A COMPARISON OF FUZZY METHODS FOR MODELING

by

Ayşe Çisel Aras

B.S., Electrical Engineering, Yıldız Technical University, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University

2008

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my thesis supervisor, Prof. Okyay Kaynak, for his invaluable guidance and help during this thesis. I owe him immense gratitude for having me shown this research area. His wide knowledge and his logical way of thinking have been provided a good basis for this thesis and have of great value for me. In addition, he was always accessible and ready to help. This thesis would be impossible without him.

I especially would like to thank Prof. Ildar Batyrshin. I am deeply grateful for his help and guidance that are invaluable for me. His direction and suggestions are very precious in the progress of this thesis.

I would like to express my sincere gratitude to the thesis committee members Prof. Kemal Cılız and Prof. Levent Akın. I deeply appreciate their insightful comments on this thesis and their patience.

I would like to thank Assist. Prof. Raşit Bilgin. I am deeply grateful for his help in correcting my English texts. I would like express my sincere thanks to laboratory assistances in mechatronics laboratory, Yeşim Öniz and Erdal Kayacan. They are always helpful and supportive in the progress of this work. I especially thank to Yeşim Öniz, her friendship is invaluable for me.

Last but not least, I want to express my deep thanks to my family for their love and invaluable support. They encourage me and take care of me while I was writing and working on this thesis at late hours.

This thesis has been supported by TUBITAK Grant No: 107E284 and Bogazici University Scientific Research Project Grant No: 08A204.

ABSTRACT

A COMPARISON OF FUZZY METHODS FOR MODELING

Type-1 Fuzzy Logic Controllers (FLCs) have been used in control applications for more than thirty years. However, traditional type-1 FLCs are deficient in dynamical unstructured environments and in many real-time applications that include large amount of uncertainties. Further to this, fuzzy logic systems work cooperatively with many optimization techniques. Conventionally, the antecedent and consequent part of the rules are tuned to obtain minimum error response. However, this situation is not desirable where the expert knowledge about the system is significant. Thus, scientists have started to seek new approaches or develop existing methods.

In literature, there are a number of noteworthy publications on type-1 fuzzy logic with parameterized t-norms. During the optimization process in this fuzzy model, the parameters of the operators and consequent part of the rules are tuned; therefore, the expert knowledge about the system is not lost or distorted. In line with this trend, the most important contribution of this thesis is that parameterized conjunctions are expanded and Constrained Fuzzy Sets (CFSs) with parameterized conjunctions are proposed. By using constrained fuzzy sets with parameterized conjunctions, both the uncertainty and the expert knowledge are taken into account. Thus, the expert knowledge about the system is not lost or distorted and the fuzzy model has more design parameters. This study has the goal of comparing the performance of four different approaches to fuzzy modeling, using parameterized conjunctions, a novel concept named Constrained Fuzzy Sets (CFSs), CFSs with parameterized conjunctions, and unnormalized interval type-2 Takagi Sugeno Kang (IT2 TSK). The theoretical and mathematical backgrounds of the four approaches are briefly described and their performances are compared in approximating a nonlinear function.

ÖZET

BULANIK METODLARININ MODELLEME İÇİN KARŞILAŞTIRILMASI

Tip-1 bulanık mantık kontrolörleri otuz yılı aşkın süredir kontrol alanında kullanılmaktadır; fakat günümüzde bu yöntem değişken çevre koşullarında ve belirsizliklerin çok olduğu gerçek zamanlı sistemlerde yetersiz kalmaktadır. Bulanık mantık sistemleri birçok optimizasyon yöntemiyle birlikte çalışmaktadır. Genellikle minimum hatayı yakalamak için, kuralların öncül ve soncul kısımları adapte edilmektedir; fakat bu yöntemle üyelik fonksiyonlarındaki uzman bilgisi kaybolabilir veya distorsiyona uğrayabilir. Bundan dolayı, bilim insanları yeni yöntemler aramaya veya varolan metdoları geliştirmeye başlamışlardır.

Parametreli t-normlar üzerine literatüre geçmiş çok önemli çalışmaları bulunmaktadır. Bu metodta optimizasyon işlemi uygulanırken, operatörlerin parametreleri ve kuralların soncul kısımlarındaki parametreler adapte edildiği için uzman bilgisinde herhangi bir kayıp ve distorsiyon meydana gelmez. Bu gelişmelere paralel olarak, bu tez çalışmasının en önemli katkısı parametreli t-normların sınırlı bulanık kümler ile birlikte kullanılmasıdır. Bu metodta üyelik fonksiyonlarındaki belirsizlikler ve uzmanlık bilgisi hesaba alınmıştır. Böylece, uzmanlık bilgisinde bozulma ya da kayıp meydana gelmez ve bulanık model daha fazla dizayn parametersine sahiptir. Bu çalışmanın amacı, dört farklı metod kullanarak bulanık sistem modellemesini gerçekleştirmektir. Bu dört farklı metod; parametreli t-normlar, sınırlı bulanık kümeler, sınırlı kümeler ile parametreli t-normlar ve normalleştirilmemiş aralık değerli tip-2 TSK. Bu dört yaklaşımın matematiksel ve teoriksel yapısı kısaca anlatılmış ve performansları lineer olmayan fonksiyon yaklaşımı uygulaması ile karşılaştırılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

A	Fuzzy set A
$\tilde{A}$	Type-2 fuzzy set
a_n^i	Coefficients of the z^i polynomial
α	The maximum value of the membership function of the intersection of two adjacent membership functions
α^*	The threshold value for the intersection of two adjacent membership functions
β	The minimum value of the membership function of the union of fuzzy intervals
c	Center of Gaussian membership function
C_i	Consequent type-1 fuzzy set
d_k	Search direction
$\tilde{C}_i$	Consequent type-2 fuzzy set
F_{obj}	Objective function
H	Positive definite symmetric matrice
$L(x, \lambda)$	Lagrangian function
M	Number of rules
μ	Membership function
$\underline{\mu}$	Lower membership function
$\overline{\mu}$	Upper membership function
$\mu(z)$	Aggregated output
nm	Number of membership functions
N	Number of values that are calculated for the function approximation
o^i	The result of i^{th} IF-THEN rule
p^{in}	The parameter of the G-conjunction operator
$\Psi(x)$	Merit function
R^i	i^{th} rule
r_i	Penalty parameter
S	S-norm

S_c	Maximum s-norm operator
S_p	Algebraic sum
S_b	Bounded sum
S_d	Drastic sum
σ	Width of the Gaussian membership function
σ_l	Sigma value for lower membership function
σ_u	Sigma value for upper membership function
T	T-norm
T_c	Minimum t-norm operator
T_p	Algebraic product
T_b	Bounded product
T_d	Drastic Product
$\overline{\omega}_i$	Upper bound of firing strength for i^{th} IF-THEN rule
$\underline{\omega}_i$	Lower bound of firing strength for i^{th} IF-THEN rule
ω^i	Firing strength of the i^{th} rule
Ω^i	Interval of firing strength of the i^{th} rule
x	The elements of X
X	Universe of discourse
x_{left}	The left most of the input space
x_{right}	The right most of the input space
z^i	Output of IF-THEN rule in Sugeno model
z	Actual output of the system
z_o	Result of defuzzification, crisp value
Z^i	Output of IF-THEN rule in Mamdani model
Z_{COS}	Center of sets type-reduction
z_r	Maximum value of z
z_l	Minimum value of z

BFGS	Broyden-Fletcher-Goldfarb-Shanno
BMFSA	Biomedical Fuzzy Systems Association
CFSs	Constrained Fuzzy Sets

CWW	Computing with Words
EC	Evolutionary Computation
FIS	Fuzzy Inference System
FL	Fuzzy Logic
FLC	Fuzzy Logic Controller
FLSs	Fuzzy Logic Systems
FLSI	Fuzzy Logic Systems Institute Iizuka
FOU	Footprint of Uncertainty
FS	Fuzzy Set
IFSA	International Fuzzy Systems Association
IT2FLSs	Interval Type-2 Fuzzy Logic Systems
IT2 TSK	Interval Type-2 Takagi Sugeno Kang
KKT	Karush-Kuhn-Tucker
LOM	Largest of Maximum
LIFE	Laboratory for International Fuzzy Engineering Research
mf	membership function
mfs	membership functions
ML	Machine Learning
NNs	Neural Networks
PR	Probabilistic Reasoning
QR	Quadratic Programming
RMSE	Root Mean Square Error
SCEI	Science Citation Expanded Index
SOFT	Japan Society for Fuzzy Theory and Systems
SOM	Smallest of Maximum
SQP	Sequential Quadratic Programming
T2FLS	Type-2 Fuzzy Logic Systems
TSK	Takagi Sugeno Kang

1. INTRODUCTION

Conventional computing, which is known as hard computing, in science is based on precise, rigorous, and quantitative analytical models. The precision and certainty in the conventional computing lead to computational burden [1]. Moreover, in many real world applications, most systems encounter many uncertainties and imprecise information due to the dynamical external environment and the inner uncertainties of the systems. Soft computing (SC) tries to imitate human mind and is able to take into consideration these uncertainties and imprecise information. The main methodologies used in soft computing are:

- Fuzzy Logic (FL)
- Neural Networks (NNs)
- Probabilistic Reasoning (PR)
- Evolutionary Computation (EC)
- Machine Learning (ML)

These methods are complementary rather than competitive and can be used together to achieve better results [1]. One of the fundamental constituents of soft computing is fuzzy logic and is the focus of this thesis.

1.1. Historical Review of Fuzzy Logic

Fuzzy logic (FL) was introduced to the scientific arena in 1965 by Prof. Lotfi A. Zadeh, who is a professor of computer science at the University of California, Berkeley, and the first industrial applications appeared in 1970s. The historical progress of the traditional fuzzy logic is given below following the documentation of fuzzyTECH® 5.3 user's manual [2]. One of the early applications of Fuzzy Logic Controller (FLC) was developed by Ebrahim Mamdani in England for controlling a steam engine. In Germany, Hans Jürgen Zimmermann applied FL to decision support systems. Another important milestone is the use of FL for cement kiln control in 1975 in Denmark.

These successful applications in Europe drew the interest of Japanese scientists in the beginning of 1980s. One of the early applications in Japan was on a water treatment plant, realized by Michio Sugeno in 1983. In 1987, fuzzy logic control was also applied to Sendai railways. After these applications FL became prevalent in Japan, and used in many industrial and consumer products, such as washing machines, cameras, etc. Because of the technological advantages and the establishment of many companies, quite a number of fuzzy societies have been founded in Japan. These include:

- International Fuzzy Systems Association (IFSA)
- Japan Society for Fuzzy Theory and Systems (SOFT)
- Biomedical Fuzzy Systems Association (BMFSA)
- Laboratory for International Fuzzy Engineering Research (LIFE)
- Fuzzy Logic Systems Institute Iizuka (FLSI)
- Center for Promotion of Fuzzy Logic at TITech.

The rapid rise of FL in Japan also influenced Europe and a great number of industrial applications of FL started to appear. About the same time, US also responded to the competition between Japan and Europe, and FL was used in new areas, such as decision support systems, hard disk controllers, memory cache, echo cancellation, network routing, and speech recognition.

Traditional FLCs have widely been used in many control applications with great success for more than three decades. In real life applications, systems are confronted with many uncertainties and imprecise information due to the inner and outer dynamics of the systems, such as highly nonlinear systems, incomplete sensory information and noise from external environment. To overcome these uncertainties, Fuzzy Logic Systems (FLSs) work collectively with some optimization techniques that enable the tuning of the system to achieve the desired performance.

Several approaches are proposed in the literature to this end [3], [4]. However, when a system is affected by both inner and outer uncertainties, the traditional type-1 fuzzy logic systems may become inadequate, and the type of optimization that is done

becomes irrelevant. To obtain the desired performance and come up with a minimum error response, some other approaches should be sought. This thesis has the goal of comparing the performance of four different approaches to fuzzy modeling, namely the traditional FLS with parameterized conjunctions, a novel concept named Constrained Fuzzy Sets (CFSs), CFSs with parameterized conjunctions, and unnormalized interval type-2 Takagi Sugeno Kang (IT2 TSK). The historical background of these methods are briefly summarized.

A FS (Fuzzy Sets) has IF-THEN type of rules. During the optimization process, both the antecedent and the consequent part of the rules can be tuned. If the linguistic terms play a major role in the design of fuzzy controller, the tuning of the membership functions may not be desirable as the linguistic interpretation can be lost due to the membership functions moving out of the domain or having large intersections with each other. In applications where the interpretation of the linguistic variable, the expert knowledge, and the rule base are important, the membership functions should therefore not be modified, at least not drastically. Therefore, it is preferable to use the G-conjunction operators. Although many parametric G-conjunction operators have been proposed in the literature, the work reported in [5] and [6] is especially interesting because the operators proposed therein are very simple, due to the fact that they are not required to have the commutativity and associativity properties. In this thesis, Constrained Fuzzy Sets (CFSs) and CFSs with parameterized conjunctions are proposed as other approaches alternative to traditional fuzzy logic.

Another approach alternative to traditional fuzzy logic is type-2 fuzzy logic. In literature, type-2 fuzzy logic was first proposed by Prof. L. A. Zadeh in 1975 as an extension of type-1 fuzzy sets, and the basic mathematical and theoretical foundations were established by him [7]. One of the most important features of type-2 fuzzy sets is the ability to incorporate uncertainties into the membership functions, and this feature makes type-2 fuzzy sets preferable when there exist significant uncertainties.

The progress of type-2 fuzzy logic since 1975 is briefly summarized below and prepared by the help of the report “Type-2 Fuzzy Logic: A Historical View” published

in 2007 [8].

The emergence of fuzzy set theory goes back to the years 1975-1981. Some notable works are those carried out by Mizumoto and Tanaka [9], [10] and Dubois and Prade [11] such as on logical connectives (AND and OR).

By the mid-1980s, type-2 interval fuzzy sets started to be developed by scientists, Gorzalczany [12], Turksen [13], Schwartz [14] and Klir and Folger [15]. Interval-valued fuzzy sets were first evolved by Gorzalczany [12], and the figure of these fuzzy sets give the impression of FOU (Footprint of Uncertainty), which are used today. Bustince's [16] and Liang and Mendel's [17] studies about approximate reasoning gave analogous outcomes. Turksen has significant studies about the logical connectives that are used for type-1 fuzzy sets, which were extended to type-2 interval fuzzy sets.

Karnik and Mendel have studies that constitute the skeleton of the type-2 interval fuzzy sets. In [18], [19], and [20] they developed type-reduction method for type-2 fuzzy sets. This means that from that time the output of the type-2 fuzzy sets can be calculated. Furthermore, they completed the type-2 fuzzy logic mathematical algorithm [18], [20] which enabled the researches to apply type-2 fuzzy logic and the first technological applications were seen in [21], [22], and [23].

In the study of Prof. L. A. Zadeh [24], fuzzy logic is defined as computing with words (CWW). In addition, Mendel [25], [26] and Turksen [27] use type-2 fuzzy logic for CWW.

Another significant establishment is the Representation Theorem by Mendel and John [28] that gives the ability to use type-1 fuzzy arithmetic for the calculations in type-2 fuzzy logic. However, the computational burden of type-2 fuzzy logic was still a problem. Karnik and Mendel developed an iterative algorithm [20] and Wu and Mendel built minimax uncertainty bounds [29], [30] to reduce the computational burden of type-2 interval fuzzy sets. After this significant step, the applications of type-2 fuzzy logic have rapidly increased.

The number of publications from 1988 to today reported at http://www.type2fuzzylogic.org/publications/statistics.php can be seen in Figure 1.1. The numbers include all types of publications.

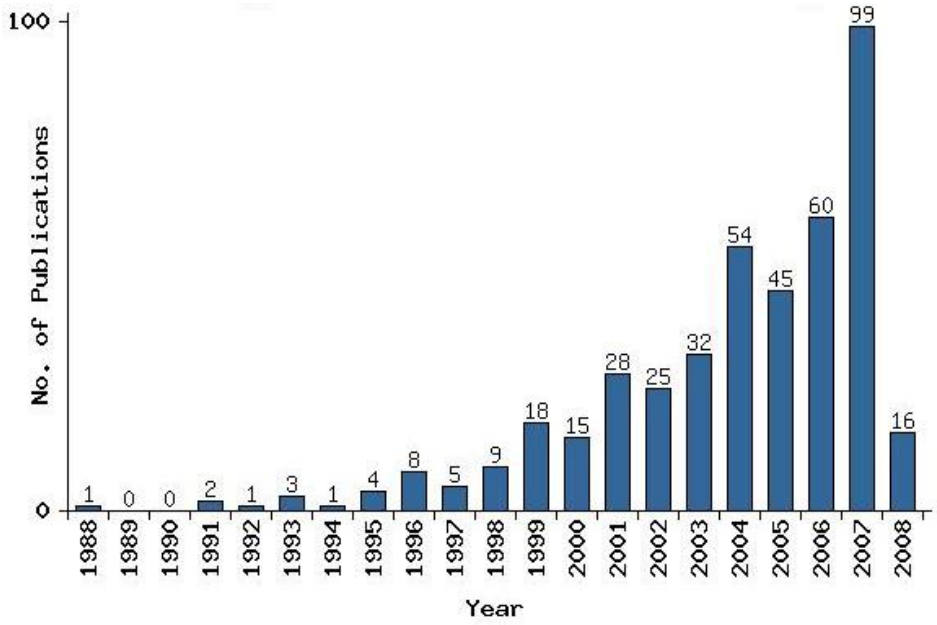

Figure 1.1. Number of publications in each year

A search in Web of Science done by entering "type-2 fuzzy" under the general search tab results in Figures 1.2 and 1.3. The number of publications those in journals cited by SCIE (Science Citation Index Expanded).

Most of the applications in this topic are about in the area of control engineering and medical science. The milestones of the control applications are: Plant Control with type-2 interval fuzzy sets by Melin and Castillo [31], type-2 interval fuzzy logic controller gives better results than type-1 under high uncertainties by Hagras [32], control of complex multi-variable liquid level process with type-2 interval fuzzy controller by Wu and Tan [33], the control of nonautonomous robots in a football game with type-2 interval fuzzy logic controller by Figueroa et al [34].

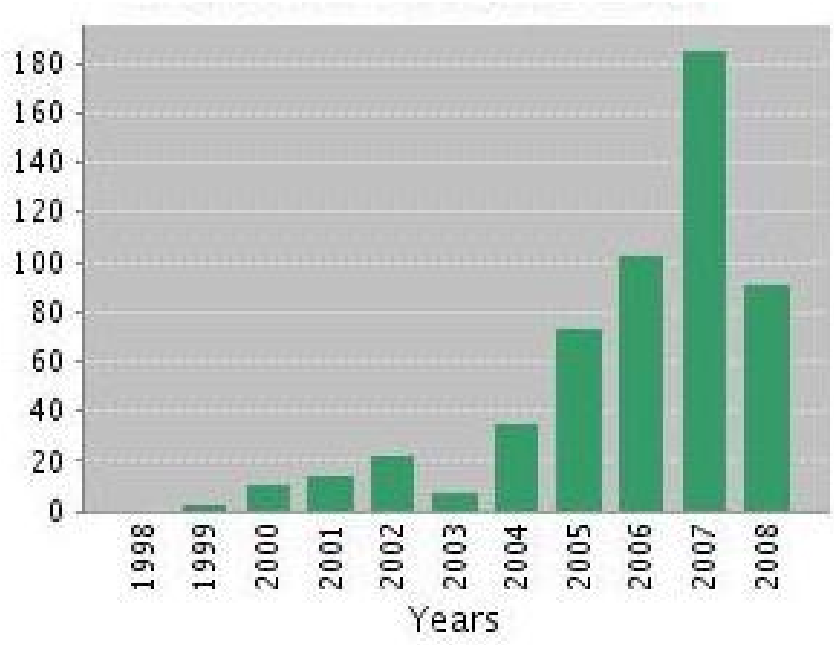

Figure 1.2. Citations in each year

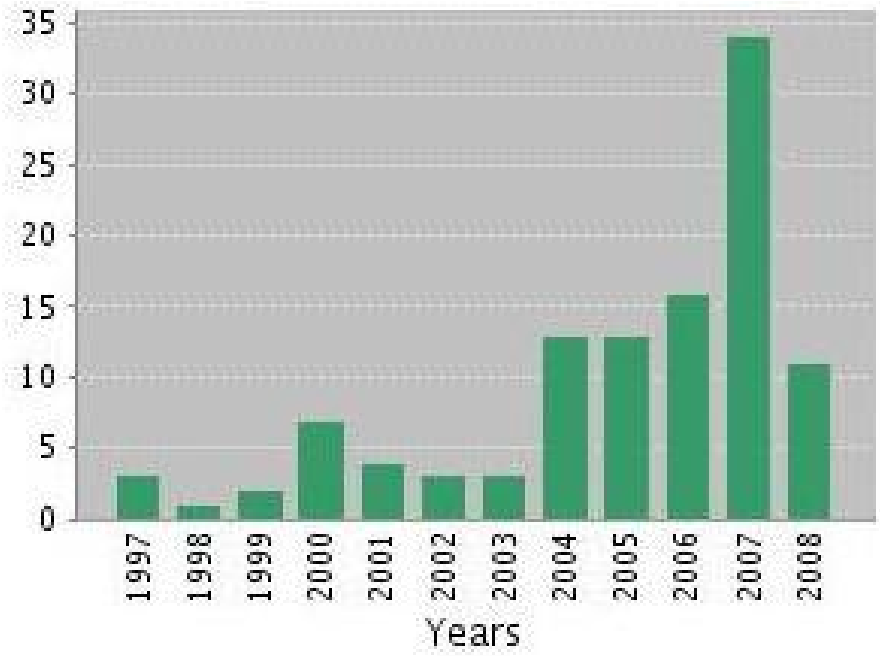

Figure 1.3. Published items in each year

1.2. The Aim of This Study

As it is mentioned earlier, traditional fuzzy logic is not efficient in many applications to problems containing great amount of uncertainty.

The aim of this thesis is a comparative study of fuzzy modelling methods that use different types of fuzzy sets and different methods of fuzzy system optimization. Based on such study, it is proposed to develop and improve alternative methods to traditional fuzzy logic and make these methods preferable in applications where the systems have great amount of uncertainty. The applied methods are:

1. Parameterized Conjunctions
2. Constrained Fuzzy Sets
3. Constrained Fuzzy Sets with Parameterized Conjunctions
4. Unnormalized Interval Type-2 TSK Fuzzy Logic Systems

These four methods are studied and applied into a nonlinear function approximation and performances are compared.

1.3. The Organization of the Thesis

In chapter one, the historical review of type-1 fuzzy logic, parameterized conjunctions, and type-2 fuzzy logic are given. In sequence, the aim of the project and the organization of the thesis are briefly presented.

In chapter two, the mathematical and theoretical backgrounds of fuzzy methods are described in detail. First, type-1 fuzzy logic is described and its usage with the simplest parameterized G-conjunction operators is explained. Second, constrained fuzzy sets are defined and a few examples are given. Then, constrained fuzzy sets with parameterized conjunctions are proposed. Lastly, interval type-2 fuzzy logic and unnormalized interval type-2 TSK fuzzy logic systems are given.

In the third chapter, the optimization method being used is defined. The applications are realized by using the optimization toolbox in MATLAB®.

In the fourth chapter the observed fuzzy relations between CFSs and interval type-2 fuzzy sets are explained. In addition, the advantages and disadvantages of

tuning membership functions are discussed.

In chapter 5, the performances of these four methods are compared in a nonlinear function approximation. This nonlinear function is a sinc function, which has a global maximum and several local maximums.

Consequently, the outcomes of the applications are analyzed and further progress in this area is discussed.

2. THE THEORETICAL AND MATHEMATICAL BACKGROUND OF FUZZY METHODS

As it was mentioned earlier traditional fuzzy logic is not sufficient for highly nonlinear systems and for the situations where uncertainties and imprecise information appear. In this work, alternative methods are examined and developed. The theoretical and mathematical background of these methods are given in the sections below.

2.1. Fuzzy Modeling with Parameterized Conjunctions

The most important feature of fuzzy logic is the ability to define human thinking and interpretation about the system by using various kinds of (e.g., Gaussian, Gbell, Triangular, Trapezoidal) membership functions and IF-THEN type of rules. In fuzzy models, in which the human expert knowledge is the key element of the design of the fuzzy model, tuning the membership functions can result in the loss or distortion of the expert knowledge. In such applications, another type of adaptation can be more appropriate than the adaptation of the membership functions [5], [6].

First of all, when we consider the traditional fuzzy logic systems, there are four main components, which can be described as:

- Fuzzification
- Fuzzy Rule-Base
- Fuzzy Inference Engine
- Defuzzification

The main structure of type-1 fuzzy logic systems is shown in Figure 2.1.

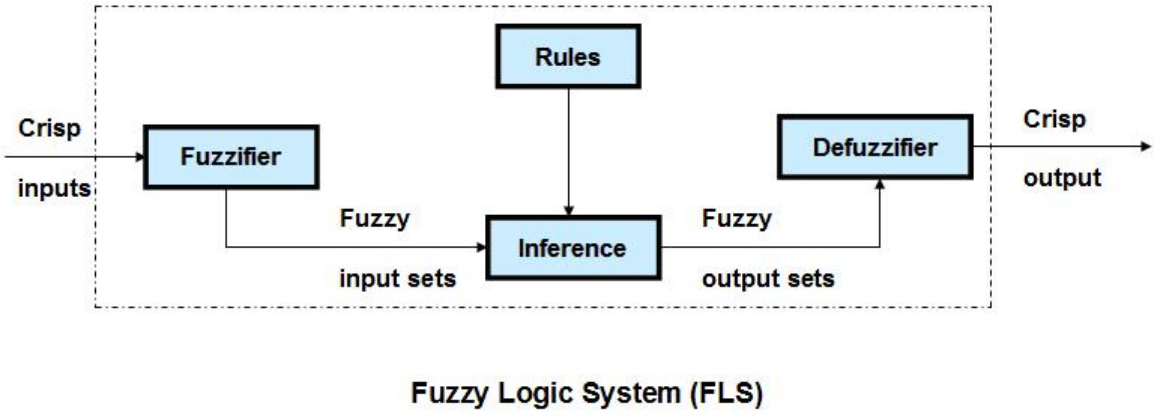

Figure 2.1. Type-1 Fuzzy Logic System (FLS)

2.1.1. Fuzzification

Initially, the crisp inputs are fuzzified by using membership functions. A fuzzy set A is defined in universe of discourse X and is indicated by a membership grade, which takes values in the closed interval 0 and 1 ([0, 1]) [3].

$$A = \{(x, \mu_A(x)) | x \in X\} \tag{2.1}$$

where x are the elements of X, and $\mu_A(x)$ is called the membership function, and indicates the degree of belonging . Every element of X maps to a membership grade taking the values between 0 and 1. The fuzzy sets can be defined by using linguistic labels such as; SMALL, LARGE, MODERATE, YOUNG, SLOW, FAST, etc. These fuzzy sets are specified by membership functions, so that mathematical computations can be performed. There are several types of membership functions. For instance, gaussian, gbell, triangular, trapezoidal, etc.

In the following several types of membership functions are shown [3].

- A Gaussian Membership Function

  A Gaussian membership function (mf) is defined as follows:

$$\text{Gaussian mf}(x, [sigma, center]) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2} \tag{2.2}$$

where c is the center and σ is the width of the membership function. x is the input of the system.

The example of Gaussian mf is shown in Figure 2.2.

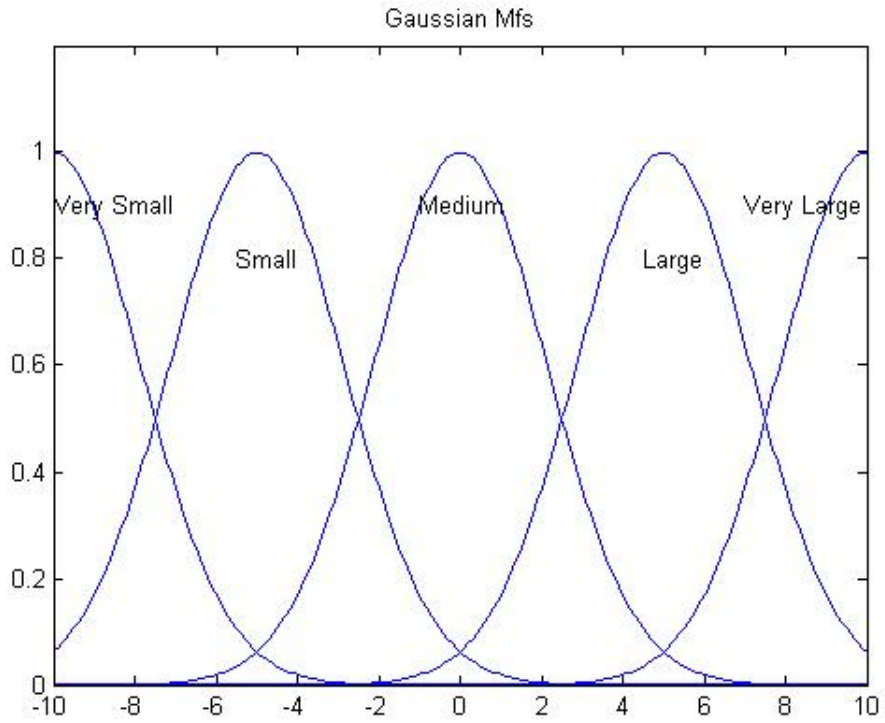

Figure 2.2. Gaussian membership functions with linguistic values "Very Small", "Small", "Medium", "Large", "Very Large"

- A Triangle Membership Function

$$\text{Triangle mf}(x, [a, b, c]) = \begin{cases} 0, & x \leq a. \\ \frac{x-a}{b-a}, & a \leq x \leq b. \\ \frac{c-x}{c-b}, & b \leq x \leq c. \\ 0, & c \leq x. \end{cases} \tag{2.3}$$

where a, b, and c define the corners of the membership function and $a \leq b \leq c$.

The example of triangle mf is shown in Figure 2.3.

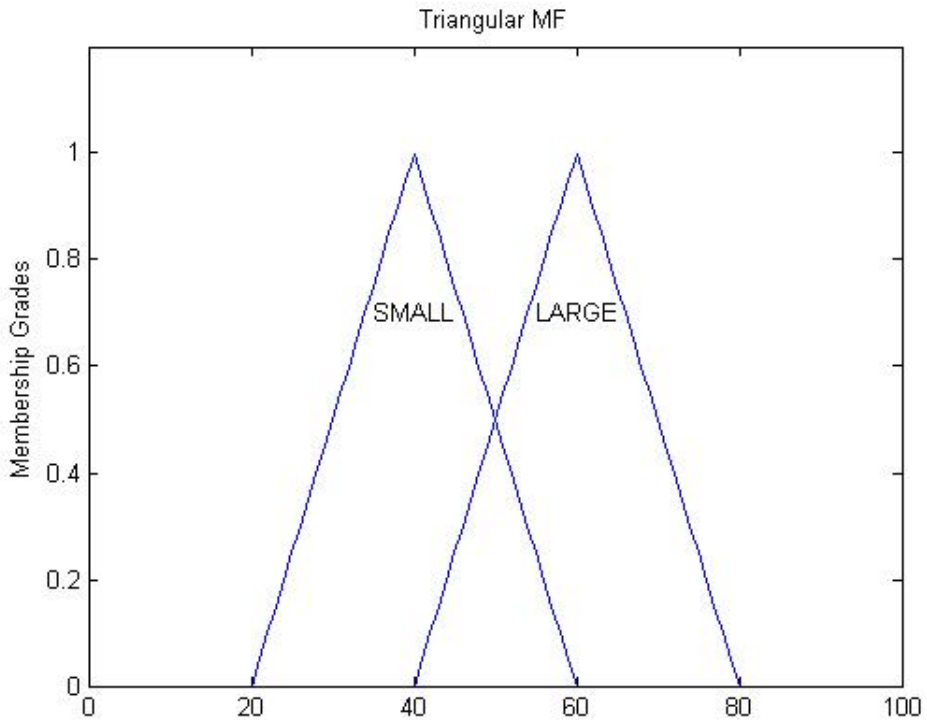

Figure 2.3. Triangular membership functions with linguistic values "Small", "Large"

- A Gbell Membership Function

$$\text{Gbell mf}(x,[a,b,c]) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \tag{2.4}$$

where a determines the width, b determines the slope and c determines the center of the membership function.

The example of Gbell mf is shown in Figure 2.4.

2.1.2. Fuzzy Rule-Base and Fuzzy Inference System (FIS)

Fuzzy Inference Systems are prevalently applied in control engineering and in multidisciplinary areas. FIS involves nonlinear mapping from input data to output data and this nonlinear mapping is performed by using fuzzy if-then rules. Fuzzy Logic Systems are universal approximators and this property enables us to build optimal

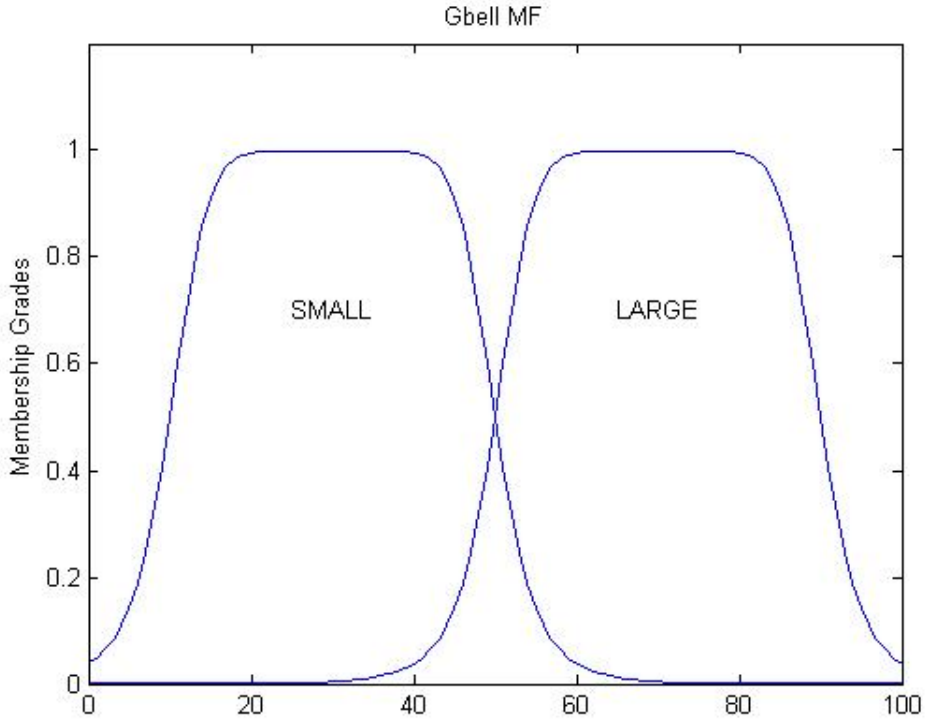

Figure 2.4. Gbell membership functions with linguistic values “Small”, “Large”

fuzzy models [6]. Traditionally, to obtain an optimal fuzzy model, the membership function parameters are tuned.

The IF part of the rule is called antecedent or premise, and the THEN part of the rule is called consequent or conclusion part of the rule. The examples of fuzzy if-then rules that are used in daily life are as follows;

IF temperature is HIGH and humidity is HIGH, THEN fan works fast.

IF the soil is DRY and the temperature is HIGH, THEN open the valve ROUNDLY.

IF X is POSITIVE LARGE and Y is POSITIVE LARGE, THEN Z is POSITIVE LARGE.

The fuzzy models differ by using different consequent membership functions, aggregation and defuzzification methods [6]. There are various types of fuzzy models;

but the most commonly used ones are:

- MAMDANI MODEL
- SUGENO MODEL (Takagi, Sugeno, Kang (TSK))

Mamdani and Sugeno model are the same in the fuzzification block and in the antecedent part of the rules; they only differ in the consequent part of the if-then rules.

Mamdani Model:

$$R^i = \text{IF } X_1 \text{ is } A_{i1} \text{ and ... and } X_n \text{ is } A_{in},$$
$$\text{THEN } Z^i = C_i$$

Sugeno Model:

$$R^i = \text{IF } X_1 \text{ is } A_{i1} \text{ and ... and } X_n \text{ is } A_{in},$$
$$\text{THEN } z^i = a_n^i x_n + a_{n-1}^i x_{n-1} + ... + a_0^i$$

where i (i = 1,2,...,M) indicates the number of rule. In these rule structures, A_{in} and C_i are the antecedent and consequent fuzzy sets, respectively. Z^i is the output of the Mamdani model and is a fuzzy set. z^i is the output of the Sugeno model, which is a first order polynomial at the consequent part of the rule structure. X_n is the input variable and n is the number of input variable.

As it is seen, both in Mamdani and Sugeno model the antecedent parts of the rules are the same, which contains antecedent fuzzy sets A_{in}'s, and inputs X_n's. They differ in the consequent part of the rules. In Mamdani Model, the consequent part is a fuzzy set, C_i. On the other hand, in Sugeno Model, the consequent is a real valued function $z^i = a_n^i x_n + a_{n-1}^i x_{n-1} + ... + a_0^i$. Depending on the degree of the polynomial, the Sugeno model is called as zero order Sugeno model, first order Sugeno model, and

so on [6], [3]. In this work, first order type-1 Takagi Sugeno Kang (TSK) fuzzy logic systems (FLSs) are considered.

The antecedent part of the rules are combined with the fuzzy operators such as AND, OR, NOT. These operators determine the firing strength (ω^i) of the rules.

Now let's consider the traditional type-1 fuzzy logic operators and assume A_{in} are fuzzy sets where i indicates the number of rules and n indicates the number of antecedent fuzzy sets.

2.1.2.1. Intersection of Fuzzy Sets (Conjunction Operators). The intersection is called as AND operator and is basically used for finding the minimum of the antecedent membership functions [3]:

$$\omega^i = min(\mu_{A_{i1}}(x_1), \mu_{A_{i2}}(x_2)) \tag{2.5}$$

Generally, instead of minimum, one can use any t-norm. T-norm is defined as a function T: [0,1] x [0,1] $\rightarrow$ [0,1] satisfying the four conditions monotonicity, commutativity, associativity, and boundary [3], [6]:

Monotonicity:

$$T(x, y) \leq T(u, v) \text{ if } x \leq u \text{ and } y \leq v \tag{2.6}$$

Commutativity:

$$T(x, y) = T(y, x) \tag{2.7}$$

Associativity:

$$T(x, T(y, z)) = T(T(x, y), z) \tag{2.8}$$

Boundary:

$$\begin{aligned} T(0, 0) &= 0, \\ T(1, x) = T(x, 1) &= x \end{aligned} \tag{2.9}$$

In literature, the most commonly used t-norm operations are minimum, algebraic product, bounded product, and drastic product that are calculated as follows, respectively [3], [6]:

$$T_c(x, y) = min(x, y) \tag{2.10}$$

$$T_p(x, y) = xy \tag{2.11}$$

$$T_b(x, y) = max\{0, (x + y - 1)\} \tag{2.12}$$

$$T_d(x, y) = \begin{cases} x, & \text{if } y = 1 \\ y, & \text{if } x = 1 \\ 0, & \text{if } x, y < 1 \end{cases} \tag{2.13}$$

The corresponding surfaces of t-norms are given in Figure 2.5 where $0 \leq x, y \leq 1$ [3].

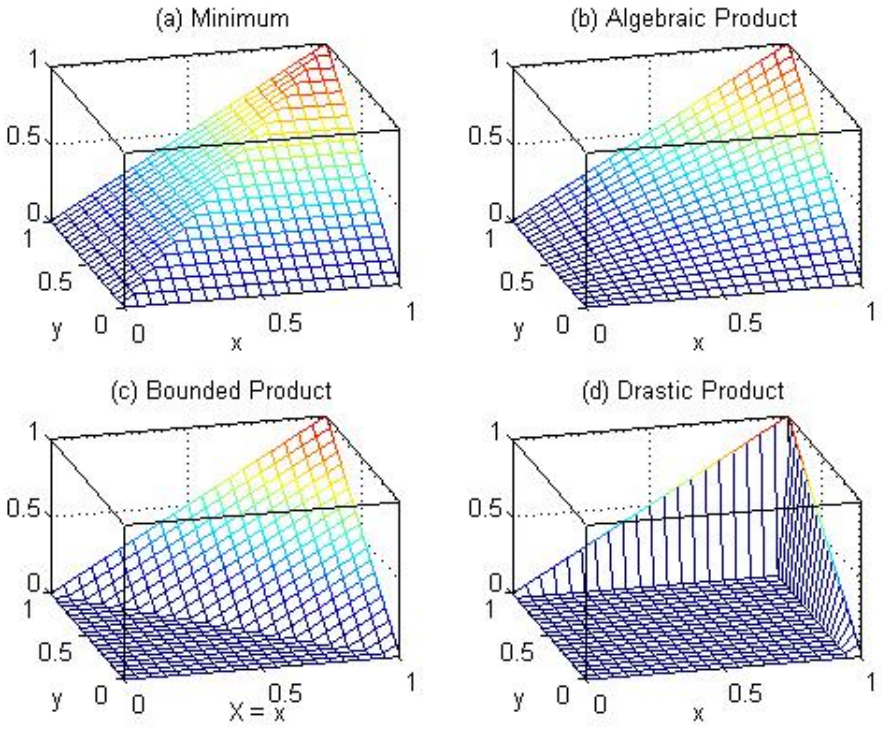

Figure 2.5. The corresponding surfaces of t-norms (a) Minimum, (b) Algebraic Product, (c) Bounded Product, (d) Drastic Product

2.1.2.2. Union of Fuzzy Sets (Disjunction). Union (disjunction) of the fuzzy sets is defined by OR operator and is calculated usually by finding the maximum of the antecedent membership functions [3]:

$$\omega^i = max(\mu_{A_{i1}}(x_1), \mu_{A_{i2}}(x_2)) \tag{2.14}$$

Generally, instead of maximum, one can use any s-norm. S-norm is defined as a function S:[0,1] x [0,1] → [0,1] satisfying the four conditions monotonicity, commutativity, associativity, and boundary [3], [6]:

Monotonicity:

$$S(x, y) \leq S(u, v) \text{ if } x \leq u \text{ and } y \leq v. \tag{2.15}$$

Commutativity:

$$S(x, y) = S(y, x) \tag{2.16}$$

Associativity:

$$S(x, S(y, z)) = S(S(x, y), z) \tag{2.17}$$

Boundary:

$$\begin{aligned} S(1, 1) &= 1, \\ S(x, 0) = S(0, x) &= x \end{aligned} \tag{2.18}$$

In literature, the most commonly used S-norms are maximum, algebraic sum, bounded sum, and drastic sum that are respectively calculated as follows [3]:

$$S_c(x, y) = max(x, y) \tag{2.19}$$

$$S_p(x, y) = x + y - xy \tag{2.20}$$

$$S_b(x, y) = min\{1, (x + y)\} \tag{2.21}$$

$$S_d(x, y) = \begin{cases} x, & \text{if } y = 0 \\ y, & \text{if } x = 0 \\ 1, & \text{if } x, y > 0 \end{cases} \tag{2.22}$$

The corresponding surfaces of S-norms are given in Figure 2.6 [3].

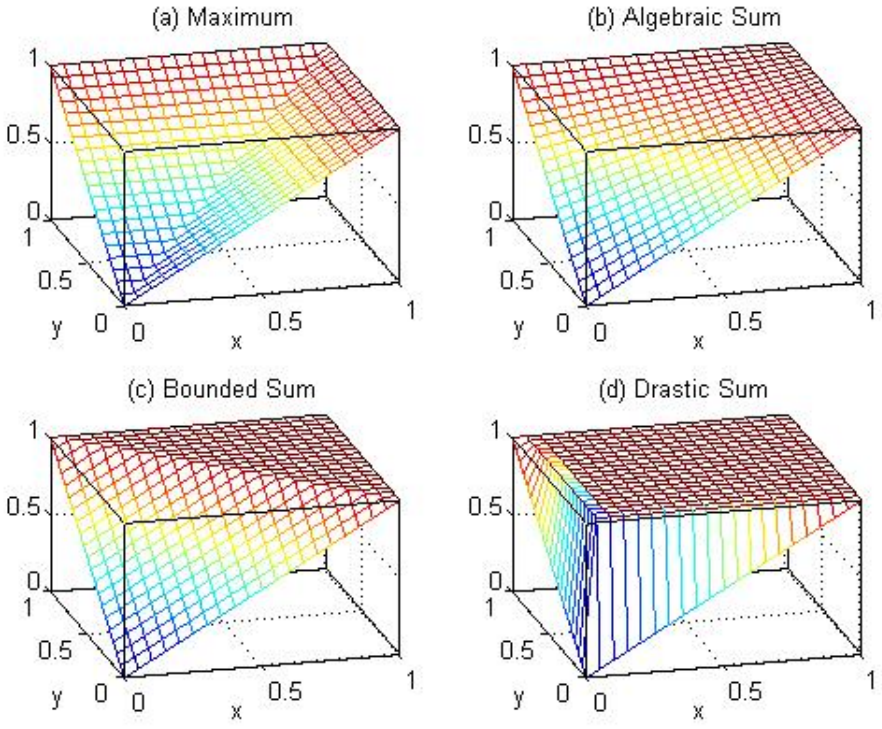

Figure 2.6. The corresponding surfaces for S-norms (a) Minimum, (b) Algebraic Sum, (c) Bounded Sum, (d) Drastic Sum

2.1.2.3. NOT (Complement) Operator. NOT operator is basically used for taking the complement of the fuzzy set A [3]:

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \tag{2.23}$$

2.1.2.4. Parameterized Conjunctions. In type-1 fuzzy logic, the conventional approach is to tune the parameters of membership functions to obtain minimum error response. This kind of adaptation results in the loss or distortion of the expert knowledge about the system and is undesirable in applications where the expert knowledge is vital. Therefore, parametric conjunction or disjunction operators are considered to be used as fuzzy operators, thus the parameters of the operators can be tuned with regarding the expert knowledge of the system. However, these operations are complicated to be

realized in hardware implementations and in optimization [6]. In [5], [6] the authors proposed to tune some simple generalized parametric conjunction operations to obtain an optimal fuzzy model. A generalized conjuction (G-conjunction) operation is defined in [6] as a function T: [0,1] x [0,1] → [0,1] satisfying the properties of binary conjunction operation:

$$T(0,0) = T(0,1) = T(1,0) = 0,$$
$$T(1,1) = 0 \tag{2.24}$$

and monotonicity condition on [0,1]:

$$T(x,y) \leq T(u,v) \text{ if } x \leq u \text{ and } y \leq v. \tag{2.25}$$

It is to be noted that such operators are not required to have the associativity and the commutativity properties and therefore some very simple parameterized operations can be derived. The simplest G-conjunction operations proposed in [6] are the following:

$$T(x,y) = x^p y^q \tag{2.26}$$

$$T(x,y) = min(x^p, y^q) \tag{2.27}$$

where $p, q \geq 0$. The corresponding surfaces for simplest G-conjunction operators are given in Figure 2.7 [6].

In this study, for the example of function approximation by tuning the parameters of operators, first order type-1 Takagi Sugeno Kang (TSK) fuzzy logic systems (FLSs) are considered. The IF-THEN rule of the TSK FLS is described as:

$$R^i = \text{IF } X_1 \text{ is } A_{i1} \text{ and ... and } X_n \text{ is } A_{in},$$
$$\text{THEN } z^i = a_n^i x_n + a_{n-1}^i x_{n-1} + ... + a_0^i$$

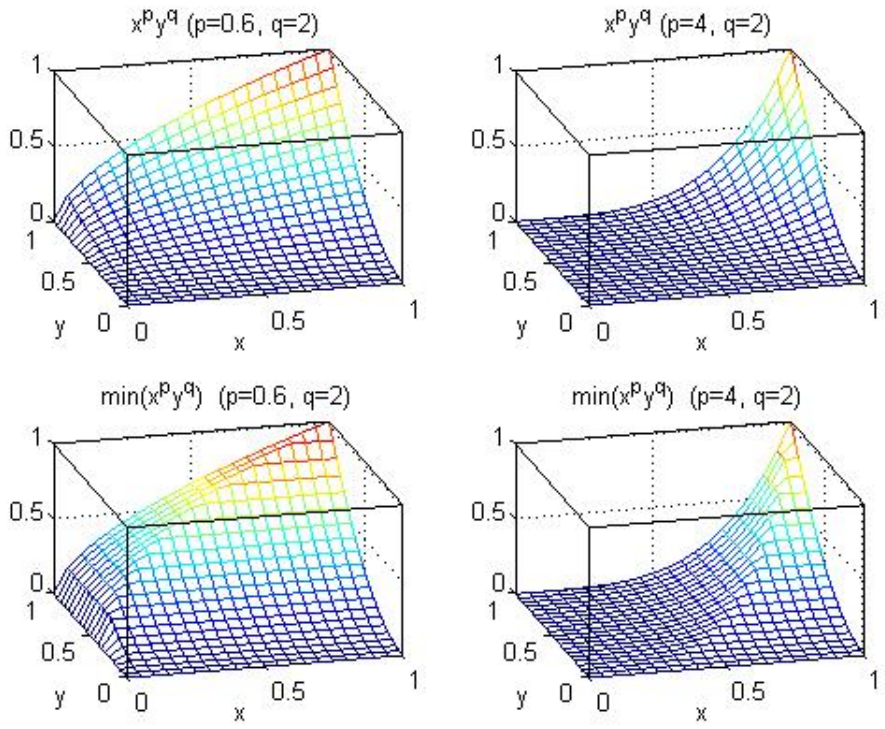

Figure 2.7. The corresponding surfaces for parameterized conjunctions T(x,y)=$x^p y^q$ and T(x,y)=min(x^p, y^q)

where i (i = 1, 2,...,M) indicates the number of rules and n is the number of the antecedent parameters. In this rule structure, X_n's (i=1,...,n) are the inputs of the system, A_{in}'s (i=1,...,n) are the fuzzy sets, and z^i is the output of the each rule. a_n^i's are the coefficients of the consequent part of the rule, which is a first order polynomial.

The firing strengths of the rules are calculated by using one of the G-conjunction operations in Equation (2.26) and (2.27). In the simulations, the first G- conjunction operator is used as AND operator and defined as follows:

$$\begin{aligned} \omega^i &= T(\mu_{A_{i1}}(x_1), \cdots, \mu_{A_{in}}(x_n)) \\ &= \mu_{A_{i1}}(x_1)^{p_{i1}} \cdots \mu_{A_{in}}(x_n)^{p_{in}} \end{aligned} \tag{2.28}$$

where i (i = 1, 2, ..., M) indicates the number of rules. ω^i is the firing strength of each rule. $\mu_{A_{in}}(x_n)$ is the membership function value of the fuzzy set A_{in} and p_{in} is the parameter of the G-conjunction operator.

After computing the firing strengths (the degree specified by antecedent membership functions), the implication method is applied to find out the result of each rule. The most commonly used implication methods are product and minimum operations.

$$\text{product} \ \rightarrow o^i = \omega^i z^i \tag{2.29}$$

$$\text{minimum} \ \rightarrow o^i = min(\omega^i, z^i) \tag{2.30}$$

In this study, to find the result of each rule, the product implication method, which involves multiplying the firing strength with the consequent part of the rules, is used in this application.

$$o^i = \omega^i z^i \tag{2.31}$$

2.1.3. Weighted Average Calculation in TSK Model

In TSK FLS, there is no need to defuzzify the results of the rules; since they are already a crisp output [3]. Their weighted average is calculated as:

$$z = \frac{\sum_{i=1}^{M} \omega^i z^i}{\sum_{i=1}^{M} \omega^i} \tag{2.32}$$

M is the number of rules (i = 1, 2, ..., M) and z is the actual output of the system.

2.1.4. Mamdani Fuzzy Inference and Defuzzification Methods

As it was stated earlier, the antecedent parts of the rules are same for both Mamdani and Sugeno model. However, in Mamdani model "compositional rule of inference" is carried out, and can be defined as max-min composition of fuzzy sets [3]. If A_{in} are the antecedent membership functions and C_i is the consequent membership

function, the max-min composition is calculated as:

$$\text{max-min composition} = max(min(A_{i1}, ..., A_{in}, C_i)) \tag{2.33}$$

In addition, compositional rule of inference can be used as the combination of max and product, for example, t-norm and t-conorm operators [3].

After finding each result of the rule, these results are aggregated by using one of the aggregation methods; such as maximum, sum, probabilistic or [35].

Each result of the rule that is calculated by implication method is a fuzzy set. Defuzzification method converts the fuzzy sets into a crisp value. First of all, the qualified fuzzy sets are aggregated, and then by using appropriate defuzzification method the crisp output is derived [3].

In Mamdani model, there are five types of defuzzification methods;

1. Center of Area
2. Bisector of Area
3. Small of Maximum
4. Middle of Maximum
5. Large of Maximum

2.1.4.1. Center of Area (Centroid) Defuzzification Method. Center of area method is the most commonly used defuzzification method in Mamdani models. In this method, the center of gravitiy of the aggregated output membership function is found and is calculated as follows [3]:

$$z_o = \frac{\int_z \mu(z) z dz}{\int_z \mu(z) dz} \tag{2.34}$$

where z_o is the centroid of the area, a crisp value, z is the output variable, and $\mu(z)$ indicates the aggregated output of the membership functions. An example of centroid method is shown in Figure 2.8.

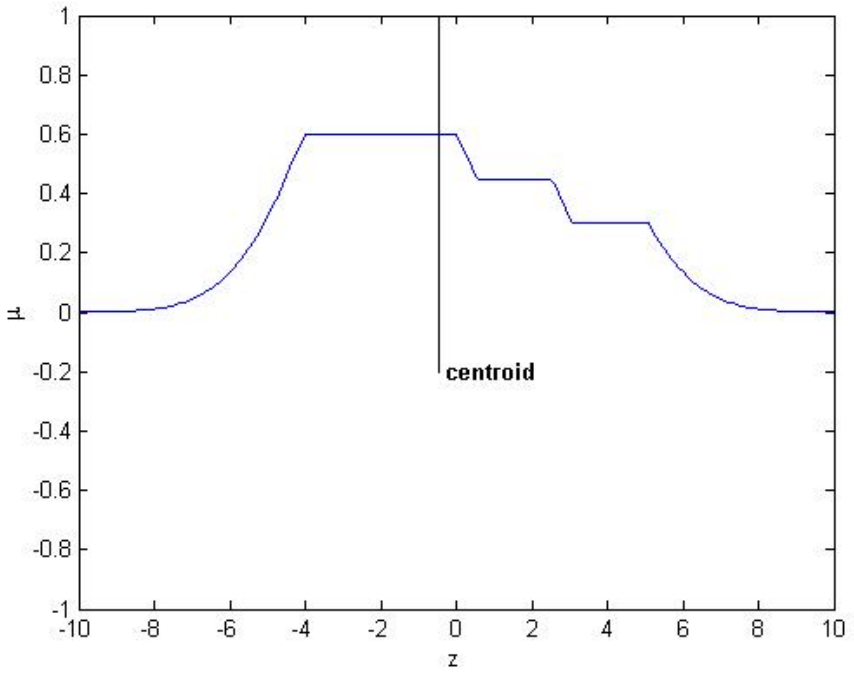

Figure 2.8. Center of area (Centroid) defuzzification method

2.1.4.2. Bisector of Area Defuzzification Method. In bisector of area method the vertical line divides the aggregated region in two equal areas, and z_o satisfies the following equation [3]:

$$\int_{\alpha}^{z_o} \mu(z)dz = \int_{z_o}^{\beta} \mu(z)dz \tag{2.35}$$

where $\alpha = min\{z|z \in Z\}$ and $\beta = max\{z|z \in Z\}$. An example of bisector of area method is shown in Figure 2.9.

2.1.4.3. Smallest of Maximum (SOM) Defuzzification Method. SOM, z_o, is the smallest value where value z takes on maximum [3]. An example of SOM defuzzification method is shown in Figure 2.10.

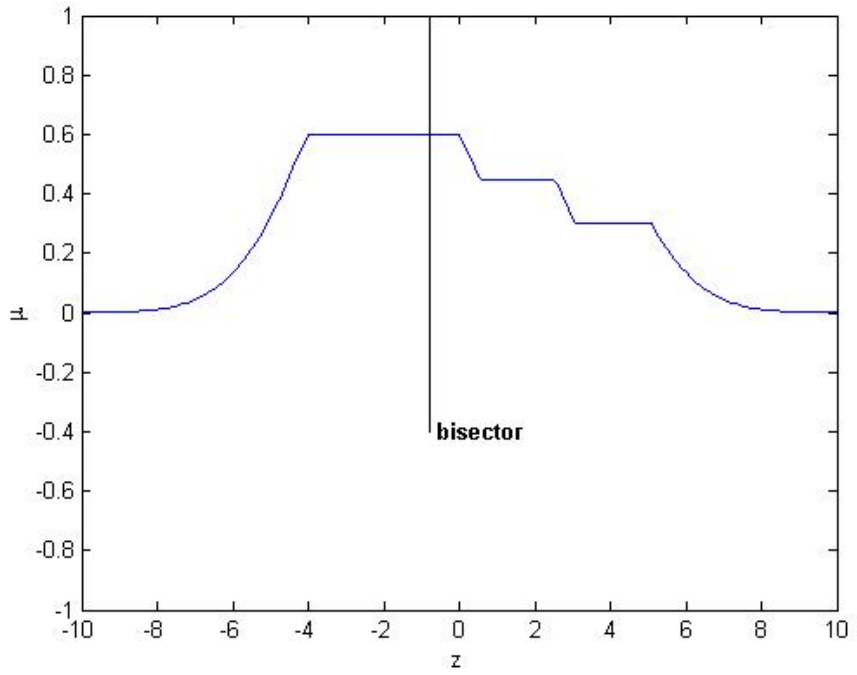

Figure 2.9. Bisector of area defuzzification method

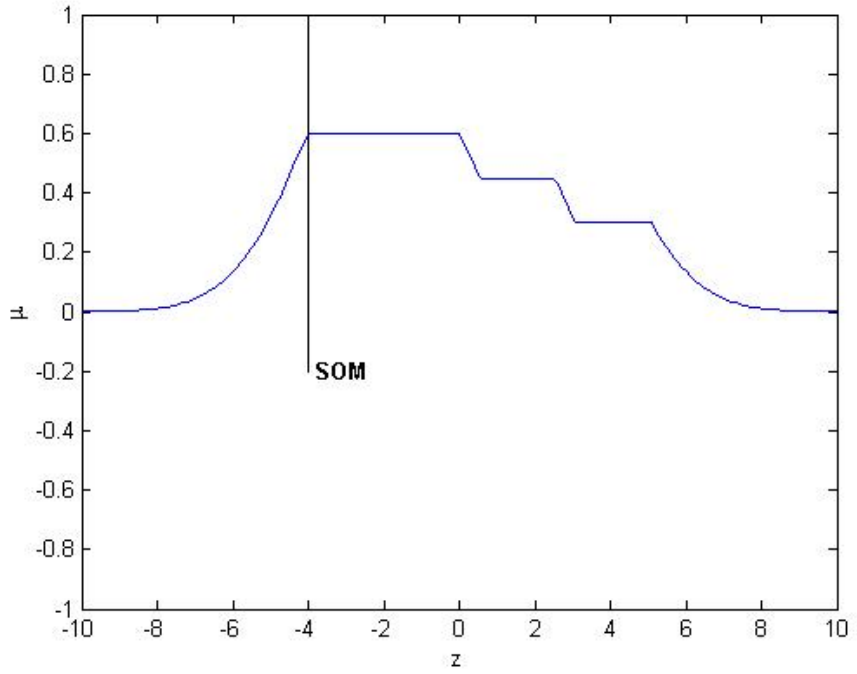

Figure 2.10. Smallest of Maximum (SOM) defuzzification method

2.1.4.4. Largest of Maximum (LOM) Defuzzification Method. The largest of the maximum, z_o, is the largest corresponding value to the largest z value [3]. An example of LOM defuzzification method is given in Figure 2.11.

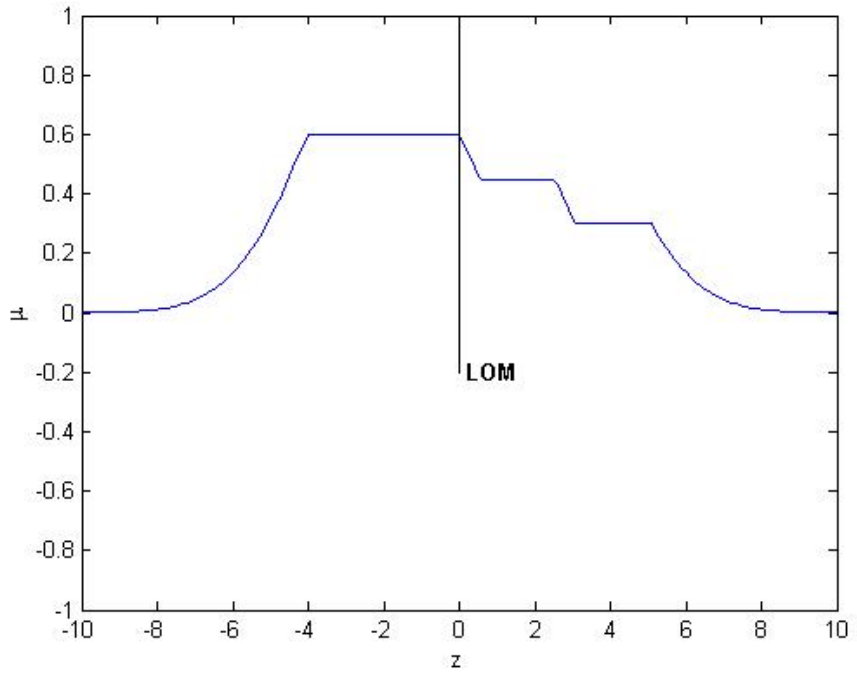

Figure 2.11. Largest of Maximum (LOM) defuzzification method

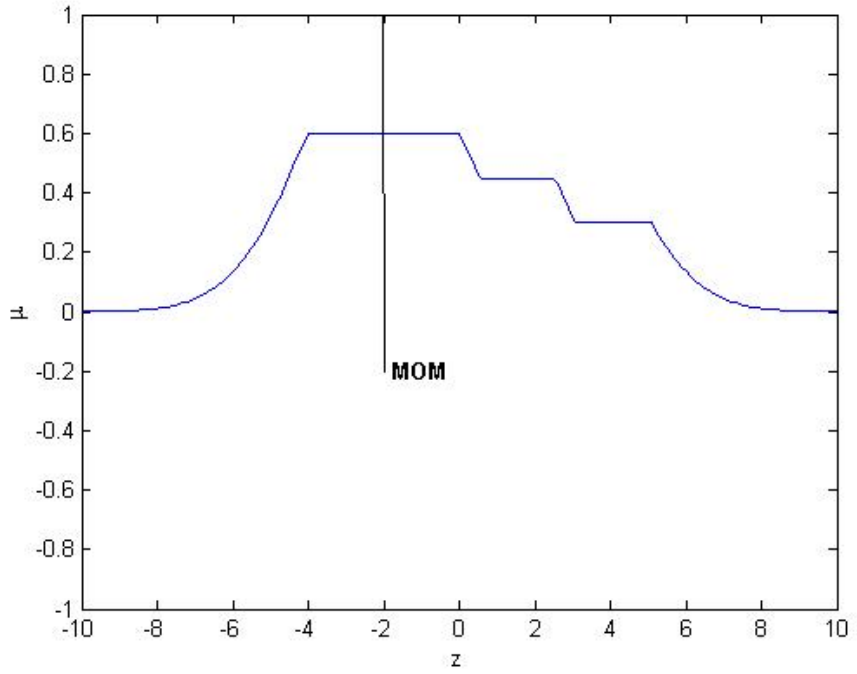

Figure 2.12. Mean of Maximum (MOM) defuzzification method

2.1.4.5. Mean of Maximum (MOM) Defuzzification Method. Mean of maximum is the mean value of the SOM and LOM [3]. An example of mean of maximum method is shown in Figure 2.12.

For better understanding, the defuzzification methods described above are shown in Figure 2.13.

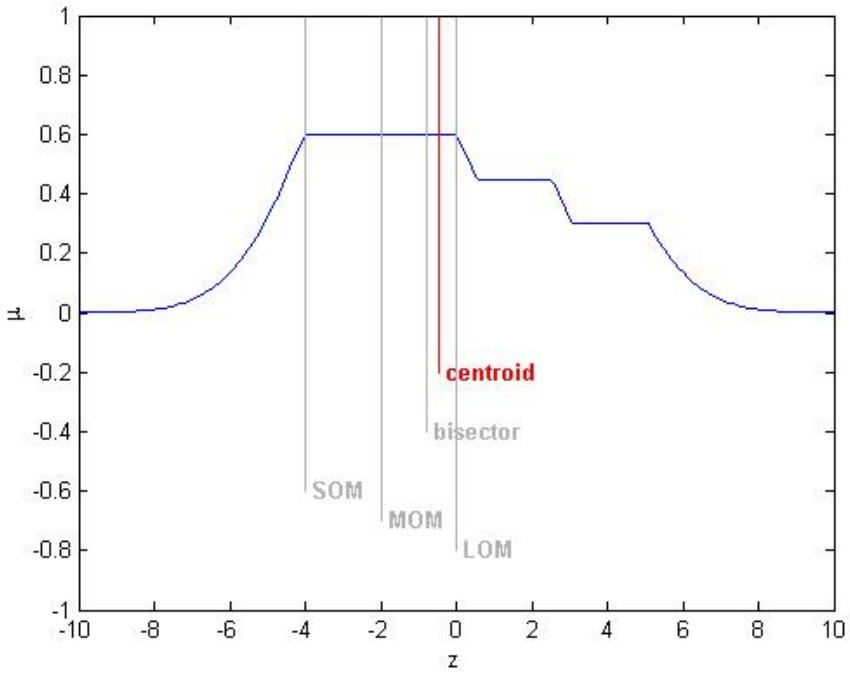

Figure 2.13. Five defuzzification methods

2.2. Constrained Fuzzy Sets (CFSs)

The traditional type-1 fuzzy sets can turn out to be insufficient in handling the uncertainties that are caused by the external environment, imprecise sensory information, or highly nonlinear systems. In order to reach the desired performance levels, one common approach used is the tuning of the membership functions. However, as stated earlier, this may result in drastic changes in the membership functions and can prevent some constraints being used, reflecting the linguistic interpretation of these membership functions by the expert. The fuzzy sets that have these kinds of constraints are called Constrained Fuzzy Sets (CFSs) [36]. Making use of the properties of membership functions various kinds of constraints can be defined. Some of the membership function properties are:

1. The membership functions are distributed on the input domain.

2. The $\alpha - \beta$ fuzzy partition is defined as follows [36]:

$$\sup_{j \neq k} (\sup_{x \in X}((A_j \bigcap A_k)(x))) = \alpha \tag{2.36}$$

and

$$\inf_{x \in X}((\bigcup_{k=1}^{m} A_k)(x)) = \beta \tag{2.37}$$

where $A_j < A_k$. α is the maximum value of the membership function of the intersection of two adjacent membership functions, and β is the minimum value of the membership function of the union of fuzzy intervals ($0 \leq \alpha < 1$ and $0 < \beta \leq 1$). α^* is a threshold value for the intersection of two adjacent membership functions. $\alpha \leq \alpha^*$, the intersection of adjacent membership functions (mfs) should not exceed the threshold value where $\alpha^* \in [0, 1]$ and $\alpha^* < 1$. $\beta \geq \beta^*$, the union of mfs should cover the input domain at least on level β^*, where $\beta^* \in [0, 1]$ and $\beta^* > 0$ [36].

3. The membership functions are normal, continuous, convex, and α-cuts are closed crisp intervals [36].

In real life, systems are usually encountered with disturbances and noise resulting from the external environment. In this study, to represent the uncertainties in the expert knowledge of the systems, the interval type constrained fuzzy sets will be used.

In this study, during the simulation studies with CFSs, a first order type-1 TSK FLS and the same rule structure as the previous method is used. The firing strengths of the each rule are calculated by means of the product t-norm as follows:

$$\begin{aligned} \omega^i &= T(\mu_{A_{i1}}(x_1), \cdots, \mu_{A_{in}}(x_n)) \\ &= \mu_{A_{i1}}(x_1) \cdots \mu_{A_{in}}(x_n) \end{aligned} \tag{2.38}$$

$\mu_{A_{in}}(x_n)$ is the membership function value of the fuzzy set A_{in}. The implication method used and the actual output of the system is calculated as before, i.e. as in (2.31) and (2.32) respectively.

2.3. Constrained Fuzzy Sets (CFSs) with Parameterized Conjunctions

In this approach, in addition to the use of Constrained Fuzzy Sets, the G-conjunction operators are used to calculate the firing strength of the fuzzy model. The rule structure and the algorithm are the same as the fuzzy modeling with parameterized G-conjunctions; but the fundamental difference is that both the membership function parameters and the parameters of the G-conjunction operators are tuned. The membership functions of the system are adjusted within the bounds of the constraints; so that the expert knowledge about the system is not lost. There are various design approaches for CFSs with parameterized conjunctions:

1. First, tune the membership functions and then tune the parameters of G- conjunction operators.
2. Tune the membership functions and parameters of G conjunction operators simultaneously.
3. In each tuning step, tune the membership functions and then tune the parameters of G-conjunction operators.

In this study, the second approach is used. The design parameters of the fuzzy model are tuned together by using one of the nonlinear programming methods.

2.4. Interval Type-2 Fuzzy Logic Systems (IT2FLSs)

Traditional type-1 fuzzy logic controllers have been widely used in many fields for more than thirty years. However, in many real time applications, systems encounter large degrees of uncertainties. Traditional type-1 fuzzy sets lack the ability to define these uncertainties in the membership functions since traditional type-1 fuzzy membership functions are precise. Scientists have started to seek a new approach for fuzzy

systems to handle such uncertainties. In 1975, type-2 fuzzy logic was first proposed by Prof. L. A. Zadeh as an extension of type-1 fuzzy sets, and the basic mathematical and theoretical foundations were established by him. One of the most important features of type-2 fuzzy sets is the ability to incorporate uncertainties in the membership functions, and this feature makes type-2 fuzzy sets preferable when there exists significant uncertainties. The uncertainties that are faced in real world applications could be listed as follows [17], [37]:

- The meaning of the words that are used in antecedent and consequent part of the rules can mean different things to different people.
- The input measurements of the system has uncertainties according to the environmental conditions (such as wind, rain, humidity etc.), and sensors that are effected by high noise levels from various sources.
- Using noisy training data stimulates uncertainties.
- Uncertainties in consequents of the system may occur due to the change of actuator characteristics.
- The uncertainties in antecedent and consequent arise due to the changing operation conditions.

Type-2 fuzzy logic controllers are able to overcome these uncertainties; because type-2 fuzzy logic sets have a fuzzy-fuzzy relation in membership functions. If the uncertainties are ignored, the type-2 fuzzy sets will be transformed to type-1 fuzzy sets [38].

A typical example of type-1 fuzzy sets is seen in Figure 2.14. A precise value corresponds to an input value. The uncertainties that the systems encounter, are not taken into consideration, which is the reason that the type-1 fuzzy sets are not good enough in highly nonlinear systems and real life applications. As it is seen in Figure 2.15, when the type-1 membership functions are blurred to the left and right, the FOU (Footprint of Uncertainty), 2-D diagram, is obtained. It is clear that a closed interval value corresponds to an input value, and furthermore, this interval has a secondary membership grade, which constitutes the third dimension of the type-2 membership functions. For interval type-2 membership functions, this value is either zero or one,

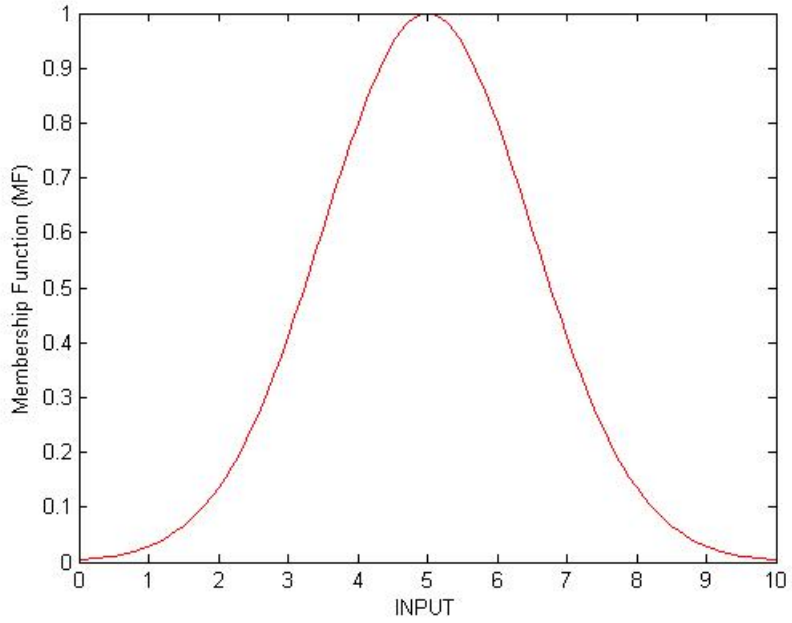

Figure 2.14. A Gaussian type-1 membership function (mf)

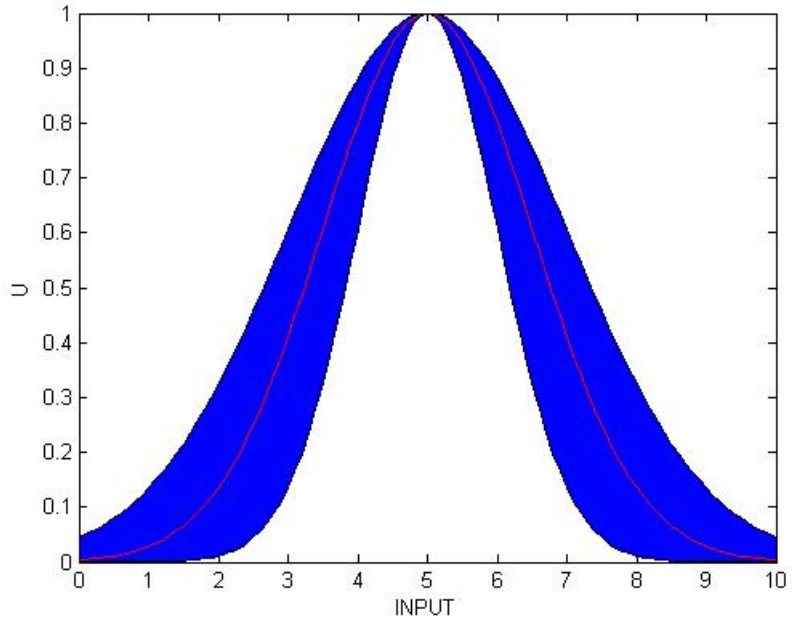

Figure 2.15. Diagram of Footprint of Uncertainty (FOU)

and its example is seen in Figure 2.16 [17].

In this thesis, interval type-2 fuzzy sets are used because of its simplicity and ease of use.

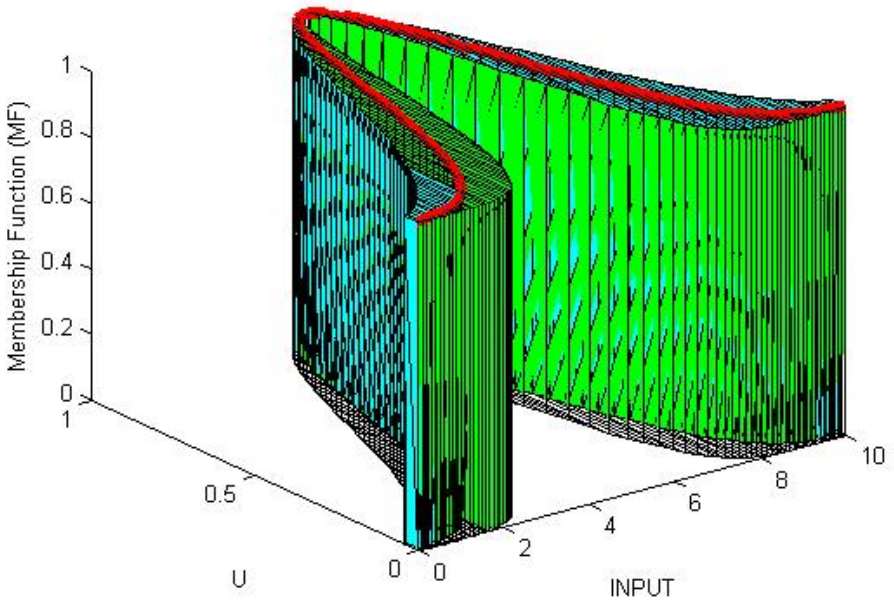

Figure 2.16. Interval Type-2 Gaussian membership function with uncertain standard deviation

A type-2 fuzzy set and an interval type-2 fuzzy set are defined as follows, respectively [17], [28]:

$$\tilde{A} = \int_{x \in X} \int_{u \in J_x \subseteq [0,1]} \mu_{\tilde{A}}(x,u)/(x,u) \tag{2.39}$$

The secondary membership grade for interval type-2 membership functions is either zero or one, and an interval type-2 fuzzy set is defined as follows:

$$\tilde{A} = \int_{x \in X} \int_{u \in J_x \subseteq [0,1]} 1/(x,u) \tag{2.40}$$

x is called the primary variable. X is the input domain. u is the secondary variable. $\tilde{A}$ is the type-2 fuzzy set.

J_x are the primary membership functions. The union of all primary membership functions constitutes the footprint of uncertainty (FOU) [17], [28]. This region is

bounded with the lower and upper membership functions. These membership functions are type-1, which makes the use of type-1 fuzzy arithmetic in calculations for type-2 fuzzy sets [38].

The structure of a type-2 fuzzy logic system (T2FLS) is shown in Figure 2.17.

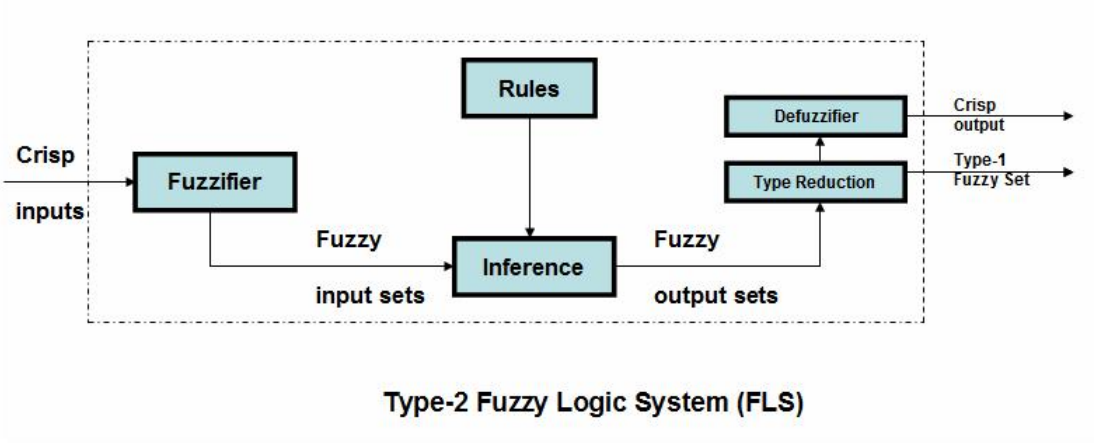

Figure 2.17. Block Diagram of Type-2 Fuzzy Logic Systems (T2FLSs)

As it can be seen, the block diagram of T2FLS is similar to type-1 fuzzy logic system. The main difference is in the last block. This block is composed of two steps: type reduction and defuzzification. The main components of type-2 FLS are:

- Fuzzification
- Fuzzy Inference System and Rule Base
- Type-Reduction
- Defuzzification

These four main components are briefly described in the next subsection.

2.4.1. Fuzzification

First of all, the inputs are fuzzified in the fuzzification block. If a fuzzy logic system contains even one type-2 fuzzy set, this system is called type-2 fuzzy logic system. Some examples of interval type-2 fuzzy sets are given in the following.

2.4.1.1. Gaussian Primary Membership Function with Uncertain Standard Deviation. A Gaussian primary membership function with uncertain standard deviation is specified by three parameters c, σ_u, σ_l where σ_u is the sigma value for upper membership function, σ_l is the sigma value for lower membership function and c indicates the center value. The mathematical representation is as follows [17]:

$$\mu_{\tilde{A}_{in}}(x) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2} \tag{2.41}$$

where $\sigma \in [\sigma_l, \sigma_u]$. i indicates the number of rules (i=1,...,M) and n is the number of antecedents.

The upper membership is:

$$\overline{\mu}_{\tilde{A}_{in}}(x) = e^{-\frac{1}{2}(\frac{x-c}{\sigma_u})^2} \tag{2.42}$$

The lower membership function is:

$$\underline{\mu}_{\tilde{A}_{in}}(x) = e^{-\frac{1}{2}(\frac{x-c}{\sigma_l})^2} \tag{2.43}$$

An example of Gaussian primary membership function with uncertain standard deviation is indicated in Figure 2.18.

2.4.1.2. Gaussian Primary Membership Function with Uncertain Mean. A Gaussian primary membership function with an uncertain mean is specified by three parameters c_1, c_2, σ, where center of the membership function changes in the interval $[c_1, c_2]$, and σ is the standard deviation of the membership function. The mathematical represen-

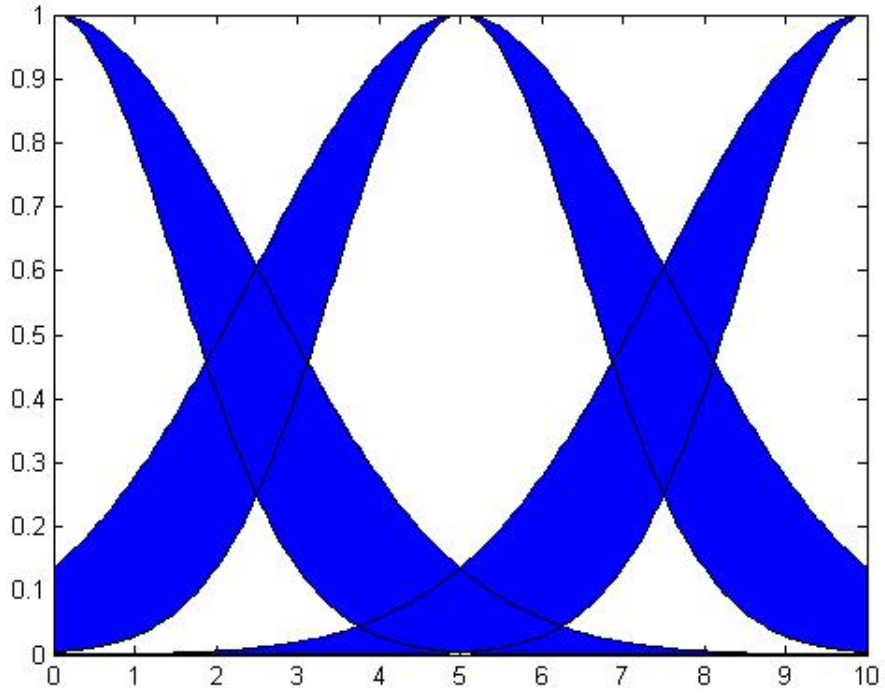

Figure 2.18. Gaussian primary membership function with uncertain standard deviation

tation is as follows [17]:

$$\mu_{\tilde{A}_{in}}(x) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2} \tag{2.44}$$

where $c \in [c_1, c_2]$. i indicates the number of rules (i=1,...,M) and n is the number of antecedents.

The upper membership is:

$$\overline{\mu}_{\tilde{A}_{in}}(x) = \begin{cases} e^{-\frac{1}{2}(\frac{x-c_1}{\sigma})^2}, & x < c_1 \\ 1, & c_1 \leq x \leq c_2 \\ e^{-\frac{1}{2}(\frac{x-c_2}{\sigma})^2}, & x > c_2 \end{cases} \tag{2.45}$$

The lower membership function is:

$$\underline{\mu}_{\tilde{A}_{in}}(x) = \begin{cases} e^{-\frac{1}{2}(\frac{x-c_2}{\sigma})^2}, & x \leq \frac{c_1+c_2}{2} \\ e^{-\frac{1}{2}(\frac{x-c_1}{\sigma})^2}, & x > \frac{c_1+c_2}{2} \end{cases} \tag{2.46}$$

An example Gaussian primary membership function with uncertain mean is shown in Figure 2.19.

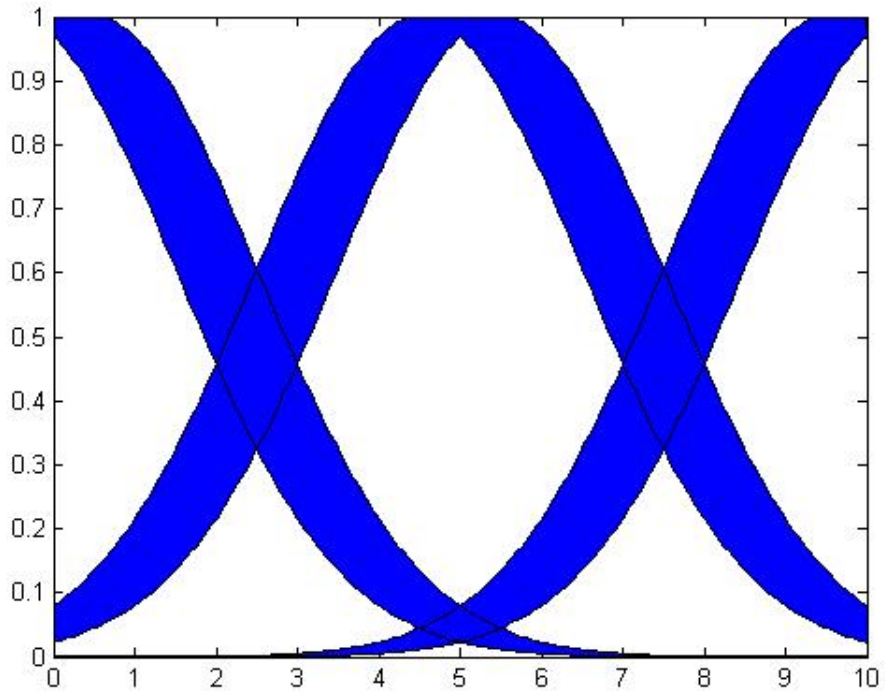

Figure 2.19. Gaussian primary membership function with uncertain mean

As it was mentioned earlier, due to the computational complexity in this work interval singleton type-2 fuzzy sets are taken into consideration.

2.4.2. Fuzzy Inference Engine and Rule Base

The fuzzified inputs get into the fuzzy inference block, which works collectively with the rule base of the FLS. The rule structure of type-2 fuzzy logic is as follows:

A General type-2 FLS:

$$R^i = \text{IF } X_1 \text{ is } \tilde{A}_{i1} \text{ and ... and } X_n \text{ is } \tilde{A}_{in},$$
$$\text{THEN } Z^i = \tilde{C}_i$$

A T2 TSK FLS:

$$R^i = \text{IF } X_1 \text{ is } \tilde{A}_{i1} \text{ and ... and } X_n \text{ is } \tilde{A}_{in},$$
$$\text{THEN } z^i = a_n^i x_n + a_{n-1}^i x_{n-1} + ... + a_0^i$$

In this rule structure, i indicates the number of rules (i = 1, 2,...,M), A_{in} are antecedent type-2 fuzzy sets and are indicated with tildes. Z^i is output of the system for general type-2 FLS and $\tilde{C}_i$ is the type-2 fuzzy set at the consequent part of the rule. z^i is the output of the T2 TSK FLS and is a polynomial. In literature, J. M. Mendel proposed several kinds of type-2 TSK FLSs, and in this work unnormalized interval type-2 TSK FLS is studied.

2.4.2.1. The Meet and Join Operators. The antecedent parts of the rules are combined by using meet and join fuzzy operations which are recently defined for type-2 fuzzy sets in [17]. The join and meet operators are defined as:

$$\underline{\omega}^i = \underline{\mu}_{\tilde{A}_{i1}}(x_1) * \cdots * \underline{\mu}_{\tilde{A}_{in}}(x_n) \tag{2.47}$$

$$\overline{\omega}^i = \overline{\mu}_{\tilde{A}_{i1}}(x_1) * \cdots * \overline{\mu}_{\tilde{A}_{in}}(x_n) \tag{2.48}$$

In the case of join operation for interval singleton type-2 fuzzy sets "*" indicates the maximum.

In the case of meet under product or minimum t-norm operation for interval singleton type-2 fuzzy sets, the "*" indicates the minimum or product operation.

In this thesis, meet under product t-norm for interval singleton type-2 fuzzy sets are applied in the simulations.

2.4.3. Type-Reduction

In type-1 fuzzy logic systems, the output of the system is a type-1 fuzzy set and this fuzzy set is defuzzified to obtain crisp output by using one of the defuzzification methods described in the previous section. On the other hand, in type-2 case, the output is a type-2 fuzzy set. First of all, type-2 fuzzy set should be reduced to a type-1 fuzzy set. This procedure was developed by Karnik-Mendel and is called the "type-reduction method". This method is an extension of type-1 defuzzification methods. There are several types of type-reduction methods such as; centroid, center of sets,..., etc. The most commonly used type-reduction method is center of sets type-reduction method and is described in the following [17]:

$$Z_{COS}(Z^1,...,Z^M,\Omega^1,...,\Omega^M) = [z_l, z_r] = \int_{z^1}\cdots\int_{z^M}\int_{\omega^1}\cdots\int_{\omega^M} 1/\frac{\sum_{i=1}^{M}\omega^i z^i}{\sum_{i=1}^{M}\omega^i} \quad (2.49)$$

The result of the type-reduction procedure, Z_{COS}, equals to interval set, $[z_l, z_r]$. where $z^i \in Z^i = [z_l^i, z_r^i]$ and $\omega^i \in \Omega^i = [\underline{\omega}^i, \overline{\omega}^i]$.

z is calculated as follows:

$$z = \frac{\sum_{i=1}^{M}\omega^i z^i}{\sum_{i=1}^{M}\omega^i} \quad (2.50)$$

where for any $z \in Z_{COS}$.

The maximum value of z is z_r and calculated as:

$$z_r = \frac{\sum_{i=1}^{M} \omega_r^i z_r^i}{\sum_{i=1}^{M} \omega_r^i} \tag{2.51}$$

The minimum value of z is z_l and calculated as:

$$z_l = \frac{\sum_{i=1}^{M} \omega_l^i z_l^i}{\sum_{i=1}^{M} \omega_l^i} \tag{2.52}$$

The computational procedure of Karnik-Mendel is as follows [17]:

For the maximum value of z is:

1. reorder z_r^i for i=1,...,M in ascending order.
2. z_r is calculated by using Equation (2.51) and $\omega_r^i = \frac{\underline{\omega}^i + \overline{\omega}^i}{2}$ for i=1,...,M, and $z_r^{'} = z_r$.
3. Find R ($1 \leq R \leq M-1$) such that $z_r^R \leq z_r^{'} \leq z_r^{R+1}$
4. Find z_r by using Equation (2.51) $\omega_r^i = \underline{\omega}^i$ for $i \leq R$ and $\omega_r^i = \overline{\omega}^i$ for $i > R$. Let $z_r^{''} = z_r$.
5. If $z_r^{''} \neq z_r^{'}$ then go to step 6. If $z_r^{''} = z_r^{'}$, then stop and set $z_r = z_r^{''}$.
6. $z_r^{'} = z_r^{''}$, and go back to step 3.

For the minimum value of z is:

1. reorder z_l^i for i=1,...,M in ascending order.
2. z_l is calculated by using Equation (2.52) and $\omega_l^i = \frac{\underline{\omega}^i + \overline{\omega}^i}{2}$ for i=1,...,M and $z_l^{'} = z_l$.
3. Find R ($1 \leq L \leq M-1$) such that $z_l^L \leq z_l^{'} \leq z_l^{L+1}$
4. Find z_l by using Equation (2.52) $\omega_l^i = \underline{\omega}^i$ for $i \leq L$ and $\omega_l^i = \overline{\omega}^i$ for $i > L$. Let $z_l^{''} = z_l$.
5. If $z_l^{''} \neq z_l^{'}$ then go to step 6. If $z_l^{''} = z_l^{'}$, then stop and set $z_l = z_l^{''}$.
6. $z_l^{'} = z_l^{''}$, and go back to step 3.

2.4.4. Defuzzification

After the type-reduction procedure, the obtained result is an interval set $[z_l, z_r]$. This interval set is defuzzified by taking the weighted average and calculated in the following [17]:

$$z = \frac{z_l + z_r}{2} \tag{2.53}$$

2.4.5. Interval Type-2 TSK Fuzzy Logic Systems (FLSs)

There are two kinds of TSK FLSs in the interval type-2 fuzzy systems: Interval TSK fuzzy logic systems and unnormalized interval type-2 fuzzy logic systems. Interval type-2 TSK FLSs are divided into three models depending on the types of membership functions on antecedent and consequent parts of the IF-THEN rules. In the next subsection these models are briefly presented [39].

2.4.5.1. Model 1. In this model, interval type-2 fuzzy sets are used in the antecedent part of the rules and type-1 fuzzy sets are used for the consequent part of the rules. An example of first order interval type-2 TSK rule structure is shown for the first model as follows [39]:

$$R^i = \text{IF } X_1 \text{ is } \tilde{A}_{i1} \text{ and } ... \text{ and } X_n \text{ is } \tilde{A}_{in},$$
$$\text{THEN } Z^i = C_{in}x_n + C_{i(n-1)}x_{n-1} + ... + C_{i0}$$

where $\tilde{A_{in}}$ is an interval type-2 fuzzy set, C_{in} is type-1 fuzzy set. X_n is the input and Z^i is the output of the system.

2.4.5.2. Model 2. In the second model, the antecedent part of the rule is interval type-2 fuzzy set and consequent part of the rule is a first order polynomial [39]. The rule

structure is given in the following:

$$R^i = \text{IF } X_1 \text{ is } \tilde{A}_{i1} \text{ and } ... \text{ and } X_n \text{ is } \tilde{A}_{in},$$
$$\text{THEN } z^i = a_n^i x_n + a_{n-1}^i x_{n-1} + ... + a_0^i$$

where z^i is the output of the system and is a first order polynomial. a_n^i are coefficients of the polynomial. X_n is the input and $\tilde{A}_{in}$ are interval type-2 fuzzy sets.

2.4.5.3. Model 3. In this model, both antecedent and consequent part of the rules are type-1 fuzzy sets. Here the assumption is that the fuzzy sets in the consequent part of the rules carry the uncertainties, and they are fuzzy numbers [39]. An example rule structure is given in the following:

$$R^i = \text{IF } X_1 \text{ is } A_{i1} \text{ and } ... \text{ and } X_n \text{ is } A_{in},$$
$$\text{THEN } Z^i = C_{in} x_n + C_{i(n-1)} x_{n-1} + ... + C_{i0}$$

where A_{in} are antecedent and C_{in} are consequent type-1 fuzzy sets. X_n is the input and Z^i is the output of the system.

2.4.6. Unnormalized Interval Type-2 TSK FLSs

For the unnormalized interval type-2 TSK fuzzy model, there is no need for type reduction. The algorithm of this method is briefly given in the following:

The fourth modeling approach used in this thesis is the unnormalized interval type-2 TSK fuzzy logic system. In this model the antecedent part of the rules are interval type-2 fuzzy sets and consequent part of the rules are type-0 fuzzy (crisp) sets. The i^{th} rule structure for M rules is;

$$R^i = \text{IF } X_1 \text{ is } \tilde{A}_{i1} \text{ and } ... \text{ and } X_n \text{ is } \tilde{A}_{in},$$
$$\text{THEN } z^i = a_n^i x_n + a_{n-1}^i x_{n-1} + ... + a_0^i$$

The interval type-2 antecedent fuzzy sets are indicated with tildes. Consequent part of the rule is a first order polynomial. This model is not the only model to realize an interval type-2 TSK fuzzy logic system. In literature, J. M. Mendel has proposed other interval Type-2 TSK models [39], which are briefly described in previous subsection.

Using unnormalized interval type-2 TSK reduces the computational complexity, when compared to the other interval type-2 TSK fuzzy logic algorithms. The output of the general unnormalized interval type-2 TSK fuzzy logic is [40]:

$$Z_{TSK,2}(x) = [z_l, z_r] = \int_{z^1 \in [z_l^1, z_r^1]} \cdots \int_{z^M \in [z_l^M, z_r^M]} \int_{\omega^1 \in [\underline{\omega}^1, \overline{\omega}^1]} \cdots \int_{\omega^M \in [\underline{\omega}^M, \overline{\omega}^M]} 1 / \sum_{i=1}^{M} \omega^i z^i \tag{2.54}$$

As it is seen above, the result is an interval type-1 fuzzy set. The algorithm of the unnormalized interval Type-2 TSK Fuzzy Logic is [40]:

- The firing strength of the each rule is calculated by using "meet under the product t-norm [4]." In the calculations of interval type-2 fuzzy sets, the upper and lower membership functions are used and they are type-1 fuzzy sets. Thus, type-1 fuzzy arithmetic can be used in the calculations of interval type-2 fuzzy sets. General formulation of the meet of interval type-2 fuzzy sets is:

$$\prod_{j=1}^{n} A_j = [\underline{\omega}^i, \overline{\omega}^i] \tag{2.55}$$

  where n indicates the antecedent number (j = 1, 2,..., n) and i indicates the rule number. A_j is the interval type-1 fuzzy set [4]. The meet of interval type-2 fuzzy sets under product t-norm is basically computed by using the lower membership functions and upper membership functions. The lower and upper bounds of the firing strength are calculated as follows:

$$\underline{\omega}^i = \underline{\mu}_{\tilde{A}_{i1}}(x_1) \cdots \underline{\mu}_{\tilde{A}_{in}}(x_n) \tag{2.56}$$

$$\overline{\omega}^i = \overline{\mu}_{\tilde{A}_{i1}}(x_1) \cdots \overline{\mu}_{\tilde{A}_{in}}(x_n) \tag{2.57}$$

- The output of each rule is an interval set $[z_r, z_l]$. The calculation of the minimum value of z is based on multiplying the lower firing strength of each rule ($\underline{\omega}^i$) with the consequent of the first order polynomial. The maximum value of z is calculated by multiplying the upper firing strength of each rule ($\overline{\omega}^i$) by the consequent of the first order polynomial.

  The minimum value of the output is:

$$z_l = \sum_{i=1}^{M} \underline{\omega}^i z^i \tag{2.58}$$

  The maximum value of the output is:

$$z_r = \sum_{i=1}^{M} \overline{\omega}^i z^i \tag{2.59}$$

$$z^i = a_n^i x_n + a_{n-1}^i x_{n-1} + ... + a_0^i \tag{2.60}$$

3. The defuzzified output of the system is:

$$z = \frac{z_l + z_r}{2} \tag{2.61}$$

3. OPTIMIZATION METHOD

In this study, the objective function is Root Mean Square Error (RMSE), and is defined in the following:

$$F_{obj} = \sqrt{\frac{\sum_{k=1}^{N}(z_D^k - z^k)^2}{N}} \tag{3.1}$$

where z_D^k is the desired output and z^k is the actual output of the system (k=1,...,N). N is the number of values that are used for the calculation of the function approximation.

The aim is to minimize an objective function based on the root mean square error (RMSE), by using one of the nonlinear programming methods. The parameters, which are used to calculate the actual output of the system, are adjusted to minimize the objective function. The parameters to be tuned changes for each method and is described in the following.

Optimization toolbox in MATLAB® is very useful to implement optimization methods. In MATLAB® [41], there are three methods for nonlinear optimization problem with constraints:

1. Trust-region
2. Active set sequential quadratic programming
3. Interior-point

3.1. Sequential Quadratic Programming (SQP)

In nonlinear programming, the most commonly used method is the second method, Sequential Quadratic Programming (SQP). In nonlinear programming, Karush-Kuhn-Tucker (KKT) conditions are used iteratively in SQP, which are necessary for a local optimum solution, and for special situations necessary and sufficient conditions for a

global optimum solution [42].

In this method, at each iteration, a QP problem is solved, and this QP problem is derived from the approximation of the Lagrangian function, which is defined below [41]:

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \tag{3.2}$$

The quadratic sub-problem is [41]:

$$\underset{d \in R^n}{minimize}\ \frac{1}{2} d^T H_k d + \nabla f(x_k)^T d \tag{3.3}$$

where $\nabla g_i(x_k)^T d + g_i(x_k) \leq 0$ and $\nabla g_i(x_k)^T d + g_i(x_k) = 0$

As described in MATLAB®, Sequential Quadratic Programming (SQP) is composed of three main steps [41]:

3.1.1. Updating the Hessian Matrix

In each step of iteration, Hessian of the Lagrangian matrix is calculated by BFGS (Broyden-Fletcher-Goldfarb-Shanno) [43], approximation of a positive definite quasi Newton [41].

$$H_{k+1} = H_k + \frac{q_k q_k^T}{q_k^T s_k} - \frac{H_k^T H_k}{s_k^T H_k s_k} \tag{3.4}$$

where $s_k = x_{k+1} - x_k$ and $q_k = \nabla f(x_{k+1}) + \sum_{i=1}^{n} \lambda_i \nabla g_i(x_{k+1}) - (\nabla f(x_k) + \sum_{i=1}^{n} \lambda_i \nabla g_i(x_k))$

3.1.2. Quadratic Programming Problem Solution

A Quadratic Programming (QP) sub-problem is solved in each step of the iteration [41].

$$\underset{d \in R^n}{minimize}\ q(d) = \frac{1}{2} d^T H d + c^T d \tag{3.5}$$

where $A_i d = b_i$, i=1,...,m_e and $A_i d \leq b_i$, $i = m_{e+1}, ..., m$

3.1.3. Line Search and Merit Function Calculation

The outcome of the above QP problem is a d_k vector and this vector is a search direction and is used for the new iteration [41].

$$x_{k+1} = x_k + \alpha d_k \tag{3.6}$$

The line search is realized such that an α_k value is found that minimizes the below merit $\Psi(x)$ function [41].

$$\Psi(x) = f(x) + \sum_{i=1}^{m_e} r_i g_i(x) + \sum_{i=m_e+1}^{m} r_i max\{0, g_i(x)\} \tag{3.7}$$

The penalty parameter r_i is defined as [41]:

$$r_i = (r_{k+1})_i = \underset{i}{max}\{\lambda_i, \frac{1}{2}((r_k)_i + \lambda_i)\} \tag{3.8}$$

Line search is a significant step for the convergence of the algorithms.

In this thesis, for the tuning of the parameters of the methods considered for modelling, the function "fminimax" of the MATLAB® optimization toolbox is ap-

plied. This function uses Sequential Quadratic Programming and gives us the ability to implement bounds, equality and inequality constraints. For a fair comparison, the same optimization approach is used in all cases.

4. FURTHER RELATIONS IN FUZZY SETS

4.1. Similarities and Differences between Constrained Fuzzy Sets (CFSs) and Type-2 Fuzzy Sets

In constrained fuzzy sets, the linguistic terms are bounded depending on the constraints. During the optimization process, the membership functions that represent the linguistic interpretation move in the uncertainty bound, and this constitutes a continuum set of interval type-2 fuzzy sets. For instance, a Gaussian membership function carries the expert knowledge about the system and has an uncertainty interval. During the optimization process, the center and sigma values are changing $[c_1, c_2]$ and $[\sigma_1, \sigma_2]$ in this interval. This constitutes a continuum of interval type-2 fuzzy set. On the other hand, when Gaussian membership function with uncertain standard deviation is taken into consideration, sigma value is in the interval of $[\sigma_l, \sigma_u]$, and has a single center value c.

In real life applications, systems encounter with many disturbances and noise from the external environment. type-1 fuzzy sets are not efficient in real-world applications with high amount of uncertainty; because they use crisp and precise membership functions. Today, it is better to use CFSs or type-2 fuzzy sets depending on the type of the application.

4.2. Advantages and Disadvantages of Tuning the Membership Functions

Fuzzy controllers give us the flexibility to choose design parameters and to determine which parameters are to be tuned. If the linguistic terms play a major role in the design of fuzzy controller, by tuning the membership functions the linguistic interpretation will be lost or distorted, as the membership functions may move out of the domain or may have large intersections with each other. Especially in industrial process control, this will give rise to drastic changes that will affect the performance of the process adversely.

In fuzzy models, in which the human expert knowledge is the key element of the design of a fuzzy model, tuning the membership functions will result in the lost or distortion of the expert knowledge. In such applications, another adaptation will be more appropriate than the adaptation of the membership functions.

Furthermore, by tuning the membership functions, a membership function can move so far that it may become a subset of another one, so the interpretation of the linguistic variables and the structure of the rules will be corrupted.

In the applications, where the interpretation of linguistic variable, expert knowledge, and rule base are important to obtain an optimal fuzzy model, the membership functions should not be modified. To obtain a desired approximation and come up with a minimum error response, other approaches should be sought. As a new approach, the tuning of the membership functions can be restricted with constraints on fuzzy sets, and also the parameters of operators can be tuned to get a proper approximation results.

5. COMPARISON OF FOUR FUZZY METHODS FOR APPROXIMATION OF SINC FUNCTION

The performances of the four modeling approaches described above are tested on the Sinc function, the equation of which is given below:

$$z = f(x_1, x_2) = \left| \frac{sin[\pi(x_1 - 3)]}{\pi(x_1 - 3)} \right| \left| \frac{sin[\pi(x_2 - 3)]}{\pi(x_2 - 3)} \right| \tag{5.1}$$

This function has a global maximum at $x_1 = 3.0$ and $x_2 = 3.0$ and also has several local maximums. The input range is in between [0, 8] ($x_1, x_2 \in [0, 8]$). Four Gaussian membership functions are used for each antecedent.

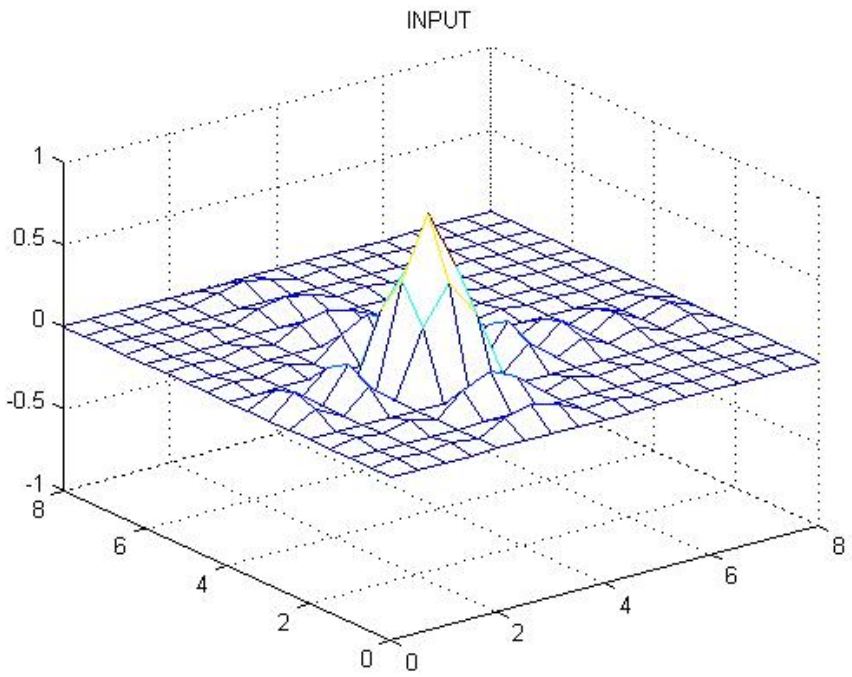

Figure 5.1. The sinc function that has global maximum at $x_1 = 3.0, x_2 = 3.0$

5.1. Approximation of Sinc Function by Using Parameterized Conjunctions

For the application of this method, the input space is first partitioned equally $x_1, x_2 = (0,\ 0.5, \cdots, 8)$ and 17x17 = 289 values of the function is calculated. It is

assumed that the membership functions carry the expert knowledge about the system. Four Gaussian membership functions for each antecedent, which are shown in Figure 5.2, are distributed on the input domain and the rule base is composed of 16 rules. The i^{th} rule is described as follows:

$$R^i = \text{IF } X_1 \text{ is } A_{i1} \text{ and } X_2 \text{ is } A_{i2},$$
$$\text{THEN } z^i = a_2^i x_2 + a_1^i x_1 + a_0^i$$

In this rule structure, A_{i1} and A_{i2} are the fuzzy sets, defining the linguistic variables (e.g., small, medium, large, etc.). z^i is the output of the each rule and a_2^i, a_1^i, and a_0^i are the coefficients of the first order polynomial. To calculate the firing strength for each rule the G-conjunction operators with the parameters p_i, and q_i are used as AND operator and have the following form:

$$\omega^i = T(\mu_{A_{i1}}(x_1), \mu_{A_{i2}}(x_2)) = \mu_{A_{i1}}(x_1)^{p_i}.\mu_{A_{i2}}(x_2)^{q_i} \tag{5.2}$$

The actual output of the system is:

$$z = \frac{\sum_{i=1}^{M} \omega^i z^i}{\sum_{i=1}^{M} \omega^i} \tag{5.3}$$

where i = 1, 2,..., M and M indicates the number of rules (M=16). z is the actual output of the system and z^i is the output of each rule. To tune the parameters p_i, and q_i, Root Mean Square Error (RMSE) is minimized by using one of the most commonly used nonlinear programming approaches, Sequential Quadratic Programming (SQP). Since the membership functions are assumed to characterize the linguistic interpretation, their parameters are kept fixed. The objective function, RMSE is:

$$F_{obj} = \sqrt{\frac{\sum_{k=1}^{N} (z_D^k - z^k)^2}{N}} \tag{5.4}$$

z_D is the desired output and N = 289 is the number of values that are calculated

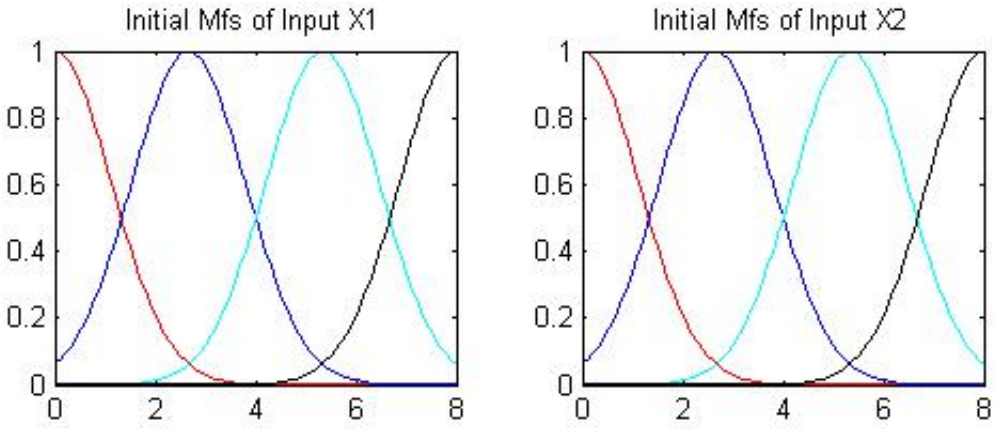

Figure 5.2. The membership functions that carry the expert knowledge

for the approximation of the function. During the optimization process, for the left sides of the rules, 32 (16x2=32) parameters, and for the right sides of the rules, 48 (16x3=48) parameters are tuned; as a result, a total of 80 parameters are adjusted to minimize the RMSE (since the membership functions are assumed to characterize the linguistic interpretation, their parameters are kept fixed.) The bound constraints are applied to the right and left sides of the rules, and respectively are $-49.9 \leq a_2^i, a_1^i, a_0^i \leq 50$ and $0.1 \leq p_i, q_i \leq 50$. The obtained results are shown on Figure 5.3. The approximation error obtained is 0.03455.

5.2. Approximation of Sinc Function by Using Constrained Fuzzy Sets (CFSs)

In this approach, the parameters of the membership functions are tuned within some constraints so that any expert knowledge present in them is not lost. A first order type-1 TSK fuzzy logic model with 16 rules, having the same rule structure as in the previous method is used. Initially, four Gaussian membership functions for each antecedent that are indicated in Figure 5.4 are distributed in the input domain, and defined as:

$$\text{Gaussian membership function} = e^{\frac{1}{2}(\frac{x-c}{\sigma})^2} \tag{5.5}$$

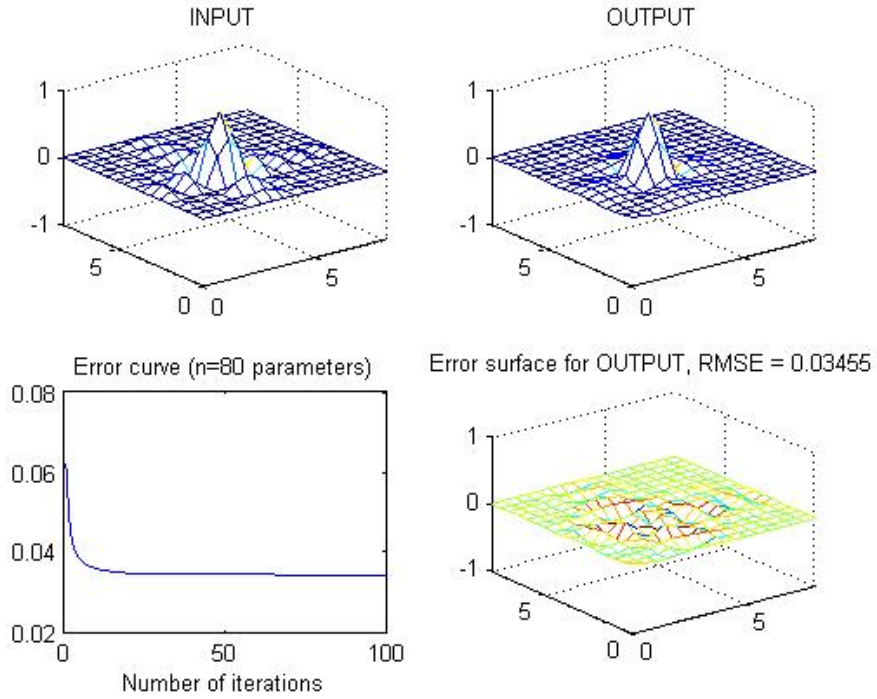

Figure 5.3. The result of approximating two input sinc function by using parameterized conjunction

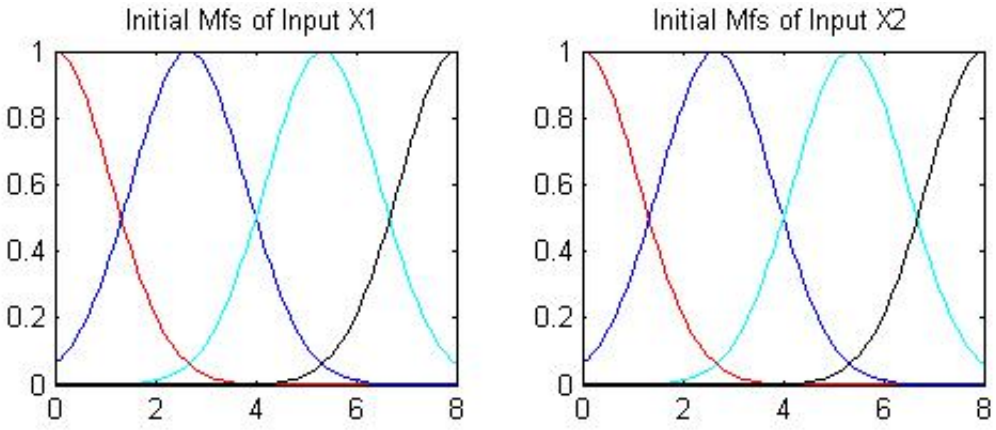

Figure 5.4. The Initial membership functions before tuning CFSs

Sigma (σ) determines the width and c determines the center of the membership functions. The expert knowledge about the system is not certain about $\mp 0,40$. Since, the model has a total of eight membership functions and there are two parameters for each membership function, 16 parameters of the left sides of the rules are adjusted within the constraints by using SQP optimization method. The 48 parameters of right sides of the rules, a_2^i, a_1^i, a_0^i, are tuned together with the left side of the rules. The final membership functions and the approximated fuzzy model are shown in Figure 5.5, and

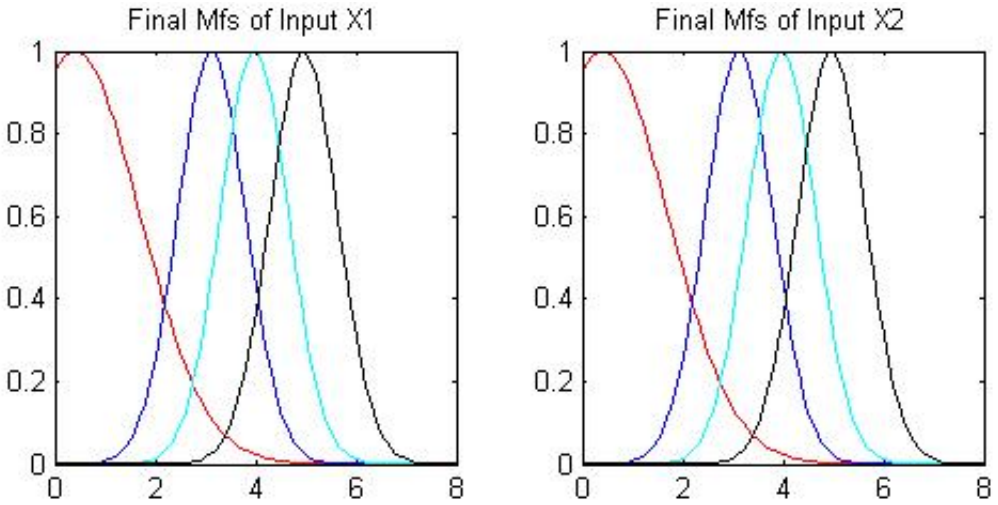

Figure 5.5. The membership functions after tuning CFSs

Figure 5.6, respectively. As it is seen, the membership functions are moved within the constraints and thus any expert knowledge about the system is not lost. The error for this approximation method is 0.04175.

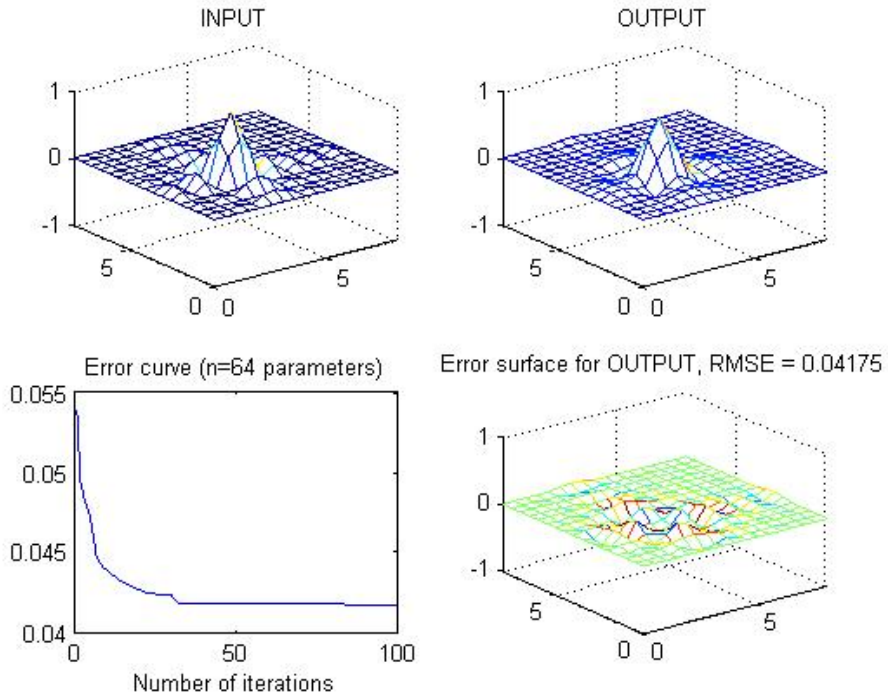

Figure 5.6. The result of approximating two input sinc function by using CFSs

5.3. Approximation of Sinc Function by Using Constrained Fuzzy Sets (CFSs) with Parameterized Conjunctions

In this approach, CFSs are used with the G-conjunction operators. The rule structure, and the fuzzy model is the same as in the previous approaches; but the major difference is that the G-conjunction operator is used as AND operator, described as:

$$\omega^i = T(\mu_{A_{i1}}(x_1), \mu_{A_{i2}}(x_2)) = \mu_{A_{i1}}(x_1)^{p_i}.\mu_{A_{i2}}(x_2)^{q_i} \tag{5.6}$$

where i = 1,2, ..., 16. The initial Gaussian membership functions are distributed as in Figure 5.7. In this model, the objective function is the RMSE, and SQP nonlinear

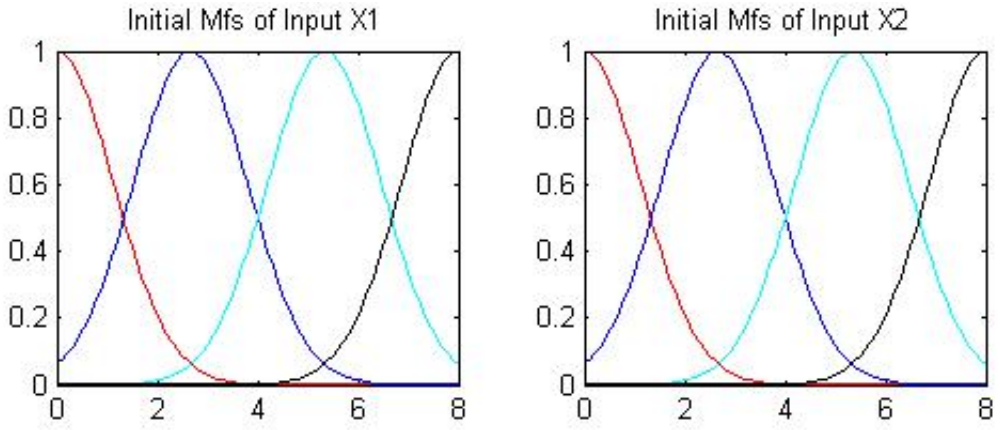

Figure 5.7. The Initial membership functions before tuning CFSs with parameterized conjunctions

optimization method is used to tune a total of 96 parameters (for the left side of the rules 48 and right sides of the rules 48 parameters). The right and left sides of the rules are tuned together. The membership functions after tuning are shown in Figure 5.8, and the experimental results that are obtained after tuning are shown in Figure 5.9. The obtained approximation error (RMSE) is 0.021074.

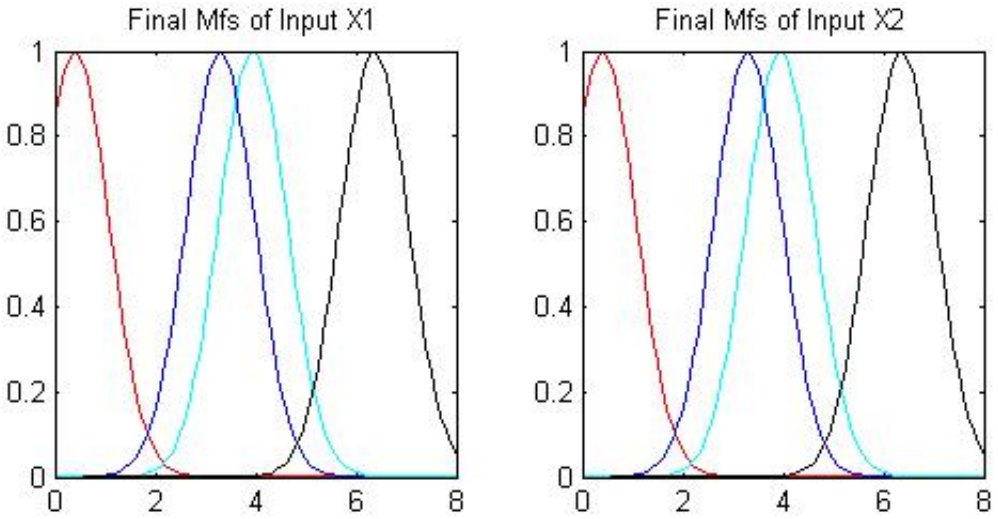

Figure 5.8. The membership functions after tuning CFSs with parameterized conjunctions

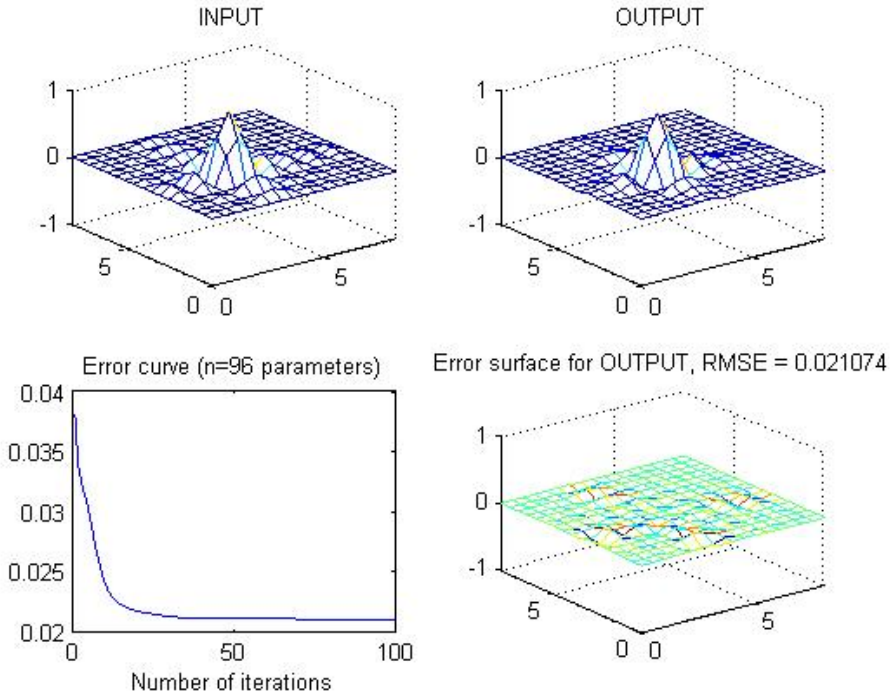

Figure 5.9. The result of approximating two input sinc function by using CFSs with parameterized conjunctions

5.4. Approximation of Sinc Function by Using Unnormalized IT2 TSK FLSs

One of the most important features of the interval type-2 fuzzy sets is the ability to incorporate uncertainties in the membership functions, and this feature makes type-

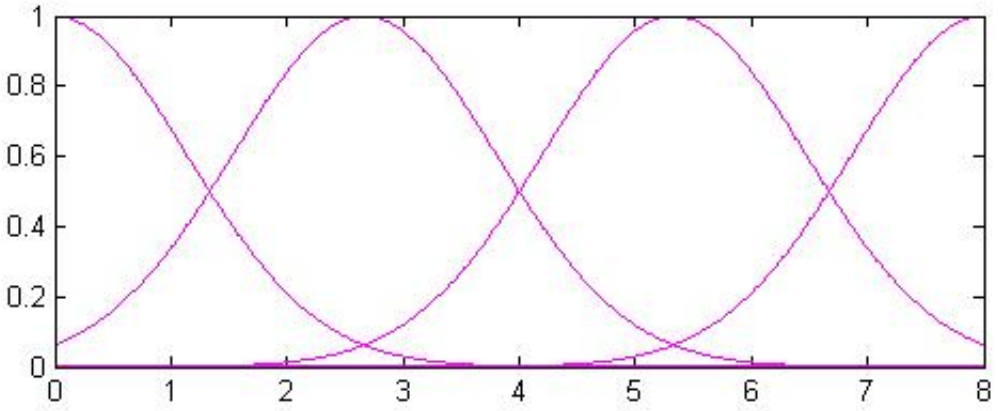

Figure 5.10. Type-1 membership functions

2 fuzzy sets preferable when there exists significant uncertainties. In this example, the approximation of sinc function is obtained by using unnormalized IT2 TSK FLS. Type-1 fuzzy sets used to identify the initial membership functions of interval type-2 fuzzy sets. First of all, as it is seen in Figure 5.10, type-1 Gaussian membership functions are distributed on the input domain with intersection of the alpha value $\alpha = 0.5$.

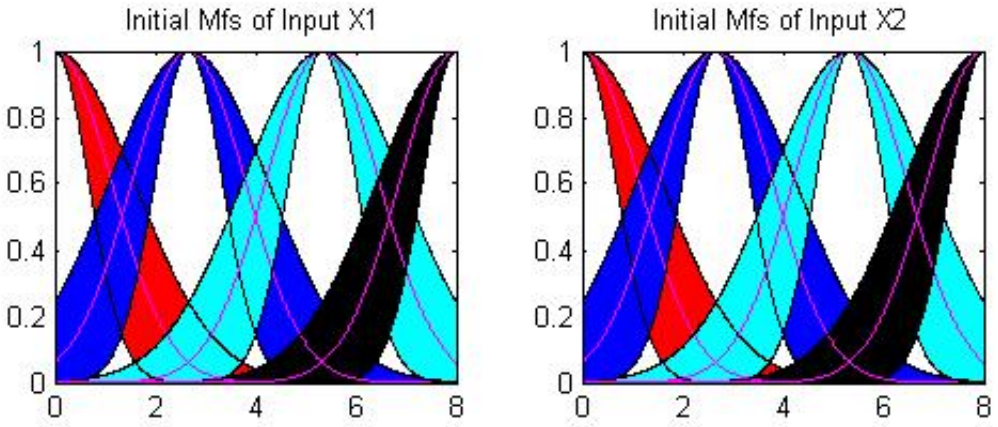

Figure 5.11. Initial interval type-2 Gaussian membership functions with uncertain standard deviation

In this application, Gaussian primary membership functions with uncertain standard deviations are used. Standard deviations of the Gaussian membership functions have the uncertainty interval ∓ 0.40. The sigma and center values of each antecedent upper and lower membership functions for uncertainty interval are 0.40. As it is seen in Figure 5.11, type-1 membership functions are blurred to the left and right by changing

the sigma value in the uncertainty interval of 0.40. The i^{th} rule is given below:

$$R^i = \text{IF } X_1 \text{ is } \tilde{A}_{i1} \text{ and } X_2 \text{ is } \tilde{A}_{i2},$$
$$\text{THEN } z^i = a_2^i x_2 + a_1^i x_1 + a_0^i$$

A_{i1} and A_{i2} are interval type-2 fuzzy sets, which describe the linguistic values. In the realization of unnormalized IT2 TSK fuzzy logic, the antecedent membership functions are interval type-2 Gaussian membership functions with uncertain standard deviations and the consequent membership functions are type-0. SQP optimization method is used to minimize the error. The objective function to be minimized is:

$$F_{obj} = \sqrt{\frac{\sum_{k=1}^{N}(z_D^k - z^k)^2}{N}} \tag{5.7}$$

N equals to 289 and is the number of values that are calculated for the function approximation. During the approximation of the nonlinear function, both the parameters of membership functions, σ_{upper}^i, σ_{lower}^i, and c^i, and the right sides of the rules, a_2^i, a_1^i, a_0^i are tuned together. The centers of the membership functions are bounded in the interval of input domain not to move out of the domain and also the parameters at right sides of the rules are bounded in the interval $-49.9 \leq a_2^i, a_1^i, a_0^i \leq 50$. As it is seen in Figure 5.12, by tuning the membership functions, the expert knowledge is somewhat lost; but the uncertainties are taken into account. Figure 5.13 depicts the modelling performance. The resulting approximation error is 0.030219.

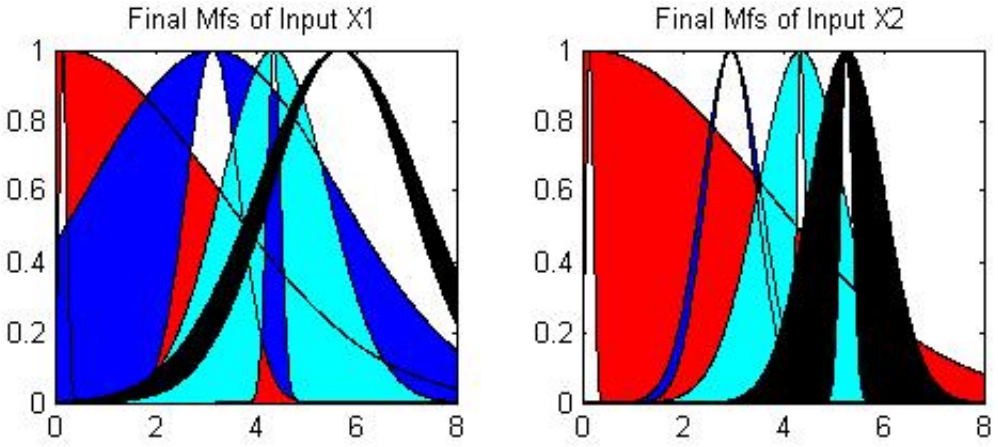

Figure 5.12. The interval type-2 gaussian membership functions with uncertain deviation after tuning with 0.40 uncertainty bound

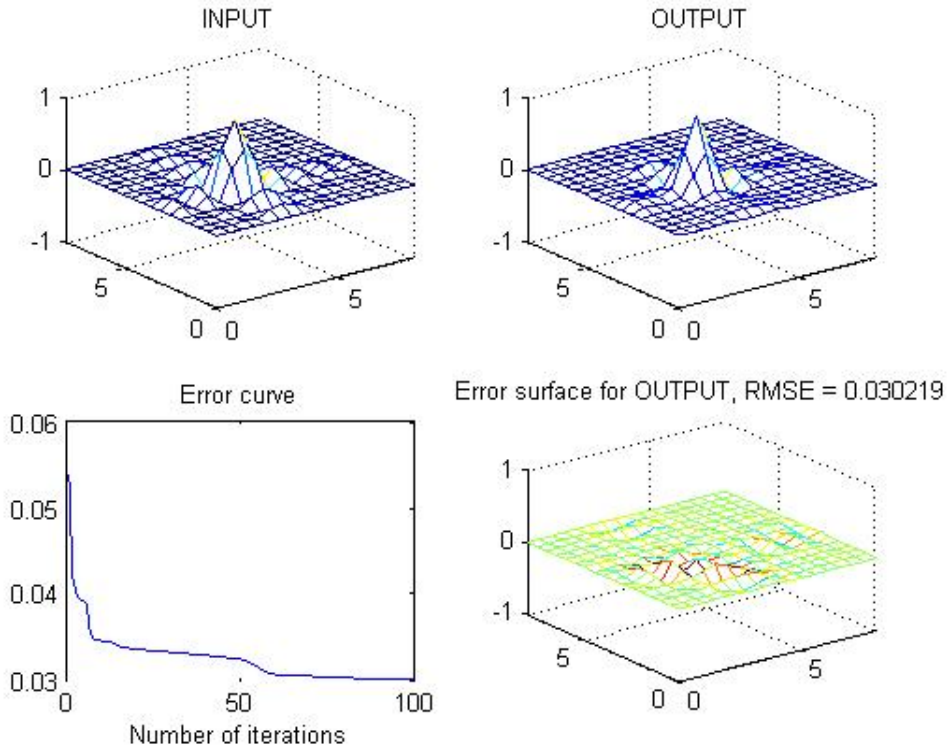

Figure 5.13. The result of approximating two input sinc function by using IT2 TSK Unormalized Fuzzy Logic

6. CONCLUSION

This thesis focuses on four different methods for fuzzy modeling and on their mathematical and theoretical background. One of the methods (CFSs) is a novel one and aims to reach a compromise between type-1 fuzzy sets and type-2 fuzzy sets for handling the uncertainties in the membership functions. The performances of these four methods are examined and compared for the approximation of a nonlinear function. The resulting RMSEs together with the number of parameters tuned are given in Table 6.1.

Table 6.1. Comparison of the Four Methods

	Method 1	Method 2	Method 3	Method 4
RMSE	0.03455	0.04175	0.021074	0.030219
No of Parameters	80	64	96	72
Optimization Method	SQP	SQP	SQP	SQP
iteration no	100	100	100	100

Fuzzy modeling with simple parameterized conjunctions that do not carry associativity and commutativity properties is proposed in [5] and [6]. This proposed model was evolved in this study, and fuzzy modeling with CFSs that include parameterized G-conjunctions is proposed. The first three methods are significant in the applications where the expert knowledge is crucial. In addition, comparing these three methods, the best result is obtained by the CFSs with parameterized conjunctions. The reason is due to this model taking into account the uncertainties of the linguistic values and having more design parameters such as the parameters of membership functions, the parameters of operators, and coefficients at the consequent part of the rules. On the other hand, in the application of parameterized conjunctions, the membership functions carry the expert knowledge, and are kept fixed; but the coefficient parameters and parameter of operations are tuned together. As a result, in this model the uncertainties in the expert knowledge are disregarded. Thus, the modelling is not as good as the third approach; but better than the second approach. In the second approach with CFSs, the uncertainties are described in the membership functions as constraints.

Although the parameters of the consequent part are tuned, the operators used are non-parameterized and this limits the modeling performance and the highest RMSE value is obtained when compared to the other approaches.

CSFs and interval type-2 fuzzy sets give the designer the flexibility to define uncertainties in the fuzzy sets. CFSs use fuzzy intervals, same as interval type-2 fuzzy sets; but CFSs move in that interval and a specified input takes on a precise value. In contrast, in the interval type-2 fuzzy sets, specified input takes on an interval value [4]. Furthermore, while tuning the CFSs, the membership function parameters, both sigma and center, move in the boundary of the constraints which constitutes a continuum set of interval type-2 fuzzy sets. Comparing the simulation results of CFSs with parameterized conjunctions and the unnormalized IT2 TSK FLSs, the formal preserves expert knowledge within the boundary of uncertainty and also has more design parameters which gives the designer more degrees of freedom. On the other hand, the latter has fewer design parameters and also in the tuning of the parameters, the expert knowledge is lost as it defines the uncertainty in terms of the interval type-2 fuzzy sets.

As a conclusion, based on the experiences obtained in this work with regards to the modelling of a complex function with two inputs, it can be stated that CFSs with parameterized G-conjunctions give better results when compared to the other methods considered. In this model, the expert knowledge is conserved within the uncertainty bounds of the knowledge. The parameters at the left side of the rules (membership function parameters and parameters of operations), and the right side of the rules (the coefficients of the first order polynomial) are tuned together.

The future work in this area will be on different type of constraints to define the uncertain expert knowledge in CFSs and CFSs with simple parameterized conjunctions. In addition, the other models of interval type-2 FLSs will be studied and the results will be compared. The expected outcome is that CFSs with parameterized conjunctions are preferable in the applications where the expert knowledge is significant.

APPENDIX A: MATLAB CODE FOR METHOD 1


```
% Sinc Function which has a global maximum at x = 3, y = 3
% Parameterized T-norm (the simplest G-conjunction operators)
clc
clear all
% Input x (X1)
x1 = 0; hx = .5; x2 = 8;
x = (x1:hx:x2)';
nmx=4;
% Input y (X2)
y1 = 0; hy = .5; y2 = 8;
y =(y1:hy:y2)';
nmy=4;
% Partitioning the inputs

[X,Y] = meshgrid(x,y);

% Nonlinear Function

ZD = sinc_new(X).*sinc_new(Y);

% Plotting the Nonlinear Function
figure;
mesh(X,Y,ZD);
title('INPUT')
axis([x1 x2 y1 y2 -1 1])
figure;
subplot(2,2,1);mesh(X,Y,ZD);
title('INPUT')
axis([x1 x2 y1 y2 -1 1])

lz = size(ZD)
nz=lz(1)*lz(2)

% initial gaussian membership parameters for input 1 and input 2

alpha1=0.5*ones(nmx,1);
alpha2=0.5*ones(nmy,1);

for i=1:1:nmx
    p1(i,2) = ((i-1)*((x2-x1)/(nmx-1)))+x1;
    p1(i,1) = (x2-x1)/(2*(nmx-1)*sqrt(-2*log(alpha1(i,1))));
end
for i=1:1:nmy
    p2(i,2) = ((i-1)*((y2-x1)/(nmy-1)))+y1;
    p2(i,1) = (y2-y1)/(2*(nmy-1)*sqrt(-2*log(alpha2(i,1))));
end

mx1 = gaussmf(X,[p1(1,1) p1(1,2)]);
mx2 = gaussmf(X,[p1(2,1) p1(2,2)]);
mx3 = gaussmf(X,[p1(3,1) p1(3,2)]);
mx4 = gaussmf(X,[p1(4,1) p1(4,2)]);

mx=[mx1 mx2 mx3 mx4];

my1 = gaussmf(Y,[p2(1,1) p2(1,2)]);
```

```
my2 = gaussmf(Y,[p2(2,1) p2(2,2)]);
my3 = gaussmf(Y,[p2(3,1) p2(3,2)]);
my4 = gaussmf(Y,[p2(4,1) p2(4,2)]);

my=[my1 my2 my3 my4];


p=ones(16,2);               % initial parameters of operations
r=zeros(16,3);              % initial parameters of right sides of rules:
A=[p r];                    % initial parameters before iterations
lb = zeros(size(A))+0.1;    % lower bounds for parameters
lb(:,3:5)=-50+ lb(:,3:5);   % lower bounds for parameters
ub = 50*ones(size(A));      % upper bounds for parameters

error = []; %errors of training data

%B=indexes of premises in k-th rules:
%B(k,1) - xk; B(k,2) - yk
% k=1: if mx1 and my1 then
% k=2: if mx1 and my2 then...

B=[ 1 1
    1 2
    1 3
    1 4
    2 1
    2 2
    2 3
    2 4
    3 1
    3 2
    3 3
    3 4
    4 1
    4 2
    4 3
    4 4
    ];

%global mx my B X Y ZD lz Z


npar=length(A);
Z=0;
f8=sinc3fminifunnew(A,mx, my, B, X, Y, ZD, lz);
err = f8;   % initial error
serror =err;
global Z serror kiter kc

errstring=num2str(err);             % used for plotting

%===below is optimization !!!===============
error = [];                         % collect errors
options = optimset('MaxFunEvals',250*npar,'MaxIter',250*npar,'TolFun',1e-8);
```

```
sa=size(A);
npar=sa(1)*sa(2);

tic                                % time of beginning
for kc=1:100
    kiter=0
    A1 = fminimax(@sinc3fminifunnew,A,[],[],[],[],lb,ub,[], options,mx, my, B, X, Y,↙
ZD, lz);
    f8=sinc3fminifunnew(A1,mx, my, B, X, Y, ZD, lz);    % error
    error2 = f8
    error=[error error2];  % collect errors
    A=A1;
end
toc % time of end

errstring=num2str(error2);


subplot(2,2,2);mesh(X,Y,Z);title('OUTPUT');
axis([x1 x2 y1 y2 -1 1])

subplot(2,2,3);
plot(error)
title('Error curve (n=80 parameters)')
xlabel('Number of iterations')
f88=Z-ZD;
titstr=['Error surface for OUTPUT, RMSE = ' errstring];
subplot(2,2,4);mesh(X,Y,f88);
title(titstr);
axis([x1 x2 y1 y2 -1 1])
figure;
mesh(X,Y,Z);title('OUTPUT');
axis([x1 x2 y1 y2 -1 1])
% Plotting Initial membership functions
x = 0:.1:8;y=0:.1:8;
% INPUT x
figure;
for j=1:1
    subplot(1,2,1);
    plot(x,gaussmf(x,[p1(j,1) p1(j,2)]),'r')
    hold on
    title('Initial Mfs of Input X1')
end
for j=2:2
    subplot(1,2,1);
    plot(x,gaussmf(x,[p1(j,1) p1(j,2)]),'b')
    hold on
    title('Initial Mfs of Input X1')
end
for j=3:3
    subplot(1,2,1);
    plot(x,gaussmf(x,[p1(j,1) p1(j,2)]),'c')
    hold on
    title('Initial Mfs of Input X1')
end
```

```
for j=4:4
    subplot(1,2,1);
    plot(x,gaussmf(x,[p1(j,1) p1(j,2)]),'k')
    hold on
    title('Initial Mfs of Input X1')
end
% INPUT y
for j=1:1
    subplot(1,2,2);
    plot(y,gaussmf(y,[p2(j,1) p2(j,2)]),'r')
    hold on
    title('Initial Mfs of Input X2')
end
for j=2:2
    subplot(1,2,2);
    plot(y,gaussmf(y,[p2(j,1) p2(j,2)]),'b')
    hold on
    title('Initial Mfs of Input X2')
end
for j=3:3
    subplot(1,2,2);
    plot(y,gaussmf(y,[p2(j,1) p2(j,2)]),'c')
    hold on
    title('Initial Mfs of Input X2')
end
for j=4:4
    subplot(1,2,2);
    plot(y,gaussmf(y,[p2(j,1) p2(j,2)]),'k')
    hold on
    title('Initial Mfs of Input X2')
end
```

```
function f = sinc3fminifunnew(A,mx, my, B, X, Y, ZD, lz)


global Z serror kiter kc
kiter=kiter+1; % number of iterations

sz=lz(2);
nz=lz(1)*lz(2);

w=zeros(lz); sw=w; swf=w;%firing values of rules

%In Rule k:
%If x=x1 and y=y1 then f=A(k,3)*x+A(k,4)*y+A(k,5)
%firing value of rule calculated as:
%mx^A(k,1)*my^A(k,2)

for k = 1:16 % 16 rules
   %firing value w of k-th rule:
   k1=B(k,1);                   %  index of mx1...mx4
   i1=1+(k1-1)*sz;              % columns of mxk1 in mx
   j1=k1*sz;                    % --"--"--"
   k2=B(k,2);                   % index of my1...my4
   i2=1+(k2-1)*sz;              % columns of myk2 in my
   j2=k2*sz;                    % --"--"--"
   w=(mx(:,i1:j1).^A(k,1)).*(my(:,i2:j2).^A(k,2));
   sw=sw+w;                     %sum of weights
   F=X*A(k,3)+Y*A(k,4)+A(k,5);  %right sides
   swf=swf+w.*F;                %sum of weighted outputs
end

Z=swf./sw; %total output for each point
ff=Z-ZD;
f = sqrt(sum(sum(ff.*ff))./nz);
disp(['kc= ' num2str(kc) ' kit= ' num2str(kiter) ' err= ' num2str(f)])

serror=[serror f];
```

APPENDIX B: MATLAB CODE FOR METHOD 2


```
% Sinc Function which has a global maximum at x = 3, y = 3
% CFSs method2
clc
clear
% Input x (X1)
x1 = 0; hx = .5; x2 = 8;
x = (x1:hx:x2)';
nmx=4;
% Input y (X2)
y1 = 0; hy = .5; y2 = 8;
y =(y1:hy:y2)';
nmy=4;
% Partitioning the inputs

[X,Y] = meshgrid(x,y);

% Nonlinear Function

ZD = sinc_new(X).*sinc_new(Y);

% Plotting the Nonlinear Function
figure;
mesh(X,Y,ZD);
title('INPUT')
axis([x1 x2 y1 y2 -1 1])
figure;
subplot(2,2,1);mesh(X,Y,ZD);
title('INPUT')
axis([x1 x2 y1 y2 -1 1])

lz = size(ZD)
nz=lz(1)*lz(2)

% initial gaussian membership parameters for input 1 and input 2

alpha1=0.5*ones(nmx,1);
alpha2=0.5*ones(nmy,1);

for i=1:1:nmx
    p1(i,2) = ((i-1)*((x2-x1)/(nmx-1)))+x1;
    p1(i,1) = (x2-x1)/(2*(nmx-1)*sqrt(-2*log(alpha1(i,1))));
end
for i=1:1:nmy
    p2(i,2) = ((i-1)*((y2-y1)/(nmy-1)))+y1;
    p2(i,1) = (y2-y1)/(2*(nmy-1)*sqrt(-2*log(alpha2(i,1))));
end

p = [p1;p2]; % Antecedent membership function parameters


r=zeros(8,6);              % initial parameters of right sides of rules:
A=[p r];                    % initial parameters before iterations
lb = zeros(size(A))+0.1;    % lower bounds for parameters
lb(:,1:1)=p(:,1:1)*0.60;
lb(1,2)=-0.4;lb(5,2)=lb(1,2);
```

```
lb(2,2)=1.6;lb(6,2)=lb(2,2);
lb(3,2)=3.2;lb(7,2)=lb(3,2);
lb(4,2)=4.8;lb(8,2)=lb(4,2);
lb(:,3:8)=-50+ lb(:,3:8);
ub(:,1:1)=p(:,1:1)*1.40;    % upper bounds for parameters
ub(1,2)=0.4;ub(5,2)=ub(1,2);
ub(2,2)=3.7334;ub(6,2)=ub(2,2);
ub(3,2)=7.4666;ub(7,2)=ub(3,2);
ub(4,2)=11.2;ub(8,2)=ub(4,2);
ub(:,3:8)=ones(8,6)*50;

error = []; %errors of training data

%B=indexes of premises in k-th rules:


B=[ 1 1
    1 2
    1 3
    1 4
    2 1
    2 2
    2 3
    2 4
    3 1
    3 2
    3 3
    3 4
    4 1
    4 2
    4 3
    4 4
    ];

global B X Y ZD Z lz


npar=length(A);
Z=0;
f8=sinc8334funnew(A, B, X, Y, ZD, lz);
err = f8;   % initial error
serror =err;
global Z serror kiter kc

errstring=num2str(err);             % used for plotting

%===below is optimization !!!================
error = [];                         % collect errors
options = optimset('MaxFunEvals',500*npar,'MaxIter',500*npar,'TolFun',1e-8);

sa=size(A);
npar=sa(1)*sa(2);

tic                                 % time of beginning
for kc=1:100
```

```
    kiter=0
    A1 = fminimax(@sinc8334funnew,A,[],[],[],[],lb,ub,[], options,B, X, Y, ZD, lz);
    f8=sinc8334funnew(A1, B, X, Y, ZD, lz);    % error
    error2 = f8
    error=[error error2];  % collect errors
    A=A1;
end
toc % time of end

errstring=num2str(error2);


subplot(2,2,2);mesh(X,Y,Z);title('OUTPUT');
axis([x1 x2 y1 y2 -1 1])

subplot(2,2,3);
plot(error)
title('Error curve (n=64 parameters)')
xlabel('Number of iterations')
f88=Z-ZD;
titstr=['Error surface for OUTPUT, RMSE = ' errstring];
subplot(2,2,4);mesh(X,Y,f88);
title(titstr);
axis([x1 x2 y1 y2 -1 1])
figure;
mesh(X,Y,Z);title('OUTPUT');
axis([x1 x2 y1 y2 -1 1])
% Plotting Initial membership functions
x = 0:.1:8;y=0:.1:8;
% INPUT x
figure;
for j=1:1
    subplot(1,2,1);
    plot(x,gaussmf(x,[p1(j,1) p1(j,2)]),'r')
    hold on
    title('Initial Mfs of Input X1')
end
for j=2:2
    subplot(1,2,1);
    plot(x,gaussmf(x,[p1(j,1) p1(j,2)]),'b')
    hold on
    title('Initial Mfs of Input X1')
end
for j=3:3
    subplot(1,2,1);
    plot(x,gaussmf(x,[p1(j,1) p1(j,2)]),'c')
    hold on
    title('Initial Mfs of Input X1')
end
for j=4:4
    subplot(1,2,1);
    plot(x,gaussmf(x,[p1(j,1) p1(j,2)]),'k')
    hold on
    title('Initial Mfs of Input X1')
end
```

```
% INPUT y
for j=1:1
    subplot(1,2,2);
    plot(y,gaussmf(y,[p2(j,1) p2(j,2)]),'r')
    hold on
    title('Initial Mfs of Input X2')
end
for j=2:2
    subplot(1,2,2);
    plot(y,gaussmf(y,[p2(j,1) p2(j,2)]),'b')
    hold on
    title('Initial Mfs of Input X2')
end
for j=3:3
    subplot(1,2,2);
    plot(y,gaussmf(y,[p2(j,1) p2(j,2)]),'c')
    hold on
    title('Initial Mfs of Input X2')
end
for j=4:4
    subplot(1,2,2);
    plot(y,gaussmf(y,[p2(j,1) p2(j,2)]),'k')
    hold on
    title('Initial Mfs of Input X2')
end
figure;
for j=1:1
    subplot(1,2,1);plot(x,gaussmf(x,[A(j,1) A(j,2)]),'r')
    title('Final Mfs of Input X1')
    hold on
end
for j=2:2
    subplot(1,2,1);plot(x,gaussmf(x,[A(j,1) A(j,2)]),'b')
    title('Final Mfs of Input X1')
    hold on
end
for j=3:3
    subplot(1,2,1);plot(x,gaussmf(x,[A(j,1) A(j,2)]),'c')
    hold on
    title('Final Mfs of Input X1')
end
for j=4:4
    subplot(1,2,1);plot(x,gaussmf(x,[A(j,1) A(j,2)]),'k')
    hold on
    title('Final Mfs of Input X1')
end
axis([0 8 0 1])

for j=5:5
    subplot(1,2,2);plot(y,gaussmf(y,[A(j,1) A(j,2)]),'r')
    title('Final Mfs of Input X2')
    hold on
end
for j=6:6
    subplot(1,2,2);plot(y,gaussmf(y,[A(j,1) A(j,2)]),'b')
```

```
    title('Final Mfs of Input X2')
    hold on
end

for j=7:7
    subplot(1,2,2);plot(y,gaussmf(y,[A(j,1) A(j,2)]),'c')
    hold on
    title('Final Mfs of Input X2')
end
for j=8:8
    subplot(1,2,2);plot(y,gaussmf(y,[A(j,1) A(j,2)]),'k')
    hold on
    title('Final Mfs of Input X2')
end
axis([0 8 0 1])
```

```
function f = sinc8334funnew(A, B, X, Y, ZD,lz)


global Z serror kiter kc
kiter=kiter+1; % number of iterations



nz=lz(1)*lz(2);

w=zeros(lz); sw=w; swf=w;%firing values of rules


for k = 1:16 % 16 Rules
    %firing value w of k-th rule:
    k1=B(k,1);
    k2=B(k,2);
    w=(gaussmf(X,[A(k1,1) A(k1,2)]).*(gaussmf(Y,[A(k2+4,1) A(k2+4,2)])));
    sw=sw+w;                      %sum of weights
    if k<=8
        F=X*A(k,3)+Y*A(k,4)+A(k,5); %right sides
    elseif k==9 | k==10 | k==11 | k==12 | k==13 | k==14 | k==15 | k==16

        F=X*A(k-8,6)+Y*A(k-8,7)+A(k-8,8);

    end
    swf=swf+w.*F;                 %sum of weighted outputs
end

Z=swf./sw; %total output for each point

ff=Z-ZD;
f = sqrt(sum(sum(ff.*ff))./nz);
disp(['kc= ' num2str(kc) ' kit= ' num2str(kiter) ' err= ' num2str(f)])

serror=[serror f];
```

APPENDIX C: MATLAB CODE FOR METHOD 3


```
% Sinc Function which has a global maximum at x = 3, y = 3
% CFS with Parameterized method3
clc
clear all
% Input x (X1)
x1 = 0; hx = .5; x2 = 8;
x = (x1:hx:x2)';
nmx=4;
% Input y (X2)
y1 = 0; hy = .5; y2 = 8;
y =(y1:hy:y2)';
nmy=4;
% Partitioning the inputs

[X,Y] = meshgrid(x,y);

% Nonlinear Function

ZD = sinc_new(X).*sinc_new(Y);

% Plotting the Nonlinear Function
figure;
subplot(2,2,1);mesh(X,Y,ZD);
title('INPUT')
axis([x1 x2 y1 y2 -1 1])

lz = size(ZD)
nz=lz(1)*lz(2)

% initial gaussian membership parameters for input 1 and input 2

alpha1=0.5*ones(nmx,1);
alpha2=0.5*ones(nmy,1);

for i=1:1:nmx
    p1(i,2) = ((i-1)*((x2-x1)/(nmx-1)))+x1;
    p1(i,1) = (x2-x1)/(2*(nmx-1)*sqrt(-2*log(alpha1(i,1))));
end
for i=1:1:nmy
    p2(i,2) = ((i-1)*((y2-y1)/(nmy-1)))+y1;
    p2(i,1) = (y2-y1)/(2*(nmy-1)*sqrt(-2*log(alpha2(i,1))));
end

p = [p1;p2]; % Antecedent membership function parameters

param1=ones(8,2); % initial parameters of parameterized conjunctions
param2=ones(8,2); % initial parameters of parameterized conjunctions
r=zeros(8,6);              % initial parameters of right sides of rules:
A=[p r param1 param2];                    % initial parameters before iterations
lb = zeros(size(A))+0.1;    % lower bounds for parameters
lb(:,1:1)=p(:,1:1)*0.60;
lb(1,2)=-0.4;lb(5,2)=lb(1,2);
lb(2,2)=1.6;lb(6,2)=lb(2,2);
lb(3,2)=3.2;lb(7,2)=lb(3,2);
lb(4,2)=4.8;lb(8,2)=lb(4,2);
```

```
lb(:,3:8)=-50+ lb(:,3:8);
ub(:,1:1)=p(:,1:1)*1.40;    % upper bounds for  parameters
ub(1,2)=0.4;ub(5,2)=ub(1,2);
ub(2,2)=3.7334;ub(6,2)=ub(2,2);
ub(3,2)=7.4666;ub(7,2)=ub(3,2);
ub(4,2)=11.2;ub(8,2)=ub(4,2);
ub(:,3:12)=ones(8,10)*50;

error = []; %errors of training data

% B=indexes of premises in k-th rules:

B=[ 1 1
    1 2
    1 3
    1 4
    2 1
    2 2
    2 3
    2 4
    3 1
    3 2
    3 3
    3 4
    4 1
    4 2
    4 3
    4 4
    ];

global B X Y ZD Z lz


npar=length(A);
Z=0;
f8=sincparam8334funnew(A, B, X, Y, ZD, lz);
err = f8;   % initial error
serror =err;
global Z serror kiter kc

errstring=num2str(err);             % used for plotting

%===below is optimization !!!===============
error = [];                         % collect errors
options = optimset('MaxFunEvals',500*npar,'MaxIter',500*npar,'TolFun',1e-8);

sa=size(A);
npar=sa(1)*sa(2);

tic                                 % time of beginning
for kc=1:100
    kiter=0
    A1 = fminimax(@sincparam8334funnew,A,[],[],[],[],lb,ub,[], options,B, X, Y, ZD,↙
lz);
    f8=sincparam8334funnew(A1, B, X, Y, ZD, lz);
```

```
    error2 = f8
    error=[error error2];  % collect errors
    A=A1;
end
toc % time of end

errstring=num2str(error2);


subplot(2,2,2);mesh(X,Y,Z);title('OUTPUT');
axis([x1 x2 y1 y2 -1 1])

subplot(2,2,3);
plot(error)
title('Error curve (n=96 parameters)')
xlabel('Number of iterations')
f88=Z-ZD;
titstr=['Error surface for OUTPUT, RMSE = ' errstring];
subplot(2,2,4);mesh(X,Y,f88);
title(titstr);
axis([x1 x2 y1 y2 -1 1])
figure;
mesh(X,Y,Z);title('OUTPUT');
axis([x1 x2 y1 y2 -1 1])

% Plotting Initial membership functions
x = 0:.1:8;y=0:.1:8;
% INPUT x
figure;
for j=1:1
    subplot(1,2,1);
    plot(x,gaussmf(x,[p1(j,1) p1(j,2)]),'r')
    hold on
    title('Initial Mfs of Input X1')
end
for j=2:2
    subplot(1,2,1);
    plot(x,gaussmf(x,[p1(j,1) p1(j,2)]),'b')
    hold on
    title('Initial Mfs of Input X1')
end
for j=3:3
    subplot(1,2,1);
    plot(x,gaussmf(x,[p1(j,1) p1(j,2)]),'c')
    hold on
    title('Initial Mfs of Input X1')
end
for j=4:4
    subplot(1,2,1);
    plot(x,gaussmf(x,[p1(j,1) p1(j,2)]),'k')
    hold on
    title('Initial Mfs of Input X1')
end
% INPUT y
for j=1:1
```

```
    subplot(1,2,2);
    plot(y,gaussmf(y,[p2(j,1) p2(j,2)]),'r')
    hold on
    title('Initial Mfs of Input X2')
end
for j=2:2
    subplot(1,2,2);
    plot(y,gaussmf(y,[p2(j,1) p2(j,2)]),'b')
    hold on
    title('Initial Mfs of Input X2')
end
for j=3:3
    subplot(1,2,2);
    plot(y,gaussmf(y,[p2(j,1) p2(j,2)]),'c')
    hold on
    title('Initial Mfs of Input X2')
end
for j=4:4
    subplot(1,2,2);
    plot(y,gaussmf(y,[p2(j,1) p2(j,2)]),'k')
    hold on
    title('Initial Mfs of Input X2')
end
% Final Mfs Input 2
figure;
for j=1:1
    subplot(1,2,1);plot(x,gaussmf(x,[A(j,1) A(j,2)]),'r')
    title('Final Mfs of Input X1')
    hold on
end
for j=2:2
    subplot(1,2,1);plot(x,gaussmf(x,[A(j,1) A(j,2)]),'b')
    title('Final Mfs of Input X1')
    hold on
end
for j=3:3
    subplot(1,2,1);plot(x,gaussmf(x,[A(j,1) A(j,2)]),'c')
    hold on
    title('Final Mfs of Input X1')
end
for j=4:4
    subplot(1,2,1);plot(x,gaussmf(x,[A(j,1) A(j,2)]),'k')
    hold on
    title('Final Mfs of Input X1')
end
axis([0 8 0 1])
% Final Mfs Input 2
for j=5:5
    subplot(1,2,2);plot(y,gaussmf(y,[A(j,1) A(j,2)]),'r')
    title('Final Mfs of Input X2')
    hold on
end
for j=6:6
    subplot(1,2,2);plot(y,gaussmf(y,[A(j,1) A(j,2)]),'b')
    title('Final Mfs of Input X2')
```

```
    hold on
end

for j=7:7
    subplot(1,2,2);plot(y,gaussmf(y,[A(j,1) A(j,2)]),'c')
    hold on
    title('Final Mfs of Input X2')
end
for j=8:8
    subplot(1,2,2);plot(y,gaussmf(y,[A(j,1) A(j,2)]),'k')
    hold on
    title('Final Mfs of Input X2')
end
axis([0 8 0 1])
```

```
function f = sincparam8334funnew(A, B, X, Y, ZD,lz)

global Z serror kiter kc
kiter=kiter+1; % number of iterations

nz=lz(1)*lz(2);

w=zeros(lz); sw=w; swf=w;%firing values of rules

% In Rule k:
% If x=x1 and y=y1 then f=Ak3*x+A4*y+A5
% firing value of rule calculated as:
% w = m(Ak1)*m(Ak2)

for k = 1:16 % 16 rules
    %firing value w of k-th rule:
    k1=B(k,1);
    k2=B(k,2);

    if k<=8
        w=(gaussmf(X,[A(k1,1) A(k1,2)]).^A(k,9)).*(gaussmf(Y,[A(k2+4,1) A(k2+4,2)]).↙
^A(k,10));
        sw=sw+w;                    %sum of weights
        F=X*A(k,3)+Y*A(k,4)+A(k,5); %right sides
    elseif k==9 | k==10 | k==11 | k==12 | k==13 | k==14 | k==15 | k==16
         w=(gaussmf(X,[A(k1,1) A(k1,2)]).^A(k-8,11)).*(gaussmf(Y,[A(k2+4,1) A(k2+4,↙
2)]).^A(k-8,12));
         sw=sw+w;                    %sum of weights
        F=X*A(k-8,6)+Y*A(k-8,7)+A(k-8,8);

    end
    swf=swf+w.*F;                %sum of weighted outputs
end

Z=swf./sw; %total output for each point

ff=Z-ZD;
f = sqrt(sum(sum(ff.*ff))./nz);
disp(['kc= ' num2str(kc) ' kit= ' num2str(kiter) ' err= ' num2str(f)])
serror=[serror f];
```

APPENDIX D: MATLAB CODE FOR METHOD 4


```
% Sinc Function which has a global maximum at x = 3, y = 3
% by fminmax in Matlab 7.1
% IT2
clc
clear all

x1=0; x2=8; hx=.5;
x = (x1:hx:x2)';
lx=length(x);
nmx=4;              % number of membership values on X (X1)

y1=0; y2=8; hy=.5;
y = (y1:hy:y2)';

ly= length(y);
nmy= 4;             % number of membership values on Y (X2)

[X,Y]= meshgrid(x,y);
ZD = sinc_new(X).*sinc_new(Y);  % my sinc function

optim=1;            % we use optim=0 to check function and not to optimize
                    % we use optim=1 to optimize

if optim ==0
    figure;
    mesh(X,Y,ZD);title('INPUT')
    axis([x1 x2 y1 y2 -1 1])

else
    figure;
    subplot(2,2,1);mesh(X,Y,ZD);title('INPUT')
    axis([x1 x2 y1 y2 -1 1])

    lz=size(ZD)
    nz=lz(1)*lz(2)


    %upper membership function parameters for input 1

    mf_upper_1=[1.5854 0;
              1.5854 2.6667;
              1.5854 5.3333;
              1.5854 8];

   %upper membership function parameters for input 2
    mf_upper_2=[1.5854 0;
              1.5854 2.6667;
              1.5854 5.3333;
              1.5854 8];



    %lower membership function parameters for input 1
   mf_lower_1=[0.6794 0;
              0.6794 2.6667;
```

```
             0.6794 5.3333;
             0.6794 8];
  %lower membership function parameters for input 2
  mf_lower_2=[0.6794 0;
            0.6794 2.6667;
            0.6794 5.3333;
            0.6794 8];
  mf_lower_11=0.6794*ones(4,1);
  mf_lower_22=0.6794*ones(4,1);
  mf_up =[mf_upper_1;mf_upper_2];
  mf_low = [mf_lower_11;mf_lower_22 ];
    r=zeros(8,6);
    A=[mf_up mf_low r];% initial parameters of right sides of rules:

    lb = zeros(size(A))-49.9;    % low bounds for parameters
    lb(:,1:1) = -49.9*ones(8,1);
    lb(:,2:2) = zeros(8,1);
    lb(:,3:3) = -49.9*ones(8,1);
   ub = 50*ones(size(A));      % upper bounds for parameters
    ub(:,1:1) = 50*ones(8,1);
    ub(:,2:2) = 8*ones(8,1);
    ub(:,3:3) = 50*ones(8,1);
    error = []; %errors of training data

    %B=indexes of premises in k-th rules:


   B=[ 1 1
    1 2
    1 3
    1 4
    2 1
    2 2
    2 3
    2 4
    3 1
    3 2
    3 3
    3 4
    4 1
    4 2
    4 3
    4 4
    ];


    global B X Y ZD Z lz

    npar=length(A);
    Z=0;                % function value in b8funnew
    f8=sincT2B3funnew(A,B, X, Y, ZD, lz);
    err = f8;   % initial error
    serror =err;
    global Z serror kiter kc
```

```
    errstring=num2str(err);             % used for plotting

    %===below is optimization !!!===============
    error = [];                         % collect errors
    options = optimset('MaxFunEvals',300*npar,'MaxIter',300*npar,'TolFun',1e-8);

    sa=size(A);
    npar=sa(1)*sa(2);
    %options(14)=10*npar;               % by default 100*npar

    tic                                 % time of beginning
    for kc=1:100
        kiter=0
        A1 = fminimax(@sincT2B3funnew,A,[],[],[],[],lb,ub, [],options,B, X, Y, ZD,↙
lz);
        f8=sincT2B3funnew(A1, B, X, Y, ZD, lz);    % error
        error2 = f8
        error=[error error2];  % collect errors
        A=A1;
    end
    toc % time of end

    errstring=num2str(error2);


    subplot(2,2,2);mesh(X,Y,Z);title('OUTPUT');
    axis([x1 x2 y1 y2 -1 1])

    subplot(2,2,3);
    plot(error)
    title('Error curve')
    xlabel('Number of iterations')
    f88=Z-ZD;
    titstr=['Error surface for OUTPUT, RMSE = ' errstring];
    subplot(2,2,4);mesh(X,Y,f88);
    title(titstr);
    axis([x1 x2 y1 y2 -1 1])
end % optim
figure;
mesh(X,Y,Z);title('OUTPUT');
axis([x1 x2 y1 y2 -1 1])

x=0:.01:8;
y=0:.01:8;
x=x';y=y';
figure;
plot(x,gaussmf(x,[1.1324 0]),'m')
hold on
plot(x,gaussmf(x,[1.1324 2.6667]),'m')
hold on
plot(x,gaussmf(x,[1.1324 5.3333]),'m')
hold on
plot(x,gaussmf(x,[1.1324 8]),'m')

figure;
```

```
for j=1:1
    subplot(1,2,1);plot2d1(gaussmf(x,[mf_upper_1(j,1) mf_upper_1(j,2)])',gaussmf(x,↙
[mf_lower_1(j,1) mf_lower_1(j,2)])',x')
    hold on
    title('Initial Mfs of Input X1')
    plot(x,gaussmf(x,[1.1324 0]),'m')
    hold on
end
for j=2:2
    subplot(1,2,1);plot2d11(gaussmf(x,[mf_upper_1(j,1) mf_upper_1(j,2)])',gaussmf(x,↙
[mf_lower_1(j,1) mf_lower_1(j,2)])',x')
    hold on
    title('Initial Mfs of Input X1')
    plot(x,gaussmf(x,[1.1324 2.6667]),'m')
    hold on
end
for j=3:3
    subplot(1,2,1);plot2d111(gaussmf(x,[mf_upper_1(j,1) mf_upper_1(j,2)])',gaussmf(x,↙
[mf_lower_1(j,1) mf_lower_1(j,2)])',x')
    hold on
    title('Initial Mfs of Input X1')
    plot(x,gaussmf(x,[1.1324 5.3333]),'m')
    hold on
end
for j=4:4
    subplot(1,2,1);plot2d1112(gaussmf(x,[mf_upper_1(j,1) mf_upper_1(j,2)])',gaussmf↙
(x,[mf_lower_1(j,1) mf_lower_1(j,2)])',x')
    hold on
    title('Initial Mfs of Input X1')
    plot(x,gaussmf(x,[1.1324 8]),'m')
    hold on
end

for j=1:1
    subplot(1,2,2);plot2d1(gaussmf(x,[mf_upper_2(j,1) mf_upper_2(j,2)])',gaussmf(x,↙
[mf_lower_2(j,1) mf_lower_2(j,2)])',x')
    hold on
    plot(x,gaussmf(x,[1.1324 0]),'m')
    hold on
    title('Initial Mfs of Input X2')
end
for j=2:2
    subplot(1,2,2);plot2d11(gaussmf(x,[mf_upper_2(j,1) mf_upper_2(j,2)])',gaussmf(x,↙
[mf_lower_2(j,1) mf_lower_2(j,2)])',x')
    hold on
    plot(x,gaussmf(x,[1.1324 2.6667]),'m')
    hold on
    title('Initial Mfs of Input X2')
end
for j=3:3
    subplot(1,2,2);plot2d111(gaussmf(x,[mf_upper_2(j,1) mf_upper_2(j,2)])',gaussmf(x,↙
[mf_lower_2(j,1) mf_lower_2(j,2)])',x')
    hold on
    plot(x,gaussmf(x,[1.1324 5.3333]),'m')
    title('Initial Mfs of Input X2')
```

```
end
for j=4:4
    subplot(1,2,2);plot2d1112(gaussmf(x,[mf_upper_2(j,1) mf_upper_2(j,2)])',gaussmf↙
(x,[mf_lower_2(j,1) mf_lower_2(j,2)])',x')
    hold on
    title('Initial Mfs of Input X2')
    plot(x,gaussmf(x,[1.1324 8]),'m')
    hold on
end

figure;
for j=1:1
     subplot(1,2,1);plot2d1(gaussmf(x,[A(j,1) A(j,2)])',gaussmf(x,[A(j,3) A(j,2)])',↙
x')
    hold on
    title('Final Mfs of Input X1')
end
for j=2:2
     subplot(1,2,1);plot2d11(gaussmf(x,[A(j,1) A(j,2)])',gaussmf(x,[A(j,3) A(j,2)])',↙
x')
    hold on
    title('Final Mfs of Input X1')
end
for j=3:3
     subplot(1,2,1);plot2d111(gaussmf(x,[A(j,1) A(j,2)])',gaussmf(x,[A(j,3) A(j,↙
2)])',x')
    hold on
    title('Final Mfs of Input X1')
end
for j=4:4
     subplot(1,2,1);plot2d1112(gaussmf(x,[A(j,1) A(j,2)])',gaussmf(x,[A(j,3) A(j,↙
2)])',x')
    hold on
    title('Final Mfs of Input X1')
end
for j=5:5
    subplot(1,2,2);plot2d1(gaussmf(y,[A(j,1) A(j,2)])',gaussmf(y,[A(j,3) A(j,2)])',↙
y')
    hold on
    title('Final Mfs of Input X2')
end
for j=6:6
    subplot(1,2,2);plot2d11(gaussmf(y,[A(j,1) A(j,2)])',gaussmf(y,[A(j,3) A(j,2)])',↙
y')
    hold on
    title('Final Mfs of Input X2')
end
for j=7:7
    subplot(1,2,2);plot2d111(gaussmf(y,[A(j,1) A(j,2)])',gaussmf(y,[A(j,3) A(j,2)])',↙
y')
    hold on
    title('Final Mfs of Input X2')
end
for j=8:8
    subplot(1,2,2);plot2d1112(gaussmf(y,[A(j,1) A(j,2)])',gaussmf(y,[A(j,3) A(j,↙
```

```
2)])',y')
    hold on
    title('Final Mfs of Input X2')
end
```

```
function f = sincT2B3funnew(A, B, X, Y, ZD,lz)


global Z serror kiter kc
kiter=kiter+1; % number of iterations


sz=lz(2);
nz=lz(1)*lz(2);

w_upper=zeros(lz); sw_upper=w_upper; swf_upper=w_upper;%firing values of rules
w_lower=zeros(lz); sw_lower=w_lower; swf_lower=w_lower;%firing values of rules
% In Rule k:
% If x=x1 and y=y1 then f=Ak3*x+A4*y+A5
% firing value of rule calculated as:
% w = m(Ak1)*m(Ak2)

for k = 1:16 % 16 rules
    %firing value w of k-th rule:
    k1=B(k,1);
    k2=B(k,2);
    w_upper=(gaussmf(X,[A(k1,1) A(k1,2)]).*gaussmf(Y,[A(k2+4,1) A(k2+4,2)]));
    w_lower=(gaussmf(X,[A(k1,3) A(k1,2)]).*gaussmf(Y,[A(k2+4,3) A(k2+4,2)]));
    sw_upper=sw_upper+w_upper;                    %sum of weights
    sw_lower=sw_lower+w_lower;
    if k<=8
        F=X*A(k,4)+Y*A(k,5)+A(k,6); %right sides
    elseif k==9 | k==10 | k==11 | k==12 | k==13 | k==14 | k==15 | k==16

        F=X*A(k-8,7)+Y*A(k-8,8)+A(k-8,9);

    end
    swf_upper=swf_upper+w_upper.*F;                %sum of weighted outputs
    swf_lower=swf_lower+w_lower.*F;
end

Z_upper=swf_upper; %total output for each point
Z_lower=swf_lower; %total output for each point
Z=(Z_upper+Z_lower)/2;
ff=Z-ZD;
f = sqrt(sum(sum(ff.*ff))./nz);
disp(['kc= ' num2str(kc) ' kit= ' num2str(kiter) ' err= ' num2str(f)])

serror=[serror f];
```

REFERENCES

1. Zadeh, L. A., "Soft Computing and Fuzzy Logic", *IEEE Software*, Vol. 11, No. 6, pp. 48-56, November 1994.

2. *fuzzyTECH® 5.3 User's Manual*, ©1999 INFORM Gbmh / Inform Software Corp., July 15 1999.

3. Jang, J.-S. R., C.-T. Sun and E. Mizutani,*Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Upper Saddle River, NJ: Prentice Hall, 1997.

4. Mendel, J. M., *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*, Upper Saddle River, NJ: Prentice Hall, 2001.

5. Batyrshin, I. and O. Kaynak, "Parametric Classes of Generalized Conjunction and Disjunction Operations for Fuzzy Modeling", *IEEE Trans. on Fuzzy Systems*, Vol. 7, No. 5, pp. 586-596, October 1999.

6. Batyrshin, I., O. Kaynak and I. Rudas, "Fuzzy Modelling Based on Generalized Conjunction Operations", *IEEE Trans. on Fuzzy Systems*, Vol. 10, No. 5, pp. 678-683, October 2002.

7. Zadeh, L. A., "The Concept of a Linguistic Variable and its Appplication to Approximate Reasoning", *Information Sciences*, Vol. 8, pp. 199-249, 1975.

8. John, R. J. and S. Coupland, "Type-2 Fuzzy Logic: A Historical View", *IEEE Computational Intelligence Magazine*, Vol. 2, No. 1, pp. 57-62, February 2007.

9. Mizumoto, M. and K. Tanaka, "Fuzzy Sets of Type 2 Under Algebraic Product and Algebraic Sum", *Fuzzy Sets and Systems*, Vol. 5, pp. 277-290, 1981.

10. Mizumoto, M. and H.-J. Zimmerman, "Some Properties of Fuzzy Set of Type 2",

Information and Control, Vol. 31, pp. 312-340, 1976.

11. Dubois, D. and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, New York, 1982.

12. Gorzalczany, M. B., "A Method of Inference in Approximate Reasoning Based on Interval-Valued Fuzzy Sets", *Fuzzy Sets and Systems*, Vol. 21, pp. 1-17, 1987.

13. Turksen, I. B., "Fuzzy Normal Forms", *Fuzzy Sets and Systems*, Vol. 69, pp. 319-346, 1995.

14. Schwarz, D. G., "The Case for An Interval-Based Representation of Linguistic Truth", *Fuzzy Sets and Systems*, Vol. 17, pp. 153-165, 1985.

15. Klir G. J. and T. A. Folger, *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall, 1988.

16. Bustince, H., "Indicator of Inclusion Grade for Interval-Valued Fuzzy Sets. Application to Approximate Reasoning Based on Interval-Valued Fuzzy Sets", *International Journal of Approximate Reasoning*, Vol. 23, No. 3, pp. 137-209, 2000.

17. Liang, Q. and J. M. Mendel, "Interval Type-2 Fuzzy Logic Systems: Theory and Design", *IEEE Transactions on Fuzzy Systems*, Vol. 8, pp. 535-549, 2000.

18. Karnik, N. N. and J. M. Mendel, "Introduction to Type-2 Fuzzy Logic Systems", *In Proc. IEEE World Congress on Computational Intelligence*, pp. 915-920, Anchorage, Alaska, USA, 1998.

19. Karnik, N. N. and J. M. Mendel, "Type-2 Fuzzy Logic Systems: Type Reduction", *In Proceedings IEEE Systems, Man and Cybernatics*, pp. 2046-2051, 1998.

20. Karnik, N. N. and J. M. Mendel, "Centroid of A Type-2 Fuzzy Set", *Information Sciences*, Vol. 132, pp. 195-220, 2001.

21. John, R. I., "Type-2 Fuzzy Sets for Community Transport Scheduling", *In Proceedings of the Fourth European Congress on Intelligent Techniques and Soft Computing-EUFIT'96*, Vol. 2, pp. 1369-1372, 1996.

22. John, R. I., "Type-2 Fuzzy Sets for Knowledge Representation and Inferencing", Research Monograph 10, *School of Computing Sciences*, De Montfort University, 1998.

23. Karnik, N. N. and J. M. Mendel, "Application of Type-2 Fuzzy Logic System to Forecasting of Time Series", *Information Sciences*, Vol. 120, pp. 89-111, 1999.

24. Zadeh, L. A., "Fuzzy Logic = Computing with Words", *IEEE Transactions on Fuzzy Systems*, Vol. 4, pp. 103-111, 1996.

25. Mendel, J. M., "The Perceptual Computer: An Architecture for Computing with Words", *In Proc. FUZZ-IEEE 2001*, Melbbourne, Australia, 2001.

26. Mendel, J. M., "Fuzzy Sets for Words: A New Beginning", *In Proc. FUZZ-2003*, pp. 37-42, St. Louis, MO, USA, 2003.

27. Turksen, I. B., "Type 2 Representation and Reasoning for CWW", *Fuzzy Sets and Systems*, Vol. 127, pp. 17-36, 2002.

28. Mendel, J. M. and R. I. John, "Type-2 Fuzzy Sets Made Simple", *IEEE Transaction on Fuzzy Systems*, Vol. 10, No. 2, pp. 117-127, 2002.

29. Wu, H. and J. M. Mendel, "Introduction to Uncertainty Bounds and Their Use in The Design of Interval Type-2 Fuzzy Logic Systems", *In Proceedings of FUZZ-IEEE 2001*, Melbourne, Australia, 2001.

30. Wu, H. and J. M. Mendel, "Uncertainty Bounds and Their Use in The Design of Interval Type-2 Fuzzy Logic Systems", *IEEE Transactions on Fuzzy Systems*, pp. 622-639, Oct. 2002.

31. Melin, P. and O. Castillo, "Intelligent Control of Non-Linear Dynamic Plants Using Type-2 Fuzzy Logic and Neural Networks", *In Proc., FUZZ-IEEE 2004*, Budapest, Hungary, July 2004.

32. Hagras, H., "A Hierarchical Type-2 Fuzzy Logic Control Architecture for Autonomous Mobile Robots", *IEEE Trans. Fuzzy Systems*, Vol. 12, pp. 524-539, 2004.

33. Wu, D. and W. W. Tan, "A Type-2 Fuzzy Logic Controller for the Liquid-Level Process", *In Proc. FUZZ-IEEE 2004*, pp. 953-958, Budapest, Hungary, July 2004.

34. Figueroa, J., J. Posada, J. Soriano, M. Melgarejo, and S. Rojas, "A Type-2 Fuzzy Controller for Tracking Mobile Objects in the Context of Robotic Soccer Games", *In Proc. FUZZ-IEEE 2005*, pp. 359-364, Reno, Az, USA, May 2005.

35. *MATLAB® Fuzzy Logic Toolbox Tutorial*, ©1994-2005 The MathWorks, Inc.

36. Batyrshin, I. and M. Wagenknecht, "Towards a linguistic description of dependencies in data", *International Journal of Applied Mathematics and Computer Science. Special Issue on Computing with Words and Perceptions (ed. by D. Rutkowska, J. Kacprzyk, L.A. Zadeh)*, Vol. 12, No. 3, 391-401, 2002.

37. Hagras, H., "Type-2 FLCs: A New Generation of Fuzzy Controllers", *IEEE Computational Intelligence Magazine*, Vol. 2, No. 1, pp. 30-43, February 2007.

38. Mendel, J. M., "Type-2 Fuzzy Sets and Systems: An Overview", *IEEE Computational Intelligence Magazine*, Vol. 2, No. 1, pp. 20-29, February 2007.

39. Liang, Q. and J. M. Mendel, "An Introduction to Type-2 TSK Fuzzy Logic Systems", *IEEE InternationalFuzzy Systems Conference Proceedings*, 1999.

40. Liang, Q. and J. M. Mendel, "Equalization of Nonlinear Time-Varying Channels Using Type-2 Fuzzy Adaptive Filters", *IEEE Transactions on Fuzzy Systems*, Vol. 8, No. 5, October 2000.

41. *MATLAB® Optimization Toolbox Tutorial*, ©1994-2005 The MathWorks, Inc.

42. Walsh, G. R., *Methods of Optimization*, A Wiley-Inrescience Publication, John Wiley & SONS, 1975.

43. Chong E. K. P. and S. H. Zak, *An Introduction to Optimization*, John Wiley & SONS, INC , Second Edition, USA, NY, 2001.