frequently applied in clustering, pattern recognition, inference, systems, and control. They also have applications in the fields known as "soft sciences," such as psychology, medicine, economics and sociology.

Fuzzy relation R from set X to set Y (or between X and Y) is a fuzzy set in the direct product $X \times Y=\{(x,y) \mid x \in X, y \in Y\}$, and is characterized by a membership function μ_R:

$$\mu_R: X \times Y \rightarrow [0,1]. \qquad (4.1)$$

Especially when X = Y, R is known as a fuzzy relation on X.

Example 4.1. Let X be a real number set. For $x, y \in X$, the relation "y is much larger than x," x<<y, is a fuzzy relation R and can be characterized by the following membership function:

$$\mu_R(x,y) = \begin{cases} 0, & X \geq Y \\ \dfrac{1}{1+\left(\dfrac{10}{Y-X}\right)^2}; & X < Y \end{cases}$$

Example 4.2. If x and y are people, relationships like "x and y look alike" and "x is much taller than y" are fuzzy relations.

Fuzzy relation R is expressed as follows by using the notation of fuzzy sets:

$$R = \int_{X>Y} \mu_R(x,y)/(x,y); \qquad x \in x;\ y \in Y. \qquad (4.2)$$

Example 4.3. Let X be a real number set. If ≈ means "x and y are almost equal," ≈ is

$$\approx = \int_{XxX} e^{-a\,x-y}/(x,y); \qquad a>0.$$

As a generalization of fuzzy relations, the n-ary fuzzy relation R in X_1 x X_2 x ...x X_n is given by

$$R = \int_{X_1 x X_2 x \ldots x X_n} \mu_R(x_1,x_2,\ldots,x_n)/(x_1,x_2,\ldots,x_n); \qquad x_i \in X_i \qquad (4.3)$$

and we get the following membership function.

$$\mu_R: X_1 x X_2 x \ldots x X_n \rightarrow [0,1].$$

When n = 1, R is an unary fuzzy relation, and this clearly a fuzzy set in X_1. Other ways of expressing fuzzy relations include matrices and graphs.

4.2.2. Fuzzy Matrices and Fuzzy Graphs

Given finite sets $X = \{x_1,x_2,\ldots,x_m\}$, $Y = \{y_1,y_2,\ldots,y_n\}$, a fuzzy relation in X x Y can be expressed by an m x n matrix like the one in Fig. 4.1.

This kind of matrix, which expresses a fuzzy relation, is called a fuzzy matrix. Since μ_R has values within the interval [0,1], the elements of the fuzzy matrix also have values within [0,1].

In order to express fuzzy relation R in a graph, for $\mu_R(x_i,y_j)$, we make x_i,y_j vertices and add the grade $\mu_R(x_i,y_j)$ to the arc from x_i to y_j. This graph is called a fuzzy graph.

Example 4.4. When fuzzy relation R on X = { a,b,c } is

R=0.2/(a,a)+1/(a,b)+0.4/(a,c)+0.6/(b,b)+0.3/(b,c)+1/(c,b)+0.8/(c,c) (4.4)

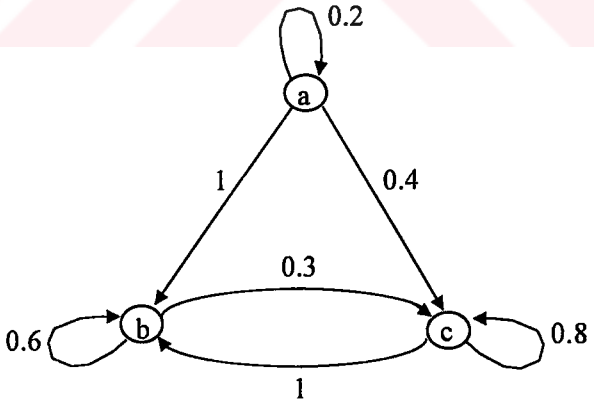
the fuzzy matrix and fuzzy graph for R are as shown in Fig. 4.2.

$$R = \begin{bmatrix} \mu_R(x_1,y_1) & \mu_R(x_1,y_2) & .... & \mu_R(x_1,y_n) \\ \mu_R(x_2,y_1) & \mu_R(x_2,y_2) & .... & \mu_R(x_2,y_n) \\ & . & & \\ & . & & \\ \mu_R(x_m,y_1) & \mu_R(x_m,y_2) & .... & \mu_{R0}(x_m,y_n) \end{bmatrix}$$

Fig. 4.1. Fuzzy Matrix for Fuzzy Relation R

	a	*b*	*c*
a	0.2	1	0.4
b	0	0.6	0.3
c	0	1	0.8

(a) Fuzzy matrix



(b) Fuzzy graph

Fig 4.2. Fuzzy Matrix and Graph For Equation (4.4)

4.2.3. Basic Properties Of Fuzzy Relations

Table 4.1. contains a list of the basic properties of operations with fuzzy relations, especially composition and converse fuzzy relation. Since the properties of unions ($\cup$), intersection ($\cap$) and complements ($\bar{}$) for fuzzy relations are the same as those for fuzzy sets, we will not repeat them here.

4.2.4 Fuzzy Relations And Fuzzy Reasoning

We will now try to give a somewhat detailed discussion of fuzzy reasoning (approximate reasoning) as an example of one application of the composition of fuzzy relations "o" which plays a major role in areas such as fuzzy control, fuzzy diagnosis, and fuzzy expert systems.

Table 4.1. Basic Properties Of Fuzzy Relations

For Composition (o)	For Inverse Relations (°)
(1) $R \circ I = I \circ R = R$	(11) $(R \cup S)^\circ = R^\circ \cup S^\circ$
(2) $R \circ O = O \circ R = O$	$(R \cap S)^\circ = R^\circ \cap S^\circ$
(3) In general, $R \circ S \neq S \circ R$	$(RoS)^\circ = S^\circ \circ R^\circ$
(4) $(RoS) \circ T = R \circ (S \circ T)$	(12) $(R^\circ)^\circ = R$, $\bar{R}^\circ = \bar{R}^\circ$
(5) $R^{m+1} = R^m \circ R$, $R^\circ = I$	(13) $R \subseteq S \rightarrow R^\circ \subseteq S^\circ$, $\bar{R} \supseteq \bar{S}$
(6) $R^m \circ R^n = R^{m+n}$	
(7) $(R^m)^n = R^{mn}$	
(8) $R \circ (S \cup T) = (R \circ S) \cup (R \circ T)$	
(9) $R \circ (S \cap T) = (R \circ S) \cap (R \circ T)$	
(10) $R \subseteq T \rightarrow R \circ S, R \circ T$	

First let us take a look at what could be considered a simple example of fuzzy reasoning:

premise 1	If x is A then y is B	
premise 2	x is A′	(4.8)
conclusion	y is B′,	

where x and y are objects and A, A′ and B, B′ are fuzzy sets in the universe sets U,U and V, V, respectively.

In this fuzzy reasoning form, A and A′ and B and B′ are not necessarily equal, but if A′=A and B′=B, the reasoning form in (4.8) is reduced to what is called the modus ponens.

The following is an example of a fuzzy reasoning form along the lines of (4.8).

If a tomato is red then the tomato is ripe.
This tomato is very red.

∴ This tomato is very ripe.

Since the fuzzy condition in (4.8), "If x is A then y is B" (for simplicity expressed by A→B), expresses some kind of relation between A and B, the following is given by Zadeh as a method for translating the fuzzy condition A→B into a fuzzy relation following the implication rule a→b=1∧ (1-a+b):

$$R_a = A \rightarrow B$$
$$=(\bar{A}xV)\oplus(UxB)$$
$$\int_{UxV} 1 \wedge \left(1 - \mu_A(u) + \mu_B(v)\right)/(u,v). \qquad (4.9)$$

The ⊕ means bounded sum, which is defined as a⊕b=1∧(a+b).

Conclusion B′ in (4.8) can be obtained by taking the composition of fuzzy set A′ and fuzzy condition A→B (the compositional rule of inference), and in general is given as follows:

$$B' = A' o (A \rightarrow B)$$

$$\mu_{B'} = \bigvee_u \{\mu_{A'}(u) \wedge \mu_{A \rightarrow B}(u,v)\}. \qquad (4.10)$$

For example, with Zadeh's method, R_a is given by

$$B' = A' o R_a = A' o[(\bar{A} x V) \oplus (U x B)]$$

and $$B' = A o R_a = \int_V \frac{1+\mu_B(v)}{2} \Big/ v \quad at \quad A' = A.$$

However, this case does not satisfy the following modus ponens, which is an appropriate requirement for fuzzy reasoning:

If x is A then y is B

x is A (4.11)

---

y is B

What, then, would happen with Mamdani's method, which is often used in fuzzy control ?

He employs the direct product of A and B for A→B. In other words, we get

$$A \rightarrow B = A \times B$$

$$= \int_{UxV} \mu_A(u) \wedge \mu_B(v)/(u,v). \qquad (4.12)$$

Thus, when A′ = A, we get the following:

$$B' = A o (A x B)$$

$$\mu_{B'}(v) = \bigvee_u \{\mu_A(u) \wedge (\mu_A(u) \wedge \mu_B(v))\}$$

$$= \bigvee_u \{\mu_A(u) \wedge \mu_B(v)\}$$

$$= \bigvee_u \mu_A(u) \wedge \mu_B(v).$$

If we assume that A is normal (the maximal grade of A is 1), we get the following:

$$= 1 \wedge \mu_B(V) = \mu_B(V),$$

and we can see that B′=B. In other words, we can see that it satisfies the modus ponens in (4.11).

Table 4.2. is a list of implication rules frequently used in fuzzy reasoning. R_a and R_m are Zadeh's method, and R_c is Madani's method.

Table 4.2. Various Implication Formulae

R_a	$a \rightarrow b = 1 \wedge (1-a+b)$
R_m	$a \rightarrow b = (a \wedge b) \vee (1-a)$
R_c	$a \rightarrow b = a \wedge b$
R_s	$a \rightarrow b = \begin{cases} 1; & a \le b \\ 0; & a > b \end{cases}$
R_g	$a \rightarrow b = \begin{cases} 1; & a \le b \\ b; & a > b \end{cases}$
R_b	$a \rightarrow b = (1-a) \vee b$
R_Δ	$a \rightarrow b = \begin{cases} 1; & a \le b \\ \frac{b}{a}; & a < b \end{cases}$

Thus far we have used the max - min composition in the compositional rule of inference B′= A′o(A→B), but besides the composition in (4.10) there are others (in general the max - *composition) we can consider. Namely, we have

B′= A′o(A→B)

$$\mu_{B'}(v) = \bigvee_u \{\mu_{A'}(u) * \mu_{A \to B}(u, v)\}.$$

For example, let the compositions using Θ (bounded - product) and Δ (drastic product) be used for *, that is,

$$x \Theta y = 0 \vee (x+y-1)$$

$$x \text{ Á } y = \begin{cases} x; & y = 1 \\ y & x = 1 \\ 0 & x, y < 1, \end{cases}$$

and let the max - Θ and max - Á composition be "□" and "▲," respectively. If we use these for R_a in Equation (4.9) when A′=A, the conclusion B′ becomes B:

$$B' = A \square R_a = A \blacktriangle R_a = B.$$

In other words, it satisfies the modus ponens. Using new compositions like this, we see that we can obtain inference results that match intuition even from the R_a method, which was unsatisfaction when based on the max-min composition. This can also be said for other methods based on the implication rules, as in Table 4.2. This is shown in Table 4.3. In this table, besides A′=A, conclusions from

$$A' = \text{very } A = \int_U \mu_A(u)^2 / u$$

$$A' = \text{more or les } A = \int_U \sqrt{\mu_A(u)} / u$$

$$A' = \text{not } A = \int_U 1 - \mu_A(u) / u$$

are given. It should be clear that conclusions that match intuition can be obtained based on new composition methods other than A′ = A for A′.

The following form of fuzzy reasoning is the one most commonly used in fuzzy control, here the conditional parts of the fuzzy conditions are tied together with the "and" of two fuzzy propositions:

premise 1	If x is A and y is B then z is C
premise 2	x is A′ and y is B′
conclusion	z is C′

Here, A, A′ ⊆ U ; B, B′ ⊆ V; C, C′ ⊆ W are fuzzy sets.

The fuzzy proposition connected by "and," "x is A and y is B," is shown by A∩B for convenience.

$$X \text{ is A and } y \text{ is B} \equiv A \cap B,$$

That is, A∩B expresses the intersection of A x V and U x B and equals the direct product A x B.

$$A\cap B=(A\times V)\cap(U\times B)$$
$$=A\times B \qquad (4.13)$$
$$=\int_{UxV}\mu_A(u)\wedge\mu_B(v)/(u,v).$$

The fuzzy condition "If x is A and y is B then z is C" (for simplicity, A∩B→C) in premise 1 is replaced by fuzzy relation R(A,B;C) in direct Product U x V x W. For example, noting that for the Mamdani's method R_c in Table 4.2. the extension of the implication $a\rightarrow b=(a\wedge b)\wedge b^2$ is

$$(a\wedge b)\rightarrow c=(a\wedge b)\wedge c,$$

and the fuzzy condition A∩B → C is replaced by the ternary fuzzy relation

$$R_a(A,B;C)=\int_{UxVxW}\mu_A(u)\wedge\mu_B(v)\wedge\mu_C(w)/(u,v;w). \quad (4.14)$$

In the same way, we get

$$R_a(A,B;C)=(\overline{A\cap B}xW)\oplus(UxVxC)$$
$$=\int_{UxVxW}1\wedge\left[1-(\mu_A(u)\wedge\mu_B(v))+\mu_C(w)\right]/(u,v,w)$$

for the R_a method in Table 4.2. Those that follow are found in the same way. The conclusion of equation (4.12) is found in the following way using the compositional rule of inference, where "o" is the max - min composition.

$$C'=(A'\cap B')oR(A,B;C)$$
$$\mu_{C'}(W)=\bigvee_{u,v}\{(\mu_{A'}(u)\wedge\mu_{B'}(v))\wedge\mu_{R(A,B;C)}(u,v,w)\}.$$

For example, we get

$$C'=(A'\cap B')oR_C(A,B;C)$$
$$\mu_{C'}(W)=\bigvee_{u,v}\{(\mu_{A'}(u)\wedge\mu_{B'}(v))\wedge\left[\mu_A(u)\wedge\mu_B(v)\wedge\mu_C(w)\right]\}$$
$$=\bigvee_u\{(\mu_{A'}(u)\wedge\mu_A(u)\wedge\mu_C(w)\wedge\bigvee_v\left[\mu_{B'}(v)\wedge\mu_B(v)\wedge\mu_C(w)\right]\}$$
$$=\bigvee_u\{\mu_{A'}(u)\wedge\mu_A(u)\wedge\mu_C(w)\wedge\mu_{B'oR_C(B;C)}(w)\}$$
$$=\mu_{A'oR_C(A;C)}(w)\wedge\mu_{B'oR_C(B;C)}(w)$$

for R_C. In other words, we get

$$C'=(A'\cap B')oR_C(A,B;C)$$
$$=\left[A'oR_C(A,C)\right]\cap\left[BoR_C(A,C)\right]$$

Table 4.3. Results of Reasoning

Composition	Max - Min Composition				Max - Composition				Max - Composition			
a→b	A	Very A	More or Less A	Not A	A	Very A	More or Less A	Not A	A	Very A	More or Less A	Not A
1∧(1-a+b)	$\frac{1+\mu_B}{2}$	(*1)	(*3)	1	μ_B	μ_B	(*5)	1	μ_B	μ_B	$\sqrt{\mu_B}$	1
(a∧b)∨(1-a)	0.5∨μ_B	(*2)	(*4)	1	μ_B	μ_B	$\frac{1}{4}\vee\mu_B$	1	μ_B	μ_B	μ_B	1
a∧b	μ_B	μ_B	μ_B	0.5∧μ_B	μ_B	μ_B^2	μ_B	0	μ_B	μ_B^2	μ_B	0
$\begin{cases}1; & a\le b\\ 0; & a>b\end{cases}$	μ_B	μ_B^2	$\sqrt{\mu_B}$	1	μ_B	μ_B	$\sqrt{\mu_B}$	1	μ_B	μ_B	$\sqrt{\mu_B}$	1
$\begin{cases}1; & a\le b\\ b & a>b\end{cases}$	μ_B	μ_B	$\sqrt{\mu_B}$	1	μ_B	μ_B	$\sqrt{\mu_B}$	1	μ_B	μ_B	$\sqrt{\mu_B}$	1
(1-a) ∨b	0.5∨μ_B	(*2)	(*4)	1	μ_B	μ_B	$\frac{1}{4}\vee\mu_B$	1	μ_B	μ_B	μ_B	1
$\begin{cases}1, & a\le b\\ \frac{b}{a}; & a>b\end{cases}$	$\sqrt{\mu_B}$	$\mu_B^{2/3}$	$\mu_B^{1/3}$	1	μ_B	μ_B	$\sqrt{\mu_B}$	1	μ_B	μ_B	$\sqrt{\mu_B}$	1

(Note) (*1) $\frac{3+2\mu_B-\sqrt{5+4\mu_B}}{2}$ (*2) $\frac{3-\sqrt{5}}{2}\vee\mu_B$ (*3) $\frac{\sqrt{5+4\mu_B}-1}{2}$ (*4) $\frac{\sqrt{5}-1}{2}\vee\mu_B$ (*5) $\begin{cases}\mu_B+\frac{1}{4}; & \mu_B\le\frac{1}{4}\\ \sqrt{\mu_B}; & \mu_B\ge\frac{1}{4}\end{cases}$

In this instance, $R_C(A;C)$ is fuzzy relation for condition $A \rightarrow C$ obtained by means of method R_C. In other words, with Mamdani's R_C, the conclusion of (4.12) is expressed as the intersection of the individual conclusion. Thus, we have

$$(A' \cap B') o (A, B \rightarrow C) = [A' o (A \rightarrow C)] \cap [B' o (B \rightarrow C)].$$

Example 4.6 When $A' = A$ and $B' = B$, the conclusion obtained by means of R_C, is

$$(A \cap B) o (A, B \rightarrow C) = [A o (A \rightarrow C)] \cap [B o (B \rightarrow C)].$$
$$= C \cap C,$$

and the following appropriate reasoning form is satisfied:

If x is A and y is B then z is C
x is A and y is B (4.15)

---

z is C

R_a, R_s, R_g, R_b and R_Δ are among the implication rules in Table 4.2. that union ($\cup$) of the individual conclusions:

$$(A' \cap B') o (A \cap B \rightarrow C) = [A' o (A \rightarrow C)] \cup [B' o (B \rightarrow C)].$$

Example 4.7 When $A' = A$ and $B' = B$, the conclusion of R_a is

$$(A \cap B) o (A, B \rightarrow C) = \int_W \frac{1 + \mu_C(w)}{2} \Big/ w \cup \int_W \frac{1 + \mu_C(w)}{2} \Big/ w$$
$$= \int_W \frac{1 + \mu_C(w)}{2} \Big/ w (\neq C),$$

and we can see that it does not satisfy Equation (4.15).(Note that that we can see that other compositions, such as "□" and "▲" satisfy (4.15).),

Finally, let us take a look at a slightly more complicated fuzzy reasoning:

premise 1	If x is A_1 and y is B_1 then z is C_1 else	
premise 1	If x is A_2 and y is B_1 then z is C_2 else	
	.	
	.	(4.16)
premise n	If x is A_n and y is B_n then z is C_n. Else	
premise n+1	x is A′ and y is B′	
conclusion	z is C′	

In this case, the conclusion C' using Mamdani's R_C is given by

$$C'=(A'\cap B')o[(A_1\cap B_1\rightarrow C_1)\cup(A_2\cap B_2\rightarrow C_2)\cup\ldots)\cup(A_n\cap B_n\rightarrow C_n)$$

$$=[A'o(A_1\rightarrow C_1)\cap B'o(B_1\rightarrow C_1)]\cup\ldots\cup[A'o(A_n\rightarrow C_n)\cap B'o(B_n\rightarrow C_n)],$$

where "else" is interpreted as the intersection for R_a, R_s, R_g, R_b and R_Δ, which satisfy $(a\wedge b)\rightarrow c=(a\rightarrow c)\vee(b\rightarrow c)$, and conclusion C' is given by

$$C'=(A'\cap B')o[(A_1\cap B_1\rightarrow C_1)\cup(A_2\cap B_2\rightarrow C_2)\cup\ldots)\cup(A_n\cap B_n\rightarrow C_n)$$

$$\subseteq[A'o(A_1\rightarrow C_1)\cup B'o(B_1\rightarrow C_1)]\cap\ldots\cap[A'o(A_n\rightarrow C_n)\cup B'o(B_n\rightarrow C_n)].$$

These types of fuzzy reasoning are frequently used in fuzzy control and fuzzy diagnosis, but, especially fuzzy control, the A', B' in (4.16)(x is A' and y is B' of premise n+1) is generally a fixed value u_o, v_o rather than a fuzzy set (for example u_o = error, v_o = chance in error).

4.2.5. Fuzzy Relational Equations

Let us now discuss fuzzy relational, which play an important role in areas such as system analysis, the planning of fuzzy controllers, and the four fundamental fuzzy operations.

As in the previous section, if we let A be a fuzzy set in set X and R be the fuzzy relation in X x Y, the composition of A and R, AoR, is defined as

$$\mu_{A\circ R}(y)=\bigvee_x\{\mu_R(x,y)\} \qquad (4.17)$$

and is a fuzzy set in Y. We call this B. In other words,

$$A\circ R=B \qquad (4.18)$$

Example 4.8 If we express fuzyy set A and fuzzy relation R in terms if vectors and matrices and let them be as shown below, the composition of A and R comes out as follows:

$$A \circ R = \begin{matrix} & x_1 & x_2 & x_3 \end{matrix} \begin{bmatrix} 0.3 & 0.7 & 1 \end{bmatrix} \circ \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix} \begin{matrix} y_1 & y_2 & y_3 \end{matrix} \begin{bmatrix} 0.5 & 0.7 & 0.8 \\ 0.6 & 0.6 & 0.4 \\ 1 & 0.5 & 0.3 \end{bmatrix}$$

$$= \begin{matrix} y_1 & y_2 & y_3 \end{matrix} \begin{bmatrix} 1 & 0.6 & 0.4 \end{bmatrix} = B.$$

If we view A as the fuzzy input, B as the fuzzy output, and R as the fuzzy system, we can think of (4.18) as expressing a fuzzy system with fuzzy input and fuzzy output. In other words, this shows that fuzzy system R gives a fuzzy output of B given a fuzzy input of A. A diagrammatic version would given us something like Fig. 4.3.

If the fuzzy input A and the fuzzy relation R are given, the fuzzy output B can be found simply by taking the composition of A and R. On the other hand, let us take a look at the following problems:

(1) Given A and B, finding R.

(2) Given B and R, finding A.

(1)Corresponds to finding the fuzzy system when the fuzzy input and fuzzy output are known, and (2) corresponds to finding the fuzzy input when the fuzzy output and fuzzy system are known.

When we are dealing with these types of problems, (4.18) can be though of as expressing one type of equation and from this,

$$A \circ R = B \tag{4.19}$$

is called fuzzy relational equation.

In preparation, let us introduce the following operation. For any a, b, $\in$ [0,1], let

$$a \alpha b = \begin{cases} 1; & a \le b \\ b; & a > b \end{cases} \tag{4.20}$$

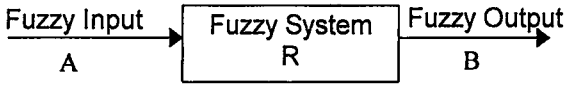


Fig. 4.3. Fuzzy System with Fuzzy Input and Output

When A and B are fuzzy sets in X and Y respectively, the fuzzy relation AαB in X x Y is given by the following using the α operation:

$$A\Phi B \leftrightarrow \mu_{A\Phi B}(x,y) = \mu_A(x)\alpha\mu_B(y) \tag{4.21}$$

We thus obtain the following properties:

$$A \circ (A\ \Phi\ B) \subseteq B$$

$$R \subseteq A\ \Phi (A \circ R)$$

Property 4.1. Given fuzzy sets A and B, the largest R that satisfies fuzzy relational equation AoR=B is the $\hat{R}$ below:

$$\hat{R} = A\ \Phi\ B \tag{4.22}$$

Example 4.9 If we find AΦB for fuzzy sets $A = \begin{matrix} x_1 & x_2 & x_3 \end{matrix}\begin{bmatrix} 0.3 & 0.7 & 1 \end{bmatrix}$ and $B = \begin{matrix} y_1 & y_2 & y_3 \end{matrix}\begin{bmatrix} 1 & 0.6 & 0.4 \end{bmatrix}$ from Example 4.8, we get

$$\hat{R} = \begin{matrix} \\ x_1 \\ x_2 \\ x_3 \end{matrix}\begin{matrix} \begin{matrix} y_1 & y_2 & y_3 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0.6 & 0.4 \\ 1 & 0.6 & 0.4 \end{bmatrix} \end{matrix}.$$

This clearly satisfies A o $\hat{R}$ = B, we also have direct product A x B ($\leftrightarrow \mu_A(x) \wedge \mu_B(y)$. The above example comes out as follows, and we have A x B $\subseteq \hat{R}$:

$$AxB = \begin{matrix} \\ x_1 \\ x_2 \\ x_3 \end{matrix}\begin{matrix} \begin{matrix} y_1 & y_2 & y_3 \end{matrix} \\ \begin{bmatrix} 0.3 & 0.3 & 0.3 \\ 0.7 & 0.6 & 0.4 \\ 1 & 0.6 & 0.4 \end{bmatrix} \end{matrix}.$$

Property 4.2. Given fuzzy relation R and fuzzy set B the largest fuzzy set A that satisfies AoR=B is $\hat{A} = R\ \Phi\ B$, (4.23)

where we have

$$\mu_{R \Phi B}(x) = \bigwedge_{y} \{\mu_R(x,y) \alpha \mu_B(y)\},$$

which is guaranteed because of the following:

$$(R \Phi B) \circ R \subseteq B, \qquad A \subseteq R \Phi (A \circ R),$$

Example 4.10. Using the fuzzy relation R and fuzzy set b from Example 4.8, we get

$$\hat{A} = R \Phi B = \begin{matrix} \\ x_1 \\ x_2 \\ x_3 \end{matrix} \begin{matrix} \begin{matrix} y_1 & y_2 & y_3 \end{matrix} \\ \begin{bmatrix} 0.5 & 0.7 & 0.8 \\ 0.6 & 0.6 & 0.4 \\ 1 & 0.5 & 0.3 \end{bmatrix} \end{matrix} \Phi \begin{matrix} \\ \begin{bmatrix} 1 \\ 0.6 \\ 0.4 \end{bmatrix} \end{matrix} \begin{matrix} \\ y_1 \\ y_2 \\ y_3 \end{matrix}$$

$$= \begin{bmatrix} 1 \wedge 0.6 \wedge 0.4 \\ 1 \wedge 1 \wedge 1 \\ 1 \wedge 1 \wedge 1 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 1 \\ 1 \end{bmatrix} \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix}$$

for $\hat{A}$, and it is clear that we can confirm that $\hat{A} \circ R = B$, $A \subseteq \hat{A}$.

4.2.6 Various Types Of Fuzzy Relations

By adding restrictions to fuzzy relations, we can obtain various types of fuzzy relations. Examples of these are "similarity relations" and "fuzzy order relations." To place such limitations on fuzzy relations, we first need to take a look at the basic properties of those fuzzy relations.

Let R be a fuzzy relation in X x Y. For any x, y, z ∈ X, we have

(1) reflective:$\mu_R(x,x)=1$;

(2) irreflective: $\mu_R(x,x)=0$;

(3) symmetric: :$\mu_R(x,y)=\mu_R(y,x)$;

(4) antisymmetric: $\mu_R(x,y)>0$, $\mu_R(y,x)>0 \rightarrow x = y$,

in other words, $x \neq y$, :$\mu_R(x,y)>0 \rightarrow$:$\mu_R(y,x)=0$;

(5) transitive: $\vee_y \{\mu_R(x,y) \wedge \mu_R(y,z)\} \leq \mu_R(x,z)$.

If we express these rules in terms of fuzzy operations, we come up with the following:

(1)′ $R \supseteq I$ (reflexive)

(2)′ $R \subseteq I$ (irreflexive)

(3)′ $R = R^{\circ}$ (symmetrical)

$$\underset{(a)\ \text{Re flexive}}{\begin{bmatrix} 1 & 0.2 & 0 \\ 0.3 & 1 & 0.1 \\ 0.9 & 0.7 & 1 \end{bmatrix}} \quad \underset{(b)\ \text{Inreflexive}}{\begin{bmatrix} 0 & 0.5 & 0.4 \\ 0.3 & 0 & 0.2 \\ 1 & 0.9 & 0 \end{bmatrix}} \quad \underset{(c)\ \text{Symmetric}}{\begin{bmatrix} 0.3 & 0.8 & 0.5 \\ 0.8 & 1 & 0.2 \\ 0.5 & 0.2 & 0 \end{bmatrix}} \quad \underset{(d)\ \text{Antisymmetric}}{\begin{bmatrix} 0.3 & 0 & 0.5 \\ 0 & 0.6 & 0.7 \\ 0 & 0 & 0.1 \end{bmatrix}} \quad \underset{(e)\ \text{Transitive}}{\begin{bmatrix} 0.2 & 1 & 0.4 \\ 0 & 0.6 & 0.3 \\ 0 & 1 & 0.3 \end{bmatrix}}$$

Fig. 4.4 Various Fuzzy Relations

(4)′ $R \cap R^{c} \subseteq I$ (antisymmetric);

(5)′ $RoR \subseteq I$ (transitive).

Examples of fuzzy relations that satisfy these rules are shown in Fig 4.4.

The transitivity in Fig.4.4 (e) clearly as follows:

$$R = \begin{bmatrix} 0.2 & 1 & 0.4 \\ 0 & 0.6 & 0.3 \\ 0 & 1 & 0.3 \end{bmatrix} \supseteq \begin{bmatrix} 0.2 & 0.6 & 0.3 \\ 0 & 0.6 & 0.3 \\ 0 & 0.6 & 0.3 \end{bmatrix} = RoR.$$

Examples 4.11. The fuzzy relation "x and y look alike" is reflexive and symmetrical; "x and y do not look alike" is irreflexive and symmetrical.

Examples 4.12. The fuzzy relation "x<<y" is irreflexive, antisymmetrical, and transitive.

Fuzzy relations with the above limitations placed on them have the following properties.

Property 4.3. In the case of reflexivity:

(1) If R is reflexive and S is any fuzzy relation, $RoS \supseteq S$ and $SoR \supseteq S$.

(2) If R is reflexive, $R \subseteq RoR$.

(3) If R and S are reflexive, $R \cup S$, $R \cap S$, and RoS are also reflexive.

Property 4.4. In the case of reflexivity:

(1) If R and S are symmetrical, $R \cup S$, $R \cap S$, and R^m are also symmetrical.*

(2) If R and S are symmetrical and RoS = SoR, RoS is also symmetrical.

Property 4.5. In the case of transitivity:

(1) If R and S are transitive, $R \cap S$ is also transitive. $R \cup S$ is not always transitive.

(2) If R and S are transitive and RoS=SoR, RoS is transitive.

(3) If R is both symmetrical and transitive, $\mu_R(x,y) \leq \mu_R(x,x)$.

(4) If R is both reflexive and transitive, RoR = R.

Note: "* $R^m = \underbrace{RoRo...oR}_{m}$"

Property 4.6. In the case of transitive closure:

(1) For any fuzzy relation R or X, when X has n number of elements the transitive closure of R, $\hat{R}$, is transitive:

$$\hat{R} = R \cup R^2 \cup R^n.$$

(2) When R is reflexive ($R \supseteq I$),

$$I \subseteq R \subseteq R^2 \subseteq ... \subseteq R^{n-1} \subseteq R^n$$

and therefore we have

$$\hat{R} = R^{n-1}.$$

(3) When R is transitive, we know $R \supseteq R^2 \supseteq R^3$ by definition, and $\hat{R}$ satisfies $\hat{R} = R$.

(4) When R is reflexive and transitive, we get $R=R^2=R^3=...$, and $\hat{R}$ satisfies $\hat{R} = R$.

(5) For any fuzzy relation R,

$$(I \cup R)^m = I \cup R \cup R^2 \cup \ldots \cup R^m$$

$$\bigcup_{i=p}^{q} R^i = R^p o (I \cup R)^{q-p}.$$

If we let p=1 and q=n, $\hat{R}$ can be expressed by

$$\hat{R} = R \cup R^2 \cup \ldots \cup R^n = Ro(I \cup R)^{n-1}.$$

4.3. Fuzzy Control

For most engineering systems, there are two important information sources: sensors which provide numerical measurements of variables, and human experts who provide linguistic instructions and descriptions about the system. We call the information from sensors numerical information and the information from human experts linguistic information. Numerical information is represented by numbers, for example 0.25,1.44, and so on, whereas linguistic information is represented by words like small, large, very large, and so forth. Conventional engineering approaches can only make use of numerical information and have difficulty incorporating linguistic information. Because so much human knowledge is represented in linguistic terms, incorporating it into engineering systems in a systematic and efficient manner is very important.

Why is linguistic information usually represented in fuzzy terms? We think that there are at least three reasons, First, we usually find it more convenient and efficient to communicate our knowledge in fuzzy terms. This is understandable because if we insist on using only crisp terms, then we must first have precise

definitions of these crisp terms. This in turn may result in a chain of definitions-a very inefficient and inconvenient procedure which clearly does not happen in our everyday lives. Second, our knowledge about many problems is essentially fuzzy. For example, when we first learn a new theory, we often find that we understand something about the theory, for example, its motivation, basic ideas, advantages, disadvantages, and so on, but we are not sure about some of the details. Now if we are asked to introduce the theory to another person, then that person can only get a fuzzy picture of the theory. The interesting point is that although the picture is not clear, it may serve the purpose quite well-for example, knowing the motivation, basic ideas, advantages, and disadvantages may be sufficient for a higher-level manager. Third, many systems are too complex to describe in crisp terms. For example, our knowledge about a complex chemical process may only be represented in fuzzy terms, for example, "if the temperature is high, then the reaction is intense." The important point here is that although this kind of linguistic information is not precise, it provides important information about, the system and sometimes it may be the only information source. We should make use of this fuzzy information in a scientific way.

4.3.1. Classification Of Fuzzy Logic Systems

Fuzzy logic systems is a name for the systems which have a direct relationship with fuzzy concepts (like fuzzy sets, linguistic variables, and so on) and fuzzy logic. The most popular fuzzy logic systems in the literature may be classified into three types [5]:

- pure fuzzy logic systems,
- Takagi and Sugeno's fuzzy system,
- fuzzy logic systems with fuzzifier and defuzzifier,

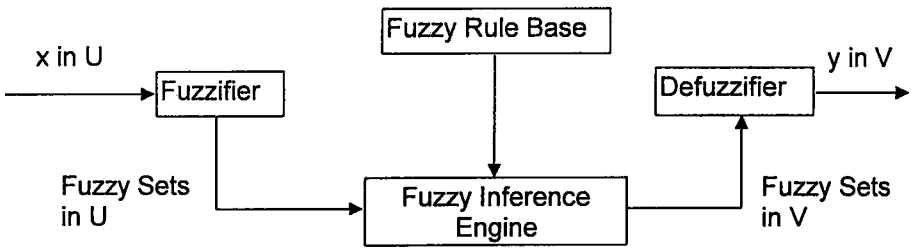


Fig. 4.5 Basic configuration of fuzzy logic system with fuzzifier and defuzzifier.

4.3.1.1. Pure Fuzzy Logic Systems

The basic configuration of a pure fuzzy logic system is shown in Fig. 4.6 where the fuzzy rule base consists of a collection of fuzzy IF-THEN rules, and the fuzzy inference engine uses these fuzzy IF-THEN rules to determine a mapping from fuzzy sets in the input universe of discourse $U \subset R^n$ to fuzzy sets in the output universe of discourse $V \subset R$ based on fuzzy logic principles. The fuzzy IF-THEN rules are of the following form:

$$R(l) : \text{IF } x_1 \text{ is } F^l_1 \text{ and } .... \text{ and } x_n \text{ is } F^l_n, \text{ THEN y is } G^l \qquad (4.24)$$

where F_i^l and G^l are fuzzy sets, $x = (x_1, ... ,x_n)^T \in U$ and $y \in V$ are input and output linguistic variables, respectively, and $l = 1, 2, . . . , M$. Practice has shown that these

fuzzy IF-THEN rules provide a convenient framework to incorporate human experts' knowledge. Each fuzzy IF-THEN rule of (4.24) defines fuzzy set $F^l_1 \times \cdots \times F^l_n \rightarrow G^l$ in the product space U x V. The most commonly used fuzzy, logic principle in the fuzzy inference engine is the so-called sup-star composition. Specifically, let A' be an arbitrary fuzzy set in U; that is, A' is the input to the pure fuzzy logic system of Fig. 4.6, then the output determined by each fuzzy IF-THEN rule of (4.24) is a fuzzy set A' o $R^{(1)}$ in V whose membership function is

$$\mu_{A' \circ R(1)}(y) = \sup_{x \in u} [\mu_{A'}(x) * \mu_{Fl1x...x Fln \rightarrow Gl}(x,y)] \qquad (4.25)$$

where the "*" operator is "min," "product," or others. We use μ_A to represent the membership function of fuzzy set A. The final output of the pure fuzzy logic system is a fuzzy set A' o ($R^{(1)}$,. . . , $R^{(M)}$) in V which is a combination of the M fuzzy sets of (4.25); that is,

$$\mu_{A' \circ (R(1)\cdots R(M))}(y) = \mu_{A'oR(1)}(y) +' \ldots +' \mu_{A'oR(M)}) \qquad (4.26)$$

where the " +' " operator is "max," "algebraic sum" (x+y = x + y - xy), or others. If there is feedback as shown by the dashed arrow line in Fig. 4.6, we have the so-called fuzzy dynamic systems, that is, pure fuzzy logic systems whose inputs depend on their outputs.

The pure fuzzy logic system constitutes the essential part of fuzzy logic systems. It is a general framework in which linguistic information from human

experts is quantified and fuzzy logic principles are used to make systematic use of linguistic information. A main disadvantage of a pure fuzzy logic system is that its inputs an outputs are fuzzy sets, whereas in most engineering systems the inputs and outputs of a system are real valued variables. To overcome this disadvantage, Takagi and Sugeno [18] proposed another fuzzy logic system whose inputs and outputs are real-valued variables.

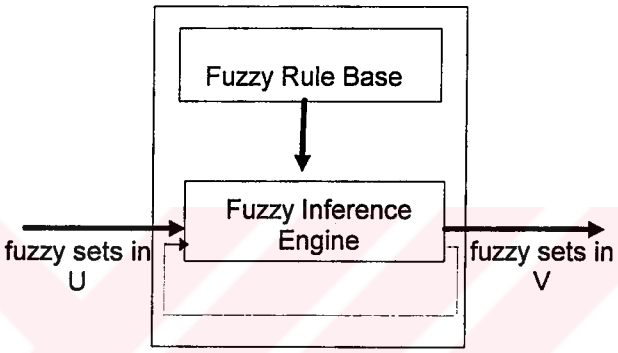


Fig. 4.6 Basic configuration of pure fuzzy logic system.

4.3.1.2. Takagi And Sugeno's Fuzzy System

Instead of considering the fuzzy IF-THEN rules in the form of (4.24), Takagi and Sugeno [15] proposed to use the following fuzzy IF-THEN rules:

$$L^{(l)} : \text{IF } x1 \text{ is } F^l_1 \text{ and } \ldots\ldots \text{and } x_n \text{ is } F^l_n \text{ THEN } y^l = c^l_0 + \ldots + c^l_n x_n \qquad (4.27)$$

where F_i^l are fuzzy sets, c_i are real-valued parameters, y^l is the system output due to rule $L^{(l)}$, and $l = 1, 2, \ldots, M$. That is, the considered rules whose IF part is fuzz but whose THEN part is crisp- the output is a linear combination of input variables.

For a real-valued input vector $x = (x_1, \ldots, x_n)^T$, the output y(x) of Takagi and Sugeno's fuzzy system is a weighted average of the y^l's:

$$y(y) = \frac{\sum_{l=1}^{M} w^l y^l}{\sum_{l=1}^{M} w^l} \tag{4.28}$$

where the weight w^l implies the overall truth value of the premise of rule $L^{(l)}$ for the input and is calculated as

$$w^l = \prod_{i=1}^{n} m_{f_i^l}(x_i) \tag{4.29}$$

The configuration of Takagi and Sugeno's fuzzy system is shown in Fig. 4.7. Takagi and Sugeno's fuzzy system has been successfully applied to many practical problems. The advantage of this fuzzy logic system is that it provides a compact system equation (4.28) and, therefore, parameter estimation and order determination methods can be developed to estimate the parameters c_i^l and the order M. A weak point of this fuzzy logic system is that the THEN part of the rule is not fuzzy; therefore it does not provide a natural framework to incorporate fuzzy rules from human experts. That is, modification of the pure fuzzy rule (4.24) (for example, change "y is G^l" to "y = c_0^l" where c_0^l is the center of μ_G) must be performed in order to incorporate it into the fuzzy logic system.

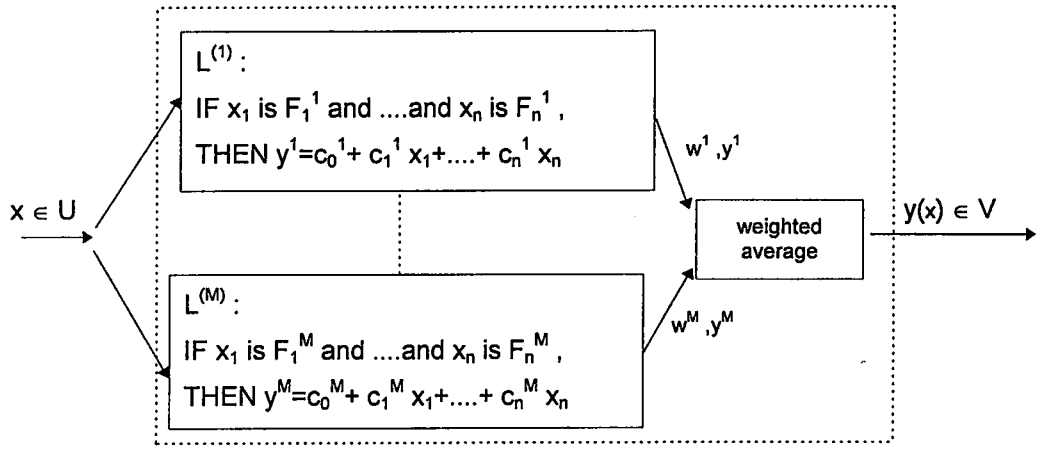


Fig. 4.7. Basic configuration of Takagi and Sugeno's fuzzy system

4.3.1.3. Fuzzy Logic Systems With Fuzzifier and Defuzziffer (Mamdani Type)

In order to use the pure fuzzy logic system shown in Fig. 4.6 in engineering systems where inputs and outputs are real-valued variables, the most straightforward way is to add a fuzzifier to the input and a defuzzifier to the output of the pure fuzzy logic system. The basic configuration of fuzzy logic systems with fuzzifier and defuzzifier is shown in Fig. 4.5 The fuzzifier maps crisp points in U to fuzzy sets in U, and the defuzzifier maps fuzzy sets in V to crisp points in V. The fuzzy rule base and fuzzy inference engine are the same as those in the pure fuzzy logic system. In the literature, this fuzzy logic system is often called the fuzzy logic controller since it has been mainly used as a controller. It was first proposed by Mamdani [23] and has been successfully applied to a variety of industrial processes and consumer products.

The fuzzy logic system with fuzzifier and defuzzifier has many attractive features. First, it is suitable for engineering systems because its inputs and outputs re real-valued variables. Second, it provides a natural framework to incorporate fuzzy IF-THEN rules from human experts. Third, there is much freedom in the choices of fuzzifier, fuzzy inference engine, and defuzzifier, so that we may obtain the most suitable fuzzy logic system for a particular problem. Finally, we can develop different training algorithms for this fuzzy logic system so that it provides an effective framework to integrate numerical and linguistic information.

Training algorithms for Fuzzy Logic systems [5].

- Back propagation
- Orthogonal least squares
- Table-lookup scheme
- Neighborhood clustering

4.3.2. Fuzzy Rule Base

A fuzzy rule base consists of a collection of fuzzy IF-THEN rules in the following form:

$$R(l) : \text{IF } x_1 \text{ is } F^l_1 \text{ and } \ldots\ldots \text{ and } x_n \text{ is } F^l_n \text{ THEN } y \text{ is } G^l \qquad (4.30)$$

where F^l_i and G^l are fuzzy sets in $U_i \subset R$ and $V \subset R$, respectively, and $x = (x_1, \ldots, x_n)^T \in U_1 \times \cdots \times U_n$ and $y \in V$ are linguistic variables. Let M be the number of

fuzzy IF-THEN rules in the form of (4.30) in the fuzzy rule base; that is, l=1, 2, . . , M in (4.30). The x and y are the input and output to the fuzzy logic system, respectively. Without loss of generality, we consider multi-input-single-output fuzzy logic systems, since a multi-output system can always be decomposed into a group of single-output systems.

The fuzzy rule base is the heart of the fuzzy logic system in the sense that all other three components are used to interpret these rules and make them usable for specific problems. Practice has shown that fuzzy IF-THEN rules in the form of (4.30) provide a very convenient framework for human experts to express their domain knowledge.

We may have the following basic questions concerning the fuzzy rule base:

- Are the rules in the form of (4.30) general enough to include other types of linguistic information?
- Where do these fuzzy IF-THEN rules come from?
- How are the membership functions for F_i^l and G^l determined?

We answer the first question by showing that the rules in the form of (4.30) include many other types of fuzzy rules as special cases.

Fact 4.1. The rules in the form of (4.30) include the following "incomplete-IF-part rule" as a special case:

IF x_1 is F^l_1 and $\cdot\cdot$ and x_m is F^l_m THEN y is G^l where $m < n$.

Proof. Clearly, the preceding incomplete-IF-part rule is equivalent to

IF x_1 is F^l_1 and $\cdots$ and x_m is F^l_m and x_{m+1} is I1 and. and .x_n is I1, THEN y is G^l

where I1 is a fuzzy set in R with $\mu_{I1}(x) = 1$ for all $x \in R$. The preceding rule is in the form of (4.30); thus, this fact is true. Q.E.D.

Fact 4.2. The rules in the form of (4.30) include the following "OR rule" as a special case:

IF x1 is F^l_1 and $\cdots$ and x_m is F^l_m or x_{m+1} is F^l_{m+1} and $\cdots$ and x_n is F^l_n THEN y is G^l

Proof. Based on intuitive meaning of the logic operator "or," the preceding OR rule is equivalent to the following two rules:

IF x_1 is F^l_1 and $\cdots$ and x_m is F^l_m, T HEN y is G^l

IF x_{m+1} is F^l_{m+1} and $\cdots$ and x_n is F^l_n, THEN y is G^l

From Fact 2.1 we have that the preceding two rules are special cases of (4.30); therefore, this fact is true. Q.E.D.

Fact 4.3. The rules in the form of (4.30) include the fuzzy statement:

$y \cdot \text{is } G^{l}$

as a special case.

Proof. Clearly, the preceding fuzzy statement is equivalent to

IF x_1 is I1 and $\cdots$ and x_n is I1, T HEN y is G^{l}

which is in the form of (4.30). Q.E.D.

Fact 4.4. The rules in the form of (4.30) include the following "gradual rule" as a special case:

The smaller the x, the bigger the y

Proof. Let S be a fuzzy set representing "smaller," for example, $\mu_s(x) = 1/(1 + \exp(5(.x- + 2)))$, and B be a fuzzy set representing "bigger," for example, $\mu_B(y) = 1/(1 + \exp(-5(y -2)))$, then the preceding "gradual rule" is equivalent t0

IF x is S. THEN $y \cdot$ is B

which is a special case of (4.30). Q.E.D.

Fact 4.5. The rules in the form of (4.30) include the following "unless rule" as a special case:

$$y \text{ is } G^l \text{ unless } x_1 \text{ is } F^l_1 \text{ and } \cdots \text{and } x_n \text{ is } F^l_n$$

Proof. Based on the intuitive meaning of unless, the preceding rule is equivalent to

$$\text{IF not } (x_1 \text{ is } F^l_1 \text{ and } \ldots \text{ and } .x_n \text{ is } F^l_n), \text{ THEN } y \text{ is } G^l$$

which, based on the De Morgan's Law, is equivalent to

$$\text{IF } x_1 \text{ is not } F^l_1 \text{ or } \cdots \text{or } x_n \text{ is not } F^l_n, \text{ THEN } y \text{ is } G^l$$

View not F^l_i as a single fuzzy set, then from Fact 2.2 the preceding rule is a special case of (4.30). Q.E.D.

Fact 4.6. The rules in the form of (4.30) include non-fuzzy rules (i.e., conventional production rules) as a special case.

Proof. If the membership functions of F^l_i and G^l can only take values 1 or 0, then the rules (4.30) become non-fuzzy rules. Q.E.D.

For the second question, there are two principal ways of obtaining fuzzy IF-THEN rules:

(1) asking human experts,

(2) using training algorithms based on measured data.

The first way is the most straightforward method for obtaining rules, but human experts in many cases may not provide a sufficient number of rules.

For the third question, there are also two principal ways depending upon where the rules come from. If the rules are provided by human experts, then the membership functions of F^l_i and G^l should be specified by the experts because these functions are an integrated part of the expert knowledge. For example, if an expert says that: "IF the error is large, THEN the control is large," he or she should tell what the large means by specifying the fuzzy membership functions for the large. If the rules are determined by numerical data, then the first task is to determine the functional forms for μ_{Fi} and μ_{Gl}. The most commonly used functional forms are Gaussian, triangular, and trapezoid. Gaussian functions have a smooth transition property, whereas triangular and trapezoid functions are simpler to compute. After the functional forms of μ_{Fi} and μ_{Gi} are fixed, the problem becomes how to determine the parameters in μ_{Fi} and μ_{Gi} based on measured data. There are number of parameter estimation methods for μ_{Fi} and μ_{Gi} .

4.3.3. Fuzzy Inference Engine

In a fuzzy inference engine, fuzzy logic principles are used to combine the fuzzy IF-THEN rules in the fuzzy rule base into a mapping from fuzzy sets in U =

$U_1 x \cdots x U_n$ to fuzzy sets in V. The first question for the fuzzy inference engine is: How do we interpret a fuzzy IF-THEN rule in the form of (4.30)?

4.3.3.1. Interpretations of A Fuzzy If-Then Rule

A fuzzy IF-THEN rule (4.30) is interpreted as a fuzzy implication $F^l_1 x \cdots x F^l_n \rightarrow G^l$ in U x V. Let a fuzzy set A' in U be the input to the fuzzy inference engine; then each fuzzy IF-THEN rule (4.30) determines a fuzzy set B^l in V using the sup-star composition (4.31). That is,

$$\mu_{Bl}(y) = \sup_{x \in u} [\mu_{Fl1 x \ldots x Fln \rightarrow Gl}(x, y) * \mu_{A'}(x)] \quad (4.31)$$

In the following, we show some commonly used interpretations for the fuzzy IF-THEN rule. For simplicity, we denote $F^l_i x \cdots x F^l_n = A$ and $G^l = B$, and the rule (4.30) is therefore denoted by $A \rightarrow B$.

- Mini-operation rule of fuzzy implication:

$$\mu_{A \rightarrow B}(x,y) = \min\{\mu_A(x), \mu_B(y)\} \quad (4.32)$$

- Product-operation rule of fuzzy implication:

$$\mu_{A \rightarrow B}(x,y) = \mu_A(x)\, \mu_B(y) \quad (4.33)$$

- Arithmetic rule of fuzzy implication:

$$\mu_{A \rightarrow B}(x,y) = \min\{1, 1-\mu_A(x) + \mu_B(y)\} \quad (4.34)$$

- Maxmin rule of fuzzy implication:

$$\mu_{A\to B}(x,y) = \max\{\min[\mu_A(x), \mu_B(y)], 1-\mu_A(x)\} \quad (4.35)$$

- Boolean rule of fuzzy implication:

$$\mu_{A\to B}(x,y) = \max\{1-\mu_A(x), \mu_B(y)\} \quad (4.36)$$

- Goguen's rule of fuzzy implication:

$$\mu_{A\to B}(x,y) = \begin{cases} 1 & \mu_A(x) \le \mu_B(y) \\ \mu_B(y)/\mu_A(x) & \mu_A(x) > \mu_B(y) \end{cases} \quad (4.37)$$

In (4.32)-(4.37), $\mu_A(x) = \mu_{Fi\ x..\ x\ Fn}(x)$ is defined either according to the mini-operation rule:

$$\mu_{Fi\ x..\ x\ Fn}(x) = \min\{\mu_{Fi}(x_1),...., \mu_{Fn}(x_n)\} \quad (4.38)$$

or according to the product-operation rule:

$$\mu_{Fi\ x..\ x\ Fn}(x) = \mu_{Fi}(x_1)..... \mu_{Fn}(x_n) \quad (4.39)$$

We note that the mini-operation rule follows from the fuzzy conjunction (4.30) by using the fuzzy intersection operator for *; the product-operation rule follows from the fuzzy conjunction by using the algebraic product for *. The arithmetic rule follows from the material implication (4.32) by using the bounded sum for $+'$ ($\mu_{A\to B}(x,y) = \mu_{A+'B} = \min\{1,1-\mu_A+\mu_B\}$): the maxmin rule follows from the propositional calculus (4.33) by using the fuzzy intersection for * and fuzzy union for $+'$ ($\mu_{A\to B}(x,y) = \mu_{A+'(A*B)} = \max\{1-\mu_A, \min[\mu_A, \mu_B]\}$)· Boolean rule

follows from the material implication by using the fuzzy union for +'($\mu_{A\to B}(x,y) = \mu_{A+'B}=\max\{1-\mu_A, \mu_B\}$) and Goguen's rule follows from the generalization of modus ponens by using the algebraic product for * ($\mu_{A\to B} = \sup\{c \in [0,1] \mid \mu_A c \leq \mu_B\} = 1$ if $\mu_A \leq \mu_B$ and $= \mu_B/\mu_A$ if $\mu_A > \mu_B$)

4.3.3.2. Overall Mapping of The Fuzzy Inference Engine

For an input A' (a fuzzy set in U) the output of the fuzzy inference engine can take two forms:

(1) M fuzzy sets B^l (l=1,2,....,M) in the form of (4.31), with each one determined by one fuzzy IF-THEN rule in the form of (4.30),

(2) one fuzzy set B' which is the union of the M fuzzy sets B^l. That is,

$$\mu_{B'}(y) = \mu_{B'}(y) +' \ldots . +' \mu_{BM}(y) \qquad (4.40)$$

that is, for a single input, the output of the fuzzy inference engine can either be a collection of M fuzzy sets or be the union of the M fuzzy sets. For these different types of outputs, we use different defuzzifiers to defuzzify them into a single point in the output space V.

4.3.4. Fuzzifler

The fuzzifier performs a mapping from a crisp point $x = (x_1, \ldots, x_n)^T \in U$ into a fuzzy set A' in U. There are (at least) two possible choices of this mapping:

- Singleton fuzzifier: A' is a fuzzy singleton with support x, that is, $\mu_{A'}(x') = 1$ for $x' = x$ and $\mu_{A'}(x') = 0$ for all other $x' \in U$ with $x' \neq x$.
- Non singleton fuzifier: $\mu_{A'}(x') = 1$ and $\mu_{A'}(x')$ decreases from 1 as x' moves away from x, for example, $\mu_{A'}(x') = \exp [-(x'-x)^T(x'-x)/ \sigma^2]$, where σ^2 is a parameter characterizing the shape of $\mu_{A'}(x')$.

It seems that only the singleton fuzzifier has been used. We think that the non singleton fuzzifier may be useful if the inputs are corrupted by noise. This issue needs further studying [4].

4.3.5. Defuzzifier

The defuzzifier performs a mapping from fuzzy sets in V to a crisp point $v \in V$. One can choose the best among the following methods [8];

- Center of area / gravity
- Modified center of area / gravity
- Center of sums
- Center of largest area

- First of maxima
- Middle of maxima
- Height

although there are others for special cases the equations for the most preferred ones are as follows ;

- First of maxima

$$y = \text{argsup}_{y \in V}(\mu_{A'}(x')) \tag{4.41}$$

where $\mu_{A'}(x')$ is given by (4.40).

- Center of gravity defuzziffier, defined as

$$y = \frac{\sum_{l=1}^{M} \bar{y}^l (m_{B'}(\bar{y}^l))}{\sum_{l=1}^{M} (m_{b'}(\bar{y}^l))} \tag{4.42}$$

where $\bar{y}^l$ is the center of the fuzzy set G^l, that is, the point in V at which μ_{Gl} (y) achieves its maximum value, and μ_{Bl} (y) is given by (4.31).

- Modified center of gravity defuzzifier, defined as

$$y = \frac{\sum_{l=1}^{M} \bar{y}^l (m_{B'}(\bar{y}^l)/\delta^l)}{\sum_{l=1}^{M} (m_{b'}(\bar{y}^l)/\delta^l)} \tag{4.43}$$

where δ^l is a parameter characterizing the shape of μ_{GI} (y) such that the narrower the shape of μ_{GI} (y), the smaller is δ^l for example, if μ_{GI} (y) $= \exp[-(\frac{y-\bar{y}^l}{\delta^l})^2]$, then δ^l is such a parameter.

The modified center average defuzzifier is justified as follows. Common sense indicates that the sharper the shape of μ_{GI} (y), the stronger is our belief that the output y should be nearer to the center of G^l (according to the rule $R^{(l)}$ of (4.30)). The standard center average defuzzifier, (4.42), is a weighted average of the $\bar{y}^l$'s, and the weights $\mu_{BI}(\bar{y}^l)$ determined by (4.31) do not take the shape of μ_{GI} (y) into consideration. This is clearly not satisfactory based on our common sense. An obvious improvement is the modified center average defuzzifier (4.43).

4.3.6. Membership Function Selection

In the membership function choice one has to solve a few problems: how to choose general parameters like a number of classes (membership functions) to describe all the values of linguistic variable on the Universe of discourse, the position of different membership functions on the Universe of discourse, the width of the membership functions, and concrete parameters like the shape of a particular membership function. The membership function is conducted on a base of the index which is called the whole overlap [1] which can be calculated according to the formula (4.44) (see Fig. 4.8):

$$WO = \frac{\int_x Min(\mu_1(x), \mu_2(x))}{\int_x Max(\mu_1(x), \mu_2(x))} \tag{4.44}$$

where $\mu_A(x)$ and $\mu_B(x)$ are two adjacent membership functions.

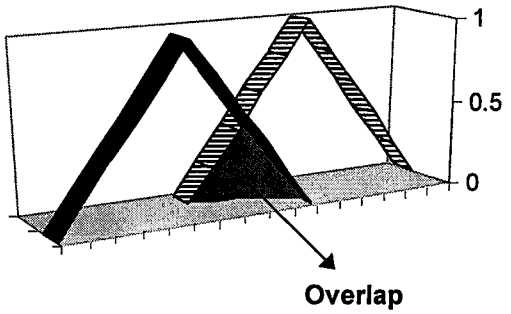


Fig. 4.8. Whole overlap parameter evaluation

Some guidelines about membership functions; [1]

1) Initial choice of the membership function choice should provide the whole overlap parameter of about 12-14 %,

2) Use of narrower membership functions results in the faster response (smaller response time)

3) Larger oscillations, overshoot and settling time appear when narrower membership functions are used,

4) Use of narrower membership functions produces the system with lower steady-state error but with a very narrow function the steady-state can possibly not be reached at all,

5) The chose of defuzzification method in many practical cases does not significantly influence the system performance characteristics,

6) The presence of small noise and disturbances generally keeps the statements made above valid.

Comparing an action of a fuzzy controller with the action of a conventional PID controller, one can state, that decreasing a membership function width (decreasing the WO ratio), one increases the differential part, and in the opposite case one emphasizes an integration performance of the controller. So we can see some analogy between a membership function choice and a PID-controller coefficients choice similar to that we established for a scaling factor choice.

4.3.7. Methodology

In the planning phase we had decided to realize two different fuzzy logic controller and test them. We implemented a Mamdani type controller "5.2.1." and a Tagaki & Sugeno type "5.2.2." controller models and test them for their step response timing and stability analysis by the help of a simulation procedure.

In the design step of the Mamdani style controller, we have covered the experience of the skilled operators by determining the input and output participation subjectively and fuzzily. In Tagaki & Sugeno style the real behavior of the system is imported into the algorithm by calculating the parameters of the fuzzy rule consequences.

4.3.7.1. Mamdani Type

Suppose we are given a set of desired input-output data pairs:

$(x_1^{(1)}, x_2^{(1)}; y^{(1)}), (x_1^{(2)}, x_2^{(2)}; y^{(2)}), \ldots\ldots$ (4.45)

Where X_1 and X_2 are inputs, and y is the output. The simple two-input one-output case is chosen in order to emphasize and to clarify the basic ideas of our approach; extensions to general multi-input multi-output cases are straightforward and will be discussed later in the section. The task here is to generate a set of fuzzy rules from the desired input-output pairs of (4.45), and use these fuzzy rules to determine a mapping

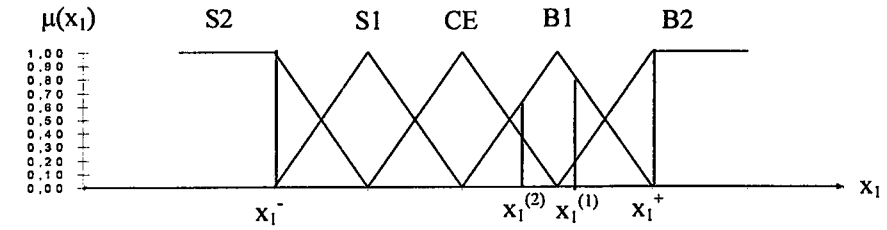
$$f : (x1, x2) \rightarrow y.$$

The approach consists of the following five steps: [2]

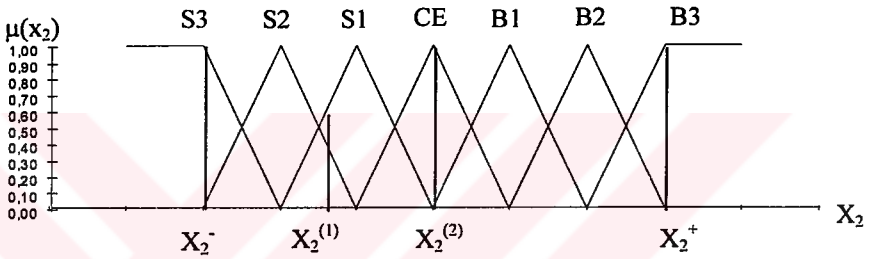
Step-1 : Divide the Input and Output Spaces into Fuzzy Regions

Assume that the domain intervals of x_1, x_2 and y are $[x_1^-, x_1^+]$,$[x_2^-, x_2^+]$ and $[y^-, y^+]$, respectively, were "domain interval" of a variable means that most probably this variable will lie in this interval (the values of a variable are allowed to lie outside its domain interval). Divide each domain interval into 2N+1 regions (N can be different for different variables, and the lengths of these regions can be equal or unequal), denoted by SN (small N), ..., S1 (small 1), CE (center), B1 (big 1), ..., BN (big N), and assign each region a fuzzy membership function. Fig. 4.9. shows an example where the domain interval of x_1 is divided into five regions (N=2), the domain region of x_2 is divided into seven regions (N=3), and the domain interval of y is divided into five regions (N=2). The shape of each membership function is triangular; one vertex lies at the center of the region and has membership value unity; the other two vertices lie at the centers of the two neighboring regions, respectively, and have membership values equal to zero. Of

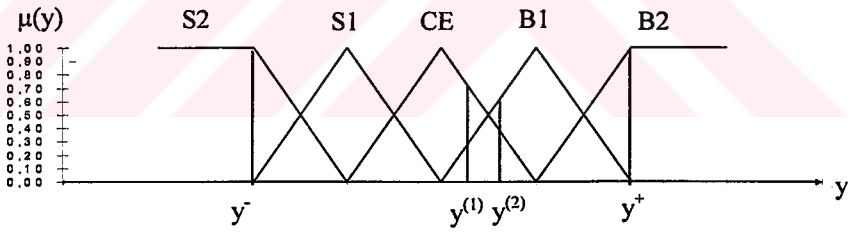
course, other divisions of the domain regions and other shapes of membership functions are possible.



(a)



(b)



(c)

Fig. 4.9 Divisions of the input and output spaces into fuzzy regions and the corresponding membership functions $\mu(x_1)$, $\mu(x_2)$, $\mu(y)$.

Step-2 : Generate Fuzzy Rules from Given Data Pairs.

First determine the degrees of given $x_1^{(i)}$,$x_2^{(i)}$ and $y^{(i)}$ in different regions. For example, $x_1^{(1)}$ in Fig. 4.9 has degree 0.8 in B1, degree 0.2 in B2, and zero

degrees in all other regions. Similarly, $x_2^{(2)}$ in Fig. 4.9 has degree 1 in CE, and zero degrees in all other regions.

Second, assign a given $x_1^{(i)}$,$x_2^{(i)}$ or $y^{(i)}$ to the region with maximum degree. For example, $x_1^{(1)}$ in Fig. 4.9 is considered to be B1, and $x_2^{(2)}$ in Fig.4.9. is considered to be CE.

Finally, obtain one rule from one pair of desired input-output data, e.g.,

$(x_1^{(1)},x_2^{(1)}; y^{(1)}) \Rightarrow [x_1^{(1)}$(0.8 in B1, max),$x_2^{(1)}$(0.7 in S1, max), $y^{(1)}$(0.9 in CE, max)]$\Rightarrow$ Rule 1:

IF x_1 is B1 and x_2 is S1, THEN y is CE ;

$(x_1^{(2)},x_2^{(2)}; y^{(2)}) \Rightarrow [x_1^{(2)}$(0.6 in B1, max),$x_2^{(2)}$(1 in CE, max), $y^{(2)}$(0.7 in B1, max)]$\Rightarrow$ Rule2:

IF x_1 is B1 and x_2 is CE, THEN y is B1 ;

The rules generated in this way are "and" rules, i.e., rules in which the conditions of the IF part must be match simultaneously in order for the result of the THEN part to occur. For the problems considered in this paper i.e., generating fuzzy rules from numerical data, only "and" rules are required since the antecedents are different components of a single input vector.

Step-3 : Assign a Degree to Each Rule :

Since there are usually lots of data pairs, and each data pair generates one rule, it is highly probable that there will be some conflicting rules, i.e., rules that have the same IF part but a different THEN part. One way to resolve this conflict is to assign a degree to each rule generated from data pairs, and accept only the rule

from a conflict group that has maximum degree. In this way not only is the conflict problem resolved, but also the number of rules is greatly reduced.

We use the following strategy to assign a degree to each rule: for the rule : "IF x_1 is A and x_2 is B THEN y is C, " the degree of this rule, denoted by D(Rule), is defined as

$$D(Rule) = \min (\mu_A(x_1), \mu_B(x_2), \mu_C(y)) \quad (4.46)$$

As examples Rule 1 has degree

$$D(Rule1) = \min (\mu_{B1}(x_1), \mu_{S1}(x_2), \mu_{CE}(y))$$

$$= \min (0.8, 0.7, 0.9) = 0.7 \quad (4.47)$$

(See Fig. 4.9) and Rule 2 has degree

$$D(Rule2) = \min (\mu_{B1}(x_1), \mu_{CE}(x_2), \mu_{B1}(y))$$

$$= \min (0.6, 1, 0.7) = 0.6 \quad (4.48)$$

In practice, we often have some a priori information about the data pairs. For example, if we let an expert check given data pairs, the experts may suggest the some are very useful and curricula, but others are very unlikely and may be caused just by measurement errors. We can therefore assign a degree to each data pair that represents our belief of its usefulness. In this sense the data pairs constitute a fuzzy set, i.e., the fuzzy set is defined as the useful measurements; a data pair belongs to this set to a degree assigned by a human expert.

Suppose the data pair $(x_1^{(l)},x_2^{(l)}: y^{(l)})$ has degree $m^{(l)}$, then we redefine the degree of Rule1 as

$$D(Rule1) = \min (\mu_{B1}(x_1), \mu_{S1}(x_2), \mu_{CE}(y), m^{(l)}) \quad (4.49)$$

i.e., the degree of a rule is defined as the minimum of the degrees of its components and the degree of the data pair that generates this rule. This is important in practical applications, because real numerical data have different reliabilities, e.g., some real data can be very bad ("wild data"). For good data we assign higher degrees, and for bad data we assign lower degrees. In this way, human experience about the data is used in a common base as other information. If one emphasizes objectivity and does not want a human to judge the numerical data, our strategy still works by setting all the degrees of the data pairs equal to unity.

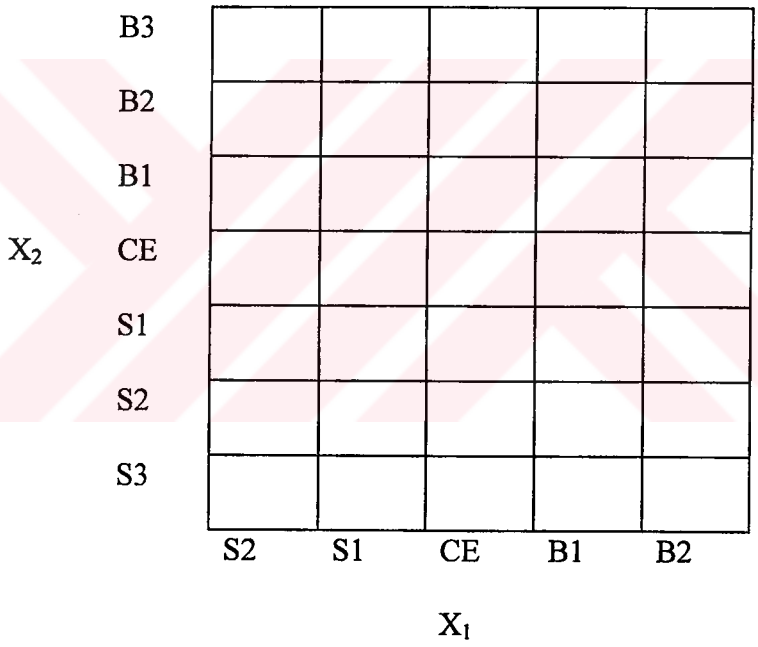


Fig. 4.10. The form of fuzzy rule base

Step-4 : Create a Combined Fuzzy Rule Base :

The form of a fuzzy rule base as illustrated in Fig. 4.10. We fill the boxes of the base with fuzzy rules according to the following strategy: A combined fuzzy

rule base is assigned rules from either those generated from numerical data or linguistic rules (we assume that a linguistic rule also has a degree that is assigned by the human experts and reflex the expert's belief of the importance of the rule); if there is more than one rule in one box of the fuzzy rule base, use the rule that has maximum degree. In this way, both numerical and linguistic information are codified into a common framework-the combined fuzzy rule base. If a linguistic rule is an "and" rule, it fills only one box of the fuzzy rule base; but, if a linguistic rule is an "or" rule (i.e., a rule for which the THEN part follows if any condition of the IF part is satisfied), it fills all the boxes in the rows or columns corresponding to the regions of the IF part. For example suppose we have the linguistic rule: " IF x_1 is S1 or x_2 is CE, THEN y is B2" for the fuzzy rule base of Fig. 4.10. then we fill the seven boxes in the column of S1 and the five boxes in the row of CE with B2. The degrees of all the B2's in these boxes equal the degree of this "or" rule.

Step-5 : Determine a Mapping Based on the Combined Fuzzy Rule Base :

We use the following defuzzification strategy to determine the output control y for given inputs (x_1,x_2) : first, for given inputs (x_1,x_2), we combine antecedents of the ith fuzzy rule using min operation to determine the degree, μ^i_{oi} of the output control corresponding (X_1,x_2) i.e.,

$$m^i_{oi} = \min (\mu_{I1}{}^i (x_1), \mu_{I2}{}^i(x_2)) \qquad (4.50)$$

where O^i denotes the output region of Rule i and I_j^i denotes the input region of Rule i for the jth component, e.g., Rule 1 gives

$$\mu^1{}_{CE} = \min (\mu_{B1}(x_1), \mu_{S1}(x_2)) \qquad (4.51)$$

Then we use the 4 defuzzification methods which are further explained in section 5.2.1. Step 5. for comparison.

From Steps 1 to 5 we see that our method is simple and straightforward in the sense that it is a one-pass build-up procedure that does not require time-consuming training; hence, it has the same advantage that the fuzzy approach has over neural approach, namely, it is simple and quick to construct.

This five step procedure can easily be extended to general multi-input multi-output cases. Steps 1 to 5 are independent of how many inputs and how many outputs there are.

If we view this five step procedure as a block, then the inputs to this block are "examples" (desired input-output data and expert rules (linguistic IF-THEN statements), and the output is a mapping from input space to output space. For control problems the input space is the state of the plant to be controlled, and the output space is the control applied to the plant. Our method essentially "Learns"' from the "Examples" and expert rules to obtain a mapping that, hopefully, has the "Generalization" property that when new inputs are presented the mapping continuous to give desired or successful outputs. Hence, our new method can be viewed as a very general model-free trainable fuzzy system for a wide range of control and signal processing problems, where : "Model-Free" means no mathematical model is required for the problem; "Trainable" means the system learns from " Examples " and expert rules, and can adaptively change the mapping when new " Examples" and expert rules are available; and, "Fuzzy" denotes the fuzziness introduced into the system by linguistic fuzzy rules, fuzziness of data, etc.

4.3.7.2. Takagi and Sugeno Type

As has been stated, in this approach, we consider a fuzzy model consisting of some number of implications that are of the format

If x_1 is A_1 and ....and x_k is A_k

then $y=p_0 + p_1\ x_1+....+ p_k\ x_k$

characterized by "and" connective and a linear equation.

For identification we have to determine the following three items by using the input-output data of an objective system [15].

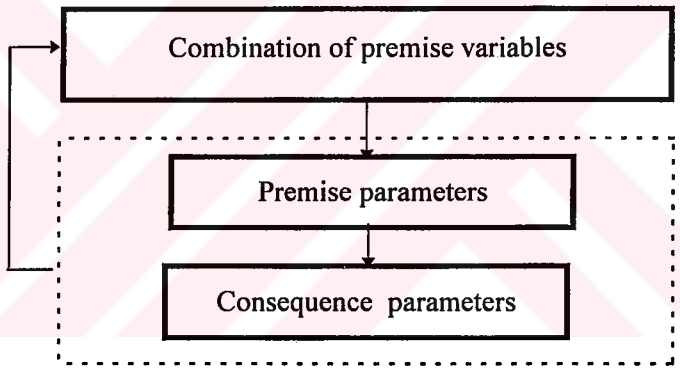


Fig. 4.11. Outline of the algorithm.

1) $x_1 +.....+ x_k$ Variables composing the premises of implications.

2) A_1 ,.....A_k Membership functions of the fuzzy sets in the premises, abbreviated as premise parameters.

3) p_0 ,.....p_k Parameters in the consequences.

Notice that all the variables in a premise may not always appear. The items 1) and 2) are related to the partition of space of input variables into some fuzzy subspaces. The item 3) is related to describing an input -output relation in each fuzzy subspace.

We can consider the relation among tree items hierarchically from 1) down to 3). The algorithm of the identification of implications is divided into three steps corresponding to the above three items. We first give a brief explanation of the algorithm at each step.

1) **Choice of Premise Variables** : First a combination of premise variables is chosen out of possible input variables sequence parameters are identified according to the steps 2) model and the output data of the objective system are calculated. We then improve the choice of the premise variables so that the performance index is decreased, which is defined as the root mean square of the output errors.

2) **Premise Parameters Identification** : In this step the optimum premise parameters are searched for the premise variables chosen at step 1). Assuming the values of premise parameters, we can obtain the optimum consequence parameters together with the performance index according to step 3). So the problem of finding the optimum premise parameters is reduced to a nonlinear programming problem minimizing the performance index.

3) **Consequence Parameters Identification** : The consequence parameters that give the least performance index are searched by the least performance index are

searched by the least squares method for the given premise variables in the step 1) and parameters in step 2). The outline of the algorithm in Fig. 4.11. From the next section we shall discuss the method in detail with an illustrative example at each step.

4.3.7.2.1. Consequence Parameters Identification

In this section we show how to determine the optimum consequence parameters to minimize the performance index, provided that both the premise variables and parameters are given. The performance index has been defined above as a root mean square of the output errors, which means the differences between the output data of an original system and those of a model.

Let a system be represented by the following implications:

R^1 If x_1 is A_1^1,....., and x_k is A_k^1

then $y=p_0^1+ p_1^1\ x_i +.....+ p_k^1\ x_k$

R^n If x_1 is A_1^n,....., and x_k is A_k^n

then $y=p_0^n+ p_1^n\ x_1 +.....+ p_k^n\ x_k$

Then the output y for the input $(x_1 ,.....,x_k)$ is obtained as

$$y= \frac{\sum_{i=1}^{n} (A_1^i(x_1)\wedge....\wedge A_k^i(x_k)).(p_0^i + p_1^i.x_1 +.....+p_k^i.x_k)}{\sum_{i=1}^{n} (A_1^i(x_1)\wedge....\wedge A_k^i(x_k))} \qquad (4.52)$$

Let β_i be

$$\beta_i = \frac{A_1^i(x_1)\wedge.....\wedge A_k^i(x_k)}{\sum_{i=1}^{n}(A_1{}^i(x_1)\wedge....\wedge A_k^i(x_k))} \tag{4.53}$$

then

$$y=\sum_i^n \beta_i(p_0^i + p_1^i \cdot x_1 + ...+ p_k^i \cdot x_k)$$

$$y=\sum_i^n (p_0^i \cdot \beta_i + p_1^i \cdot x_1\beta_i + ...+ p_k^i \cdot x_k \cdot \beta_i) \tag{4.54}$$

When a set of input-output data x_{1j}..... , x_{2j} ,...., x_{kj} y_j (j=1.....m) is given, we can obtain the consequence parameters $p_0^i, p_1^i, ..., p_k^i$ (i=1.....n) by the least squares method using (4.54).

Let X (m n(k+1) matrix), Y (m vector) and P (n (k+1) vector) be

$$X = \begin{bmatrix} \beta_{11},..., & \beta_{n1}, x_{11}.\beta_{11},..., & x_{11}.\beta_{n1},... \\ & .... & x_{k1}.\beta_{11}..., \quad x_{k1}.\beta_{n1} \\ : & & : \\ \beta_{1m},...., & \beta_{nm}, x_{1m}.\beta_{1m},...., & x_{1m}.\beta_{nm},... \\ ..... & ....., & x_{k1}.\beta_{1nm},.. \quad x_{k1}.\beta_{nm} \end{bmatrix} \tag{4.55}$$

where

$$\beta_{ij} = \frac{A_{i1}(x_{1j})\wedge.....\wedge A_{1k}(x_{kj})}{\sum_j A_{ij}(x_{ij})\wedge....\wedge A_{ik}(x_{kj})} \tag{4.56}$$

$$Y = [y_1, \ldots, y_m]^T \tag{4.57}$$

$$P = [p_0^1, \ldots, p_0^n, p_1^1, \ldots, p_1^n, \ldots, p_k^1, \ldots, p_k^n]^T \tag{4.58}$$

Then the parameter vector P is calculated by

$$P = (X^T X)^{-1} X^T Y. \tag{4.59}$$

It is noted that the proposed method is consistent with the reasoning method. In other words, this method of identification enables us to obtain just the same parameters as the original system, if we have a sufficient number of noiseless output data for the identification.

In this paper the parameter vector P is calculated by a stable-state Kalman filter. The so-called stable-state Kalman filter is an algorithm to calculate the parameters of a linear algebraic equation that gives the least squares of errors. Here we apply it to calculate the parameter vector P in (4.59).

Let the i th row vector of matrix X defined in (4.55) be x_i and the i th element of Y be y_i . Then P is recursively calculated by (4.60) and (4.61) where S_i is (n (k+1)) x (n (k+1)) matrix.

$$P_{i+1} = P_i + S_{i+1} \cdot x^T{}_{i+1} \cdot (y_{i+1} - x_{i+1} \cdot P_i) \tag{4.60}$$

$$S_{i+1} = S_i - \frac{S_i \cdot x^T{}_{i+1} \cdot x_{i+1} \cdot S_i}{1 + x_{i+1} \cdot S \cdot x_{i+1}^T} \qquad i=0,1,\ldots\ldots,m-1 \tag{4.61}$$

$$P = P_m \tag{4.62}$$

where the initial values of P_0 and S_0 are set as follows.

$$P_0 = 0 \tag{4.63}$$

$$S_0 = \alpha \cdot I \ (\alpha = \text{big number}) \tag{4.64}$$

where I is the identity matrix.

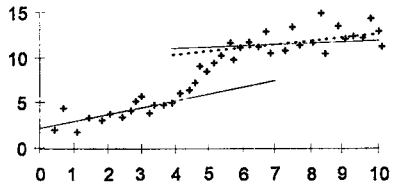


Fig.4.12 Results of identification.

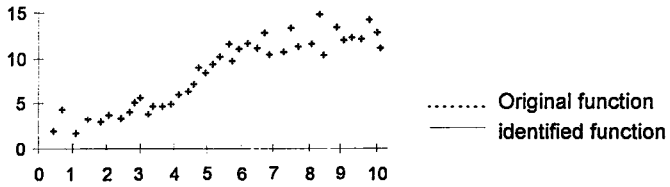
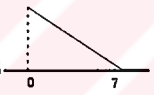
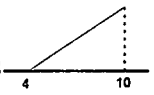


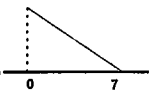
Fig.4.13 Input-Output data.

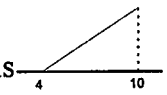
Example 5 : Suppose in a system

If x_1 is  then y= 0.6x + 2

If x_1 is  then y= 0.2x + 9

Under the condition that the premise of the model are fixed to those of the original system, the consequences are identified from input-output data as follows, where the noises are added to the data.

If x_1 is  then y= 0.56x + 2.17

If x_1 is 

then y= 0.11x + 9.60

Fig.4.12 shows the noised input-output data, the original consequences, and the identified consequences.

4.3.7.2.2. Premise Parameters Identification

In this section we show how to identify the fuzzy sets in the premises, that is, how to partition the space of premise variables into fuzzy subspaces, provided that the premise variables are chosen.

For example, looking at input-output data shown in Fig.4.13, we can see that the input-output characteristics change as the input x increases. So dividing the space of x into two fuzzy subspaces such that x is small or x is big, we have a model with the following two implications:

If x is small then $y=a_1x+b_1$

If x is big then $y=a_2x+b_2$

We next have to determine the membership functions of "small" and "big" as well as the parameters a_1,b_1,a_2 and b_2 in the consequences.

As it is easily seen, to divide the spaces into some fuzzy subspaces is to determine the membership functions of the fuzzy sets in the premises. The problem is thus to find the optimum premise parameters of their membership functions by which the performance index is minimized.

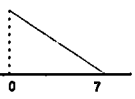
We call this procedure "premise parameter identification". The algorithm is as follows.

1) Assuming the parameters of the fuzzy sets in the premises, we can obtain the optimum parameters in the consequences that minimize the performance index as discussed in the previous section.
2) The problem of finding the performance index is reduced to a nonlinear programming problem. In this study we use the well-known complex method for the minimization. Each fuzzy set in the premises is determined by two parameters that give the greatest grade 1 and the least grade 0, since a fuzzy set is assumed to have a linear membership function.

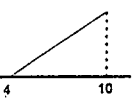
Example 6: This example shows the identification using the input-output data gathered from the preassumed system with noises. The standard deviation of the noises is five percent of that of the outputs. It has to be also noted that we can identify just the same parameters of the premises as the original system if noises do not exist.

It is of great importance to point out the above fact. If it is not the case, we cannot claim the validity of an identification algorithm together with a fuzzy system description language.

Suppose the original system exists with the following two implications:

If x_1 is 

then y= 0.6x + 2

If x_1 is 

then y= 0.2x + 9

The functions in the consequences of the implications and the noised input-output data are shown in Fig.4.14.

The identified premise parameters are as follows. We can see that almost the same parameters have been derived.

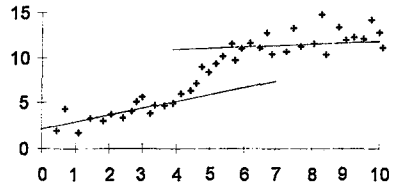
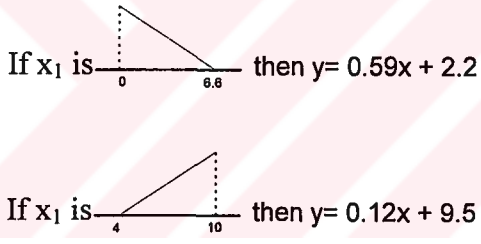


Fig.4.14 Consequences and noised data.



4.3.7.2.3. Choice Of Premise Variables

In this section suggest an algorithm to choose premise variables from the considerable input variables. As has been stated previously, all the variables of the consequences do not always appear in the premises. There are two problems concerned with the algorithm. One is the choice of variables: to choose a variable in the premises implies that its space is divided. The whole problem is a

combinatorial one. So in general there seems no theoretical approach available. Here we just take heuristic search method described in the following steps.

Suppose that we build a fuzzy model of a k-input $x_1,\ldots,x_k$ and single output system.

Step 1 : The range of x_1 is divided into two fuzzy subspaces "big" and "small" and the range of other variables $x_2,\ldots,x_k$ are not divided, which means that only x_1 appears in the premises of the implications. This model of two implications is thus

If x_1 is big_1 then .....

If x_1 is $small_2$ then .....

It is called model 1-1. Similarly, a model in which the range of x_2 is divided and the ranges of the other variables $x_1,x_3,\ldots,x_k$ are undivided is called model 1-2. In this way we have k-modes, each of which is composed of two implications. In general, the model 1-i is of the form

If x_i is big_i then .....

If x_1 is $small_i$then .....

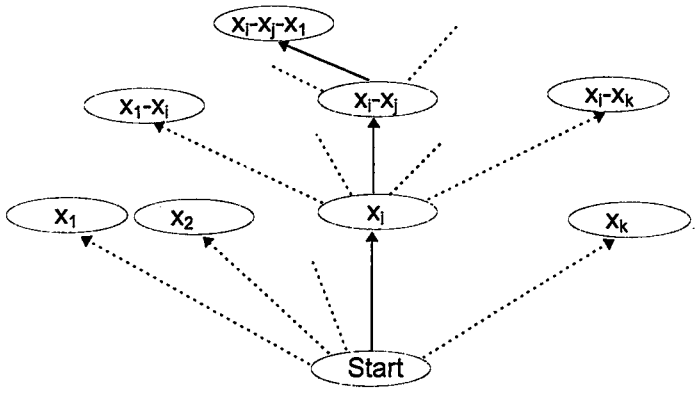


Fig. 4.15 Choice of premise variables.

Step 2: For each model the optimum premise parameters and consequence parameters are found by the algorithm described in the previous sections. The optimum model with the least performance index is adopted out of the k-models. It is called a stable state.

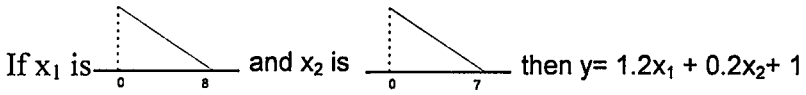
Step 3: Starting from a stable state at step 1, say model 1-i, where only variable x_i appears in the premises, take all the combinations of x_i-x_j (j=1,2,…,k) and divide the range of each variable into two fuzzy subspaces. For the combination x_i-x_i, the range of x_i is divided into four subspaces, for example "big", "medium big", "medium small" and "small". Thus we get k-models each of which is named model 2-j. Each consists of 2x2 implications. Then find again a model with the least performance index just as in step 2 that is also called a stable state at this step.

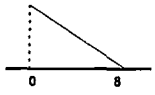
Step 4: Repeat step 3 in a similar way by putting another variable into the premise. The search is stopped if either of the following criteria is satisfied.

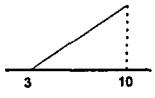
1) The performance index of a stable state becomes less than the predetermined value.
2) The number of implications of a stable state exceeds the predetermined number.

The choice of the variables in the premises proceeds as it is shown in Fig.4.15.

Example 7: We show an example of identification. The original system is also a fuzzy system with two inputs and single output expressed by the implications as are shown below.



If x_1 is 

and x_2 is 

then $y = 2.5x_1 + 2.1x_2 + 4$

In this system, the range of x_1 is divided into four fuzzy subspaces and that of x_2 into two fuzzy subspaces. So the number of implications is 4x2 altogether.

CHAPTER 5

APPLICATION

Our implementation is in the field of military and with this performance we aimed to meet a specific need of the Logistics Department of the Turkish Army. On the other hand, if this way of a control gain a considerable success in this field then some other and more complex control problems will be forwarded for solution. Unfortunately, despite benefits of fuzzy controllers on solving complex control problems and simulating the experienced operator actions, there exist no example of it in the army. With this implementation, we tried to supply an example of such a controller and we wish that this will be the beginning of fuzzy controllers in the Turkish Army on the fields that the traditional control methods are not or partially able to tackle.

5.1. Problem Definition

The usage of multi barreled rocket launchers in the army does not go back more than eight years. This weapon is our national design and in military sense, the effectiveness and the usability of it have been ensured. With the usage of multi

barreled rocket launcher units, there arose a need for transportation and manipulation of rocket ammunition. The ammunition is packed in eight pieces together and it weights slightly more than 2160 Kg.

The Logistics Department of the army had supplied trucks, which were equipped with a trailer and a crane, for transportation and manipulation of the ammunition. Recently, the department plans to allow the usage of the truck and the crane for the manipulations other than this specific ammunition, but telescopic arm of the crane Fig. 5.1 has some security limitations for different weights of loads. In Table 5.1. the calculated length-weight relation is shown. Mechanical engineers who had designed the crane stated that those values can surely be used for interpolating the length-weight relation which falls between any two items. As can be derived from the table the maximum weight that can be lifted is 20.000 kg. for which the maximum length of the arm ought to be 3 meters and at the same instance, this length is the possible minimum length of the telescopic arm. On the other hand the maximum length of the arm is 12.51 m. with which only 3840 kg. can be lifted.

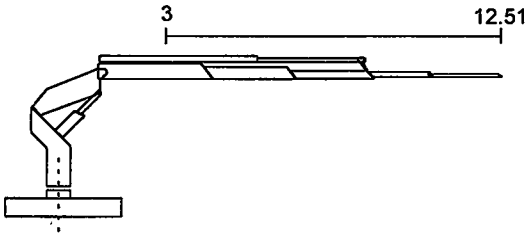


Fig. 5.1. Telescopic arm of the crane

Table 5.1. Length-Weight Relation for Interpolation

Length (m.)	Weight (Kg.)
3	20000
4.21	13815
6.11	9230
8.31	6780
10.41	4900
12.51	3840

By the help of four extension arms, lifting surface of the crane is adjusted to the horizontal level, for eliminating the angular effects of the ground.

Under the field circumstances it is not always possible to know the exact weight of the load, so there should be some means of control to keep the arm in secure limits. At present the crane has a ruler which is mounted on the arm for measuring the arm length. But this way of a measurement is found unsuitable by the operators, since they have to keep an eye on the ruler while operating the crane. Although there are some maintenance problems on keeping the visual measurement devices tuned, under the heavy conditions of field and warfare. So, the department guaranties the secure usage of the crane only with an experienced operator. Experienced operators infer the weight of the load by analogy and sense the security violence by the help of the vibrations of the hydraulic pump. More over, due to the obligatory service system of the Turkish Armed Forces, it is difficult to find an experienced operator for each truck. The operators of the trucks are chosen among the personnel who are serving in the army. Depending on the limited service period of the operators, the maximum service time for an experienced operator can't be more than 14 months.

We aimed to produce a more usable and more secure tool for the logistics department of the army, which will automatically utilize the experiences gained by the operators. As a matter of fact, the traditional control methods need the mathematical model of the system should be defined precisely, and it is absolutely not a feasible task to define a mathematical model for this problem. We decided to implement a fuzzy logic controller which we thought to be more efficient and less time consuming solution.

In formally speaking, we tried to implement a fuzzy controller which will run under the following modes.

- Just showing the sensed weight and maximum secure length of the arm (no fuzzy control)
- By indicating the required adjustment value to the operator, and the operator should take proper action, but the controller alarm the operator when secure limits are violated.(half fuzzy control)
- In this mode fuzzy controller is responsible for taking the required action.(full fuzzy control)

We have studied the last mode (full fuzzy control), for the other two modes can be achieved by putting some constraints to the last one.

The inputs to the controller are considered to be the weight of the load and the length of the crane lifting arm, both of which are sensed by electronic devices. With having these inputs the controller should find the maximum secure length of the arm for the inputted weight, by the help of interpolation on Table 5.1. After determining the maximum secure length, the controller should produce appropriate commands for the valve which controls the flow of the hydraulic liquid in/out of the extension piston, if the current position of the arm is longer than the calculated secure length. Forward and backward movement of the arm is done by the help of a hydraulic piston. The control device for this piston is a valve and it has 6 adjustable positions (between 0^0 and 90^0 by 15^0 in each step) on one direction forward or backward. As a total of 13 positions including 0^0 which means no action and also falls just between the forward and backward movements. The

movement of the arm shows different characteristics in each direction. Forward movement is slower than backward movement. So, in the time domain the same angle value for the valve, but in different directions, cause different amount of change in the length of the arm.

Experienced operator commands differ from each other, with respect to operator, while adjusting the arm to a specified length. We have directed several experiments on three of those operators and collected the data shown in Table 5.2. As can be seen on the results of experiments the control commands vary in the last ±10 cm. of adjustment.

Table 5.2. Results of experiments

Experiment #	Error*	Valve Angle		
		Operator # 1	Operator # 2	Operator # 3
1	+ 10	+90	+90	+90
2	+10	+90	+90	+90
3	+9	+90	+75	+90
4	+8	+75	+75	+75
5	+7	+75	+75	+60
6	+6	+75	+75	+60
7	+5	+60	+60	+45
8	+4	+45	+45	+45
9	+3	+45	+30	+30
10	+2	+30	+30	+15
11	+1	+15	+15	+15
12	0	0	0	0
13	-1	-15	-15	-15
14	-2	-30	-15	-30
15	-3	-30	-15	-30
16	-4	-30	-30	-30
17	-5	-45	-30	-45
18	-6	-45	-30	-45
19	-7	-60	-45	-60
20	-8	-60	-60	-60
21	-9	-75	-60	-75
22	-10	-75	-75	-90
23	- (-10)	-90	-90	-90

* Error = Desired length - Current length

5.2. Implementation

In the implementation phase we have realized a Mamdani type and a Takagi & Sugeno type fuzzy controller, and we have built a simulation procedure to simulate the behavior of the real system. Although the rules, which causes the crane to extend the telescopic arm when the weight is light, make no sense at the first glance, the necessity of them is hidden in the propagation of the arm when reaching the stable condition. General configuration of the controller is shown in Fig. 5.2. The error between the desired position and the current position is calculated as

Error = Desired Position - Current Position

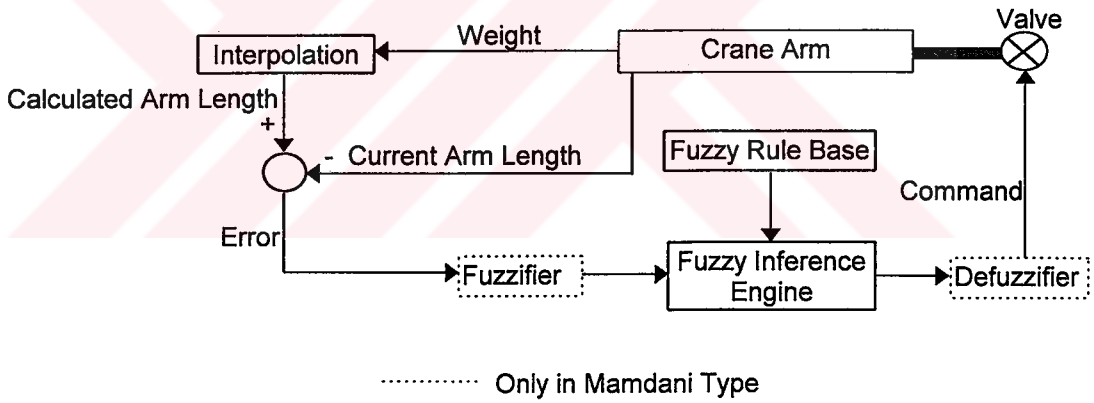


Fig. 5.2. General configuration of the controller

5.2.1. Mamdani Type

Step 1 : Divide the Input and Output Spaces into Fuzzy Regions.

After interpolating the maximum length of the arm depending on the sensed weight, the error between the current position and the calculated length is

derived. This error value is constrained in to a maximum of ±10 cm. For the movement of the arm is very slow with respect to the frequency of adjustment commands, the errors which have greater absolute values than 10 are rounded to nearest ±10. By applying this constraint to the error we bounded the input space within ±10. The linguistic variables and subjectively defined membership functions for the input error and the output value adjustment commands are shown in Fig. 5.3.

PL : Positive Large

PM : Positive Medium

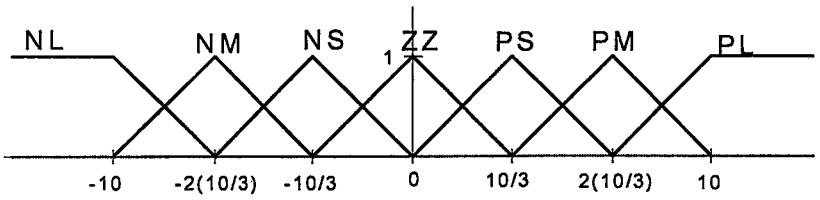
PS : Positive Small

ZZ : Zero

NS : Negative Small

NM : Negative Medium

NL : Negative Large



(a) Linguistic Variables and Membership Functions for the input Error.

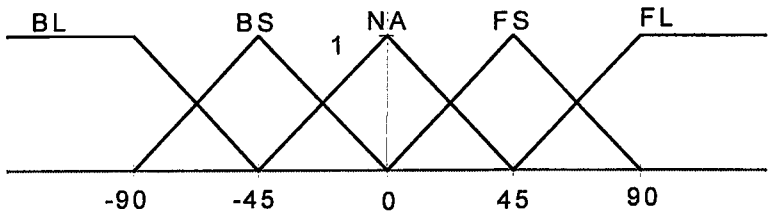
FL : Forward Large

FS : Forward Small

NA : No Action

BS : Backward Small

BL : Backward Large



(b) Linguistic Variables and Membership Functions for Valve Commands

Fig.5.3. Membership Functions for Input and Output variables

Step 2 : Generate Fuzzy Rules From Given Data Pairs.

We built the following tables (Table 5.3.-9) from the results of experiments in Table 5.2. In these tables the fuzzy subspaces in which the error and the command of operators fall are shown with their membership values in the respective fuzzy sets. The highlighted cells indicete the data pair which satisfies the criteria stated in step 3. If there are more than one data pair which gives the same conclusion with the same weighting, then we mark the first cell on the way from top to bottom by row vise in the table.

The data pairs of which the error part falls in the fuzzy set PL is shown in Table 5.3.

Table 5.3. The data pairs of which the error part is PL

Exp. #	Error		Command					
			Opr. # 1		Opr. # 2		Opr. # 3	
	Subspace	μ	Subspace	μ	Subspace	μ	Subspace	μ
1	PL	1.0	FL	1.0	FL	1.0	FL	1.0
2	PL	1.0	FL	1.0	FL	1.0	FL	1.0
3	PL	0.7	FL	0.6	FL	0.6	FL	1.0

Generated rules :

R1 : IF Error is PL THEN Command is FL

The data pairs of which the error part falls in the fuzzy set PM is shown in Table 5.4.

Table 5.4. The data pairs of which the error part is PM

Exp. #	Error		Command					
			Opr. # 1		Opr. # 2		Opr. # 3	
	Subspace	μ	Subspace	μ	Subspace	μ	Subspace	μ
4	PM	0.6	FL	0.6	FL	0.6	FL	0.6
5	PM	0.9	FL	0.6	FL	0.6	FS	0.6
6	PM	0.8	FL	0.6	FL	0.6	FS	0.6
7	PM	0.5	FS	0.6	FS	0.6	FS	1.0

Generated rules :

R2 : IF Error is PM THEN Command is FL

R3 : IF Error is PM THEN Command is FS

The data pairs of which the error part falls in the fuzzy set PS is shown in Table 5.5.

Table 5.5. The data pairs of which the error part is PS

Exp. #	Error		Command					
			Opr. # 1		Opr. # 2		Opr. # 3	
	Subspace	μ	Subspace	μ	Subspace	μ	Subspace	μ
7	PS	0.5	FS	0.6	FS	0.6	FS	1.0
8	PS	0.8	FS	1.0	FS	1.0	FS	1.0
9	PS	0.9	FS	1.0	FS	0.6	FS	0.6
10	PS	0.6	FS	0.6	FS	0.6	NA	0.6

Generated rules :

R4 : IF Error is PS THEN Command is FS

R5 : IF Error is PS THEN Command is NA

The data pairs of which the error part falls in the fuzzy set ZZ is shown in Table 5.6.

Table 5.6. The data pairs of which the error part is ZZ

Exp. #	Error		Command					
			Opr. # 1		Opr. # 2		Opr. # 3	
	Subspace	μ	Subspace	μ	Subspace	μ	Subspace	μ
11	ZZ	0.7	NA	0.6	NA	0.6	NA	0.6
12	ZZ	1.0	NA	1.0	NA	1.0	NA	1.0
13	ZZ	0.7	NA	0.6	NA	0.6	NA	0.6

Generated rules :

R6 : IF Error is ZZ THEN Command is NA

The data pairs of which the error part falls in the fuzzy set NS is shown in Table 5.7.

Table 5.7. The data pairs of which the error part is NS

Exp. #	Error		Command					
			Opr. # 1		Opr. # 2		Opr. # 3	
	Subspace	μ	Subspace	μ	Subspace	μ	Subspace	μ
14	NS	0.6	BS	0.6	NA	0.6	BS	0.6
15	NS	0.9	BS	0.6	NA	0.6	BS	0.6
16	NS	0.8	BS	0.6	BS	0.6	BS	0.6
17	NS	0.5	BS	1.0	BS	0.6	BS	1.0

Generated rules :

R7 : IF Error is NS THEN Command is NA

R8 : IF Error is NS THEN Command is BS

The data pairs of which the error part falls in the fuzzy set NM is shown in Table 5.8.

Table 5.8. The data pairs of which the error part is NM

Exp. #	Error		Command					
			Opr. # 1		Opr. # 2		Opr. # 3	
	Subspace	μ	Subspace	μ	Subspace	μ	Subspace	μ
17	NM	0.5	BS	0.6	BS	0.6	BS	1.0
18	NM	0.8	BS	1.0	BS	0.6	BS	1.0
19	NM	0.9	BS	0.6	BS	1.0	BS	0.6
20	NM	0.6	BS	0.6	BS	0.6	BS	0.6

Generated rules :

R9 : IF Error is NM THEN Command is BS

The data pairs of which the error part falls in the fuzzy set NL is shown in Table 5.9.

Table 5.9. The data pairs of which the error part is NL

Exp. #	Error		Command					
			Opr. # 1		Opr. # 2		Opr. # 3	
	Subspace	μ	Subspace	μ	Subspace	μ	Subspace	μ
21	NL	0.7	BL	0.6	BS	0.6	BL	1.0
22	NL	1.0	BL	0.6	BL	0.6	BL	1.0
23	NL	1.0	BL	1.0	BL	1.0	BL	1.0

Generated rules :

R10 : IF Error is NL THEN Command is BS

R11 : IF Error is NL THEN Command is BL

Step 3 : Assign A Degree To Each Rule.

We assign a degree D(Rule) to each rule, for resolving the conflicting rule groups and reducing the number of generated rules. Our strategy in assigning the degrees is as in (5.1)

$$D(Rule_n) = \min(\mu_{se}(Error_n), \mu_{ce}(Command_n), m_n) \quad (5.1)$$

where

$Rule_n$: Rule with number n

se : Fuzzy subspace that the $Error_n$ falls with maximum membership degree

ce : Fuzzy subspace that the $Command_n$ falls with maximum membership degree

m_n : Degree that represents our belief for the data pair which produces $Rule_n$

We determined m_n for $Rule_n$ by

$$m_n = 3 / Count_{ce} \quad (5.2)$$

where

$Count_{ce}$ is the count of operator commands which falls into the fuzzy set ce for the same value of error ($Error_n$).

By applying these criterias, we concluded the followings.

D(Rule1) = min(1.0, 1.0, 1.0) = 1.0

D(Rule2) = min(0.6, 0.6, 1.0) = 0.6

D(Rule3) = min(0.5, 0.6, 1.0) = 0.5

D(Rule4) = min(0.8, 1.0, 1.0) = 0.8

D(Rule5) = min(0.6, 0.6, 0.3) = 0.3

D(Rule6) = min(1.0, 1.0, 1.0) = 1.0

D(Rule7) = min(0.6, 0.6, 0.3) = 0.3

D(Rule8) = min(0.9, 0.6, 0.6) = 0.6

D(Rule9) = min(0.9, 1.0, 1.0) = 0.9

D(Rule10) = min(0.7, 0.6, 0.3) = 0.3

D(Rule11) = min(1.0, 1.0, 1.0) = 1.0

We purified the conflicting rules by the help of rule degrees.

Rule1 → IF Error is PL THEN Command is FL

max(D(Rule2), D(Rule3)) → D(Rule2) → IF Error is PM THEN Command is FL

max(D(Rule4), D(Rule5)) → D(Rule4) → IF Error is PS THEN Command is FS

Rule6 → IF Error is ZZ THEN Command is NA

max(D(Rule7), D(Rule8)) → D(Rule8) → IF Error is NS THEN Command is BS

Rule9 → IF Error is NM THEN Command is BS

max(D(Rule10), D(Rule11))→D(Rule11)→IF Error is NL THEN Command is BL

Step 4 : Create a Combined Fuzzy Rule Base.

Combined rule base is shown in Table 5.10.

Table 5.10. Combined Rule Base

Error	PL	PM	PS	ZZ	NS	NM	NL
Command	FL	FL	FS	NA	BS	BS	BL

The rules for the controller rule base can be written as ;

IF Error is PL OR PM THEN Command is FL

IF Error is PS THEN Command Is FS

IF Error is ZZ THEN Command Is NA

IF Error is NS OR NM THEN Command Is BS

IF Error is NL THEN Command Is BL

Step 5 : Determine a Mapping Based on the Combined Fuzzy Rule Base.

Defuzzification of the inference output is carried out by the following 4 methods for comparison.

1. Center of gravity weighted by matching

$$y_0 = \frac{\sum_i h_i W_i}{\sum_i h_i}$$

2. Maximum value weighted by matching

$$y_0 = \frac{\sum_i h_i G_i}{\sum_i h_i}$$

3. Center of gravity of the fuzzy set with largest matching

$B'_k = B'_i \mid hi = \text{Max}\,(h_t), \forall\, t \in \{1,...,m\}$

$y_0 = W_k$

4. Maximum value of the fuzzy set with largest matching

$B'_k = B'_i \mid hi = \text{Max}\,(h_t)\ \forall\, t \in \{1,..,m\}$

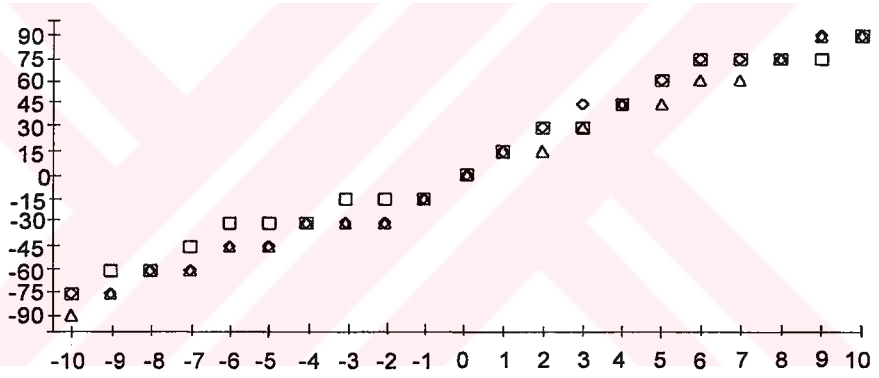
$y_0 = G_k$

where,

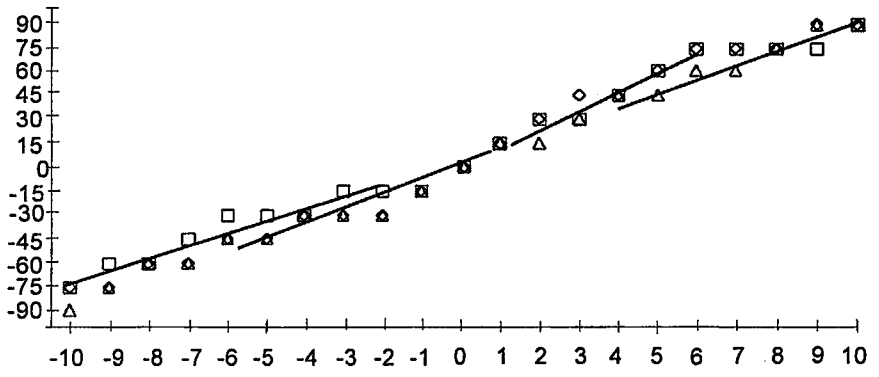
- y_0 is the global output from the inference process
- B'_i is the fuzzy set obtained as output when performing inference on rule R_i
- h_i is the matching among the inputs and the fuzzy rule R_i

W_i and G_i are the center of gravity and the maximum value of the fuzzy set obtained from the inference process performed on rule R_i

5.2.2. Takagi & Sugeno Type



(a) Input-Output Relation

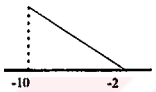


(b) Partition of Input-Output Relation

Fig.5.4.

5.2.2.1. Premise Parameter Identification

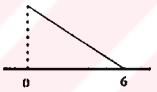
The input-output relation of the system is shown in Fig.5.4. This relation is derived from the data which were collected from the experienced operator actions (Table 5.2.). Loking at the input-output data in Fig.5.4.(a), we can derive the fuzzy subspaces of relation. In Fig.5.4.(b) the partition of relation into linear fuzzy subspaces is shown. With this four fuzzy sets of relation, we obtain the following implications.

1 : IF Error is 

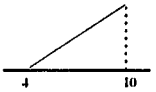
then Command is $P_0 + P_1$ x Error

2 : IF Error is 

then Command is $P_0 + P_1$ x Error

3 : IF Error is 

then Command is $P_0 + P_1$ x Error

4 : IF Error is 

then Command $P_0 + P_1$ x Error

5.2.2.2. Consequence Parameter Identification

After identification of premise parameters, we calculated $A_{ij}(X_j)$ and β_i for the stated algorithm in section 4.3.7.2.1. Where i = 1...n and j = 1...k. As an

example lets take the input -3 and the corresponding command of operator # 1 at row 15 in Table 5.2. The calculated β values for this data pair are

$\beta_1 = 0.125 / 0.625 = 0.2$

$\beta_2 = 0.5 / 0.625 = 0.8$

$\beta_3, \beta_4 = 0$

and the 15^{th} row of the matrix X is

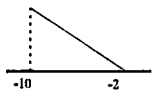
[0.2, 0.8, 0, 0, 0.2 x (-3), 0.8 x (-3), 0, 0]

The 15th index of Y should be 30 which is the corresponding command value for the input error -3.

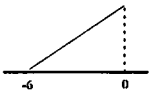
By taking α as 20 and solving the iterative formulea (4.60) and (4.61), we concluded that P is

$$P = \begin{bmatrix} 0.038, & 9.008 \\ 0.017, & 10.082 \\ 0.005, & 12.323 \\ 0.047, & 8.412 \end{bmatrix}$$

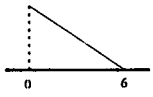
which means

1 : IF Error is 

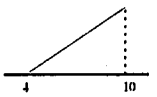
then Command is 0.038 + 9.008 x Error

2 : IF Error is 

then Command is 0.017 + 10.082 x Error

3 : IF Error is 

then Command is 0.005 + 12.323 x Error

4 : IF Error is 

then Command is 0.047 + 8.412 x Error

Final output of the Takagi and Sugeno type controller was calculated with the following weighted average formulla.

$$U' = \frac{\sum_i h_i \, u_i}{\sum_i h_i}$$

where

U' : Final output of the controller.

h_i : Matching degree for the rule i.

u_i : Output of the rule I.

5.2.3. Simulation Procedure

The real behavior of the crane arm was needed to be simulated, with the outputs for deciding on the accuracy of the controllers. So we built a simulation procedure. We had collected large amount of data from the experiments which had been carried out on the real system, and we had seen that the exact length of the arm includes some amount of error after giving a command. We analyzed the distribution of errors and reached to the conclusion that the error is proportional to the propagated distance and as can be seen in Fig.5.5. the error probabilities present normal distribution characteristics. The value of proportion is rounded to 10 % which was 9.8753 in exact averaging of the errors.

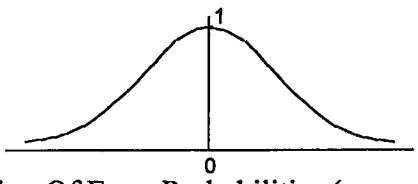


Fig.5.5. Distribution Of Error Probabilities (proportional to length)

By the help of the stochastic approaches the simulation procedure was coded with the option of frequency parameter which is used to analyze the step responses and stabilization times of the designed controller algorithms. By frequency we mean that the period between the two consecutive reading of the arm length. In analyzing the step responses of the controller, we used 10 different reading frequencies from 1 to 10 reading per unit time. As can be seen on the output graphs in the section 5.4. some frequencies give better step responses and some give better stabilization times than the others. We have carried out all the simulation processes on a PC with a Pentium 100 processor, so the exact determination of the timing unit should be calculated by using the imposed hardware restrictions, such as the speed of the controller which is to be used. The simulated and tasted beginning positions are shown in Table 5.11.

Table 5.11. Beginning Positions for Simulations.

Beginning Position (cm.)	Load (Kg.)
8.00	7000
	10250

5.3. Results

In the simulations we have tasted the behavior of the controller algorithms with 100 iterations. In Fig. 5.6. to 5.25, we showed the step responses and timing graphs of the controller outputs. The step response graphs indicate the position of

the crane arm at each reading instant and the timing graphs present the time-position relation. Specially some of the step response graphs are scaled down to 30 iterations (in the experiments the iteration count was 100) to make the differences in responses more distinguishable. In graphs each line represents a different frequency output.

In the figures "ln" and "we" indicates the starting arm position and the weight of the load respectively.

MAMDANI TYPE

Center of Gravity Weighted by Matching :

ln: 800 cm. we: 7000 Kg.

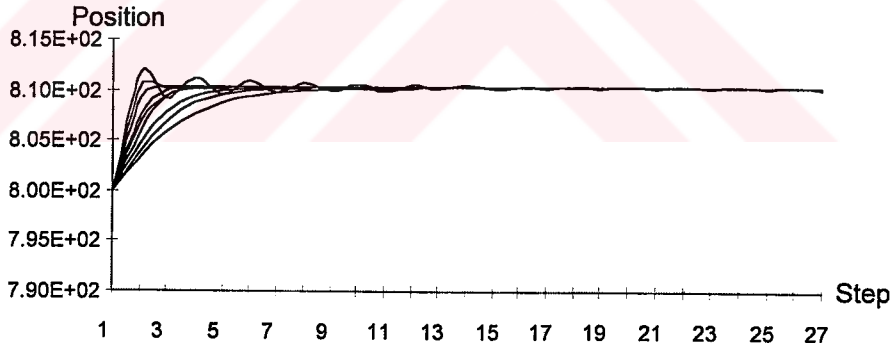


Fig 5.6. Step- Position

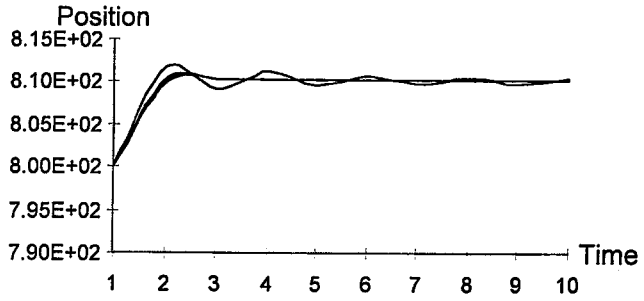


Fig.5.7. Time-Position

Center of Gravity with Largest Matching

ln: 800 cm. we: 7000 Kg .

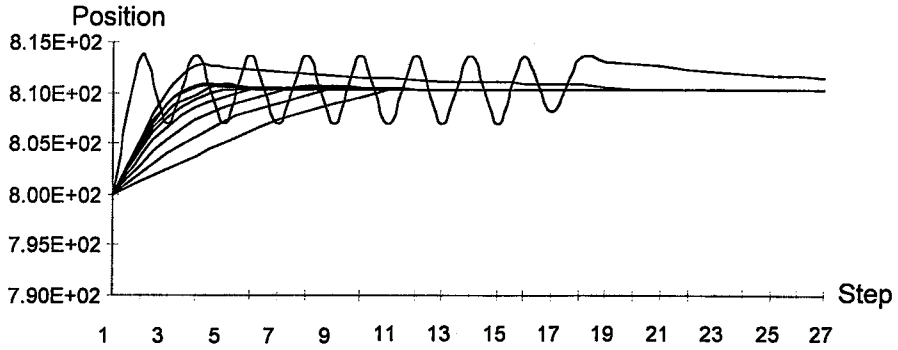


Fig.5.8. Step- Position

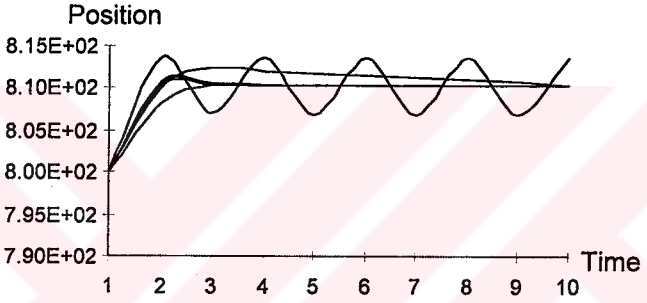


Fig.5.9. Time-Position

Maximum Value Weighted by Matching

ln: 800 cm. we: 7000 Kg .

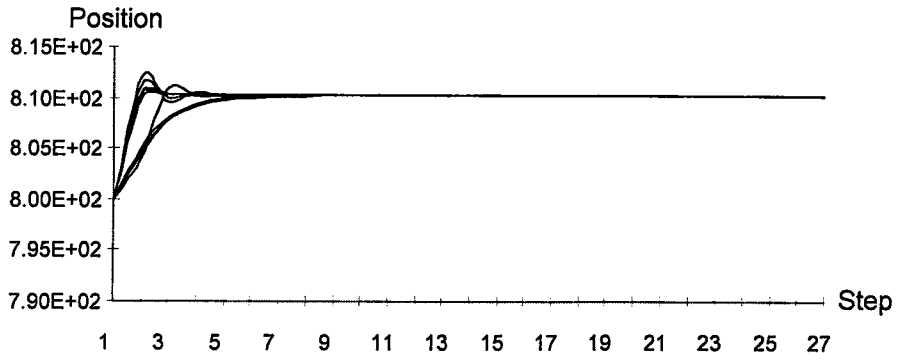


Fig.5.10. Step- Position

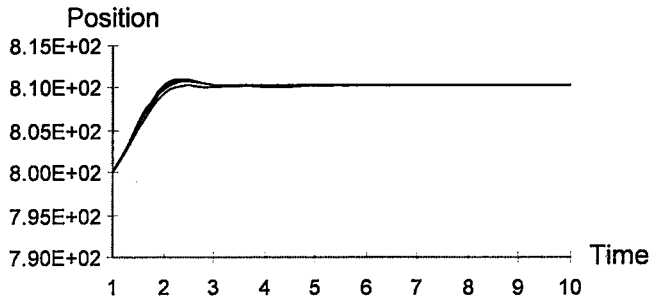


Fig.5.11. Time-Position

Maximum Value with Largest Matching

ln: 800 cm. we: 7000 Kg .

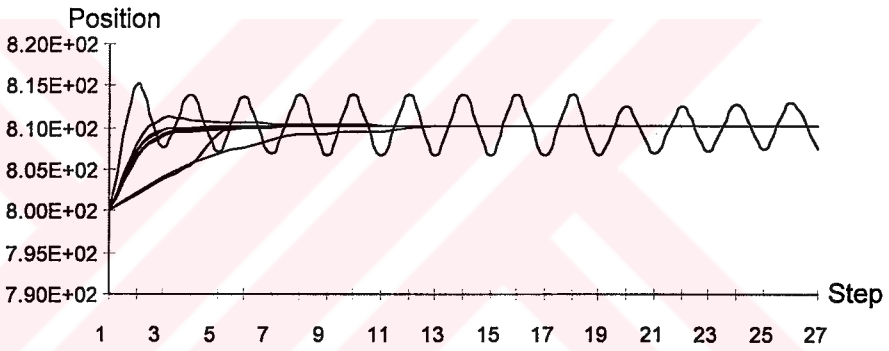


Fig.5.12. Step- Position

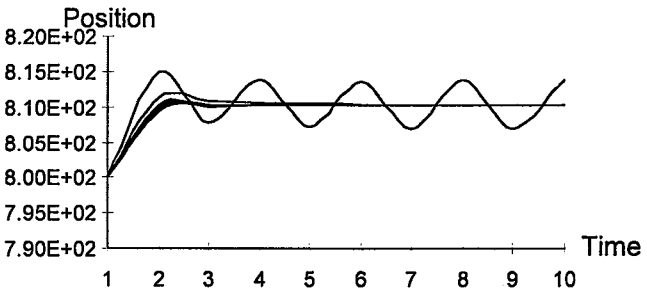


Fig.5.13. Time-Position

Center of Gravity Weighted by Matching

ln:800 cm. we:10250

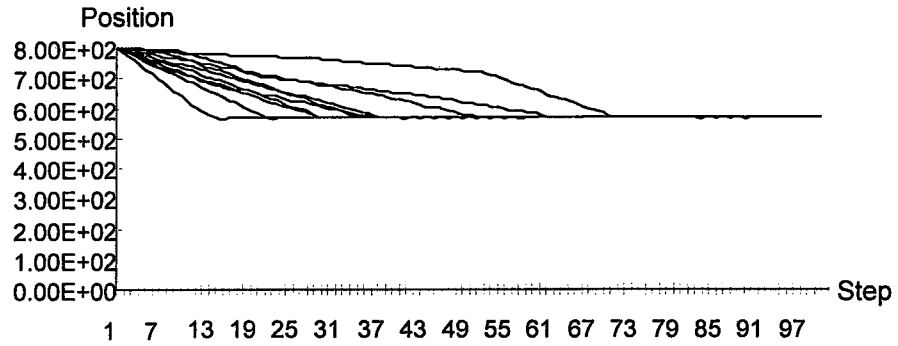


Fig.5.14. Step-Position

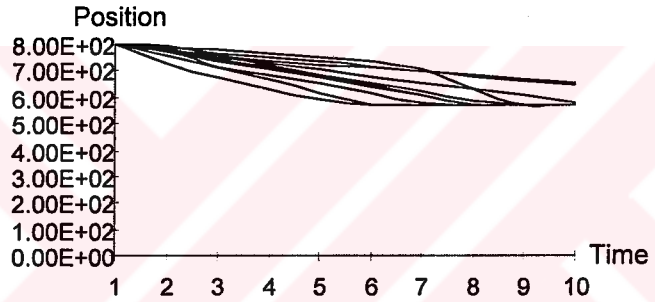


Fig.5.15. Time-Position

Center of Gravity with Largest Matching

ln:800 cm. we:10250

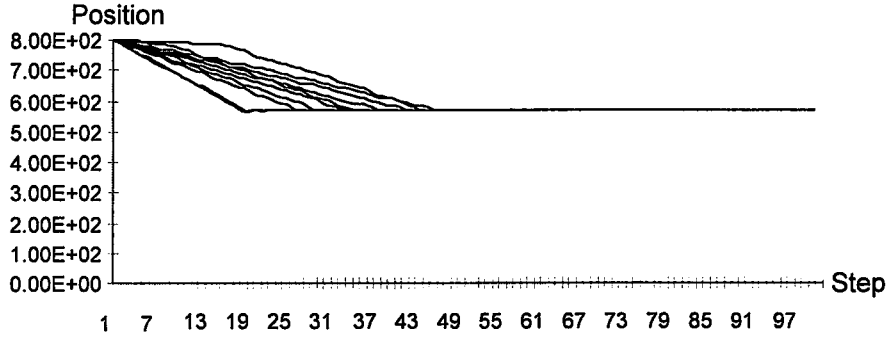


Fig.5.16. Step-Position

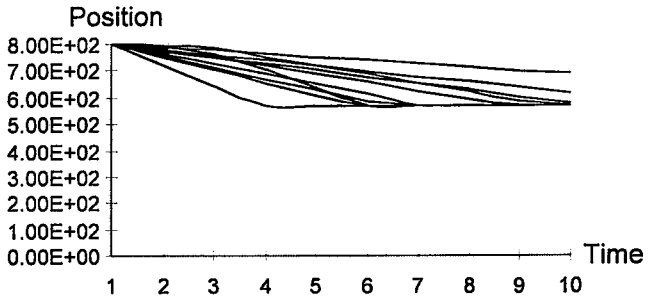


Fig.5.17. Time-Position

Maximum Value Weighted by Matching

ln:800 cm. we: 10250

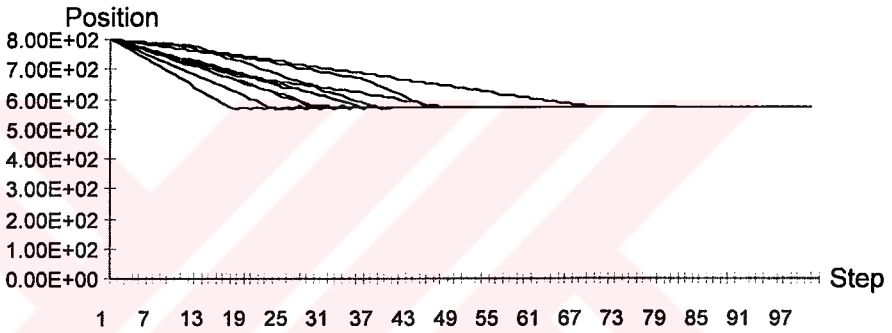


Fig.5.18. Step-Position

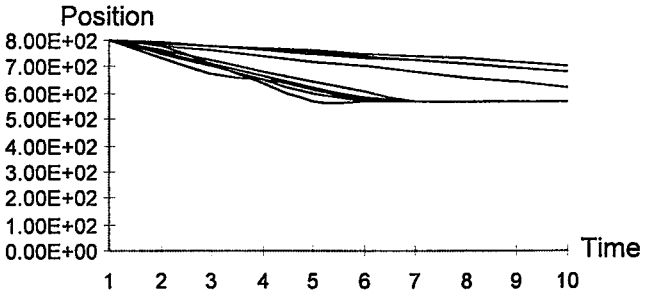


Fig.5.19. Time-Position

Maximum Value with Largest Matching

ln:800 cm. we:10250

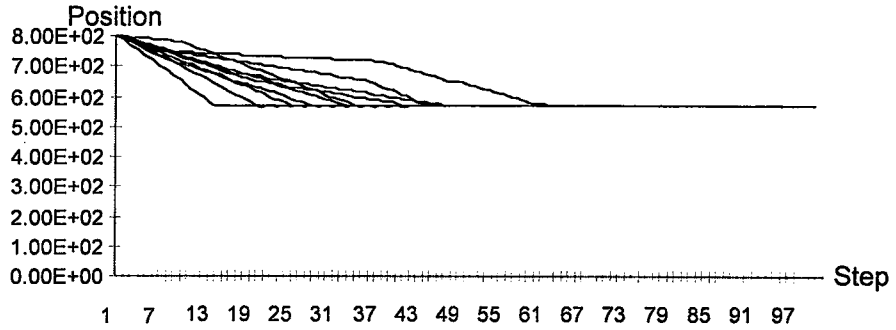


Fig.5.20. Step-Position

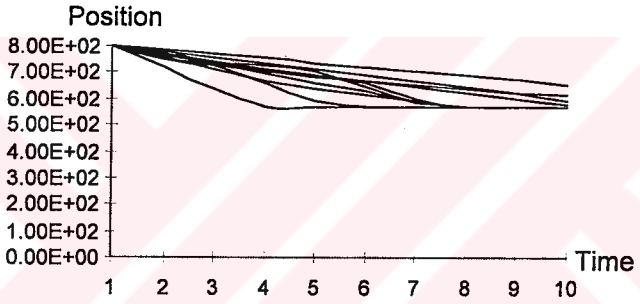


Fig.5.21. Time-Position

TAKAGI & SUGENO TYPE

ln:800 cm. we: 7000

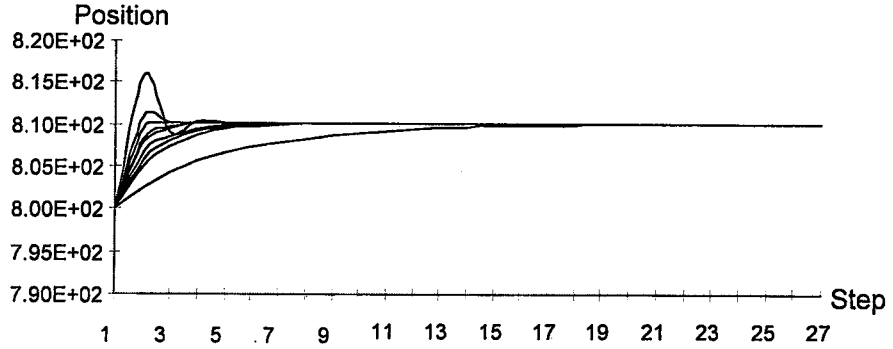


Fig.5.22. Step-Position

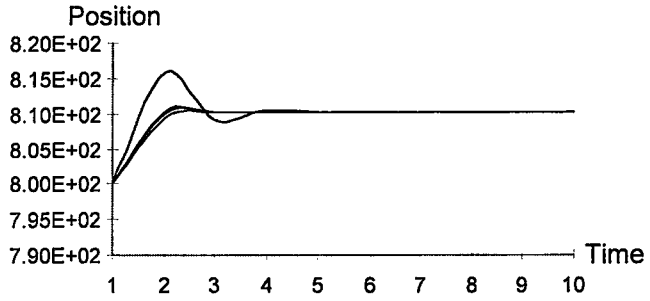


Fig.5.23. Time-Position

ln:800 cm. we: 10250

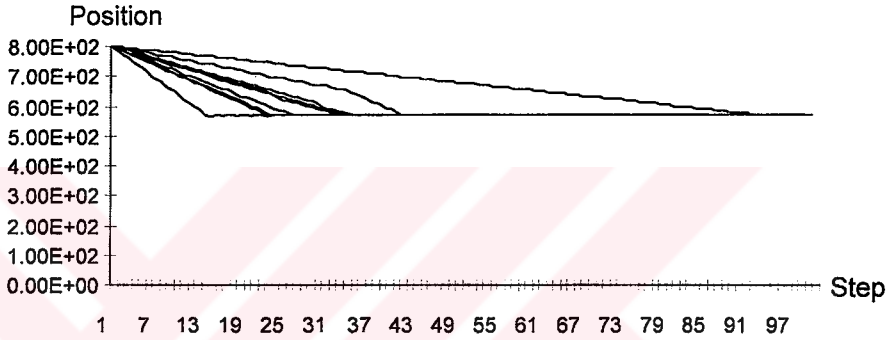


Fig.5.24. Step-Position

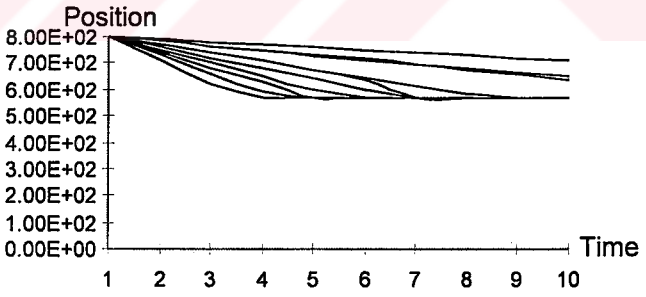


Fig.5.25. Time-Position

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

The simulations that we carried out on Mamdani type controller algorithms, showed that the weighted by matching strategy produces more accurate results than largest matching. On the other hand maximum value of the inferred fuzzy set gave better timing response than center of gravity, if weighted by matching strategy was used. As expected, in slow frequencies (1-6), the fluctuation period of the arm was longer than higher ones (7-10). Moreover, as can be seen in Fig. 5.6-9, 5.12, 5.13 the arm never reached to a stable state at frequency 1.

Takagi and Sugeno type presented the best behavior of control, on our simulations, for its fast convergence to the stable state and property of reducing the error in a linear way. In Takagi and Sugeno type, our tests showed that, even in slow frequencies it guaranties the stable state.

As the result of this work, we recommend the Takagi and Sugeno type controller which was developed in the scope of our performance, be used in the control problem of the crane arm.

For future work, we address

- The adjustment of the crane lifting surface to the horizontal level by the help of four extension arms, can easily be controlled with a controller of same type, since while operating some distortions usually take place depending on the density of the ground on which the crane operates.
- The swing control of the load can be achieved by a fuzzy controller.
- More accurate consequence parameters for Takagi and Sugeno type controller may be found by analyzing larger amount of input-output data pair than we worked on.

REFERENCES

[1] Dr. L. Reznik, Fuzzy Controller Design Methods Classification and Parameters Choice, Proc. Seventh IFSA World Congress Prague 1997.

[2] Li-Xin Wang-Jerry M. Mendel, Generating Fuzzy Rules by Learning From Examples, Proc. Seventh IFSA World Congress Prague 1997.

[3] De Silva, Clarence W., Intelligent Control, CRC Press Boca Raton 1995.

[4] Hong-Xing Li, Fuzzy Sets and Fuzzy Decision-Making, CRC Press Boca Raton 1995.

[5] Li-Xin Wang, Adaptive Fuzzy Systems and Control, PTR Prentice Hall Englewood Cliff 1994.

[6] Karl J. Astrom-Bjorn Wittenmark, Computer Controlled Systems, Prentice Hall International 1994.

[7] Viliem Novak Fuzzy Approach to, Reasoning and Decision Making, Kluwer Academic Dordrecht 1993.

[8] D. Driankov-H. Hellendorn-M. Reinfrank, An Introduction To Fuzzy Control, Springer-Verlag 1993.

[9] Toshiro Terano-Kyoji Asai-Michio Sugeno, Fuzzy Systems Theory and Its Applications, Academic Press Inc. 1992.

[10] M. Shinners Stanley, Modern Control Systems Theory and Design, John Wiley and Sons Inc. 1992.

[11] Gustaf Olsson-Gianguido Piani, Computer Systems for Automation and Control , Prentice Hall International 1992.

[12] L. Lewis Frank, Applied Optimal Control & Estimation, Prentice Hall International 1992.

[13] E. Rich- K. Knight, Artifical Intelligence, New York McGraw Hill Inc. 1991.

[14] S. Miyamoto, Fuzzy Sets In Information Retrival and Cluster Analysis, Kluwer Academic Publishers Dortrecht 1990.

[15] T. Takagi-M. Sugeno, Fuzzy Identification of Systems and Its Applications to Modeling and Control, Proc. IEEE Transactions on systems, Man, and Cybernetics Vol SMC-15, no.1, pp 116-132 1985.

[16] H. Blomberg-R. Ylinen, Algebraic Theory for Multivariable Linear Systems, New York Academic Press 1983.

[17] K. J. Astrom, Theory and Applications of Adaptive Control Automatica, 1983.

[18] M. Sugeno and T. Takagi, Multi Dimentional Fuzzy Reasoning, Fuzzy Sets and Systems Vol:9 no 2 1983.

[19] K. J. Astrom-B. Wittenmark, Self Tuning Controllers Based On Pole-Zero Placement, Proc. IEE pt D.127, 120-30 1980.

[20] H. Nyquist-M. Bode, Frequency-Response Methods In Control Systems, IEEE Press selected reprint series 1979.

[21] V. Kucera, Dicrete Linear Control , Prague Academia 1979.

[22] G. C. Goodwin-R. L. Payne, Dynamic System Identification Experiment Design and Analysis, New York Academic Press 1977.

[23] E. H. Mamdani, Appication Fuzzy Algorithms For Control of Simple Dynamic Plant, Proc. IEEE vol:121 no 12, pp 1585-1588 1976.

[24] W. M. Wonham, Linear Multivarible Control: A geometric Approach, New York Springer-Verlag 1974.

[25] K. J. Astrom-B. Wittenmark, On Self Tuning Regulators Automatica, 9, 185-99 1973.

[26] K. J. Astrom-P. Eykhoff, Systems Identification Automatica, 7,123-62 1971.

[27] H. H. Rossenbrock, State Space and Multivariable Theory London Nelson 1970.

[28] R. E. Kalman-P. L. Falb-M. A. Arbib, Topics In Mathemathical System Theory, New York McGraw Hill 1969.

[29] L. A. Zadeh, Fuzzy Sets Information and Control, Vol 8, pp 338-353 1965.

[30] L. S. Pontragin-V. G. Boltyanskiii-R. v. Gamkrelidze-E. F. Mischenke, The Mathemathical Theory of Optimal Processes, New York John Wiley 1962.

[31] R. E. Kalman, On The General Theory Of Control Systems, Proc. First IFAC Congress, Moscow Butterworths, 1, 481-92 1960.

[32] J. T. Tou, Digital and Sampled Data Control Systems, New York McGraw Hill 1959.

[33] J. R. Ragazzini-G. F. Franklin, Sampled Data Control Systems, New York NcGraw Hill 1958.

[34] E. I. Jury, Sampled Data Control Systems, New York John Wiley 1958.

[35] Y. Z. Tsypkin, Theory of Inpulse Systems Moskow : State Publisher for Phisical Mathemathical Literature 1958.

[36] R. Bellman-I. Glicksberg-O. A. Gross, Some Aspects of Mathemathical Theory of Control Process, Report R-313. Santa Monica. Califf: The RAND Corparation 1958.

[37] R. Bellman, Dynamic Programming, N. J. Princeton Universty Press 1957.

[38] J. R. Ragazzini-L. A. Zadeh, The Analsis of Sampled Data Systems, AIEE Transactions vol: 71, pp 225-34 1952.

[39] R. H. Barker, The Pulse Transfer Function and Its applications to Sampling Servosystems, Proc. IEE vol:99, pp 302-317 1952.

[40] E. I. Jury, Synthesis and Critical Study of Sampled Data Control Systems, AIEE Trans. 75. pt. II. 141-151 1952.

[41] W. K. Linvill, Sampled-Data Control Systems Studied Trough Comperison of Sampling with Amplitude Modulation, AIEE Trans. 70. pt II 1778-1788 1951.

[42] Y. Z. Tsypkin, Theory of Discontoinuous Control Automat I, Telemekh 1950.

[43] C. E. Shanon, Communication in Presence of Noise, Proc. IRE vol:37, pp 10-27 1949.

[44] R. C. Oldenburg-H. Sartorius, The Dynamics of Automatic Control, New York ASME 1949.

[45] W. Hurewicz-H. M. James-N. B. Nicholas-R. S. Philips, Filters and Servo Systems with Pulsed Data, New York McGraw Hill 1947.

[46] L. A. McColl, Fundamental Theory of Servomechanisms, New York D. Von Nostrand 1945.