The Applicability of Fuzzy Theory in Machine Learning

PhD Thesis

Abrar Hussain
Supervisor: József Dombi

Doctoral School of Computer Science
Department of Computer Algorithms and Artificial Intelligence
Faculty of Science and Informatics
University of Szeged

Szeged
June 30, 2022

Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence.
ANFIS	Adaptive Neuro-Fuzzy Inference system.
BGD	Batch Gradient Descent.
CNN	Convolutional Neural Network.
COG	Centre of Gravity.
CSTR	Continuously Stirred Tank Reactor.
DF	Distending Function.
DFIS	Distending Function-based Fuzzy Inference System.
DFNN	Deep Fuzzy Neural Network.
DNN	Deep Neural Network.
FIS	Fuzzy Inference System.
FLS	Fuzzy Logic System.
FOU	Foot Print of Uncertainty.
GDO	Generalized Dombi operator.
KM	Karnik Mendel.
LM	Levenberg-Marquadt.
LMF	Lower Membership Function.
MF	Membership Function.
MIMO	multi-input multi-output system.
MISO	multi-input single output.
ON	Ordered Normalization.

RBN	Rule-Based Neuron.
RBNN	Rule-Based Neural Network.
RMF	Retractable Membership Function.
SGD	Stochastic Gradient Descent.
t-conorm	triangular conorm.
t-norm	triangular norm.
T2DF	Type2 Distending Function.
T2FSs	Type2 Fuzzy Sets.
TS	Takagi Sugeno.
UMF	Upper Membership Function.

Chapter 1

Literature Review and Contributions

1.1 Literature Review

Around 1920 Jan Lukasiewcz and his colleagues developed a three-valued logic [74]. This development was mainly driven for philosophical reasons and the interpretation of the intermediate values was not clear. H. Reichenbach [101] proposed a probability logic that was based on a continuous scale of truth values. Later on Carnap [18] pointed out that there is a significant difference between a probability value and truth value. Lotfi Zadeh proposed fuzzy sets while trying to solve problem in information processing and pattern classification. In contrast to classical sets, the fuzzy sets are characterized by membership values. Each element in the fuzzy set is assigned a membership values in the interval [0, 1]. This value tell us the degree of belonging of a particular element to the fuzzy set.
In the sense of Zadeh, fuzzy theory can be viewed as a framework for reasoning with incomplete information or the vagueness in knowledge. Bellman and Zadeh emphasized that the truth/falseness of a statement should be evaluated based on the available vague knowledge-base [12, 13]. Vagueness in knowledge was called fuzziness by Zadeh. Fuzziness is not regarded as a limitation in fuzzy theory but rather it is a way of expressing the grades. Fuzzy theory deals with the classes which have continuous properties and unsharp (blurred) boundaries. It does not handle the uncertainty associated with the knowledge. To handle this, Zadeh introduced a separate framework called the possibility theory [145].
Since then, several sub-fields of fuzzy theory have emerged and evolved over the time. Now, we will briefly describe some of these fields:

1.1.1 Fuzzy classification

The notion of fuzzy classification was initiated by Zadeh, Bellman and Kalaba [11]. The idea was to extract the membership function of various classes from the data.

The data points which are closer to each other have the high membership value for a particular class. The partition between the classes was later defined by Ruspini [104].

1.1.2 Fuzzy events

The idea of replacing sets by fuzzy sets was quickly applied by Zadeh to the notion of an event in probability theory [139]. The probability of a fuzzy event is just the expectation of its membership function. Mundici [87] generalized De Finetti's theory of subjective probability to fuzzy events. Since then, it has become as alternative probability theory on algebras of fuzzy events.

1.1.3 Fuzzy systems

The idea of fuzzy systems was initially viewed as systems whose state equations involve fuzzy variables or parameters, and this give birth to fuzzy classes of systems [136]. Later it was suggested that fuzziness lies in the description of the approximate rules needed to make the system work. That view was the result of a convergence between the idea of system with those of fuzzy algorithms introduced earlier [138, 142], and this increased the focus of attention on the representation of natural language statements via linguistic variables. In this paper, systems of fuzzy if-then rules were first described, which paved the way to fuzzy controllers, built on human information, and they achieved great success in the early 1980's.

1.1.4 Type2 Fuzzy Sets and Interval Valued Type2 Fuzzy Sets

Type2 Fuzzy Sets (T2FSs) are an important extension of fuzzy sets which don't have crisp membership values. Instead, the membership values are fuzzy and are defined by a secondary membership function in the $[0, 1]$ interval. T2FSs were introduced by Zadeh [144]. The mathematical foundations were then laid down by Mizumoto and Tanaka [85]. The concepts were further developed by Dobios and Prade [44]. Due to the computational complexity associated with generalized T2FSs , Interval Valued Type2 Fuzzy Sets (IT2FS) were introduced and promoted by Klir [64], Turksen [115] and Schwartz [105]. Gorzalczany introduced the concept of Foot Print of Uncertainty (FOU) without naming it at that time. Karnik and Mendel proposed the Type Reduction (TR) technique for defuzzification of T2FSs [61]. They also presented the complete inference process using the T2FSs [62]. The Representation Theorem for T2FSs was provided by Mendel and John [82]. This allowed then to extend the type1 fuzzy mathematics for T2FSs . Later on the TR techniques were made more efficient and fast [127]. T2FSs have lots of applications in the fields of Machine Control and Medicine [79, 91].

1.1.5 Intuitionistic Fuzzy Sets

Intuitionistic Fuzzy Set (IFS) is a generalization of fuzzy sets proposed by Atanassov [7] which incorporates a hesitation margin, membership degree and non-membership degree. The hesitation margin is defined as 1 minus the sum of membership degree and non-membership degree. Kacprzyk and Szmidt showed that IFS can be successfully applied to problems where the degree of belongingness alone doesn't completely describe the situation [109]. IFS is an appropriate tool for utilizing the roughly stated facts [110]. De et al demonstrated the application of IFS in medical diagnosis [24]. Later Atanassov extended the theory and applications of IFS [8]. Various distance measures have been incorporated with IFS to extend its the scope of applicability [123]. IFS are frequently used in decision making, pattern recognition and machine learning.

1.1.6 Fuzzy Hesitant Sets

There are some situations where uncertainty arises due to expert hesitation in assigning proper membership values to the elements of fuzzy set. Torra introduced a novel extension of fuzzy set to model these uncertainties, called Hesitant Fuzzy Set (HFS) [113]. Since then several extensions of HFS have been proposed to model hesitation uncertainty arising from various sources. Zhu et al proposed the Dual Hesitant Fuzzy Set (DHFS) to model hesitation in both membership and non-membership degrees [153]. Chen et al proposed an Interval Valued Hesitant Fuzzy Set (IVHFS) in which a few possible interval values are assigned as a membership value to each element of the fuzzy set [21]. Qian et al presented he Generalized Hesitant Fuzzy Set (GHFS) by incorporating the concepts of IFS in HFS [98]. With this extension the membership function is a union of a few IFSs. Yu presented the Triangular Hesitant Fuzzy Set (THFS) by exploiting the fact that sometimes the experts are hesitant about giving the membership degree of the element of fuzzy set between 0 and 1 [133]. In this framework, triangular fuzzy numbers are used to express the degree of membership. Rodriguez proposed Hesitant Fuzzy Linguistic Term Set (HFLTS) to handle the hesitation that expert encounter when the already defined linguistic terms are not adequate to describe the situation [102]. This extension addresses the problem which is qualitative in nature.

1.1.7 Fuzzy Rough Sets

Rough sets were introduced by Pawlak to handle the uncertainty arising due to incomplete information [94]. Rough sets have numerous applications in data analysis, classification and feature selection [72, 108]. However the rough sets cannot process the gradual indiscernibility and quantitative data directly. To handle this shortcom-

ing, a hybridization of fuzzy theory and rough sets was proposed resulting the development of Fuzzy rough Sets (FRS). The seminal contributions to FRS came from the work of Dubios and Prade [43]. Axiomatic approach to FRS theory was developed by Morsi et al. [86] and Radzikowska et al. [99]. Variable Precision rough Sets (VPRS) were introduced to handle the noisy data [154]. In the last decade several important directions such as classification-Oriented FRS [60] and axiomatic approach to FRS [73] have been studied extensively.

1.1.8 Fuzzy Graphs

A graph is a mathematical tool that is used to represent a network. It consists of nodes (vertices) which are connected using edges. Graphs are used to describe various real world phenomena. Kauffman introduced the concept of Fuzzy Graphs (FGs) to model the uncertainty in various graph attributes [63]. FGs are based on Zadeh's fuzzy relation [141]. The theoretical notions of FGs were developed by Rosenfeld and Bhattacharya [14, 103]. Nagoor Gani and Radha defined important operations and properties such as conjunction of FGs, sequences in FGs and degree of vertex [48, 49]. Later, Akram et al. [2, 3, 4] introduced various extensions such as hyper FGs, bipolar FGs abd soft FGs. FGs is a rapidly growing field and it has numerous applications in computer networks, communication, image processing, social networking and scheduling [93].

1.1.9 Fuzzy Modeling and Control

Fuzzy theory has been an area of extensive research since its inception, nearly half a century ago, by Lotfi A. Zadeh [135, 140] and it provides applications in various areas of daily life [9, 80, 128, 152]. To design control systems for complex ill-defined non-linear processes (for which adequate analytical models are not available), is a challenging task. Novel control techniques have been proposed to solve such problems [117, 118, 119, 120]. However, if a knowledge base is available for these systems, fuzzy theory provides an adequate solution for controller design [39]. For some non-linear processes, the model parameters vary with time or they may have uncertain initial conditions. The control of such non-linear dynamic processes is called adaptive control, where the control law adapts itself to the changing dynamics in order to meet the control objectives [70, 121, 122]. An adaptive fuzzy controller organizes the rule base (type and number of rules) and it tunes the parameters of the membership functions if the process dynamics changes over time [39, 70].
The design of fuzzy logic control (FLC) is based on the set of 'If then' rules forming a rule base. The multi-input single output (MISO) fuzzy rule base has the following form:

$$\text{If } x_1 \text{ is } A_1^i \text{ and } ..... \text{ and } x_n \text{ is } A_n^i \text{ then } y \text{ is } B^i \ , \tag{1.1}$$

where $x_1, x_2, ..., x_n, y$ are the linguistic variables that take the linguistic values from the fuzzy sets $A_1, A_2, ..., A_n, B$ and $i = 1, ..., l$ is the number of fuzzy rules. The part of the rule after '*If*' is a logical expression, called the antecedent (related to the input) and the part after '*then*' is called the consequent (related to the output). The Fuzzy Rule Inference (FRI) applies a fuzzy relation to map the input and output space. Based on the FRI, various types of control techniques have been developed. The two best known are the Mamdani [117] and model-based Takagi Sugeno (TS) [17, 112] type fuzzy control systems.

In the Mamdani inference system (also called the Type-I TS system), the output of a rule is a fuzzy set. The operation of the Mamdani inference system consists of five steps. These are: $1)$ The fuzzification of crisp inputs in order to get fuzzy inputs; $2)$ The antecedent parts of the rules are based on fuzzy logic operators. These fuzzy logical expressions are evaluated to determine the applicability of the fuzzy rules; $3)$ Implication is carried out to get fuzzy outputs; $4)$ Aggregating the fuzzy outputs of all the rules; $5)$ Defuzzification of the aggregated output is carried out to get the final crisp output. The Mamdani fuzzy controller directly transforms the operator implicit knowledge or expert explicit views into fuzzy rules and generates a control law. The Mamdani approach is intuitive, works well with direct human input and various control tasks can be performed [1, 100]. The stability of a closed loop control system is one of the main objectives that should be met. Frequency domain methods are mostly used for the stability analysis of Mamdani fuzzy controllers, such as Popov's method, the circle stability criterion and hyperstability theory [120].

In the case of the TS (Type-II / Type-III) fuzzy controller, the consequent part of the rule (Eq. 1.1) is a function (mostly linear) of the inputs or (as a special case) a crisp value i.e. it is not a fuzzy set as in the Mamdani case. The TS fuzzy model consists of a membership function and a set of linear models to form a global nonlinear model. The TS fuzzy controller also has a nonlinear function approximation property [50].

In most of the cases, expert knowledge is not available or it is poorly described. So the exact description of fuzzy rules is not an easy task. If the working data of the process is available then a data-driven based design is an attractive option. In this case, the problem reduces to identifying a suitable fuzzy model which fits the given data [5, 68, 114].

Data-driven fuzzy modeling is quite similar to statistical non-parametric regression methods, especially when we are using TS systems. The technique can be extended to classification problems where we have to identify the fuzzy rules for each possible class [22]. The data-based identification of a fuzzy model can be divided into two parts namely, qualitative and quantitative identification. Qualitative identification focuses on the number and description of fuzzy rules. Quantitative identification is

concerned with the identification of parameter values. These parameters belong to membership functions and operators. In the case of qualitative identification, soft computing methodologies are used like evolutionary algorithms, genetic algorithms and swarm optimization [38, 59, 150]. Neural networks are mostly used for learning parameters (the quantitative part). Combining these two leads to the development of a Neuro-Fuzzy Inference systems [96, 116].

1.1.10 Neuro-fuzzy systems and soft computing

Due to the increase in computational power and efficient training algorithms, Deep Neural Networks (DNNs) are being used to solve complex tasks. DNNs have a lot of applications in the fields of computer vision, natural language processing, speech recognition and bio-informatics. However, DNNs are extremely sensitive to input feature noise and the accuracy decreases very rapidly with the increase in attribute noise [27, 53, 88]. In [106] Jiawei et al. showed that a change in one pixel value can significantly affect the accuracy of the DNN. Training using DNNs results in a black box model and this means that the results have no interpretation. To overcome these shortcomings, several efforts have been made. One of these involves the incorporation of fuzzy logic with a DNN. This has led to the development of (DFNNs). A lot of structures for DFNN have been proposed in the past decade. Das et al. suggested that these structures can be categorized into two main types [23]: 1) Ensemble DFNN and 2) Integrated DFNN. In ensemble DFNN, fuzzy logics is used in a parallel or sequential manner [69]. Xiaowei Gu introduced multi-layer ensemble learning model to tackle high dimension complex problems [54]. First order evolving Fuzzy Inference Systems (FISs) are used as building blocks and the overall ensemble system has demonstrated state of art performance with high transparency. In integrated DFNN, the fuzzy logic is the integral part of the training process. A single hybrid architecture is obtained by fusing a DNN with fuzzy logic. For example, when Pythagorean fuzzy numbers are used as weights [151], this leads to the development of a Pythagorean Fuzzy Deep Boltzmann machine. In [26], Deng et al. presented multi-model learning by using the membership values of the data in parallel with deep representation extraction. In [45], fuzzy logic is used to adjust the learning rate and parameters. Tabrizi et al. designed a deep Convolutional Neural Network (CNN) by feeding the fuzzified data and it has the noise reduction effect [111].
In the last twenty years, while researchers have been developing formal many-valued logics and uncertainty logics based on fuzzy sets, Zadeh rather emphasized computational and engineering issues by advocating the importance of soft computing (a range of numerically oriented techniques including fuzzy rules-based control systems, neural nets, and genetic algorithms [146]) and then introduced new paradigms about computational intelligence like granular computing [147], computing with

words [148] and perception-based reasoning [149], attempting to enlarge his original motivation for a computational approach to the way humans handle information.

1.2 Existing Problems and Proposed Solutions

Here, we briefly present some of the existing problems and our research contributions in the following areas:

1.2.1 Arithmetic-based type1 fuzzy system

The Fuzzy Rule Inference (FRI) uses a fuzzy relation to map the input and output space. Based on the FRI, various types of FISs have been developed. The two best known are the Mamdani [77] and model-based TS [17, 112] type inference systems. There are however some drawbacks with these two inference techniques. These are:

1. The input space is not completely covered by the triangular (or trapezoidal) membership functions i.e. they cover only a limited subspace. For example if we have two inputs, each with 7 categories, then 49 rules are required to cover the whole input space. Usually a few rules are applied to decrease the computation load, so a large area of the input space is not covered. If the input value falls in the area which is not covered by any rule, then no action is generated. If the number of input variables increases from two (working in a higher dimensional space), then the problem grows exponentially. Some efforts to overcome this problem have been made by L. Kóczy and K. Hirota [65], but all these procedures increase the computational cost.

2. Most of the membership functions are not analytical i.e. the derivative is not defined at every point and higher derivatives do not exist. This is a drawback because the gradient-based optimization techniques cannot be used to tune the parameters of these membership functions as they work only on analytical functions. However in this case, the gradient-free optimization techniques can still be used but these are not very fast, accurate and computationally efficient compared to gradient-based methods.

3. It is not clear how to choose a fuzzy operator system for the antecedent part of a rule. Various fuzzy operators can be chosen. Using different fuzzy operators produces different results. Hence the choice of membership function and operator system is completely arbitrary and we cannot get a proper efficient design.

4. Different implication operators are introduced. Surprisingly, the product operator is mainly used. The product operator is a strict t-norm and it is not an implication operator.

5. The result of an evaluation of the consequent part of the rule is not a membership function (it is an α-cut of the membership function). For every input value, each rule is evaluated to get the aggregated output. The aggregation of consequent parts of all the rules is a membership function which does not belong to the same class of the antecedent and consequent membership functions.

6. Centre of Gravity (COG) defuzzification usually involves the integral evaluation of the aggregated function and it is computationally expensive. Although there are various defuzzification methods that do not require integral calculations, in general these are less accurate than COG-based methods.

Using these techniques, designing an adaptive fuzzy controller is a challenging task. As we mentioned above, both of these techniques have some advantages and disadvantages. There is a need to combine these advantages into a single design approach. We attempted to solve these issues using a new approach, which has the following good features:

1. A new type of parametric membership function called the Distending Function (DF) is introduced. With a few rules, it can cover the whole input space. It has three parameters and each has a semantic meaning. The values of the two parameters are usually fixed and one parameter value is tuned during the design process. It has two types, namely the symmetric and asymmetric DF and both can be utilized for developing FIS.

2. The DF is analytical i.e. higher derivatives exist at each point. This property is used in optimization procedures to tune the parameters of the DF.

3. The general parametric fuzzy operator system is used for evaluating the antecedent part of the rule. The operator system and the DF are based on the Dombi operator. Hence, both are consistent with each other.

4. Our approach does not involve the implication step. Instead the activation strength of each rule is multiplied by the consequent DF to get the fuzzy output of each rule.

5. The consequent of each rule is a DF. Aggregation is carried out using the weighted arithmetic mean of these consequent DFs of all the rules. A linear combination is closed for DFs and so the result of aggregation is also a DF.

6. Defuzzification in this case is only a single-step calculation (finding the point that has the highest value of the aggregated DF). This is why it is simple and computationally efficient.

7. Using our proposed approach, we designed an adaptive FIS. It consists of tuning the DF parameters using gradient descent optimization. The adaptive FIS can handle the changing process dynamics with increased computational efficiency.

We combined the advantages of Mamdani and TS methods and developed the arithmetic-based FIS. Here, the antecedent and consequent parts of the rule base are fuzzy sets, so it is very close to direct human linguistic inputs. Related work for designing an adaptive fuzzy controller for a nonlinear system with unknown dynamics was carried out by Ning Wang et al. [121, 122]. However, there are two main differences with our approach: 1) Product inference (operator and implication) is used for evaluating the fuzzy rules and designing controller in [122]; 2) Adaptivity is achieved using Retractable Membership Function (RMF) in [121], which is a symmetric membership function. In our approach, we used a more general operator system which can utilize various available fuzzy operators (e.g. min/max, product, Einstein, Hamacher, Dombi, drastic). Fuzzy arithmetic is used instead of implication i.e. we used a regression-like approach. In our study, the symmetric and asymmetric DFs are used. The asymmetric DF provides more flexibility in adaptive controller design. The efficiency of this new approach is shown by designing an adaptive control system for a water tank level and vehicle lateral dynamics.

1.2.2 Arithmetic-based type2 fuzzy system

Compared to type-1, the type-2 fuzzy systems are better at handling uncertainties, produce smoother control response and are more adaptive and use a smaller rule base [71]. For practical and computational reasons, interval type-2 fuzzy systems were introduced [81]. These systems have been successfully used in control systems, data mining, cost and risk assessment, time series predictions, urban planning and human resource management [19, 51, 55, 89, 125, 126]. The design of interval type-2 fuzzy system consists of five steps: 1) Fuzzification of the inputs using type-2 MFs; 2) Calculation of rules firing strengths. The firing strength is now an interval; 3) Implication and aggregation is used to produce the outputs. These operations also produce also a type-2 fuzzy set; 4) Type reduction is applied to convert type-2 fuzzy set into type-I fuzzy sets; 5) Defuzzification is performed to get the crisp output value. This process is similar to design of type-I fuzzy system but here we have type reduction as an additional step. This step converts type-2 fuzzy sets to type-I fuzzy sets. The type reduction is achieved using the so called Karnik Mendel (KM) iterative algorithm [83]. This algorithm defines the switching points of the lower and upper

firing strengths. Using these points, the KM algorithms generates two type-I fuzzy sets. These sets are defuzzified to get a crisp output.
There are some drawbacks in the above-mentioned approach. These are:

1. The choice of type-2 membership function and its systematic connection with the uncertainty are not clear. Different type-I membership functions can be combined to generate type-2 membership function. And it is not clear which type of membership functions should be used for particular uncertainty case.

2. The type reduction step is based on the KM algorithm, which is computational expensive [132]. Due to its iterative nature, it is not suited for on-line applications. There are some alternative solutions which reduce the computation burden but these are all approximations [134].

3. Although type-2 Fuzzy Logic System (FLS) require fewer rules compared to type-I fuzzy systems, but the number of parameters is comparatively large. So optimization is not easy in this case.

4. The implication and aggregation steps also increases the computation complexity of the type-2 FLS.

Here, we solve some of these issues by proposing a new type of interval type-2 FLS. It overcomes these issues using the following unique features:

1. A type2 extension of the DF called the Type2 Distending Function (T2DF) is proposed. Different types of uncertainties can be expressed by associating it with the parameters of T2DF. It can effectively represent most of the forms of uncertainties used in type-2 fuzzy systems.

2. Fuzzy arithmetic approach is utilized here for designing type-2 fuzzy logic controller. So it has no type reduction step and it does not require the iterative algorithms. It is simple, computationally fast and suitable for on-line implementations.

3. Most of the parameters of the T2DF are fixed. Usually just the parameter associated with the uncertainty is varied. We can say that the number of parameters are the same as in type-I FLS. The optimization process is easy to perform and fast.

4. There are no implication and aggregation steps. Therefore it is computationally fast.

Because of these features, the proposed approach provides a complete framework for handling the uncertainty using type-2 fuzzy systems.

1.2.3 Data-driven arithmetic-based fuzzy type1 system

A rule base system represents the human expertise in the form of linguistic rules. These rules describe the dependencies between the input and output variables in the form of IF-THEN statements. As the number of input variables increases, the required number of rules and the system complexity increases exponentially. In most cases, expert knowledge is not available or it is poorly described. So the exact description of fuzzy rules is not an easy task. If the working data of the process is available then a data-driven based design is an attractive option. In this case, the problem reduces to identifying a suitable fuzzy model which fits the given data [5, 68, 114].
Data-driven fuzzy modeling is quite similar to statistical non-parametric regression methods, especially when we use TS systems. The technique can be extended to classification problems where we have to identify the fuzzy rules for each possible class [22]. The data-based identification of a fuzzy model can be divided into two parts namely; qualitative and quantitative identification. Qualitative identification focuses on the number and description of fuzzy rules. Quantitative identification is concerned with the identification of parameter values. These parameters belong to membership functions and operators. In the case of qualitative identification, soft computing methodologies are used like evolutionary algorithms, genetic algorithms and swarm optimization etc. [38, 59, 97, 150]. Neural networks are mostly used for learning parameters (the quantitative part). Combining these two leads to the development of a Neuro-Fuzzy Inference Systems [6, 96].
The above mentioned approaches however have some drawbacks:

1. Qualitative identification:
   The identified rule base has a so-called flat structure (curse of dimensionality) problem [78]. In most of the cases, triangular membership functions are used and the support of these functions covers a limited area of the input space (the grade of membership is zero outside this area). To cover the input space completely, a huge number of rules are required. If we have 2 input variables, each with 5 categories, then the number of rules required to cover the whole input space will be 25. If the number of input variables increases, then an exponentially large number of rules are required. Each rule is applicable only within a specific area and its strength is zero outside. If the training data of the system does not fully span the input and output space, then this will cause serious problems when modeling the system. If the input falls in these uncovered areas then the identified rule bases do not generate any action. Even if some sort of interpolation technique is applied, computation complexity will increase [96]. In summary, a global fuzzy model requires a large number of rules and the number of these rules depends exponentially on number of the input variables and this will lead to a huge complexity demand of the data-driven fuzzy

models.

2. Quantitative identification:
   The computation complexity of the quantitative part of the identified fuzzy model also increases with the number of rules. As the number of rules increases, the number of parameters of the membership function and operators also grow exponentially. The calculation of these parameters will increase the computational cost of the quantitative model.

3. Interpretability:
   In most cases, the interpretability of the identified fuzzy rule base is not clear. It is easier to interpret a few rules and get an insight into the working model. However, if the number of rules grows exponentially, then for a given set of input values, it is not possible to predict the response of the model and analyze its performance. The model tends to be more like a black box in these situations and the beauty of a fuzzy-based design fades.

4. Complexity of control design:
   The identified rule base is used to generate a fuzzy controller using the Mamdani or TSinference engines. Both of these are computationally expensive due to implication, aggregation requirements and defuzzification step. These conventional fuzzy inference procedures add complexity to the overall design of a data-driven fuzzy controller.

In this study, we propose a new methodology for a data-driven fuzzy control design. This new method has the following unique features:

1. DFs have been used to cover the input space entirely. A DF has a long tail and it is defined on [-∞, ∞]. The grade of membership always has a non-zero value and the whole input space can be covered. If we apply the Dombi operator on the DFs in the input space, the results is also a DF in the output (higher dimensional) space. So the whole area of the output is also covered by the DF. A few key points from the training data have to be selected to generate the fuzzy rule base. This fuzzy rule base, in combination with the DF and Dombi operator, spans the whole input-output space just using a few rules.

2. The DF has three parameters. Usually two parameters can be kept constant and one parameter is used for tuning. The latter increases/decreases the influence area of the DF function. Also, a single step calculation is required to calculate this parameter. Because of this, due to the smaller number of identified rules and the deterministic nature of single parameter, the computation complexity of the quantitative part is negligible.

3. The interpretability of the model increases due to the significant decrease in the number of fuzzy rules.

4. We used an arithmetic-based type1 FIS to generate the output signal. There is no implication operator involved and aggregation is based on arithmetic operations. The aggregation of fuzzy rules results in a single value that is used as a output signal. This single value arises from the defuzzification of consequent DFs, which is a single step calculation. So the overall complexity of the data-driven fuzzy controller design decreases significantly.

1.2.4 Data-driven arithmetic-based fuzzy type2 system

We presented a novel technique for fuzzy type2 modeling and control. The proposed fuzzy model called Distending Function based Inference System (DFIS) consists of rules and type2 membership functions. The rules are based on the Dombi conjunctive operator. A procedure for designing a type2 FIS using the rules is also presented. This FIS can handle various types of uncertainties (e.g., sensor noise). The whole procedure has the following unique features:

1. We used the Type2 Distending Function (T2DF) [35]. It has symmetric and asymmetric forms. With a few rules, it can completely cover the whole input space, and this helps overcome the flat structure issue.

2. T2DF has only a few parameters. Most of these parameters are kept fixed and a few are varied during the training process. It reduces the computational burden of quantitative identification.

3. Different types of uncertainties can be modeled using various parameters of T2DF. Therefore, most forms of the uncertainties in fuzzy systems can be represented using T2DF.

4. Our approach identifies a few important fuzzy rules. And we have also developed a rule reduction algorithm which can further reduce the number of identified rules and it results in an interpretable model.

5. Because only a few parameters are varied during the design process, the optimization is simple and fast.

6. We presented an arithmetic-based interval type-2 FIS. The type reduction, implication and aggregation steps are not involved in the design procedure. Therefore the type2 FIS is computationally efficient and can be implemented online.

1.2.5 Rule-Based Neural Network

DNNs and FISs seem to complement each other. It is the natural to expect that a combination of these two approaches might have the advantages of both. On the one hand, FISs can benefit from the computational learning procedures of the DNN. The parameters of the rule (sometimes the whole rule) can be learnt directly from the data. Because of this the need for the expert knowledge is no longer essential. On the other hand, the ANN can take advantages of benefits offered by an FIS. The incorporation of fuzzy logic with a DNN leads to the development of Deep Fuzzy Neural Networks (DFNN). In our research, we generalized the concept of DFNNs by presenting a Rule-Based Neural Network (RBNN). It has the following special features:

1. RBNN can be trained to solve various real world regression and classification tasks. RBNN has a similar architecture to a DNN but it has relatively few trainable parameters.

2. The input layer in RBNN has the normalization functionality. We have proposed a new type of normalization technique and it is called the Ordered Normalization (ON). ON is specially useful when the training data has an asymmetric distribution.

3. The training of RBNN is similar to that of DNN. Stochastic gradient (SG), batch gradient descent (BGD) and Levenberg-Marquadt (LM) optimization methods are used in the parameter update of back propagation. Other different variants of gradient-based optimization can also be used to train an RBNN. The output of each RBN is calculated by evaluating the rules using arithmetic operations.

4. The results of RBNN are interpretable and hence it is not a black box model. Hidden and output layers in RBNN contain (Rule-Based Neuron (RBN)). Each RBNN has a built-in FIS. After the training phase, the prediction results of the RBNN can be interpreted using simple if-else rules.

5. It is robust to (input) feature noise and compared to DNN, it produce a higher prediction accuracy even in the presence of large feature noise.

6. The performance of the RBNN on a skewed dataset is comparatively better than that of the DNN and it produces higher F1 scores for the minority (smaller) classes.

Chapter 2

Fuzzy Operators and Membership Functions

Membership Function (MF) is a key part of a fuzzy inference system. The fuzziness of a fuzzy set is represented by a MF. The MF can be of any shape and the only essential condition is that it must be between 0 and 1. The shape of the MF has effects on the performance of the inference system. Unfortunately there is no proper criteria to choose the shape of the MF. Therefore intuition and experience are the key factors in deciding the shape of the MF for a particular problem. In our research we solved this problem by proposing a new parametric MF called the Distending Function (DF). It can take various shapes depending on the values of its parameters. The parameters can be learned directly from the data of the system. Therefore the DF takes an appropriate shape depending on the nature of the problem.
Classical set theoretical operations (intersection, union, complement) have extension in fuzzy set theory and these are called t-norm, t-conorm and negation operations. Various operators have been proposed for these operations and this led to development of various operator systems (min/max, drastic, Nilpotent, parametric). In our research, we used the Dombi parametric operators to implement these fuzzy operations. The DF is based on Dombi operators. Therefore using the Dombi operators for fuzzy set operations in our research led to the development consist system.
Now we briefly describe the important fuzzy operators and T2FSs.

2.1 Fuzzy Operators

After the introduction of fuzzy set theory by the seminal paper of Zadeh [137], extensive research was carried out to extend the the set theoretic operations to fuzzy sets. Soon the important set operations such as union, intersection, complementation, set difference, quantifier and inclusion were extended to fuzzy sets. This resulted in fuzzy operators defined on unit interval such as t-norm, t-conorm, negation, sym-

metric difference, implication and ordered weighted average operators. Now we will briefly describe some of the important families of these operators.

2.1.1 Negation

Negation is an extension of the set complementation operation. Bellman and Giertz [10] defined negation N as

$$N(0) = 1, \qquad N(1) = 0 \tag{2.1}$$

$$N(N(z)) = z, \qquad z \in [0,1] \tag{2.2}$$

$$\text{and} \quad y < z \implies N(y) > N(z), \quad y, z \in [0,1] \tag{2.3}$$

If the negation is continuous and strictly decreasing then it is called a strict negation. If the strict negation is involutive (i.e. (2.2) holds), then it is called a strong negation. Some of the negations worth mentioning are:

Intuitionistic Negation

Yager [129, 131] introduced the following type of negation function called the intutionistic negation

$$N_i(z) = \begin{cases} 0 & z > 0 \\ 1 & z = 0 \, . \end{cases} \tag{2.4}$$

This negation is neither strict nor involutive.

Dual Intuionistic Negation

Ovchinnikov [90] defined a dual intuitionistic negation function of the form

$$N_d(z) = \begin{cases} 0 & z = 1 \\ 1 & z < 1 \, . \end{cases} \tag{2.5}$$

This negation is also neither strict nor involutive.

Standard Negation

Zadeh proposed a strong negation in his seminal paper [137]. It is defined as

$$N(z) = 1 - z, \qquad z \in [0,1] \, . \tag{2.6}$$

This negation is strict as well as involutive.

Sugeno Negation

Sugeno [107] proposed the parametric family of negation given by

$$N_\lambda(z) = \frac{1-z}{1+\lambda z}, \qquad z \in [0,1]\ . \tag{2.7}$$

If $\lambda \to 0$, then this negation converges to the standard negation operator.

2.1.2 Dombi Negation

Dombi introduced the parametric negation in pliant systems [34] and it is given by

$$N_\nu(x) = \frac{1}{1+\left(\frac{1-\nu}{\nu}\right)^2\left(\frac{1-x}{x}.\right)}, \qquad \nu \in [0,1]\ . \tag{2.8}$$

Where ν is the fixed point of the negation. There exists a ν value (ν_*), such that

$$N(\nu_*) = \nu_*. \tag{2.9}$$

It is called the neutral value of negation. The neutral value divides the interval in to two parts. The negation values less than ν_* are treated as negative evaluation range and those negation values greater that ν_* are treated as acceptable negation range.

2.1.3 Triangular norm and Triangular conorm

A function t ($[0,1]^2 \to [0,1]$) is called a triangular norm (t-norm), if it satisfies the following four conditions:

1.

$$t(z,1) = z \qquad (Identity) \tag{2.10}$$

2.

$$t(y,z) = t(z,y) \qquad (Commutative) \tag{2.11}$$

3.

$$t(z,t(x,y)) = t(t(z,x),y) \qquad (Associate) \tag{2.12}$$

4.

$$\begin{aligned} &\text{for } y \leq u, \quad z \leq v \\ &t(y,z) \leq t(u,v) \end{aligned} \tag{2.13}$$

Similarly a function s$([0,1]^2 \rightarrow [0,1])$ is called a triangular conorm (t-conorm) if it satisfies the following four conditions:

1.

$$s(z,0) = z \qquad (Identity) \tag{2.14}$$

2.

$$s(y,z) = s(z,y) \qquad (Commutative) \tag{2.15}$$

3.

$$s(z,s(x,y)) = s(s(z,x),y) \qquad (Associate) \tag{2.16}$$

4.

$$\begin{aligned} &\text{for } y \leq u, \quad z \leq v \\ &s(y,z) \leq s(u,v) \end{aligned} \tag{2.17}$$

The major difference between the t-norm and t-conorm is the identity property. From 2.10 and 2.13, it can be proved that

$$t(y,z) \leq min(y,z). \tag{2.18}$$

Also, from 2.14 and 2.17, it can be shown that

$$s(y,z) \geq max(y,z) \tag{2.19}$$

Together with the negation operator N, the t-norm t and t-conorms form a De Morgan triplet if it satisfies the condition

$$s(y,z) = N(t(N(y),N(z))), \qquad y,z \in [0,1] \tag{2.20}$$

Now, we will briefly introduce some of the important types of t-norms and t-conorms.

Min t-norm and Max t-conorm

Zadeh proposed the min function for the t-norm and max function for the t-conorm operator [137].

$$t(y,z) = min(y,z)$$
$$s(y,z) = max(y,z).$$

It is worth mentioning that the min is the largest possible t-norm, while max is the smallest possible t-conorm.

Product and probabilistic sum

Goguen [52] proposed the product function (Π) for the t-norm and probabilistic sum ($\Pi^{'}$) for the t-conorm.

$$p(y,z) = yz$$
$$p^{'}(y,z) = y + z - yz.$$

The Lukasiewicz t-norm and t-conorm

Lukasiewicz [75] defined the t-norm w and t-conorm $w^{'}$ using the min and max operators.

$$w(y,z) = max\{y + z - 1,0\}$$
$$w^{'}(y,z) = min\{y + z,1\}$$

The nilpotent t-norm and t-conorm

Perny [95] and Fodor [46] independently discovered the nilpotent minimum and maximum norms. These are also called the bounded norms.

$$min_0(y,z) = \begin{cases} min(y,z) & y + z > 1 \\ 0 & otherwise. \end{cases} \qquad max_1(y,z) = \begin{cases} max(y,z) & y + z < 1 \\ 1 & otherwise. \end{cases}$$

Drastic t-norm and t-conorm

The drastic t-norm (D) and t-conorm ($D^{'}$) are defined using the min and max operators.

$$D(y,z) = \begin{cases} min(y,z) & max(y,z) = 1 \\ 0 & otherwise. \end{cases} \qquad D^{'}(y,z) = \begin{cases} max(y,z) & min(y,z) = 0 \\ 1 & otherwise. \end{cases}$$

D is the weakest among all the t-norms and $D^{'}$ is the strongest among all the t-conorms.

2.1.4 Parametric t-norms and t-conorms

The parametric families of t-norm and t-conorm operator include at least one parameter whose value determines the nature and intensity of the operation. It was shown by Dombi [28] that when this parameter tends to extreme values then the t-norm operator approaches either the min or drastic product operator D. Similarly for the extreme values of the parameter, the t-conorm operator approaches either max or drastic sum operator $D^{'}$. Now we will briefly describe some of these parametric families.

Frank parametric family

Frank introduced the fundamental family of t-norms [47] and it is defined by

$$t_p(y,z) = log_p\left(1 + \frac{(p^y - 1)(p^z) - 1}{p - 1}\right), \tag{2.21}$$

where p is a real number such that $p > 0$ and $p \neq 1$. It was shown by Klement et al. that the members of the Frank family are decreasing functions of p. Using the De Morgan law, the Frank t-conorm can be defined as

$$s_p(y,z) = 1 - t_p(1-y,1-z) \tag{2.22}$$

Hamacher parametric family

Hamacher proposed the negation, t-norm and t-conorm operators [56]. These belongs to rational class of fuzzy operators.

$$N_\gamma(z) = \frac{1-z}{1+\gamma z} \qquad \gamma > -1 \tag{2.23}$$

$$t_{alpha}(y,z) = \frac{yz}{\alpha + (1-\alpha)(y+z-yz)} \qquad \alpha \geq 0 \tag{2.24}$$

$$s_\beta(y,z) = \frac{y+z+(\beta-1)yz}{1+\beta yz} \qquad \beta \geq -1 \tag{2.25}$$

The generator function for these operators is

$$f_\alpha(z) = \begin{cases} \log\left(\frac{\alpha+(1-\alpha)z}{z}\right) & \alpha = 0 \\ \frac{1-z}{z} & \alpha > 0 \end{cases} \tag{2.26}$$

Sugeno-Weber parametric family

Sugeno and Weber introduced the following parametric t-norm and t-conorm [124].

$$t_\lambda(y,z) = max\left(0, \frac{y+z-1+\lambda yz}{1+\lambda}\right), \tag{2.27}$$

$$s_\lambda(y,z) = min(1, y+z+\lambda yz). \tag{2.28}$$

Here $\lambda > -1$ and $y, z \in [0,1]$. The additive generator function for these operators is

$$f_\lambda(z) = \begin{cases} 1 - \frac{log(1+\lambda z)}{log(1+\lambda)} & \lambda \neq 0 \\ 1-z & \lambda = 0 \end{cases} \tag{2.29}$$

The Yager parametric family

Yager introduced one of the important parametric families which is significantly used in the nilpotent logic [130]. The t-norm and t-conorm of this family are strictly increasing functions of the parameter p.

$$t_p(y,z) = max\left(0, 1-((1-y)^p + (1-z)^p)^{\frac{1}{p}}\right), \tag{2.30}$$

$$s_p(y,z) = min\left(1, ((y)^p + (z)^p)^{\frac{1}{p}}\right). \tag{2.31}$$

Here $y, z \in [0,1]$ and $p \in]0, \infty[$.

The Dubios-Prade parametric family

Dubios and Prade introduced a non-Archimedean family of t-norm and t-conorm [41] operators defined by

$$t_\gamma(y,z) = \frac{yz}{max(y,z,\gamma)} \qquad y,z,\gamma \in [0,1]. \tag{2.32}$$

Perny-Fodor parametric family

Perny and Fodor proposed the following family of t-norm and t-conorm operators [46, 95]

$$t_N(y,z) = \begin{cases} min(y,z) & if y > N(z) \\ 0 & else, \end{cases} \quad s_N(y,z) = \begin{cases} 1 & if y \geq N(z) \\ max(y,z) & else, \end{cases} \tag{2.33}$$

where N is any strong negation.

The Dombi parametric family

Dombi introduced the following strict t-norm and t-conorm operators [28]

$$t_\lambda(y,z) = \frac{1}{1+\left(\left(\frac{1-x}{x}\right)^\lambda + \left(\frac{1-y}{y}\right)^\lambda\right)^{\frac{1}{\lambda}}}, \tag{2.34}$$

$$s_\lambda = 1 - t_\lambda(1-y, 1-z). \tag{2.35}$$

This parametric family has only one parameter λ. Changing its value generates the Dombi t-norms and t-conorms. These t-norms and t-conorms have the following additive generator functions:

$$f_\lambda(z) = \left(\frac{1-z}{z}\right)^\lambda, \tag{2.36}$$

$$g_\lambda(z) = \left(\frac{z}{1-z}\right)^\lambda, \tag{2.37}$$

$$\tag{2.38}$$

where $\lambda > 0$ and $y, z \in]0, \infty[$. Dombi parametric family has infinity many negations. The Dombi conjunctive, disjunctive and negation operators form the De Morgan class.

The Generalized Dombi operator (GDO)

Dombi combined three important classes of the fuzzy operator into a single parametric operator [31]. These operator classes includes the min/max operators, strict operators (Einstein, Hamacher, Frank, Product and Dombi) and drastic operators. The Generalized Dombi operator (GDO) has the following form

$$DG_\gamma^{(\alpha)}(x) = \frac{1}{1+\left(\frac{1}{\gamma}\left(\prod_{i=1}^n\left(1+\gamma\left(\frac{1-x_i}{x_i}\right)^\alpha\right)-1\right)\right)^{\frac{1}{\alpha}}}. \tag{2.39}$$

With the right set of α and γ values, the $DG_\gamma^{(\alpha)}(x)$ approaches the operators of these three classes, as shown in the Table 2.1.

2.2 Membership Functions

The Membership Function (MF) assigns the degree of belongingness to each element of the fuzzy set. The value of this degree is always between 0 and 1. The input region

S.No.	Operator	γ value	α value for t-norm	α value for t-conorm
1	Min/Max	0	∞	$-\infty$
2	Drastic	∞	$\alpha > 0$	$\alpha < 0$
3	Hamacher	$\gamma \in (0, \infty)$	1	-1
4	Einstein	2	1	-1
5	Product	1	1	-1
6	Dombi	-	$\alpha > 0$	$\alpha < 0$

Table 2.1: *Various combination of α and γ parameter values in $DG_{\gamma}^{(\alpha)}(x)$*

where the MF has a non-zero degree is called the support of the MF. The region where the MF has a degree of one is called the core of the MF. The elements in the fuzzy set that have the membership grade of 0.5 are called the crossover points. THe α-cut of a fuzzy set is a crisp set which contains those elements which have a membership grade greater than the α value. Some of the popular MF s include piece-wise linear functions (triangular and trapezoidal), Gaussian, Bell-shaped and Sigmoid MFs. In general, a MF is denoted by $\mu(x)$, where x is the input space and it is called the universe of discourse.
A MF is called normal if at least one of its elements has a membership grade of 1. If all the α-cuts of a MF are convex then the MF is also convex. The distance between the cross-over points of the normal and convex is called the bandwidth of the MF. Now, we briefly describe some of the most commonly used MFs.

2.2.1 Triangular MF

A triangular MF of a fuzzy set F is defined as

$$\mu_{tri}(x) = \begin{cases} 0 & x \leq a_1 \\ \frac{x-a_1}{a_2-a_3} & a_1 \leq x \leq a_2 \\ \frac{a_3-x}{a_3-a_2} & a_2 \leq x \leq a_3 \\ 0 & x \geq a_3. \end{cases} \tag{2.40}$$

Here, x is the input space, a_1, a_3 are the coordinates of the boundary points of the triangular MF and a_2 is the peak value coordinate of the triangular MF (Fig. 2.1).

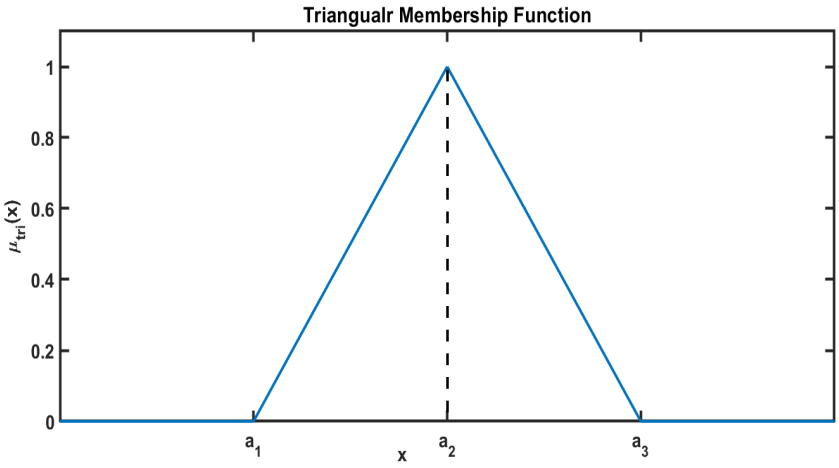

Figure 2.1: *Triangular Membership Function*

2.2.2 Trapezoidal MF

A trapezoidal MF of a fuzzy set is defined as

$$\mu_{trap}(x) = \begin{cases} 0 & x \leq a_1 \\ \frac{x-a_1}{a_2-a_1} & a_1 \leq x \leq a_2 \\ 1 & a_2 \leq x \leq a_3 \\ \frac{a_4-x}{a_4-a_3} & a_2 \leq x \leq a_3 \\ 0 & x \geq a_4. \end{cases} \tag{2.41}$$

A trapezoidal MF with boundary points a_1 and a_4 is shown in Fig. 2.2.

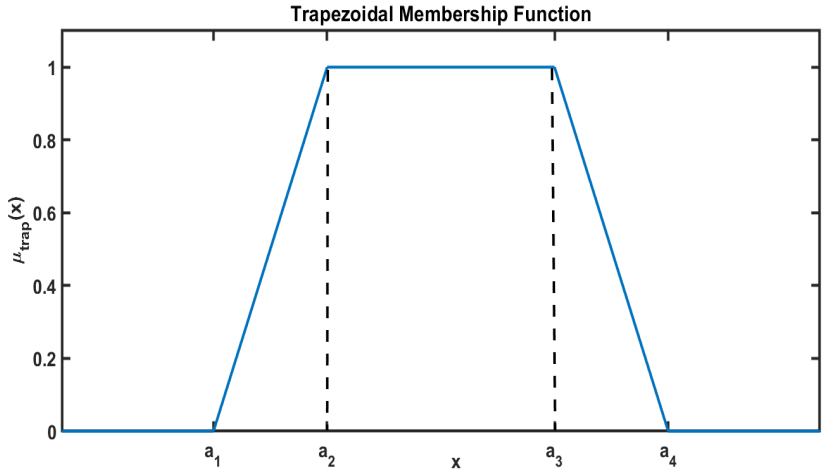

Figure 2.2: *Trapezoidal Membership Function*

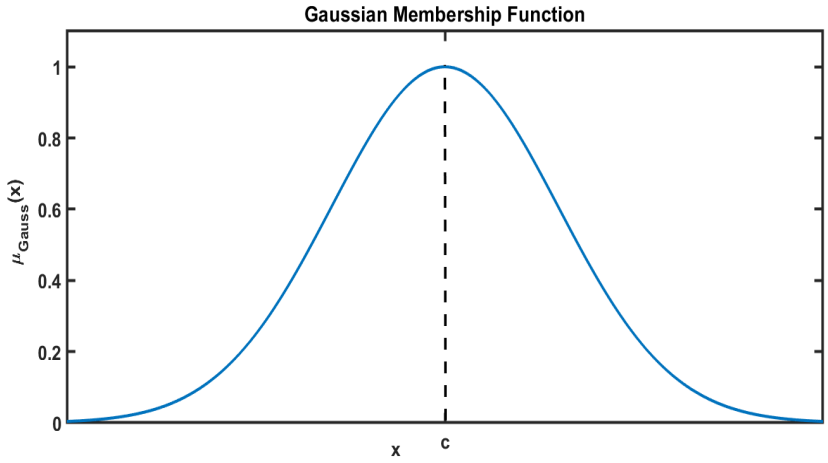

Figure 2.3: *Gaussian Membership Function*

2.2.3 Gaussian MF

Let c and σ denote the mean and standard deviation of a Gaussian function. A Gaussian MF (Fig. 2.3) of a fuzzy set is defined as

$$\mu_{Gauss}(x) = e^{\left(-\frac{1}{2}\left|\frac{x-c}{\sigma}\right|\right)} \tag{2.42}$$

2.2.4 Generalized Bell-shaped MF

A Generalized bell-shaped MF is defined as

$$\mu_{Bell}(x) = \frac{1}{1 + \left|\frac{x-c}{a_1}\right|^{2a_2}}, \tag{2.43}$$

where the parameter c controls the central point, a_1 alters the width and a_2 alters the sharpness of the bell-shaped MF, as shown in Fig. 2.4.

2.2.5 Sigmoid MF

A sigmoid MF is defined as

$$\mu_{Sig}(x) = \frac{1}{1 + e^{(-\lambda(x-c))}}, \tag{2.44}$$

where c is the coordinate of the crossover point and λ controls the slope of the MF as the crossover point (Fig. 2.5).

In our study, we developed and employed a new type of parametric MF called the

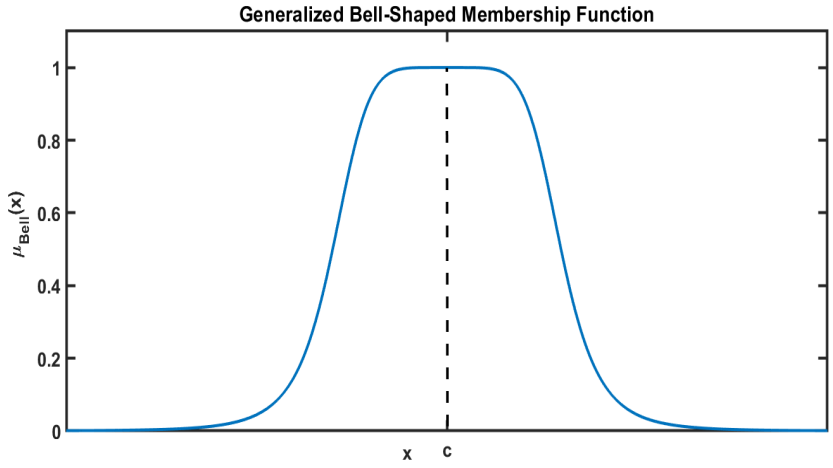

Figure 2.4: *Generalized Bell-shaped Membership Function*

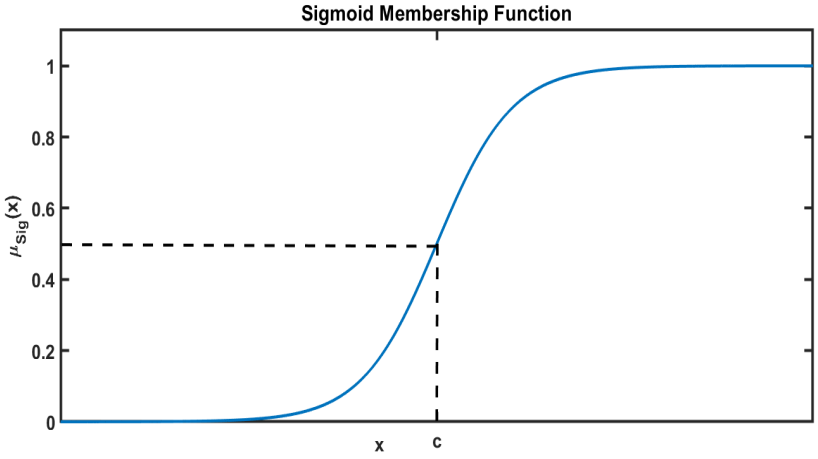

Figure 2.5: *Sigmoid Membership Function*

Distending Function. Now, we will briefly introduce this MF and discuss its unique features.

2.3 The Distending Function

The Distending Function (DF) is a continuous function which is monotonically increasing in the interval $(-\infty,\ 0)$ and monotonically decreasing in the interval $(0,\ +\infty)$ and it takes values in $[0,\ 1]$. The DF $\left(\delta_{\varepsilon,\nu}^{(\lambda)}(x)\right)$ is a parametric membership function. The parameters used are the threshold (ν), tolerance/error (ε) and sharpness (λ). Now we will introduce two types of DF namely, symmetric and asymmetric.

2.3.1 The symmetric Distending Function

The symmetric DF is symmetric around $x - c$ and it is defined as:

Definition 1. *The symmetric DF (shown in Fig. 2.6) is given by*

$$\delta^{(\lambda)}_{\varepsilon,\nu}(x-c) = \frac{1}{1+\frac{1-\nu}{\nu}\left|\frac{x-c}{\varepsilon}\right|^{\lambda}}, \tag{2.45}$$

where $\nu \in (0,1)$, $\varepsilon > 0$, $\lambda \in (1,+\infty)$ *and* $c \in \mathbb{R}$. $\delta^{(\lambda)}_{\varepsilon,\nu}(x-c)$ *will be denoted by* $\delta_s(x)$.

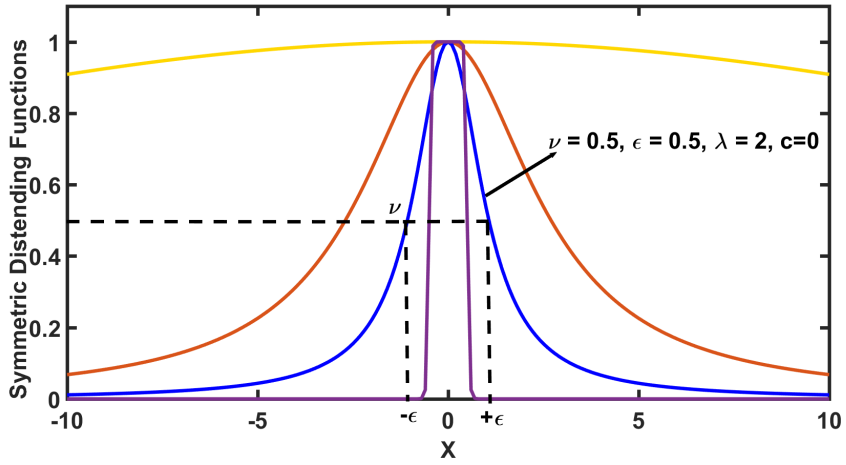

Figure 2.6: *Various shapes of symmetric Distending Functions depending on the parameter values*

The parameters of the DF have the following meanings. The DF has a peak value of 1 at $x = c$. If the input is in the interval $[-\epsilon,\ \epsilon]$, the value of the DF is greater than ν and also $\delta_s(x = \pm\varepsilon) = \nu$. When the input is in the interval $[-\epsilon,\ \epsilon]$, it is treated as a truth value. The threshold (ν) divides the $[0,\ 1]$ interval into truth and falseness regions. And the parameter λ controls the sharpness of the DF. If $\lambda \to \infty$, then the DF approaches the characteristic function. With the appropriate values of ν, ε and λ, all the existing T2FSs s (Trapezoidal, Gaussian, sigmoid, etc) can be approximated using the DF. The T2FSs can be shifted by a parameter c and we shall use the notation $\left(\delta^{(\lambda)}_{\varepsilon,\nu}(x-c)\right)$. We can interpret it when x approaches c, i.e. when x is equal to c (soft equality). Here c is the center point of the DF $(\delta^{(\lambda)}_{\varepsilon,\nu}(c) = 1)$.

2.3.2 The asymmetric Distending Function

The asymmetric type of the DF can be defined in the following way:

Definition 2. *The asymmetric DF (shown in Fig. 2.7) is given by*

$$\delta_A(x) = \frac{1}{1 + \frac{1-\nu_R}{\nu_R}\left|\frac{x-c}{\varepsilon_R}\right|^{\lambda_R}\frac{1}{1+e^{-\lambda^*(x-c)}} + \frac{1-\nu_L}{\nu_L}\left|\frac{x-c}{\varepsilon_L}\right|^{\lambda_L}\frac{1}{1+e^{\lambda^*(x-c)}}}, \tag{2.46}$$

where $\nu_R, \nu_L \in (0,1)$, $\varepsilon_R, \varepsilon_L > 0$, $\lambda_L, \lambda_R \in (1, +\infty)$, $c \in \mathbb{R}$ *and* $\lambda^* \in (1, +\infty)$. λ^* *is a*

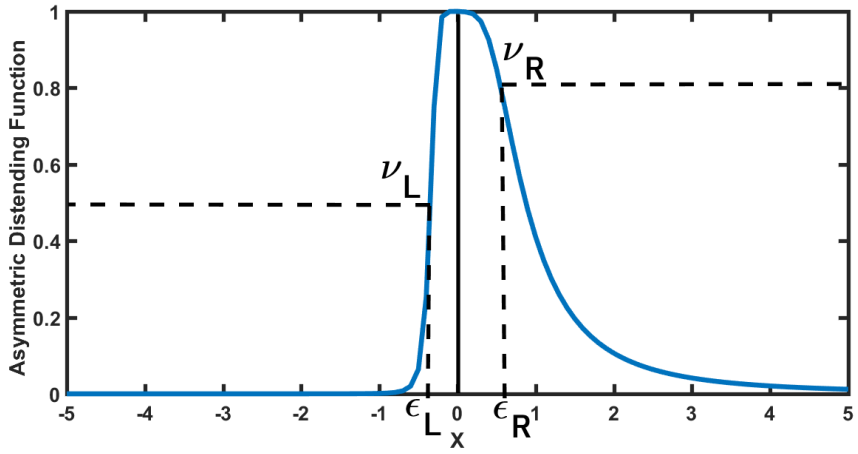

Figure 2.7: *The asymmetric Distending Function (*$\nu_L = 0.5$, $\varepsilon_L = 0.5$, $\lambda_L = 5$, $\nu_R = 0.8$, $\varepsilon_R = 0.7$, $\lambda_R = 5$, $\lambda = 5$, $c = 0$*)*

technical parameter and its value is very large compared to λ_R *and* λ_L. ν_L, ε_L *and* λ_L *are the parameters of the left hand side whereas* ν_R, ε_R *and* λ_R *are the parameters of the right hand side of the asymmetric DF. Here,* c *is the centre point i.e.* $\delta_A(c) = 1$.

The asymmetric DF provides more flexibility in control design. Here, the right and left hand sides of the asymmetric DF can be controlled independently.
Next, we will give a formula to calculate the coordinate of the Centre of Gravity (COG) of the asymmetric DF. First, we calculate the area under the DF. We will consider only one (right hand) side of the DF. Using this area, we will derive an expression for the coordinate of the COG.

2.3.3 Area Under the Distending Function

The integral of the DF can be written in the form

$$I = \int_0^{+\infty} \frac{1}{1 + \frac{1-\nu}{\nu}\left|\frac{x}{\varepsilon}\right|^{\lambda}} \mathrm{d}x, \tag{2.47}$$

where

$$I_{\frac{1}{2}} = \int_0^{\infty} \frac{1}{1 + \frac{1-\nu}{\nu}\left(\frac{x}{\varepsilon}\right)^{\lambda}} \mathrm{d}x. \tag{2.48}$$

By applying the

$$t = \left(\frac{1-\nu}{\nu}\right)^{\frac{1}{\lambda}} \frac{x}{\varepsilon} \tag{2.49}$$

substitution, the integral I can be written as

$$I = \varepsilon \left(\frac{\nu}{1-\nu}\right)^{\frac{1}{\lambda}} \int_{-\infty}^{+\infty} \frac{1}{1+t^{\lambda}} \mathrm{d}t \tag{2.50}$$

The result is

$$I = \varepsilon \left(\frac{\nu}{1-\nu}\right)^{\frac{1}{\lambda}} \frac{\frac{\pi}{\lambda}}{\sin\frac{\pi}{\lambda}}. \tag{2.51}$$

2.3.4 Center of Gravity (COG) Defuzzification

Let the function $\delta^{(\lambda)}_{\varepsilon,\nu,+}(x)$ ("+" means the right hand side) be defined as follows:

$$\delta^{(\lambda)}_{\varepsilon,\nu,+}(x) = \begin{cases} 0, & \text{if } x < 0 \\ \frac{1}{1+\frac{1-\nu}{\nu}\left|\frac{x}{\varepsilon}\right|^{\lambda}}, & \text{if } x \geq 0, \end{cases} \tag{2.52}$$

where $\nu \in (0,1)$, $\varepsilon > 0$ and $\lambda \in \mathbb{R}, \lambda > 1$. Let x^* denote the horizontal coordinate of the COG, as shown in Fig. 2.8. It is well known that

$$x^* = \frac{\int_{-\infty}^{+\infty} x\delta^{(\lambda)}_{\varepsilon,\nu,+}(x)\mathrm{d}x}{\int_{-\infty}^{+\infty} \delta^{(\lambda)}_{\varepsilon,\nu,+}(x)\mathrm{d}x}. \tag{2.53}$$

Then the coordinate x^* of the COG of the area under the curve for $\delta^{(\lambda)}_{\varepsilon,\nu,+}(x)$ is

$$x^* = \frac{1}{2}\varepsilon \left(\frac{\nu}{1-\nu}\right)^{\frac{1}{\lambda}} \frac{1}{\cos\frac{\pi}{\lambda}}. \tag{2.54}$$

Proof. Since $\delta^{(\lambda)}_{\varepsilon,\nu,+}(x) = 0$ if $x < 0$, then using Section. 2.3.3, it is trivial to prove that

Eq. (2.53) reduces to Eq. (2.54).

$$x^* = \frac{\int_0^{+\infty} x\delta^{(\lambda)}_{\varepsilon,\nu,+}(x)\mathrm{d}x}{\int_0^{+\infty} \delta^{(\lambda)}_{\varepsilon,\nu,+}(x)\mathrm{d}x} = \frac{\int_0^{+\infty} x\delta^{(\lambda)}_{\varepsilon,\nu,+}(x)\mathrm{d}x}{\varepsilon\left(\frac{\nu}{1-\nu}\right)^{\frac{1}{\lambda}}\frac{\frac{\pi}{\lambda}}{\sin\frac{\pi}{\lambda}}}. \tag{2.55}$$

Now, applying the

$$t = \left(\frac{1-\nu}{\nu}\right)^{\frac{1}{\lambda}}\frac{x}{\varepsilon} \tag{2.56}$$

substitution leads to

$$x^* = \varepsilon\left(\frac{\nu}{1-\nu}\right)^{\frac{1}{\lambda}}\frac{\sin\frac{\pi}{\lambda}}{\frac{\pi}{\lambda}}\int_0^{\infty}\frac{t}{1+t^{\lambda}}\mathrm{d}t, \tag{2.57}$$

and utilizing Eq.(2.50), we get

$$x^* = \varepsilon\left(\frac{\nu}{1-\nu}\right)^{\frac{1}{\lambda}}\frac{\sin\frac{\pi}{\lambda}}{\sin\frac{2\pi}{\lambda}} = \frac{1}{2}\frac{\varepsilon\left(\frac{\nu}{1-\nu}\right)^{\frac{1}{\lambda}}}{\cos\frac{\pi}{\lambda}}. \tag{2.58}$$

□

Using this, the coordinate x^* of the COG for both sides of the asymmetric DF is

$$x^* = \frac{\Delta_L^2 S_1 - \Delta_R^2 S_2}{\Delta_L + \Delta_R}, \tag{2.59}$$

where

$$\Delta_L = \varepsilon_L\left(\frac{\nu_L}{1-\nu_L}\right)^{\frac{1}{\lambda_L}}\frac{\frac{\pi}{\lambda_L}}{\sin\frac{\pi}{\lambda_L}} \;;\; \Delta_R = \varepsilon_R\left(\frac{\nu_R}{1-\nu_R}\right)^{\frac{1}{\lambda_R}}\frac{\frac{\pi}{\lambda_R}}{\sin\frac{\pi}{\lambda_R}}$$
$$S_1 = \frac{\sin\frac{\pi}{\lambda_L}}{\frac{2\pi}{\lambda_L}\cos\frac{\pi}{\lambda_L}} \;;\; S_2 = \frac{\sin\frac{\pi}{\lambda_R}}{\frac{2\pi}{\lambda_R}\cos\frac{\pi}{\lambda_R}}.$$

Remark. *Some special cases:*

1. *If $\nu_L = \nu_R = \nu$ and $\lambda_L = \lambda_R = \lambda$, then*

$$x^* = \frac{\varepsilon_L^2\left(\frac{\nu}{1-\nu}\right)^{\frac{1}{\lambda}} - \varepsilon_R^2\left(\frac{\nu}{1-\nu}\right)^{\frac{1}{\lambda}}}{\varepsilon_L + \varepsilon_R}\frac{1}{\cos\frac{\pi}{\lambda}}.$$

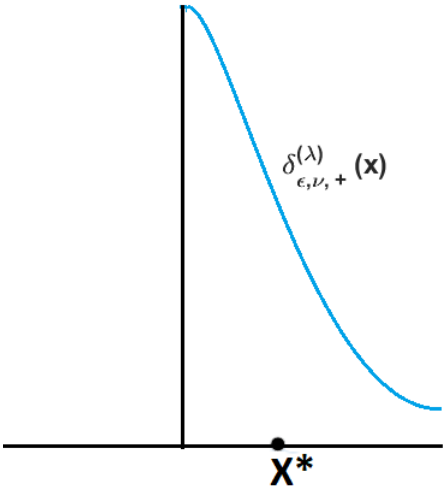

Figure 2.8: *The COG of the DF (RHS)*

2. *If* $\nu = 0.5$*, then*

$$x^* = \frac{1}{2}\left(\varepsilon_L - \varepsilon_R\right)\frac{1}{\cos\frac{\pi}{\lambda}}$$

and if $c \neq 0$*, then*

$$x^* - c = \frac{1}{2}\left(\varepsilon_L - \varepsilon_R\right)\frac{1}{\cos\frac{\pi}{\lambda}}. \tag{2.60}$$

3. *For symmetric DFs,* $\varepsilon_R = \varepsilon_L$*, so*

$$x^* = c, \tag{2.61}$$

which gives an expression for the coordinate of the COG of the symmetric DFs.

Now, we will evaluate the derivatives of DF. These will be used in the optimization process.

2.3.5 Derivatives of the Distending Function

Let

$$\delta(x) = \delta^{(\lambda)}_{\varepsilon,\nu}(x) = \frac{1}{1 + \frac{1-\nu}{\nu}\left|\frac{x}{\varepsilon}\right|^{\lambda}}. \tag{2.62}$$

Then the first derivative of Eq. (2.62) is:

$$\frac{\partial}{\partial x}\delta(x) = -\frac{\lambda}{x}\delta(x)\left(1 - \delta(x)\right).$$

The partial derivatives of the DF with respect to ε is

$$\frac{\partial}{\partial \varepsilon}\delta(x) = -\frac{\lambda}{\varepsilon}\delta(x)(1-\delta(x)).$$

It is worth mentioning that the derivatives of DF are similar to the derivatives of the sigmoid function and these derivatives can readily be calculated just using the DF.

Chapter 3

Arithmetic-based Type1 Fuzzy System

In this chapter we present the design of type1 Fuzzy Inference System (FIS) using fuzzy arithmetic and Distending Function (DF). The proposed new FIS is simple, fast and computationally efficient, compared to the classical techniques (Mamdani, Takagi Sugeno) and it can also adapt itself to the process dynamics. The unique features are: 1) DF is used as membership function; 2) A general parametric operator system is used. It utilizes most of the fuzzy operator systems for evaluating the knowledge base; 3) Inference is based on fuzzy arithmetic operations; 4) This leads to a computationally efficient single-step defuzzification. With these concepts, the paradigm of fuzzy control design changes radically. Using this technique with an optimization method, an adaptive fuzzy FIS is designed. This adaptive FIS adjusts itself to the changing dynamics of the non-linear processes by tuning the DF. The effectiveness of the proposed methodology is demonstrated on two industrial processes, namely a water tank system and continuously stirred tank reactor system.

3.1 Fuzzy Arithmetic

It was suggested by Zadeh [143] that fuzzy quantities can be combined arithmetically using the laws of fuzzy theory. This direction was then explored independently by many researchers [40, 52, 107]. Later it was established that fuzzy theory is an extension of the algebra of many-valued logic and interval analysis [42, 135]. Thus interest in the fuzzy interval domain has increased [84]. Fuzzy arithmetic can be viewed as the arithmetic of α cuts. It handles the fuzzy quantities which are obtained by mapping a real number to the real interval $[0, 1]$. We create a α cuts ($\alpha \in [0, 1]$) for each fuzzy quantity and then perform the required operation by applying the principles of interval arithmetic. Here, instead of intervals, we will use the left and right hand sides of DFs, which are defined on these intervals. This is possible in the case where two sides of the given functions are monotonously increasing or decreasing functions. The details and advantages of using fuzzy arithmetic operations in control

design are given in [37].
Next, we will show that DFs are closed under a linear combination i.e. a linear combination of DFs is also a DF.

3.1.1 Linear Combination of Distending Functions

Let

$$\delta^{(\lambda)}_{i(\nu,\varepsilon_i,c_i)}(x) = \frac{1}{1+\left(\frac{1-\nu}{\nu}\right)\left|\frac{x-c_i}{\varepsilon_i}\right|^{\lambda}},$$

where $i = 1, \ldots, n$ are the DFs that have the same ν and λ values. These n DFs can be combined using fuzzy arithmetic operations and the result is

$$\delta_R(x) = \frac{1}{1+\frac{1-\nu}{\nu}\left|\frac{x-c_R}{\varepsilon_R}\right|^{\lambda}}. \tag{3.1}$$

δ_R is the resultant DF obtained from the linear combination of n DFs.

Proposition 1. *Consider n DFs $\delta^{(\lambda)}_{1(\nu,\varepsilon_1,c_1)}(x), \delta^{(\lambda)}_{2(\nu,\varepsilon_2,c_2)}(x), \ldots, \delta^{(\lambda)}_{n(\nu,\varepsilon_n,c_n)}(x)$. The weighted linear combination of these DFs will also be a DF $\delta^{(\lambda)}_{R(\nu,\varepsilon_R,c_R)}(x)$, where*

$$\varepsilon_R = \sum_{i=1}^{n} w_i\varepsilon_i, \qquad c_R = \sum_{i=1}^{n} w_i c_i \tag{3.2}$$

Proof. Consider two DFs $\delta^{(\lambda)}_{1(\nu,\varepsilon_1,c_1)}(x)$, $\delta^{(\lambda)}_{2(\nu,\varepsilon_2,c_2)}(x)$ and two inputs x_1 and x_2. Let α be the values of δ_1 and δ_2 at the inputs x_1 and x_2 respectively, as shown in Fig. 3.1.

$$\delta^{(\lambda)}_{1(\nu,\varepsilon_1,c_1)}(x_1) = \frac{1}{1+\left(\frac{1-\nu}{\nu}\right)\left|\frac{x_1-c1}{\varepsilon_1}\right|^{\lambda}} = \alpha,$$
$$\delta^{(\lambda)}_{2(\nu,\varepsilon_2,c_2)}(x_2) = \frac{1}{1+\left(\frac{1-\nu}{\nu}\right)\left|\frac{x_2-c2}{\varepsilon_1}\right|^{\lambda}} = \alpha.$$

Then x_1 and x_2 can be expressed as

$$w_1x_1 = w_1\left(\left(\frac{\nu}{1-\nu}\frac{1-\alpha}{\alpha}\right)^{\frac{1}{\lambda}}\varepsilon_1 + c_1\right) \tag{3.3}$$

$$w_2x_2 = w_2\left(\left(\frac{\nu}{1-\nu}\frac{1-\alpha}{\alpha}\right)^{\frac{1}{\lambda}}\varepsilon_2 + c_2\right), \tag{3.4}$$

where w_1 and w_2 are the weights of the fuzzy rules applicable to the input x_1 and x_2. Let $x = w_1x_1 + w_2x_2$. Then by adding Eq. (3.3) and Eq. (3.4), we get

$$x = \left(\frac{\nu}{1-\nu}\frac{1-\alpha}{\alpha}\right)^{\frac{1}{\alpha}} ((w_1\varepsilon_1 + w_2\varepsilon_2) + (w_1c_1 + w_2c_2))$$

From this, we get

$$\delta_R(x) = \frac{1}{1 + \left(\frac{1-\nu}{\nu}\right)\left|\frac{x-(w_1c_1+w_2c_2)}{(w_1\varepsilon_1+w_2\varepsilon_2)}\right|^{\lambda}} \tag{3.5}$$

For n inputs $x_1, x_2, ..., x_n$ and n DFs $\delta^{(\lambda)}_{1(\nu,\varepsilon_1,c_1)}(x), \delta^{(\lambda)}_{2(\nu,\varepsilon_2,c_2)}(x), \ldots, \delta^{(\lambda)}_{n(\nu,\varepsilon_n,c_n)}(x)$, the result can be generalized to

$$\delta_R(x) = \frac{1}{1 + \frac{1-\nu}{\nu}\left|\frac{x-c_R}{\varepsilon_R}\right|^{\lambda}}, \tag{3.6}$$

where δ_R is the resultant DF obtained from the linear combination of n DFs with c_R and ε_R given by Eq. (3.2). □

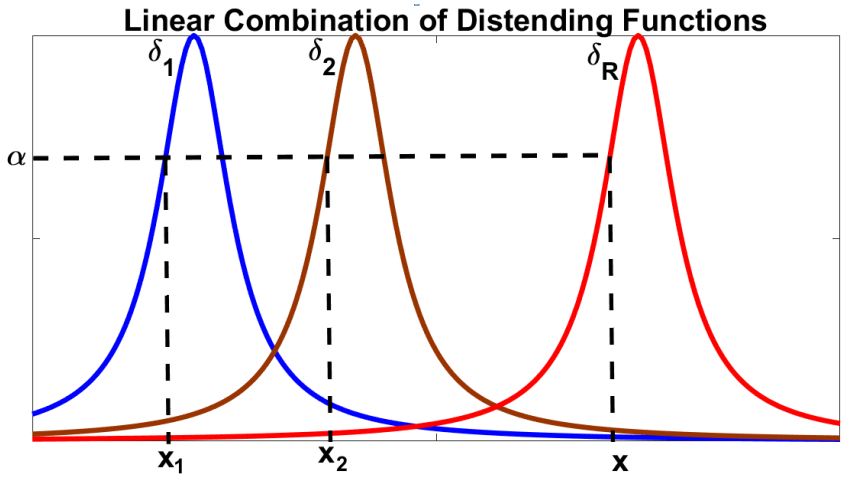

Figure 3.1: *A linear combination of two DFs δ_1 and δ_2 using fuzzy arithmetic operations on α cuts*

3.2 Arithmetic Type1 FIS

Our design methodology was motivated by a previous study where a fuzzy system was designed using fuzzy arithmetic operations [37]. It has all the desired properties,

such as:

- Range independence,
- The effect of fuzziness is fully incorporated,
- The computation speed is very high compared to conventional COG defuzzification-based FIS.

Using expert knowledge or process data, the multi-input multi-output system (MIMO) is described by:

$$\begin{aligned} &\text{If } x_1 \text{ is } A_1^i \text{ and } ... \text{ and } x_n \text{ is } A_n^i \\ &\text{then } y_1 \text{ is } B_1^i \text{ ; ... ; } y_m \text{ is } B_m^i \text{ ,} \end{aligned} \tag{3.7}$$

where $x_1, x_2, ..., x_n$ are the input linguistic variables which take the values from the input fuzzy subsets $A_1, A_2, ..., A_n$. The variables $y_1, y_2, ..., y_m$ are the output linguistic variables which take the values from the output fuzzy subsets $B_1, B_2, ..., B_m$. $i = 1, ..., l$ is the number of fuzzy rules. If the output variables are independent of each other, then each rule of the rule base given by Eq. (3.7) can be written as m multi-input single output (MISO) rules

$$\text{If } x_1 \text{ is } A_1^i \text{ and } ... \text{ and } x_n \text{ is } A_n^i \text{ then } y \text{ is } B^i \text{ .} \tag{3.8}$$

We will try to find the fuzzy inference mechanism to map the input-output space and generate a crisp output signal. In our approach, we handle the antecedent and consequent parts of the rule separately.

3.2.1 The antecedent part

The antecedent part of the ith fuzzy rule is

$$\mathcal{L}(\delta_1(x_1)^i, \delta_2(x_2)^i, \ldots, \delta_n(x_n)^i) = \hat{w}_i(x), \tag{3.9}$$

where $\hat{w}_i(x)$ is the rule applicability function. $\mathcal{L}$ is the fuzzy logical expression and it may include *and* ($x_1 \in A_1$ and $x_2 \in A_2$), *or* ($x_1 \in A_1$ or $x_2 \in A_2$) and *not* ($x_1 \notin A_1$) operators. Here, we use the Generalized Dombi operator (GDO)

$$D_\gamma(x) = \frac{1}{1 + \left(\frac{1}{\gamma}\left(\prod_{i=1}^{n}\left(1 + \gamma\left(\frac{1-\delta(x_i)}{\delta(x_i)}\right)^\alpha\right) - 1\right)\right)^{\frac{1}{\alpha}}}. \tag{3.10}$$

Most of the conjunctive or disjunctive operators used (e.g min/max, product, Einstein, Hamacher, Dombi, drastic) are covered by Eq. (3.10).

For specific input values of $\underline{x}^*$, Eq. (3.9) can be evaluated and this results in a single numeric value $\hat{w_i}(\underline{x}^*)$

$$\mathcal{L}(\delta^i_{1(x^*_1)}, \delta^i_{2(x^*_2)}, \ldots, \delta^i_{n(x^*_n)}) = \hat{w_i}(\underline{x}^*), \tag{3.11}$$

where $\hat{w_i}(x^*)$ is called the strength of the ith rule. We normalize these strengths (to compare the rules) to get the firing strengths $w_i(x^*)$. The firing strength gives the probability of the rule. The firing strength of the ith rule is

$$w_i(\underline{x}^*) = \frac{\hat{w_i}(\underline{x}^*)}{\sum_{i=1}^{l} \hat{w_i}(\underline{x}^*)}, \tag{3.12}$$

where

$$\sum_{i=1}^{l} w_i(\underline{x}^*) = 1. \tag{3.13}$$

3.2.2 The consequent part

This part of the rule is a fuzzy set represented by a single DF. The firing strength of each rule (calculated from the antecedent part) is multiplied by the consequent part. Also the fuzzy output of each rule is a DF. By combining all the rules, we can generate an aggregated output. If $w_1(\underline{x}^*), w_2(\underline{x}^*), \ldots, w_l(\underline{x}^*)$ are the firing strengths and $\delta_{1o}(x), \delta_{2o}(x), \ldots, \delta_{lo}(x)$ are the l consequents, then the aggregated output of the l fuzzy rule is given by:

$$\delta_a(x) = \sum_{i=1}^{l} w_i(\underline{x}^*)\delta_{io}(x). \tag{3.14}$$

Of course, $\delta_a(x)$ is also a DF. We calculate the parameters of the DF using Eq. 3.2. The aggregated output DF $\delta_a(x)$ has the following form

$$\delta_a(x) = \frac{1}{1 + \frac{1-\nu}{\nu}\left|\frac{x-c_a}{\varepsilon_a}\right|^{\lambda}}, \tag{3.15}$$

where

$$c_a = \sum_{i=1}^{l} w_i(\underline{x}^*)c_i, \qquad \varepsilon_a = \sum_{i=1}^{l} w_i(\underline{x}^*)\varepsilon_i, \tag{3.16}$$

where c_i and ε_i are the parameters of the ith consequent. The crisp output will be the COG of the aggregated DF $\delta_a(x)$.

Our design approach has the following unique features:

- A general parametric operator is used to calculate the firing strength of rules. This single operator can be employed to calculate the Zadeh, product, Einstein, drastic and Dombi t-norm and t-conorms with appropriate values of the parameters.
- The inputs are fuzzified using DFs. These DFs can approximate most types of bell-shaped membership functions. Also, the input member functions might have different shapes (Gaussian, Trapezoidal, Sigmoidal) at the same time for different categories.
- The aggregate functions is also a DF. This is due to the fact that a linear combination of DFs is a also a DF.
- The design procedure is simplified because it does not include implication. The aggregation is carried out using fuzzy arithmetic operations.
- Defuzzification is a single-step calculation.

Hence, the proposed approach is computationally efficient.

3.2.3 Algorithm

The whole procedure is summarized in Algorithm 1.

Algorithm 1 Algorithm for the synthesis of arithmetic-based FIS using a DF

Step 1: Define the DFs for the input and output linguistic variables.
Step 2: Fuzzify the crisp inputs using Eq. (2.45) or Eq. (2.46).
Step 3: Construct the rule base from the expert knowledge using Eq. (3.8).
Step 4: Calculate the strength of each rule using Eq. (3.10) by choosing the appropriate fuzzy conjunctive/disjunctive operators.
Step 5: Calculate the l firing strengths using Eq. (3.12).
Step 6: Calculate the parameters (c_a, ε_a) of the aggregated output DF via Eq. (3.16).
Step 7: Generate the aggregated output DF using Eq. (3.15).
Step 8: Get the crisp output signal u by calculating the COG of the aggregated output DF using Eq. (2.59) or Eq. (2.61).

3.3 Adaptive FIS Design

If the process parameters vary with time, then the adaptive FIS changes the output signal in accordance with the change in process dynamics. Therefore, the adaptive

FIS works even when the values of the parameters are outside the desired range. Here, we shall present a hybrid scheme for adaptive FIS. We call it a hybrid because it utilizes the knowledge base and an optimization method. The hybrid scheme selects one fuzzy rule from the knowledge base and it tunes the parameters of the antecedent part. This rule will be selected based on the firing strength. The optimization technique is used to tune the parameters of the antecedent part.
Here we have three tunable parameters, namely ν, λ and ε. We shall fix the values of ν and λ and we will tune the ε parameter. The knowledge base of a FIS consists of "If then" rules that have the following form:

$$\begin{bmatrix} \text{If } x_1 \text{ is } A_{11} \text{ and }, \ldots, x_n \text{ is } A_{1n} \text{ then } y \text{ is } B_1 \\ \vdots \\ \text{If } x_1 \text{ is } A_{i1} \text{ and }, \ldots, x_n \text{ is } A_{in} \text{ then } y \text{ is } B_i \\ \vdots \\ \text{If } x_1 \text{ is } A_{l1} \text{ and }, \ldots, x_n \text{ is } A_{ln} \text{ then } y \text{ is } B_l \end{bmatrix}, \tag{3.17}$$

where $i = 1, 2, \ldots, l$ is the number of rules. $A_{i1}, A_{i2}, \ldots, A_{in}$ are the input linguistic terms and B_i are the output linguistic terms. All the input and outputs are associated with corresponding DFs. Here, y is the fuzzy output. Now let $\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_l$ be the strengths of the l fuzzy rules and $c_1, c_2, \ldots, c_l$ be the COG values of l corresponding DFs. The crisp value Y (3.17) is

$$Y = \frac{c_1\hat{w}_1 + c_2\hat{w}_2 + \cdots + c_l\hat{w}_l}{\hat{w}_1 + \hat{w}_2 + \cdots + \hat{w}_l}. \tag{3.18}$$

Let the squared error function has the form

$$E = \frac{1}{2}\left(Y_{ref} - Y\right)^2, \tag{3.19}$$

where Y_{Ref} is the known reference output signal and Y is the output signal (crisp output) generated by the FIS. Now the adaptive FIS problem reduces to the following optimization task;

$$\underset{\varepsilon>0}{Minimise}\left(E\right). \tag{3.20}$$

Using the gradient descent method,

$$\varepsilon_{t+1} = \varepsilon_t - \eta_s \frac{\partial}{\partial \varepsilon}\left(E\right). \tag{3.21}$$

From Eq. (3.19), we have

$$\varepsilon_{t+1} = \varepsilon_t + 2\eta_s E \frac{\partial}{\partial \varepsilon} \left(\frac{c_1 \hat{w_1} + c_2 \hat{w_2} + \cdots + c_l \hat{w_l}}{\hat{w_1} + \hat{w_2} + \cdots + \hat{w_l}} \right), \tag{3.22}$$

where η_s is the step size of the optimization. To reduce the complexity and computational cost, we will select one rule and tune the antecedent part at the same time. The rule will be selected on the basis of the rule strengths $(\hat{w_1}, \hat{w_2}, \ldots, \hat{w_l})$. Now we will explain this selection and the tuning procedure.
Let the ith rule has the highest firing strength. The remaining $(l-1)$ strengths will be constant. Eq. (3.22) can be written as

$$\varepsilon_{t+1} = \varepsilon_t + 2\eta_s E \frac{\partial}{\partial \varepsilon_i} \left(\frac{c_i \hat{w}_i + k_1}{\hat{w}_i + k_2} \right), \tag{3.23}$$

where

$$k_1 = c_1 \hat{w}_1 + \cdots + c_{i-1} \hat{w}_{i-1} + c_{i+1} \hat{w}_{i+1} + \cdots + c_l \hat{w}_l$$
$$k_2 = \hat{w}_1 + \cdots + \hat{w}_{i-1} + \hat{w}_{i+1} + \cdots + \hat{w}_l.$$

From Eq. (3.23), we get

$$\frac{\partial}{\partial \varepsilon_i} \left(\frac{c_i \hat{w}_i + k_1}{\hat{w}_i + k_2} \right) = \frac{(\hat{w}_i + k_2)(c_i \hat{w}_i^{'}) + (c_i \hat{w}_i + k_1)\hat{w}_i^{'}}{(\hat{w}_i + k_2)^2}, \tag{3.24}$$

where

$$\hat{w}_i^{'} = \frac{\partial}{\partial \varepsilon_i}(\hat{w}_i).$$

By using Eq. (3.10),

$$\hat{w}_i^{'} = \frac{\partial}{\partial \varepsilon_i} \left(\frac{1}{1 + \left(\sum_{i=1}^{n} \left(\frac{1-\delta_i(x_i)}{\delta_i(x_i)} \right)^{\alpha} \right)^{\frac{1}{\alpha}}} \right). \tag{3.25}$$

Here $\delta_1(x_1), \ldots, \delta_n(x_n)$ are the n antecedent DFs in the ith rule. To reduce the computation cost, the ε parameter of only one antecedent DF will be tuned at a time. The DF to be tuned is selected by comparing the grade of membership of these n DFs. The DF with the highest grade is selected for tuning. The remaining $(n-1)$ antecedent DFs in the ith rule will be treated as constants. Let ε_i be the parameter of the ith

antecedent DF having the highest grade value. Then Eq. (3.25) can be written as

$$\hat{w}_i^{'} = \frac{\partial}{\partial \varepsilon_i} \left(\frac{1}{1 + \left(\left(\frac{1-\delta_i(x_i)}{\delta_i(x_i)} \right)^{\alpha} + k_3 \right)^{\frac{1}{\alpha}}} \right), \tag{3.26}$$

where

$$k_3 = \sum_{k=1}^{i-1} \left(\frac{1-\delta_k(x_k)}{\delta_k(x_k)} \right)^{\alpha} + \sum_{k=i+1}^{n} \left(\frac{1-\delta_k(x_k)}{\delta_k(x_k)} \right)^{\alpha}.$$

For the Dombi conjunctive operator, $\alpha = 1$. Since k_3 is independent of ε_i,

$$\begin{aligned} \hat{w}_i^{'} &= \frac{\partial}{\partial \varepsilon_i} \delta_i(x_i) \\ &= -\frac{\lambda}{\varepsilon_i} \delta_i(x_i) \left(1 - \delta_i(x_i)\right). \end{aligned} \tag{3.27}$$

Eq. (3.23) can be written as

$$\varepsilon_{t+1} = \varepsilon_t - \frac{2\lambda \eta_s E \delta_i(x_i) \left(1 - \delta_i(x_i)\right) \left(2c_i \hat{w}_i + c_i k_2 + k_1\right)}{\varepsilon_i (w_i + k_2)^2}, \tag{3.28}$$

which is the update for ε of the ith antecedent DF of the ith rule.

3.3.1 Algorithm

The procedure for designing the adaptive FIS is summarized in Algorithm 2.

Algorithm 2 Algorithm for the synthesis of the adaptive FIS using DF

Step 1: Define a tolerance τ as an acceptable upper bound on error E.
Step 2: Calculate the error E between the fuzzy output Y and reference output Y_{Ref}.
Step 3: If $E \geq \tau$, then perform the following steps (steps 4 to 9) else exit.
Step 4: Calculate the rule strengths $\hat{w}_1, \ldots, \hat{w}_l$ of l fuzzy rules.
Step 5: Select the rule with the highest rule strength.
Step 6: Calculate the grade of membership of n antecedent DFs within in the selected rule using Eq. (2.45) or Eq. (2.46).
Step 7: Select the antecedent DF with the highest grade of membership.
Step 8: Update the parameter ε of the selected antecedent DF using Eq. (3.28).
Step 9: Go to Step 2.

3.4 Experiments, Results and Discussion

The effectiveness of the proposed technique is demonstrated using the simulation case studies of two practical systems.

3.4.1 Water Tank Level Control

Water Tank Model

Consider a water tank system, as shown in Fig. 3.2. Water continuously flows in and out of the tank. There is also a valve at the inlet pipe to control the inflow to the tank. The rate of change of the water volume V inside the tank is

$$\frac{dV}{dt} = q_{in} - q_o, \tag{3.29}$$

where q_{in} and q_o are the inflow and outflow rates. If A_t is the area of the tank base and l is the height of the liquid in the tank, then

$$A_t \frac{dl}{dt} = q_{in} - A_o\sqrt{2gl}, \tag{3.30}$$

where A_o is the cross sectional area of the outlet and g is the acceleration due to gravity. A control signal u is sent to the valve located on the inlet pipe and the height of water column inside the tank is controlled by the change in the ratio of inlet and outlet flow rates. The level l of the water in the tank is measured using a level sensor.

Control Scenario (Changing the water level)

A fuzzy controller is designed using our approach to control the water level of the tank at the specified height (reference) by opening/closing the inlet valve. The difference in the measured level l and the reference height (called the Level Error) is fed to the fuzzy controller as an input. To control the level efficiently, the rate of change of the level in the tank is also fed to the controller as a second input. The controller then generates the control signal for opening and closing the valve. The controller rule base is shown in Table 3.1. The multiple inputs are fuzzified using the DF (Eq. (2.45) or Eq. (2.46)). DFs are also defined for the output control i.e. the valve opening/closing signal. The DFs for the antecedent and consequent parts are shown in Fig. 3.3. The control action is generated using Algorithm 1. Here, Fig. 3.5 shows the control surface of the fuzzy controller. A reference signal for changing the level of water in the tank between 5m and 15m is fed to the system. Fig. 3.6 and Fig. 3.7 show the response comparison of the proposed and the conventional Mamdani

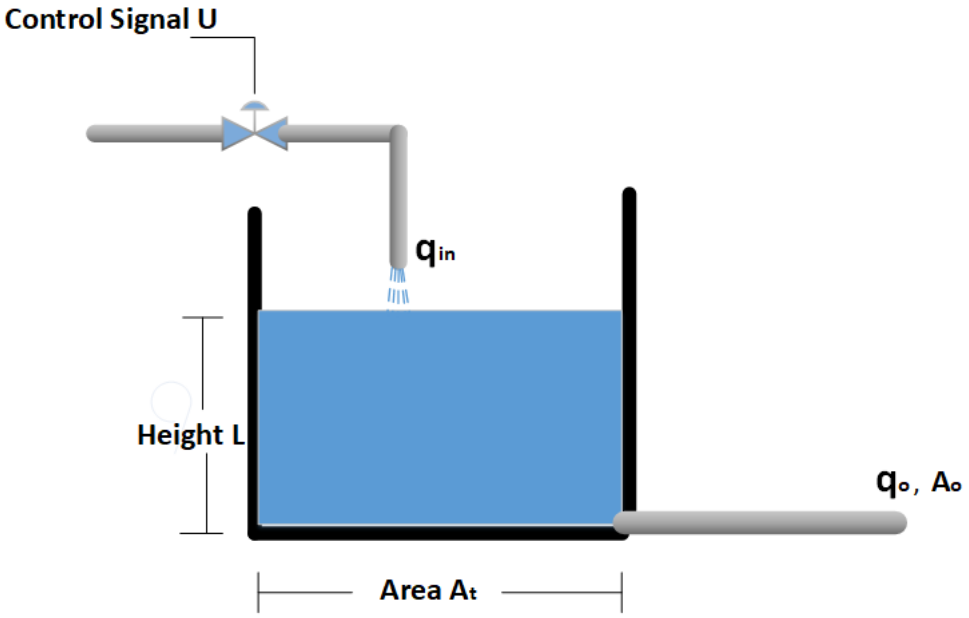

Figure 3.2: *Water Tank level system*

controllers. The conventional controller is based on Gaussian membership functions, product conjunction and implication operators, max aggregation and COG defuzzification. Fig. 3.8 shows the performance of the proposed fuzzy controllers using symmetric and asymmetric DFs. The consequent DFs for the asymmetric DF-based controller are shown in Fig. 3.4.

Level Error \ Rate of Level	Positive	Negative	Don't Care
High	-	-	Close Fast
Low	-	-	Open Fast
OK	Close Slow	Open Slow	No Change

Table 3.1: *The rule base for the Water Tank level fuzzy controller*

3.4.2 Temperature Control of the Continuously Stirred Tank Reactor (CSTR)

CSTR Dynamical Model

CSTR is a chemical reactor (shown in Fig. 3.9) which converts a hazardous chemical A into an acceptable product B, which is then disposed of in the natural environment. The reactor consists of a tank, a cooling jacket and a continuous stirring mechanism.

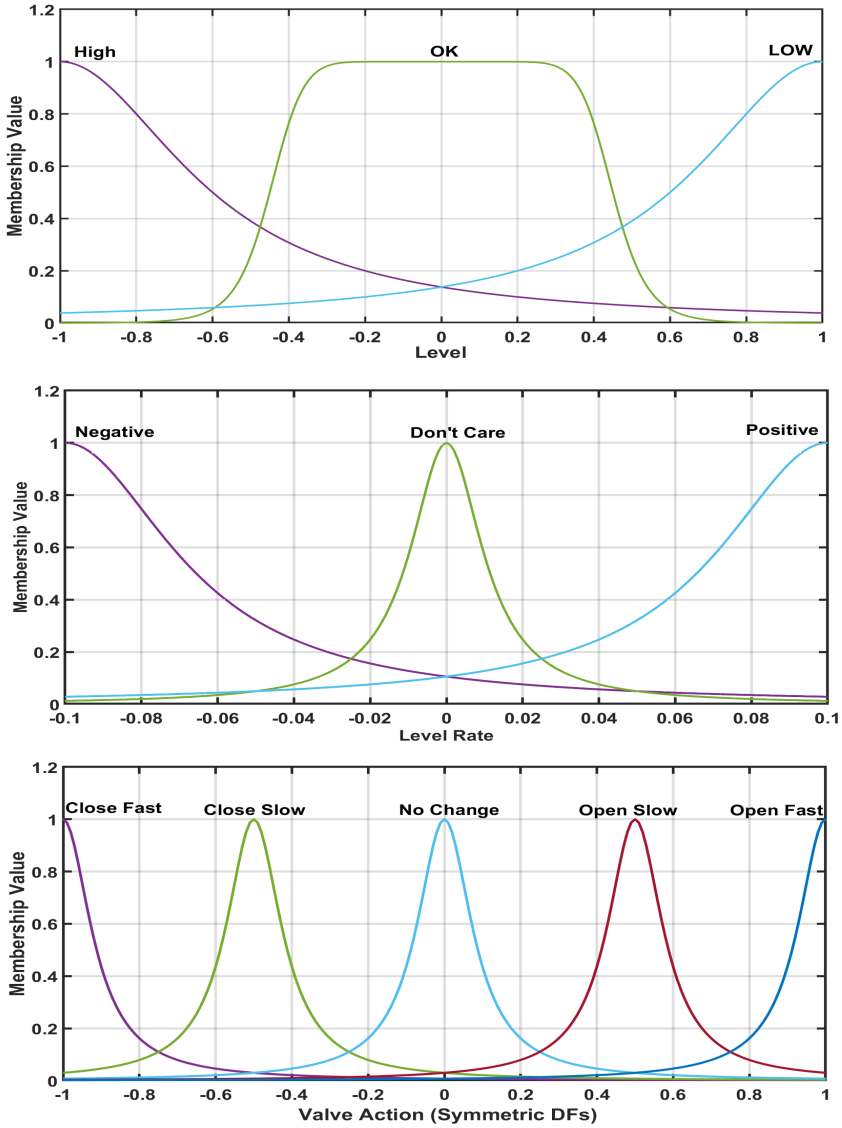

Figure 3.3: *Antecedent and Consequent DFs for the Water Tank level controller (Symmetric)*

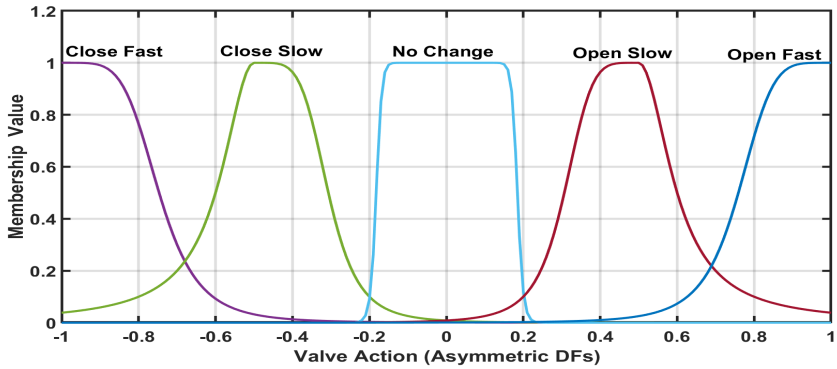

Figure 3.4: *Consequent DFs for the Water Tank level controller (Asymmetric)*

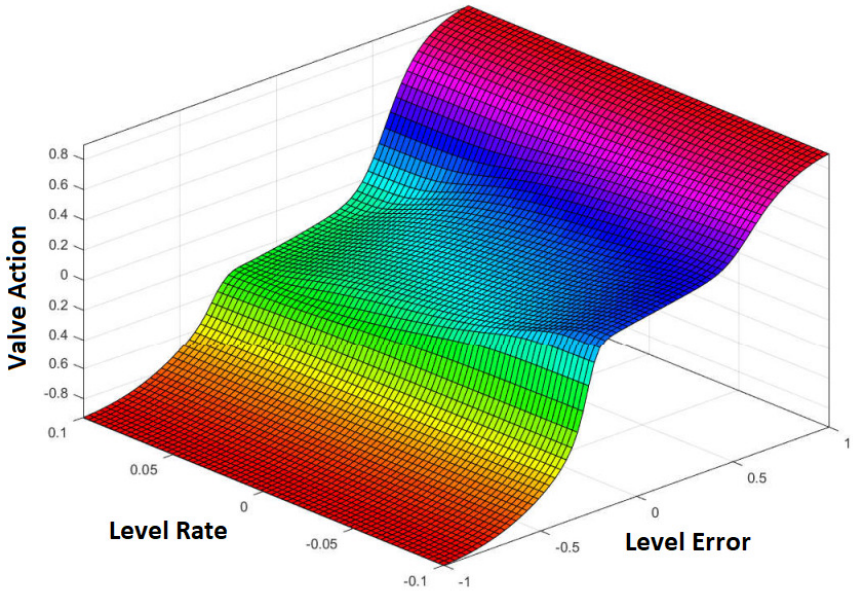

Figure 3.5: *The control surface for the Water Tank level controllers*

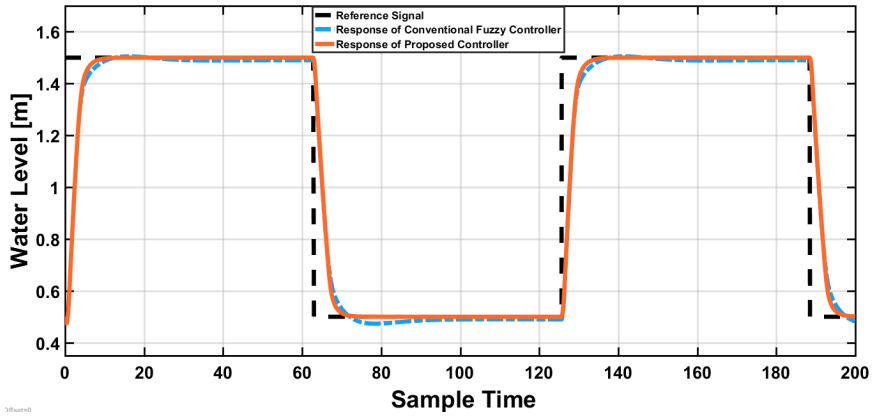

Figure 3.6: *The response comparison of our proposed controller with a conventional fuzzy controller*

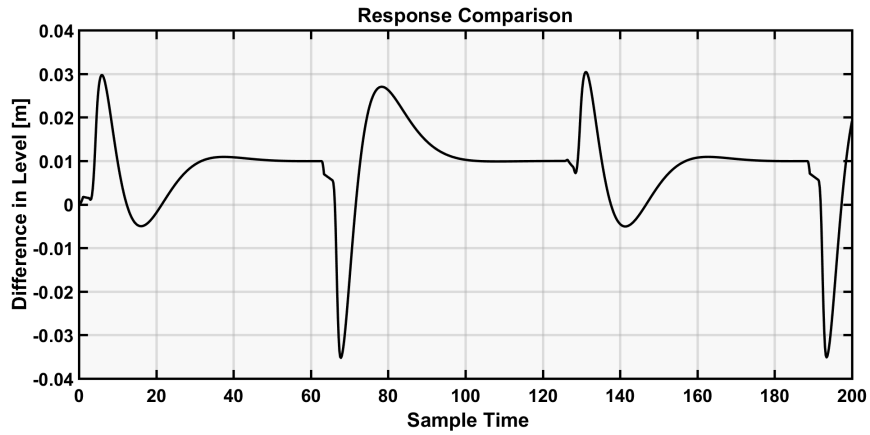

Figure 3.7: *The difference in response between the proposed controller and the conventional controller (proposed - conventional)*

The volume of the chemical inside the tank is usually kept constant. The tank temperature (T_R) and concentration C_A of the chemical A in the outlet stream are the important variables. The reaction is exothermic and irreversible. The tank is continuously stirred for proper mixing to get uniform temperature and concentration profiles. By changing the jacket temperature (T_J), the tank temperature T_R and concentration C_A can be controlled. If T_R reaches the high threshold limit then temperature runaway can occur in the reactor and result in an unsafe operation. The dynamical model of CSTR was derived using mole balance and energy balance equations. It leads to the following state equations:

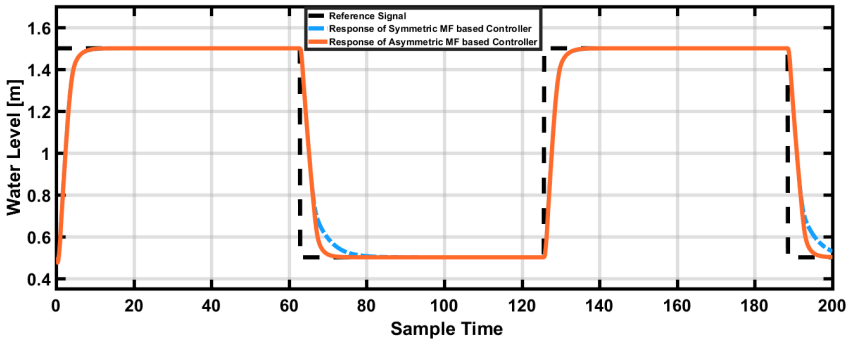

Figure 3.8: *The response comparison of the symmetric and asymmetric DF-based fuzzy controllers*

$$\begin{aligned} \dot{x_1} &= T_F - x_1 + x_2 \frac{\Delta H_r k_0 e^{\frac{-E}{R_T}}}{\rho C_p} + U_A(T_J - x_1); \\ \dot{x_2} &= \frac{q}{V}(C_{AF} - x_2) - x_2 k_0 e^{\frac{-E}{R_T}}, \end{aligned} \tag{3.31}$$

Here, x_1 and x_2 are the states of the process and they represent T_R and C_A, respectively. T_J is the control variable. The relevant parameters of this model are given in Table 3.2. An open source Matlab package was used for the simulations of CSTR [58].

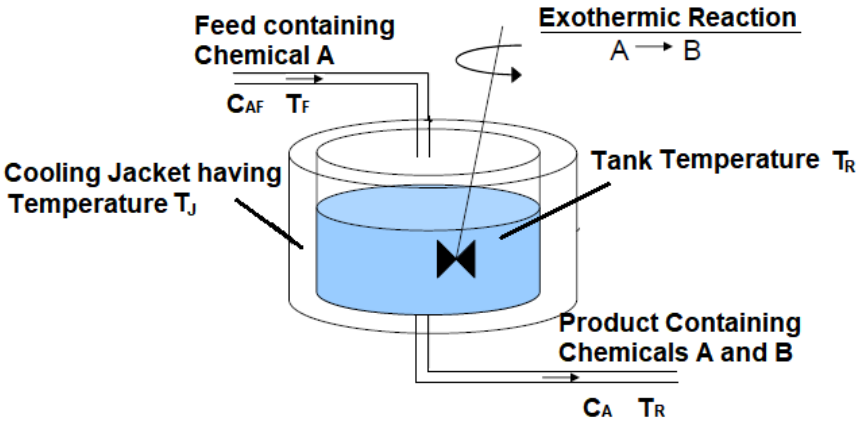

Figure 3.9: *Continuously Stirred Tank Reactor*

S.No	Paramter	Description
1	T_J	Temperature of cooling jacket (K)
2	q	Volumetric Flowrate (m^3/sec)
3	V	Volume of liquid in CSTR (m^3)
4	ρ	Density of A→B Mixture (kg/m^3)
6	C_p	Heat capacity of A-B Mixture ($J/kg-K$)
7	ΔH_r	Heat of reaction for A-¿B (J/mol)
8	k_0	Pre-exponential factor ($1/sec$)
9	U_A	Overall heat transfer coefficient ($U = W/m^2 - K$)
10	C_{AF}	Feed Concentration (mol/m^3)
11	T_F	Feed Temperature (K)
12	C_A	Concentration of A in CSTR (mol/m^3)
13	T_R	Temperature in CSTR (K)
14	R_T	Residence Time (Sec)

Table 3.2: *CSTR model parameters*

Control Scenario (Tracking the Reactor Temperature T_R)

A fuzzy controller based on DFs is designed to track T_R at a set point of $365°k$ and C_A ratio in the outlet stream below 0.3. The high threshold for the reactor temperature is $400°K$. T_R must remain below this threshold to avoid temperature runaway. The controller generates the change in T_J to achieve the desired T_R. The rule base of the controller consists of seven rules, as shown in Table 3.3. Inputs are fuzzified using the DF (Eq. (2.46)) and rules are evaluated using Dombi operators (see Eq. 3.10). The reference signal governing the reactor temperature is subtracted from the actual reactor temperature T_R of the CSTR to generate an error signal E. The error in the reactor temperature E, rate of change of the error signal dE and feed temperature T_F form the input to the fuzzy controller. The antecedents and consequent DFs are shown in Fig. 3.13. The control surface (for two inputs; E and T_F) of the fuzzy controller is shown in Fig. 3.12. A reference signal commands the controller to achieve the desired value of the temperature T_R by changing T_J. The response of the tuned PID controller (P =4, I = $.8$, D= 0.5, N=100) has also been plotted for comparison purposes. Fig. 3.10 shows the reactor temperature during the simulation scenario. The top figure shows the response of the PID and DF-based controllers when the feed temperature is $350°K$. The bottom figure shows the responses when the feed temperature is increased to $370°K$. The parameters of the PID and DF-based controller were kept the same. It is evident from the responses that the PID response exceeds the high temperature threshold ($400°K$) when the feed temperature is increased. However the DF-based controller keeps the reactor temperature within the threshold limits, but with a slightly slow response. The concentration C_A has been plotted in

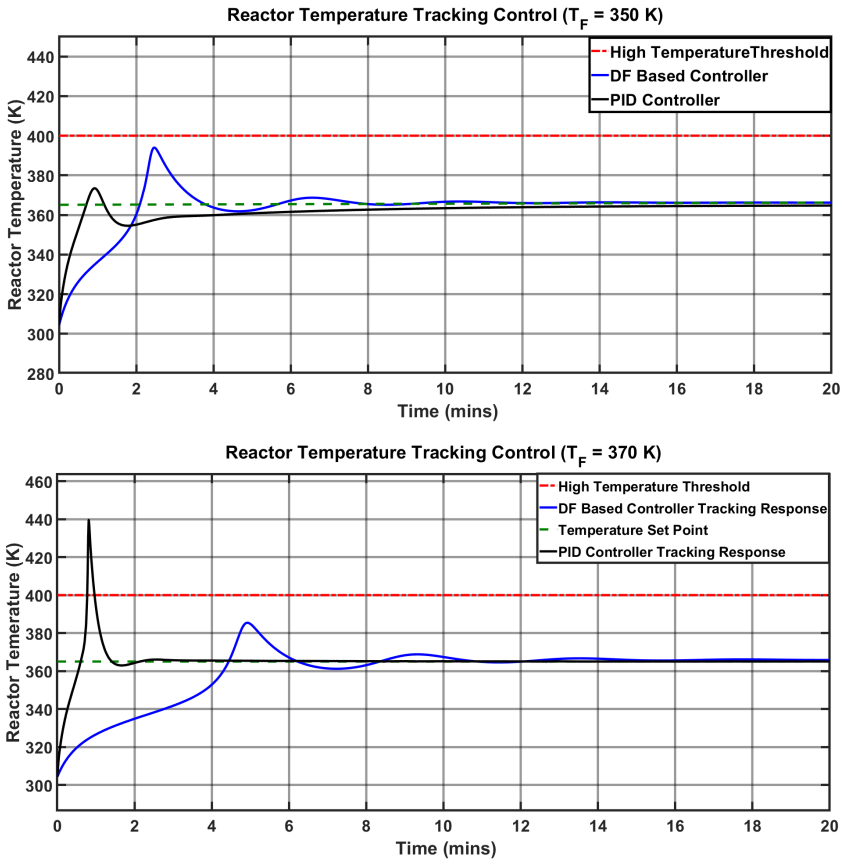

Figure 3.10: *A comparison of tracking responses generated using the PID (P =*4*, I = .*8*, D=* 0.5*, N=*100*) and DF-based controller for different feed temperatures. Both controllers track the reactor temperature when* $T_F = 350°K$ *and they are within the safe operating temperature limits (Top). Only the DF-based controller tracks and keeps the reactor temperature below the high threshold when* $T_F = 370°K$ *(Bottom)*

Fig. 3.11 (the $T_F = 350°K$ case).

3.4.3 Adaptive FIS

The effectiveness of the proposed adaptive fuzzy FIS is examined in a reactor temperature tracking situation where the process dynamics change. The volume V of liquid in the reactor is an important parameter and it must remain constant. This is usually ensured by a separate level control system which keeps the liquid in the reactor at a constant level and hence the volume remains unchanged. However it may happen that the reactor liquid volume increases or decreases due to a fault in the liquid level controller or a sudden increase/decrease in the chemical A inventory. In such cases, the reactor temperature will not follow the desired set point and it may lead to a runaway situation. Now, consider the case in which the reactor liquid volume increases

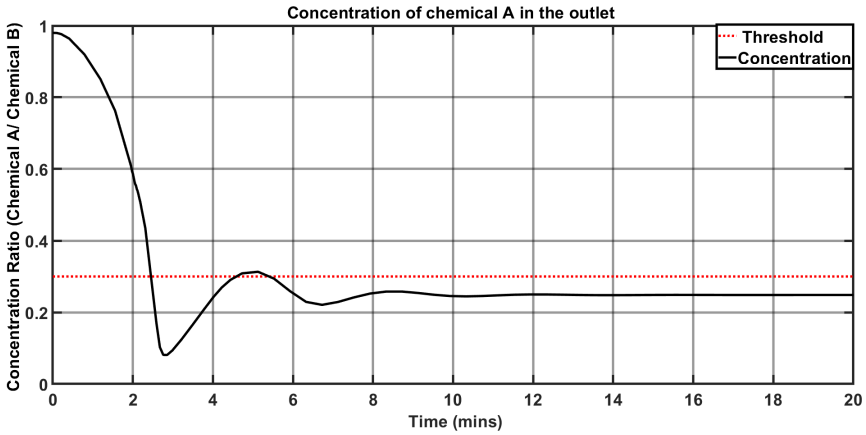

Figure 3.11: *The concentration of chemical A in the outlet ($T_F = 350°K$)*

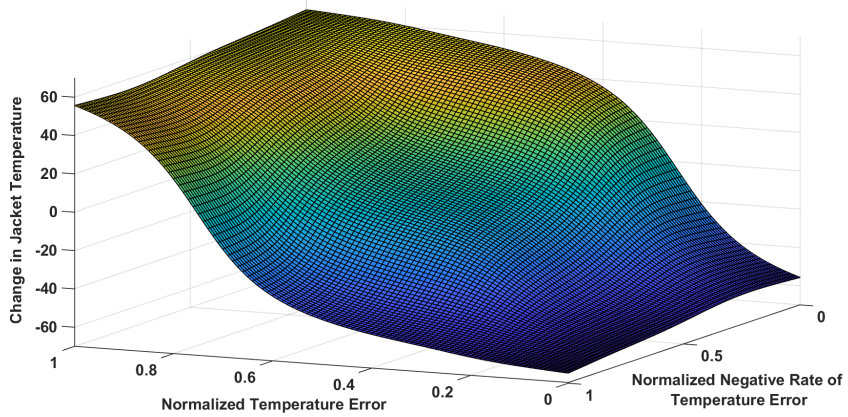

Figure 3.12: *The control surface of the CSTR fuzzy controller. Two inputs (temperature error and negative rate of temperature error) are used here.*

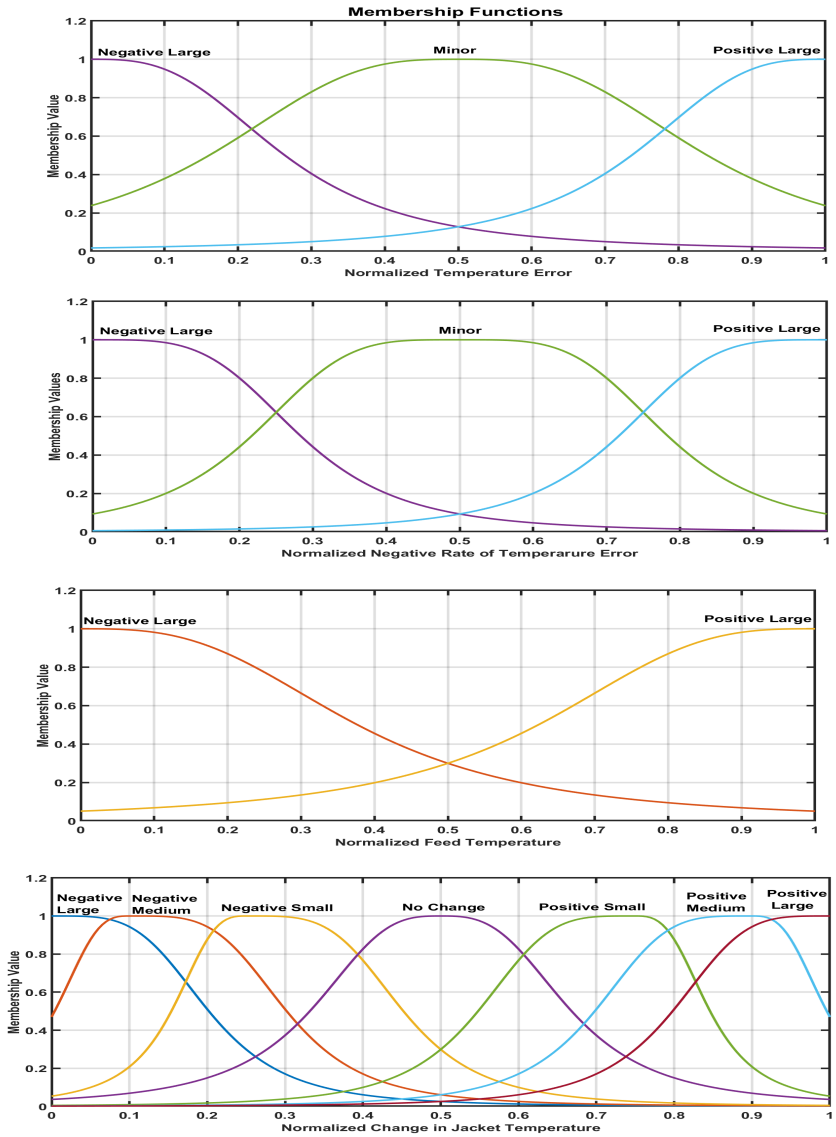

Figure 3.13: *The antecedent and consequent DFs of the CSTR controller*

Rule No.	Temperature Error	Negative Rate of Temperature Error	Feed Temperature	Change in Jacket Temperature
1	PL	-	-	PL
2	NL	-	-	NL
3	M	-	-	NC
4	M	PL	-	NM
5	M	NL	-	PM
6	PL	-	PL	NS
7	PL	-	NL	PS

Table 3.3: *Rule base of CSTR fuzzy controller. PL- Positive Large, NL- Negative Large, PM- Positive Medium, NM- Negative Medium, PS- Positive Small, NS- Negative Small, M- Minor, NC- No Change*

by 10%. Without adaptive control, the reactor temperature is shown in Fig. 3.14. It is clear that reactor temperature exceeds the higher threshold limit (resulting in unsafe operating conditions) when there is a slight change in the liquid volume. We apply Algorithm 2 to tune the parameters of DFs. The response of the adaptive controller is shown in Fig. 3.15. The adaptive algorithm is able to tune the parameters of DFs to such a degree that the reactor temperature does not reach the higher threshold and the reactor temperature follows the desired set point even when there is a change in the reactor liquid volume.

3.4.4 Discussion

From the simulation results, it can be seen that the proposed fuzzy controller follows the reference command signal very efficiently. The control surfaces indicate that the fuzzy controller has very smooth transitions for these nonlinear processes (Fig. 3.5 and Fig. 3.12). The performance is much better than the conventional fuzzy controller and PID. Compared to the conventional fuzzy controller, the proposed controller has a short rise time, peak time and percentage overshoot (Fig. 3.6 and Fig. 3.7). The symmetric and asymmetric DF-based controllers converge to a zero steady state error, but the response of the asymmetric DF-based controller is better due to its short rise and peak times (Fig. 3.8). The left and right hand sides of the asymmetric DF can be shaped independently, resulting in a more flexible control. Computation efficiency is the major advantage of the proposed method and the reasons are: 1) The grade of membership can be calculated very quickly using the DF. 2) There is no implication, aggregation is based on fuzzy arithmetic operations and defuzzification is a single step calculation. Table 3.4 shows the computation cost of the

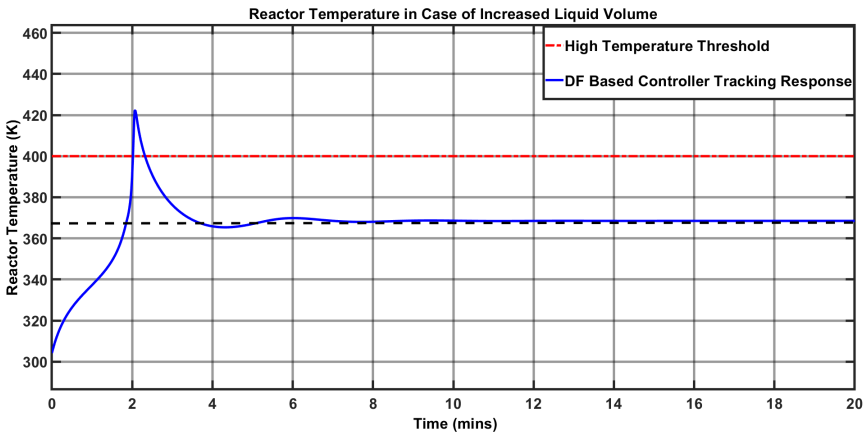

Figure 3.14: *The reactor temperature exceeded the high temperature threshold when the liquid volume increased.*

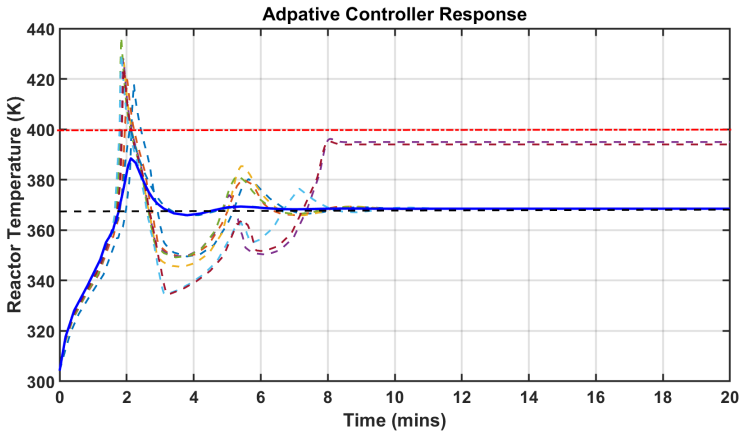

Figure 3.15: *Adaptive controller performance. The dashed lines show the results of various iterations. The solid blue line shows the final response of the adaptive controller.*

classical fuzzy controller based on the Mamdani inference model and the proposed arithmetic-based controller. The conventional controller used Gaussian membership functions, product conjunction and implication operators, max aggregation and COG defuzzification. Different cases based on the number of fuzzy rules and the numerical resolution (n) of the output control surface were considered. Each entry is the average result of 100 simulations. It was shown that when we have 8 fuzzy rules and a very high numerical resolution (i.e. n = 1001 x 1001), the DF-based fuzzy arithmetic control technique is at least 21 times faster than the conventional Mamdani controller. As shown in Fig. 3.15, the adaptive algorithm converges in only 8 iterations. The proposed adaptive controller needs less computation time and it has a rapid convergence. For adaptation convergence, an upper bound on ϵ can be defined using the range of the input signal (i.e. $\frac{Input\ range}{4}$). In our simulations, the ϵ parameter always converged within a few iterations. New rules can be added in the knowledge base if the adaptation procedure fails to find the value of ϵ between the upper and lower bounds. This would mean that the process dynamics has changed significantly and existing rules cannot control the system even after adaptation.

	n = 11 x 11			n = 101 x 101			n = 1001 x 1001		
	3 Rules	5 Rules	8 Rules	3 Rules	5 Rules	8 Rules	3 Rules	5 Rules	8 Rules
Conventional Mamdani Inference	.0712	.0748	.0932	5.235	5.735	6.8722	20.68	22.34	27.30
Arithmetic Symmetric DF	.00078	.0013	.0014	.0950	0.111	.1150	1.184	1.256	1.278
Arithmetic Asymmetric DF	.00079	.0013	.0015	.0954	0.113	.1161	1.189	1.263	1.314
Speed ratio (Mamdani / Symmetric DF)	90	55	65	55	50	60	17	17	21

Table 3.4: *A speed comparison of a conventional Mamdani controller and Arithmetic-based controllers (for both symmetric and asymmetric types)*

3.5 Summary

A novel technique for the design of a arithmetic-based FIS is proposed. It is based on a new type of parametric membership function called the DF. The DF is a continuous and differentiable function (it is analytical). It has a few parameters and it covers the input space with a few rules. A general parametric fuzzy operator is used to calculate

the firing strengths. Also, the operator system and the DF are consistent with each other. The design process is simplified by handling the antecedent and consequent parts separately. Here, the proposed approach does not include any type of implication. Aggregation is performed using fuzzy arithmetic operations, more precisely using a linear combination of the DFs. The result of aggregation is also a DF. Also, defuzzification is just a single step calculation. The technique is simple, computationally efficient and overcomes some of the drawbacks with the existing established techniques.

Based on this new arithmetic-based FIS and the gradient-based optimization method, a hybrid adaptive FIS is presented. It tunes the parameters of the DF using the gradient descent technique. The calculation is fast and the adaptation process converges within a few iterations. The adaptive FIS can satisfactorily achieve the objectives even when the process dynamics change. Computation efficiency is one of the main advantages of the DF-based FIS. It is 20 to 50 times faster than the conventional FISs. Lastly, the effectiveness of the proposed approach is demonstrated using by controlling the level of a water tank system and tank temperature of a continuously stirred reactor system.

Chapter 4

Data-Driven Arithmetic-Based Type1 Fuzzy System

Defining a proper rule system for a Fuzzy Inference System (FIS) is a difficult task. This is due to the unavailability of experts or incomplete knowledge of the process. Fortunately, rules can be extracted from the training data of the system. This allow us to design a data-driven FIS. Data-driven FIS techniques suffer from the flat structure problem i.e. the number of fuzzy rules grows exponentially as the input dimension increases. The consequence of this is the greater complexity and poorer interpretability of the fuzzy rules. In this chapter, we presents a solution to the above-mentioned problem by proposing a novel data-driven type1 FIS. Its unique features are: 1) DF is used as a membership function. 2) It helps us to identify very few and important fuzzy rules. 3) These rules cover the whole input space. 4) Dombi operators (usually the conjunctive) are employed to generate a control surface in a higher dimension. 5) Using the identified rules, a FIS based on fuzzy arithmetic operations is designed. 6) Defuzzification is a single step calculation. These features make it possible for us to design a computationally efficient data-driven FIS. Due to the small number of fuzzy rules, the complexity of the fuzzy model decreases and it becomes interpretable. The effectiveness of the proposed scheme is demonstrated using the data-driven based control of a water tank system and vehicle lateral dynamics control.

4.1 Distending Function in Higher Dimensions

Proposition 2. *Consider n different DFs in different dimensions given by $\Delta^{(\lambda_1)}_{1(\epsilon_1,\nu_1)}(x_1 - a_1)$, $\Delta^{(\lambda_2)}_{2(\epsilon_2,\nu_2)}(x_2 - a_2)\ldots,\Delta^{(\lambda_n)}_{n(\epsilon_n,\nu_n)}(x_n - a_n)$. If we apply Dombi conjunctive operator on these n DFs, then the result will also be a DF $\Delta^{(\lambda_{n+1})}_{n+1(\epsilon_{n+1},\nu_{n+1})}(x_1 - a_1)$ in $(n + 1)$ dimensions.*

Proof. Consider two DFs $\Delta^{(\lambda_1)}_{1(\epsilon_1,\nu_1)}(x_1-a_1)$, $\Delta^{(\lambda_2)}_{2(\epsilon_2,\nu_2)}(x_2-a_2)$ in x_1 and x_2 dimensions.

$$\Delta_{x_1}=\Delta^{(\lambda_1)}_{1(\epsilon_1,\nu_1)}(x_1-a_1)=\frac{1}{1+\frac{1-\nu_1}{\nu_1}\left|\frac{x_1-a_1}{\epsilon_1}\right|^{\lambda_1}}$$

$$\Delta_{x_2}=\Delta^{(\lambda_2)}_{2(\epsilon_2,\nu_2)}(x_2-a_2)\frac{1}{1+\frac{1-\nu_2}{\nu_2}\left|\frac{x_2-a_2}{\epsilon_2}\right|^{\lambda_2}}$$

Let us apply the Dombi conjunctive operator on these two DFs [29].

$$\Delta_{x_3}=C(\Delta_{x_1},\Delta_{x_2})=\frac{1}{1+\frac{1-\Delta_{x_1}}{\Delta_{x_1}}+\frac{1-\Delta_{x_2}}{\Delta_{x_2}}},$$

which results in

$$\Delta_{x_3}=\frac{1}{1+\frac{1-\nu_1}{\nu_1}\left|\frac{x_1-a_1}{\epsilon_1}\right|^{\lambda_1}+\frac{1-\nu_2}{\nu_2}\left|\frac{x_2-a_2}{\epsilon_2}\right|^{\lambda_2}}$$

where Δ_{x_3} is also a DF in the x_3 dimension, as shown in Fig. 4.1. Now for the case for n DFs, the resultant DF $\Delta_{x_{n+1}}$ in $(n+1)$ dimensions is given as

$$\Delta_{x_{n+1}}=\frac{1}{1+\frac{1-\nu_1}{\nu_1}\left|\frac{x_1-a_1}{\epsilon_1}\right|^{\lambda_1}+\cdots+\frac{1-\nu_n}{\nu_n}\left|\frac{x_n-a_n}{\epsilon_n}\right|^{\lambda_n}},$$

which can be written in compact form as

$$\Delta_{x_{n+1}}=\frac{1}{1+\sum_{i=1}^{n}\frac{1-\nu_i}{\nu_i}\left|\frac{x_i-a_i}{\epsilon_i}\right|^{\lambda_i}}. \tag{4.1}$$

□

4.2 Data-Driven arithmetic-based type1 FIS

The proposed algorithm was motivated by our previous study (presented in the Chapter 3) where a FIS was designed using arithmetic operations. Here, we suppose that the expert knowledge in the form of linguistic rules is not available. However the training (working) data of the process is available. Using the training data, a rule base for multi-input multi-output system (MIMO) system can be written as:

$$\begin{aligned}&\text{if } x_1 \text{ is } U_1^i \text{ and } \ldots\ldots \text{ and } x_n \text{ is } U_n^i\\ &\text{then } y_1 \text{ is } V_1^i \text{ ; } \ldots\ldots \text{ ; } y_m \text{ is } V_m^i \text{ ;}\end{aligned} \tag{4.2}$$

where $x_1, x_2, ..., x_n$ are input and $y_1, y_2, ..., y_m$ are 'm' output variables and the corresponding fuzzy subsets are $U_1, U_2, ..., U_n$ and $V_1, V_2, ..., V_m$ respectively. The index i

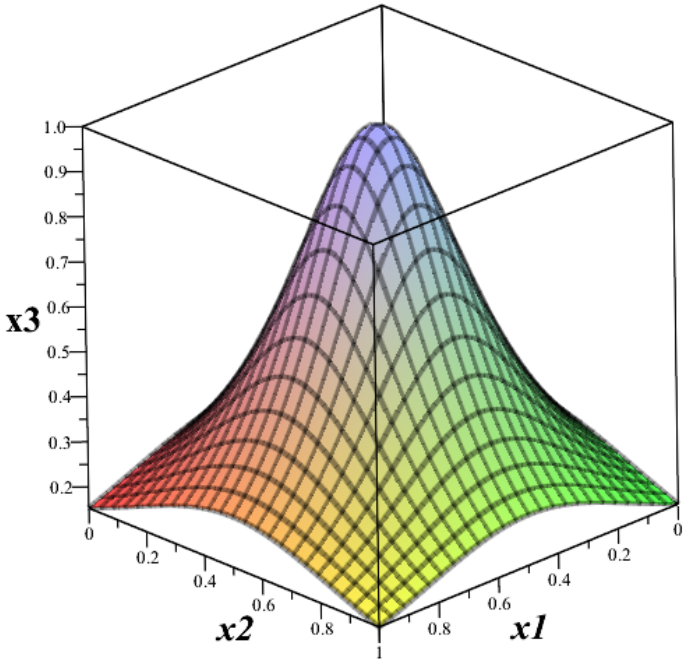

Figure 4.1: *A Distending Function in three dimensions.*

represents the rule number and there are l fuzzy rules. A MIMO system given by Eq. (4.2) with m independent outputs can always be replaced by m multi-input single output (MISO) systems combined together. The MISO system has the following form

$$\begin{aligned} &\text{if } x_1 \text{ is } U_1^i \text{ and } \ldots\ldots \text{ and } x_n \text{ is } U_n^i \\ &\qquad\qquad\qquad\qquad \text{then } y_t \text{ is } V_t^i \end{aligned} \tag{4.3}$$

where $t = 1, ..., m$ are the output of the system. For simplicity, we will consider the case where $t = 1$ and we propose a methodology to generate a crisp output signal using the training data of inputs and output. The methodology can be generalized for m independent outputs.
Let us assume that the input and output training data is given in the following form:

$$U = \begin{bmatrix} a_1^1 & a_2^1 & \ldots & a_n^1 \\ a_1^2 & a_2^2 & \ldots & a_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ a_1^l & a_2^l & \ldots & a_n^l \end{bmatrix}, \qquad V = \begin{bmatrix} b_1^1 \\ b_1^2 \\ \vdots \\ b_1^l \end{bmatrix}; \tag{4.4}$$

where U contains the l data point of each input and V contains the corresponding l output data points. The data points $a_1, a_2, \ldots, a_n$ correspond to the input fuzzy

subsets $U_1, U_2, \ldots, U_n$ and b_1 corresponds to the fuzzy subset V. n different columns of U exist in n different dimensions depending on the input variable. In a similar way V is an $(n+1)$ dimensional output space. We can divide our proposed method into the following three key steps:

4.2.1 Construction of estimated Control Surface and Rule base

From the given process data of Eq. (4.4), an estimated control surface $G^{'}$ can be constructed in the $(n+1)$ dimensional space. The columns of the training data matrix U are transformed to the $[0, 1]$ interval to get an n dimensional unit space. Next, we extract a few rows from U that contain the important values of the input variables. We choose the boundary and average values of each input variable as key values. These rows and the corresponding elements in V are then used to construct the rule base R_b. It is the so-called boundary-value rule base R_b. It is called the boundary-value because it mostly contains the extreme values of the inputs that define the boundary of the control surface. Each fuzzy rule consists of an antecedent (which processes the input data) and a consequent (which processes the output data) part. Here, the antecedent part handles a row of U and the consequent part with an element of V. The antecedent part of the ith fuzzy rule is represented by the following relation

$$\mathcal{L}(\Delta_1(x_1)^i, \Delta_2(x_2)^i, \ldots, \Delta_n(x_n)^i), \tag{4.5}$$

where $\mathcal{L}$ is the fuzzy logical expression and it may contain an AND ($x_1 \in A_1$ and $x_2 \in A_2$), OR ($x_1 \in A_1$ or $x_2 \in A_2$) and NOT ($x_1 \notin A_1$) operators. Here, we use a special class of parametric fuzzy operators [32]

$$D_\gamma(x) = \frac{1}{1 + \left(\frac{1}{\gamma}\left(\prod_{i=1}^{n}\left(1 + \gamma\left(\frac{1-\Delta(x_i)}{\Delta(x_i)}\right)^{\alpha}\right) - 1\right)\right)^{\frac{1}{\alpha}}} \tag{4.6}$$

This operator describes a wide class of fuzzy operators e.g. min/max, Einstein, Hamacher, product and drastic, but here we just use Dombi conjunctive (or disjunctive) operators.

4.2.2 Generation of the Fuzzy Control Surface and Error Surface

Now we will generate the fuzzy control surface $\hat{G}$ using the DF given in Eq. (2.45). We will construct DFs for all the input variables in the antecedent part of each rule. The parameter a of DF will have the value of the corresponding input variable in the rule. Here we let $\lambda = 2$ and $\varepsilon = 0.3$ for the input (antecedent) DFs. The common value of ν is used for all the DFs that have the same rule. This ν value will be set so as to minimize the influence of high dimensional DFs (which will be explained later).

Each rule in the R_b is evaluated using the Dombi conjunctive/disjunctive operator. Suppose that in the antecedent part of ith rule, there are n DFs corresponding to n input variables. These are called input DFs. Each input variable and its corresponding DF are in one of n different dimensions. By applying Dombi conjunctive/disjunctive operators over these n input DFs, this will result in a DF in a higher dimension i.e. $(n+1)$ dimension given by

$$\Delta_{x_{n+1}} = \frac{1}{1 + \frac{1-\nu}{\nu}\left(|\frac{x_1-a_1}{\epsilon_1}|^\lambda + \cdots + |\frac{x_n-a_n}{\epsilon_n}|^\lambda\right)}. \tag{4.7}$$

It is called output DF. For all the l rules in the rule base R_b, l output DFs in $(n+1)$ dimension will be generated. All these output DFs in $n+1$ dimension form a fuzzy control surface $\hat{G}$ in $(n+1)$ dimension. Now we choose a proper value of ν for all these output DFs. The ν value of each output DFs will be calculated based on the principle of minimum influence on all other DFs in the $(n+1)$ dimensional space. As we know well, each DF has a long tail. Each of the l output DF in the $n+1$ dimensional space should not influence other DF. This influence can never be zero but can be decreased by k times. To decrease the influence of ith output DF k times at the location of jth DF, we can derive the following equation from Eq. (4.1):

$$\frac{1}{1 + \frac{1-\nu}{\nu}\left(|\frac{x_{i1}-x_{j1}}{\epsilon}|^\lambda + \cdots + |\frac{x_{in}-x_{jn}}{\epsilon}|^\lambda\right)} = \frac{1}{k}, \tag{4.8}$$

where $x_{i1}, \ldots, x_{in}$ are the n coordinates of the ith output DF in $(n+1)$ dimension and $x_{j1}, \ldots, x_{jn}$ are the coordinates of the jth DF. The expression for required value of ν can be calculated from Eq. (4.8) as

$$\nu = \frac{1}{1 + \frac{k-1}{d}}, \tag{4.9}$$

where $d = \left(|\frac{x_{i1}-x_{j1}}{\epsilon}|^\lambda + \cdots + |\frac{x_{in}-x_{jn}}{\epsilon}|^\lambda\right)$ and it is a distance-like measure.
Based on this, we can define an error surface E as the difference between the estimated control surface G' and the fuzzy control surface $\hat{G}$

$$E_{(x1,\ldots,x_n)} = G'_{(x1,\ldots,x_n)} - \hat{G}_{(x1,\ldots,x_n)}. \tag{4.10}$$

We decrease the magnitude of E below a chosen threshold limit τ. This is achieved by making $\hat{G}$ similar to G' by adding new rules in the rule base R_b. To add a new fuzzy rule in R_b, the point of the maximum value in the error surface is located. The coordinates of the maximum error point are looked for in the training database. The corresponding row in the training database is then added to the rule base R_b as a new fuzzy rule. This new rule is evaluated to generate a new output DF in

$(n+1)$ dimension. This ν value of this new output DF will be calculated based on the principle of minimum influence to the other existing output DFs in $(n+1)$ dimension (using Eq. (4.9)). This new output DF is included in $\hat{G}$. This will modify the surface $\hat{G}$ in such a way that the magnitude of maximum error on the surface E at this location will decrease. This process is repeated iteratively to add new rules to the boundary value rule base R_b, until the error surface E is within the tolerance limit τ.

4.2.3 Designing the Arithmetic-based FIS using the Rule base R_b

The identified rule base R_b is used to generate the FIS. The fuzzy rules in R_b have two parts (Antecedent and consequent parts). We will handle the these parts separately when we design a FIS based on arithmetic operations.

Antecedent Part

The antecedent part of the ith fuzzy rule can be described by the expression

$$\mathcal{L}(\delta_1(x_1)^i, \delta_2(x_2)^i, \ldots, \delta_n(x_n)^i) = \hat{w}_i(x), \tag{4.11}$$

where $\hat{w}_i(x)$ is the rule applicability function and $\mathcal{L}$ is the fuzzy logical expression that contains the logical operators given by Eq. (4.6). For a specific set of input values x^*, Eq. (4.11) can be evaluated and it results in a single numeric value $\hat{w}_i(x^*)$

$$\mathcal{L}(\delta^i_{1(x_1^*)}, \delta^i_{2(x_2^*)}, \ldots, \delta^i_{n(x_n^*)}) = \hat{w}_i(x^*), \tag{4.12}$$

where $\hat{w}_i(x^*)$ is called the strength of the ith rule. To compare the strengths of different rules, we normalize these strengths to get the firing strength (normalized strength) $w_i(x^*)$ of ith rule.The firing strength tell us the probability of the rule. Let

$$W(x^*) = \sum_{i=1}^{l} \hat{w}_i(x^*). \tag{4.13}$$

Then the firing strength of the ith rule is defined as

$$w_i(x^*) = \frac{\hat{w}_i(x^*)}{W(x^*)}, \tag{4.14}$$

where

$$\sum_{i=1}^{l} w_i(x^*) = 1. \tag{4.15}$$

Consequent part

The consequent part of the ith fuzzy rule in R_b is a numeric value and it is an element of matrix V given in Eq. (4.4). Let $b^1, \ldots, b^l$ be consequent values and $w_1^*, \ldots, w_l^*$ be the firing strengths (determined from the antecedent part of the rule) of the l fuzzy rules of R_b. Then the crisp output C of the FIS is given by

$$C = \sum_{i=1}^{l} w_i(x^*) b^i . Ah \tag{4.16}$$

4.2.4 Algorithm

The whole procedure is summarized in Algorithm 3.

Algorithm 3 Algorithm for Data-Diven Type1 FIS

Step 1: Get the training data of the process in the form of Eq. (4.4)
Step 2: From the training database, generate the estimated control surface $G^{'}$. Use interpolation to find missing data points.
Step 3: Transform each column of U to the $[0, 1]$ interval to get a unit space in n dimensions.
Step 4: Generate a boundary value rule base R_b by selecting extreme and average value rows of each inputs in the training database.
Step 5: Assign the DF to each input point in the antecedent part of the rule base R_b. Fix the values of λ and ε.
Step 6: For each rule in R_b, calculate the output DF in $(n + 1)$ dimension using Eq. (4.7). The ν value is calculated using Eq. (4.8) and Eq. (4.9). A value of k can be chosen between 10 to 100.
Step 7: Generate the fuzzy control surface $\hat{G}$ from the the output DF of all the rules in R_b.
Step 8: Generate the error surface using Eq. (4.10). If the error surface is within the tolerance limit τ, go to step 10.
Step 9: Find the maximum value point on the error surface. Add a new rule in R_b corresponding to this training database point. Go to Step 6.
Step 10: Using the identified rule base R_b, generate the required output signal using equations (4.11) to Eq. (4.16).

4.3 Experiments, Simulations and Results Discussion

The effectiveness of the proposed data-driven strategy will be presented using simulation studies on an industrial system.

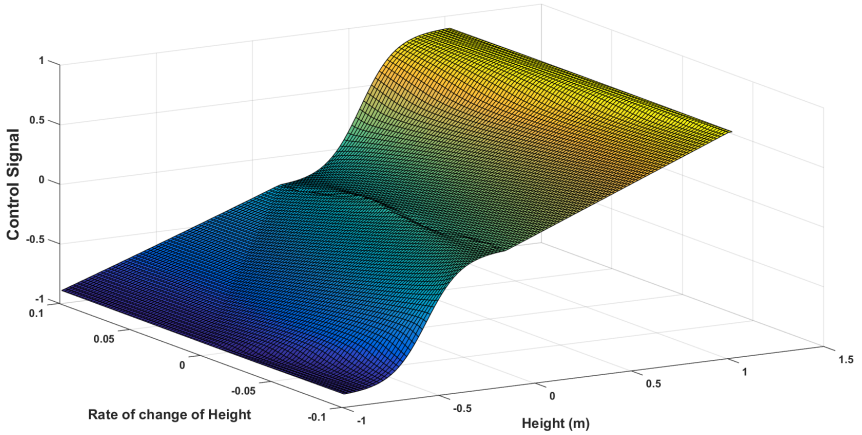

Figure 4.2: *The estimated control surface G'.*

4.3.1 Benchmark: Liquid Tank Level Control

The simulations were performed on the Water Tank system model described in section 3.4.1. In this scenario, we regulated the height of the liquid in the tank at the specified level using a data-driven fuzzy controller. First, training data was generated using a working tank-level controlled system. The data consisted of 2000 samples of the control signal u, the height h and the rate of change of height $\frac{dh}{dt}$. Using this training data an estimated control surface G' was constructed as shown in Fig. 4.2. The training data was used as input for Algorithm 3 in the form compatible with Eq. (4.4). The tolerance level τ was selected to be between $.1$ and $.2$ to minimize the number of identified fuzzy rules. The proposed approach generated a control surface $\hat{G}$ as shown in Fig. 4.3. The error surface E within the specified tolerance is shown in Fig. 4.4. In addition to the initial number of the boundary value rules in R_b, only 3 new rules were identified by the proposed algorithm to generate the control surface $\hat{G}$. The algorithm converged within a few iterations to generate a reasonably good control surface $\hat{G}$. The identified fuzzy rule base was used to design a controller to regulate the height of the liquid at reference point of the tank. The response of the data-driven fuzzy controller for regulating the height of the liquid between $1.5m$ and $.5m$ is shown in Fig. 4.5.

4.3.2 Benchmark: Vehicle Lateral Dynamic Control

Vehicle and Tire Model

To simulate the vehicle lateral dynamics, we used an open access package [25] in the Matlab environment. The non-linear bicycle vehicle model employed is shown in Fig. 4.6. The key parameters used in this model are described in Table 4.1. The whole

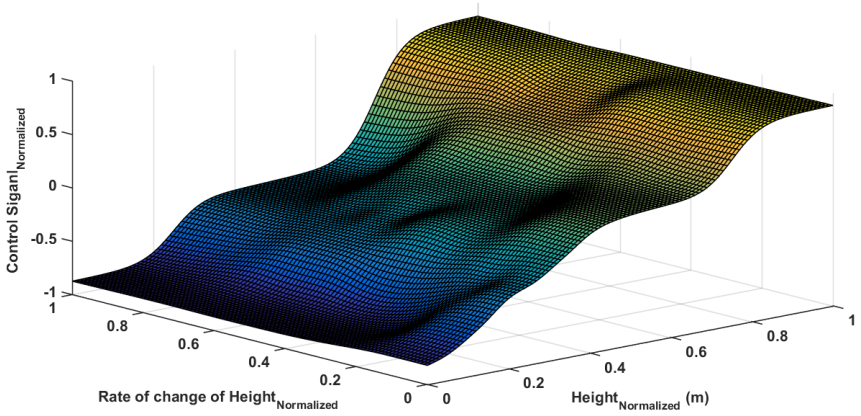

Figure 4.3: *The control surface $\hat{G}$.*

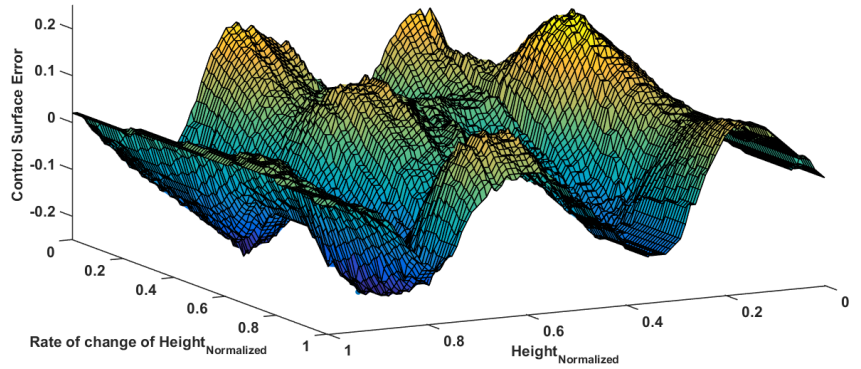

Figure 4.4: *The error surface.*

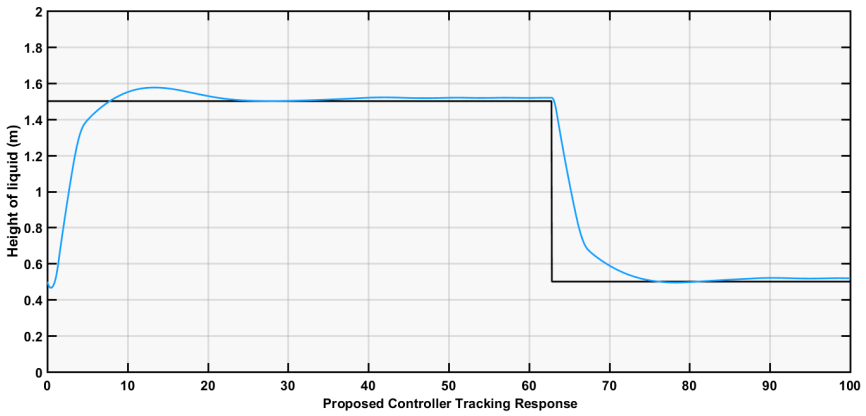

Figure 4.5: *The control response.*

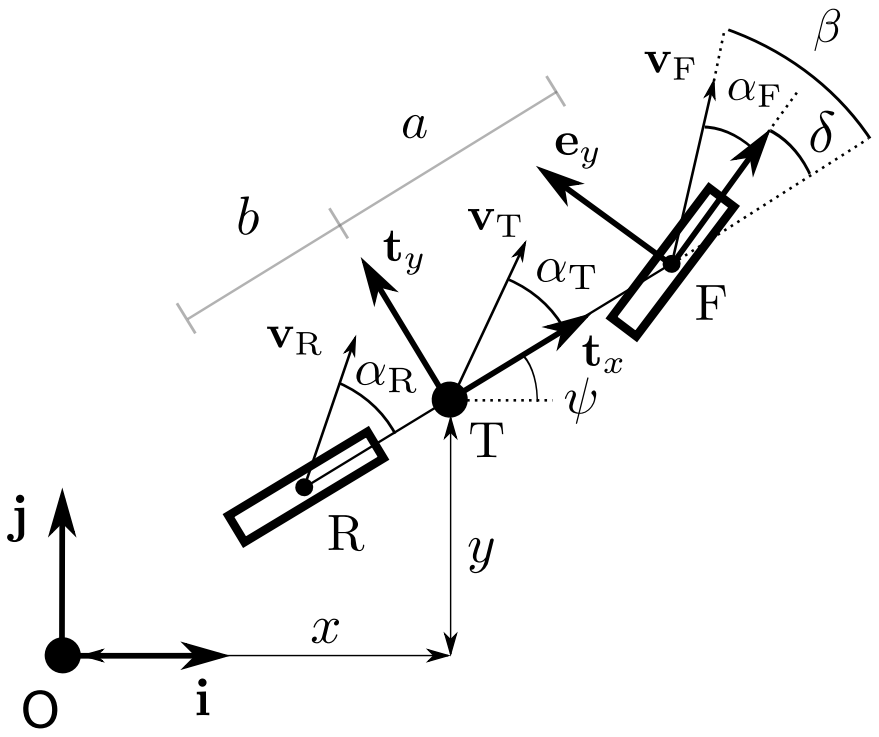

Figure 4.6: *The vehicle model.*

S.No	Parameter	Description
1	$mF0$	Mass on the front axle
2	$mR0$	Mass on the rear axle
3	mT	Mass of the vehicle
4	IT	Moment of inertia
5	T	Center of gravity
6	α_f, α_r	Front and Rear slip angles
7	Fy_f, Fy_r	Lateral forces at F and R
8	δ	Steering Angle
9	ψ	Yaw angle
10	α_T	Side slip angle
11	Fx_f, Fx_r	Longitudinal forces at F and R
12	F ,R	Front and Rear axle
13	a ,b	Distance from T to F and R

Table 4.1: *The vehicle model parameters.*

simulation model consists of a tire model and a vehicle model. The vehicle model in the state space representation is given as:

$$\dot{X} = F(x, u),$$

where

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} x \\ y \\ v_T \\ \alpha_T \\ \psi \\ \dot{\psi} \end{bmatrix}, \qquad u = \begin{bmatrix} \delta \\ Fx_R \\ Fy_R \\ Fx_f \\ Fy_f \end{bmatrix}, \tag{4.17}$$

where x is the longitudinal position, y is the lateral position, ψ is the yaw angle, $\dot{\psi}$ the yaw rate, α_T is the side slip angle and v_T is the velocity of the centre of gravity of the vehicle. u is the input to the dynamic model and δ is the steering angle. The state equations of the dynamic model are:

$$\begin{aligned}
\dot{x_1} &= x_3 \cos(x_4 + x_5), \\
\dot{x_2} &= x_3 \sin(x_4 + x_5), \\
\dot{x_5} &= x_6, \\
\dot{x_4} &= \frac{1}{m_T}(Fy_F \sin(x_4 - \delta) + Fy_R \sin(x_4), \\
\dot{x_5} &= \frac{1}{m_T x_3}(-Fy_F \cos(x_4 - \delta), \\
&\qquad + Fy_r \cos\alpha_T - m_T x_3 x_6), \\
\dot{x_6} &= \frac{1}{I_T}\left(Fy_F \cos\delta - Fy_r b\right).
\end{aligned}$$

The non-linear tire model for the simulations is derived by modeling the relationship between the vehicle lateral forces and the slip angle using the Pacejka empirical formula [92].

Control Scenario (Lateral Position Control)

In this scenario, we seek to modify the traveling path of a vehicle by changing its lateral position y using a data-driven fuzzy controller. First, a large training data set is generated using a working vehicle lateral position control system. The training data consists of samples points of steering control signal δ and the four inputs to the lateral position controller i.e. error 'e' in the lateral position y, the yaw angle ψ, velocity v_T and the side slip angle α_T as a feedback signal. The training data set is used to generate the estimated control surface G' in a higher dimension. Then the training data set is used as input for Algorithm 3 to identify the actual control surface $\hat{G}$. The tolerance level τ is set to $.07$ to achieve a compromise between the number of identified fuzzy rules and accuracy of control surface $\hat{G}$. As there are four

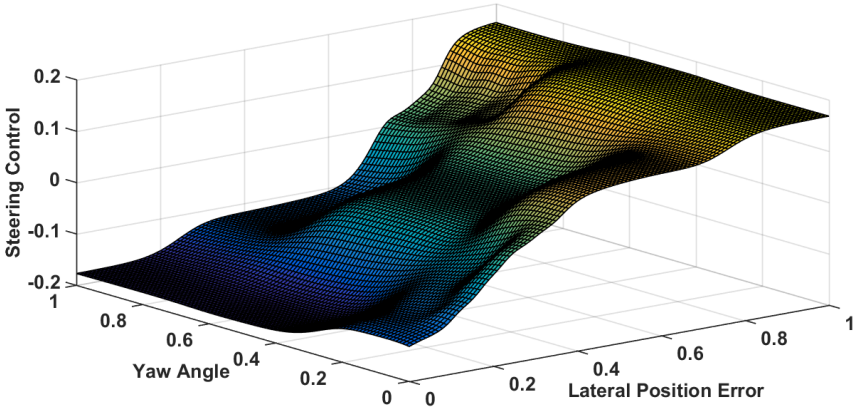

Figure 4.7: *The control Surface $\hat{G}$ for two inputs (Y and ψ).*

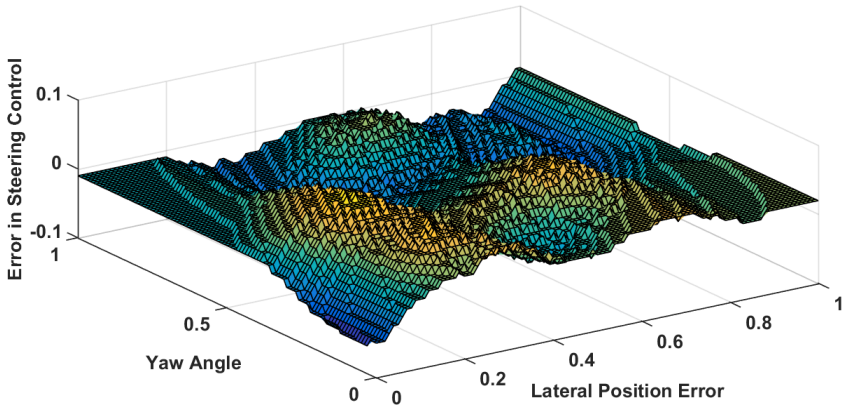

Figure 4.8: *The error surface for two inputs (Y and ψ).*

inputs in this case, the control surface $\hat{G}$ is identified in the fifth dimension using the concepts outlined in Section 4.1. Although it is not possible to visualize the control surface $\hat{G}$ in a higher (five) dimension, therefore the control surface $\hat{G}$ for only two inputs (error in the lateral position and yaw angle) in three dimension is shown in Fig. 4.7 (Similarly, the relation between the other two inputs and control surface can be also be visualized). The error surface E for the same two inputs (i.e. error in the lateral position and yaw angle) with the specified tolerance is shown in Fig. 4.8. The fuzzy rules identified by Algorithm 3 are used as a fuzzy controller to change the lateral position of the vehicle. The response of the data-driven fuzzy controller used to modify the course of the vehicle by adjusting its lateral position is shown in Fig. 4.9.

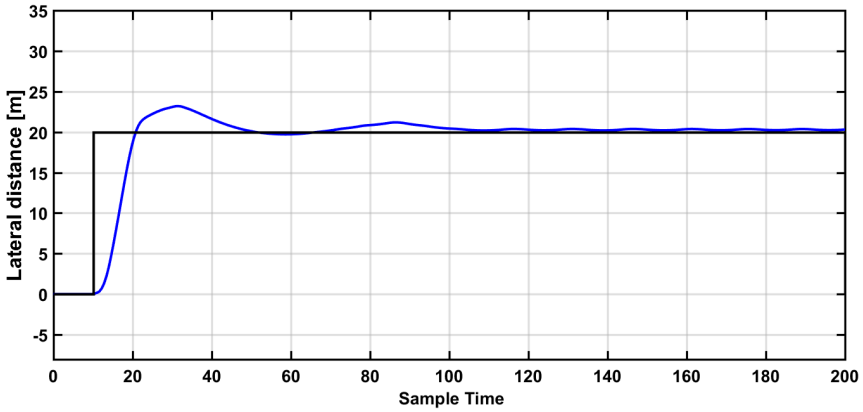

Figure 4.9: *The reference signal and actual lateral positions of the vehicle.*

4.3.3 Results discussion

It is worth noting that the magnitude of error surface is not negligible in both simulation cases. This is due to the fact that the level chosen for tolerance of the error surface is a bit higher (0.15 for water tank level controller and $.07$ for the vehicle lateral dynamics controller). A higher value was chosen intentionally to reduce the complexity of the fuzzy controller (a small number of fuzzy rules are identified). If a lower value of tolerance is chosen, then the accuracy of the design fuzzy controller increases but at the expense of a larger number of identified fuzzy rules and this results in a high complexity of the data-driven fuzzy controller. Fortunately, in most of the control scenarios, an identified control surface $\hat{G}$ using a high value of tolerance performs quite well, as shown in Fig. 4.5. So there is no need for a highly complex fuzzy controller (designed using a large number of fuzzy rules) at the expense of a negligible improvement in control performance.

4.4 Summary

A data-driven arithmetic-based FIS is designed. Distending functions are used instead of the classical membership functions. The DFs cover most of the input space using a few fuzzy rules. We calculate only one parameter of each DF. Then the complexity of the qualitative and quantitative parts of the data-driven fuzzy model is reduced. DFs in the input dimension are combined using the Dombi conjunctive operator to generate a control surface in higher dimension. Based on this, an algorithm is presented to derive a fuzzy rule base from the training data of the process. The derived rule base is then used to generate a FIS based on arithmetic operations. The effectiveness of the proposed approach is demonstrated using a data-driven control of a water tank system and vehicle lateral dynamics.

Chapter 5

Arithmetic-Based Interval Type2 Fuzzy System

In this chapter, we present a novel interval type2 FIS. An interval Type2 Distending Function (T2DF) is used. It is shown that this T2DF can be generated using various methods by adding uncertainty to its parameters. The T2DF can represent and handle different types of uncertainties. Interval type2 FIS is designed using the fuzzy arithmetic operations. The design process does not include the implication and the type reduction steps, hence it is computationally efficient. The expressions used are in closed form, and this makes it suitable for on-line implementation. The proposed design is simple, intuitive, computationally fast and handles uncertainties. The T2DF can also handle the uncertainties that are generated as a result of measurement noise. The efficiency of the proposed type2 FIS is demonstrated by designing an altitude controller for a quadcopter with a noisy feedback signal.

5.1 Introduction

The DF function consists of right hand side and left hand side functions (shown in Fig. 5.1) given by

$$\delta_L(x-c) = \frac{1}{1+\frac{1-\nu}{\nu}\left|\frac{x-c}{\varepsilon}\right|^{\lambda}\frac{1}{1+e^{(\lambda^*(x-c))}}}, \tag{5.1}$$

$$\delta_R(x-c) = \frac{1}{1+\frac{1-\nu}{\nu}\left|\frac{x-c}{\varepsilon}\right|^{\lambda}\frac{1}{1+e^{(-\lambda^*(x-c))}}}. \tag{5.2}$$

Here, λ^* is a free parameter and $\lambda^* >> \lambda$. These LHS and RHS functions can be combined using the Dombi conjunctive operator and it results in a DF like that shown in Fig. 5.1 and defined by Eq. (2.45).

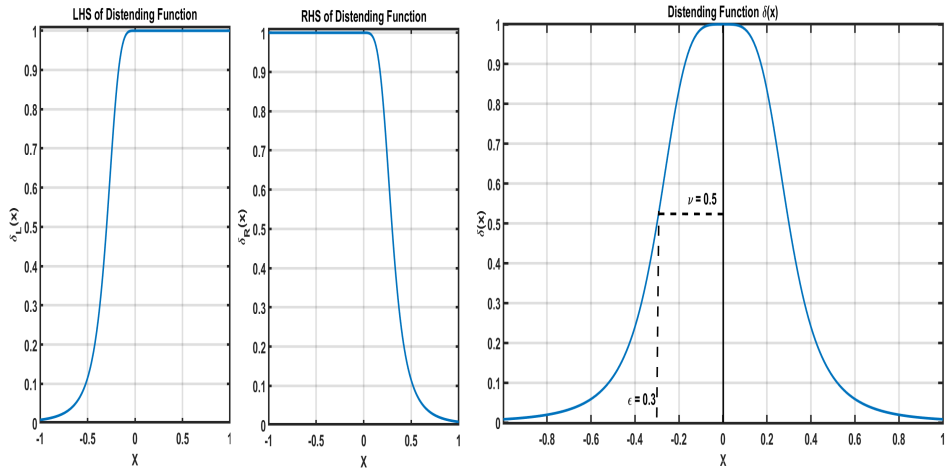

Figure 5.1: *LHS and RHS of DF (Left and Middle). Distending Function (Right).*

5.2 Uncertainty and Type-2 Distending Function (T2DF)

DF has four parameters i.e. ν, ε, λ and c. Uncertainty can appear in any of these parameters and as a result various DFs are produced. The DF with highest grade values is called the Upper Membership Function (UMF) and the one with the lowest values is called the Lower Membership Function (LMF)). The UMF, LMF and various DFs in between, can be combined to form an interval T2DF. Here we consider only the uncertainty in the c parameter. It will generate an interval T2DF with uncertain peak values.

5.2.1 Uncertainty in the peak values

If the peak value of DF becomes uncertain, then it can be represented using an interval T2DF having an uncertain 'c' value. Consider the right and left hand sides of the DF shown in Fig . 5.1. Now if the peak value becomes uncertain with a magnitude of Δ, then the this uncertainty can be added to the 'c' value of the LHS and RHS of DF as shown in Fig. 5.2. These LHS and RHS parts can be combined using the Dombi conjunctive operator. It results in a T2DF with uncertain peak values as shown in Fig. 5.3.

5.2.2 Uncertainty in boundary values

Uncertainty in the boundary values can be produced by adding the uncertainty in the ε parameter. It results in the generation of a set of various DFs. Among these DFs, the one with the highest grade value is called the Upper Membership Function (UMF) and the one with the lowest grade values is called the Lower Membership Function

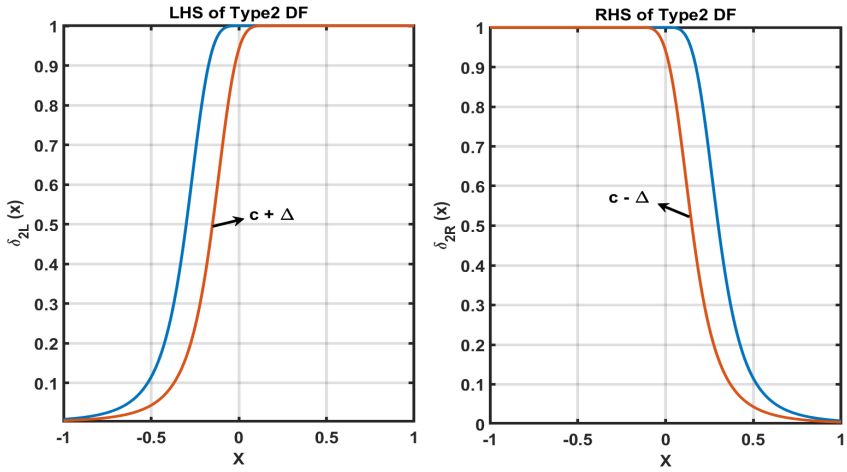

Figure 5.2: *LHS and RHS of uncertain peak values T2DF (Left and Middle). The Uncertain peak values T2DF (Right).*

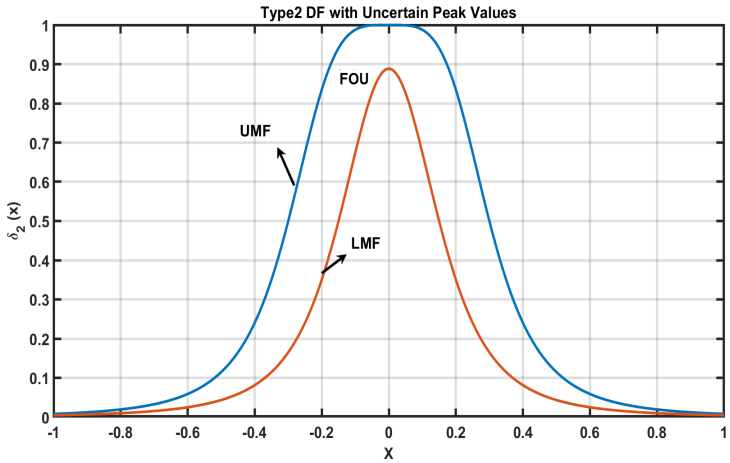

Figure 5.3: *The uncertain peak values T2DF.*

(LMF). The LMF, UMF and all the DFs in between can be combined to form a T2DF. Consider the LHS and RHS of the DF, as shown in Fig. 5.1. Adding uncertainty will generate the LHS and RHS of T2DF (see Fig. 5.4). Applying the Dombi conjunctive operator will result in a T2DF with uncertain boundary values, as shown in Fig. 5.5.

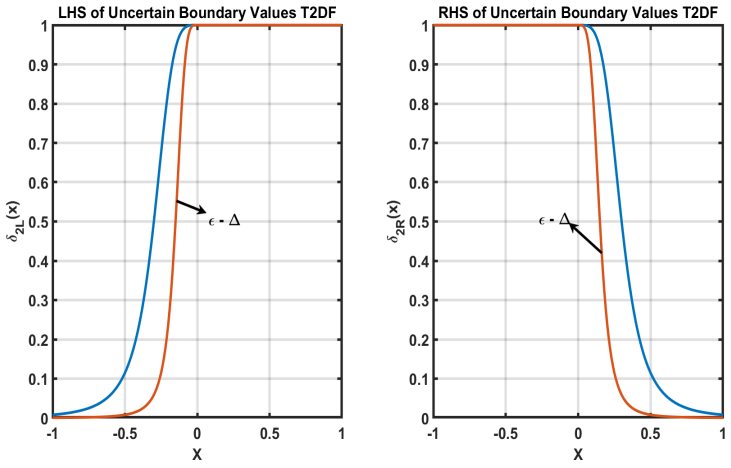

Figure 5.4: *LHS and RHS of uncertain boundary values T2DF.*

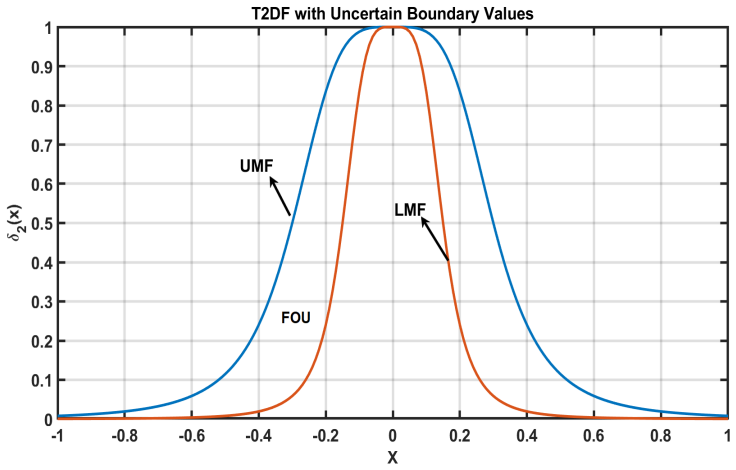

Figure 5.5: *The uncertain boundary values T2DF.*

5.2.3 Uncertainty in fuzziness measure

Uncertainty in the fuzziness measure can be obtained by adding the uncertainty in the λ parameter. It results in the generation of a set of various DFs and consequently the Upper Membership Function (UMF) and the Lower Membership Function (LMF)

are created. The LMF, UMF and all the DFs in between can be combined to form a T2DF. Consider the LHS and RHS of the DF, as shown in Fig. 5.1. Adding uncertainty will generate the LHS and RHS of the T2DF (see Fig. 5.6). Applying the Dombi conjunctive operator will result in a T2DF with uncertain fuzziness measure, as shown in Fig. 5.7.
Next, we show that the T2DF is closed under linear combination.

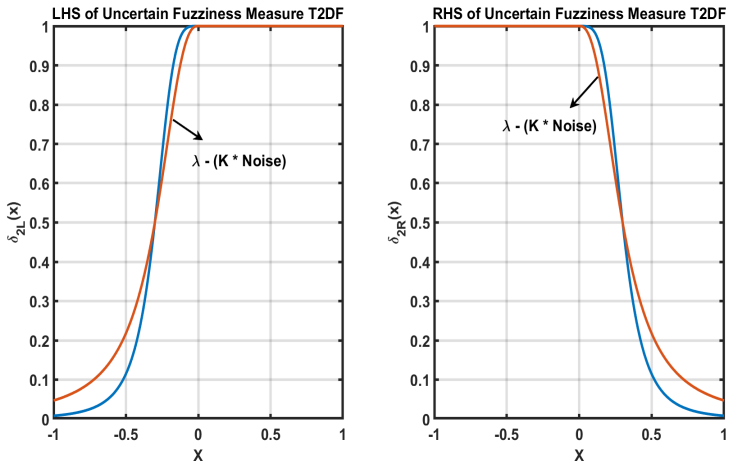

Figure 5.6: *LHS and RHS of uncertain fuzziness measure T2DF.*

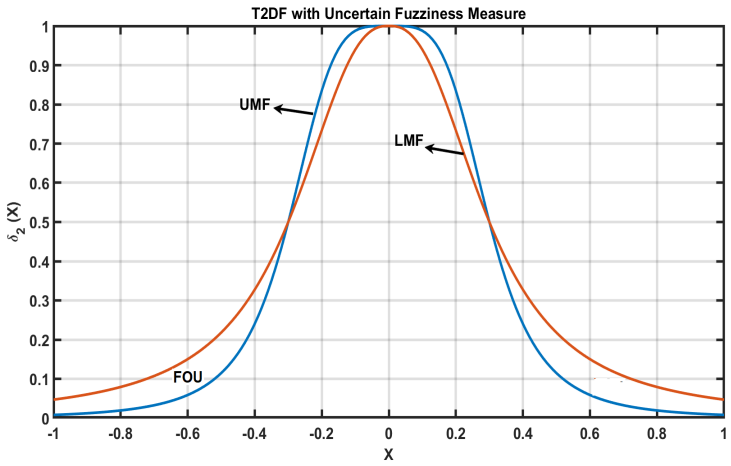

Figure 5.7: *The uncertain fuzziness measure T2DF.*

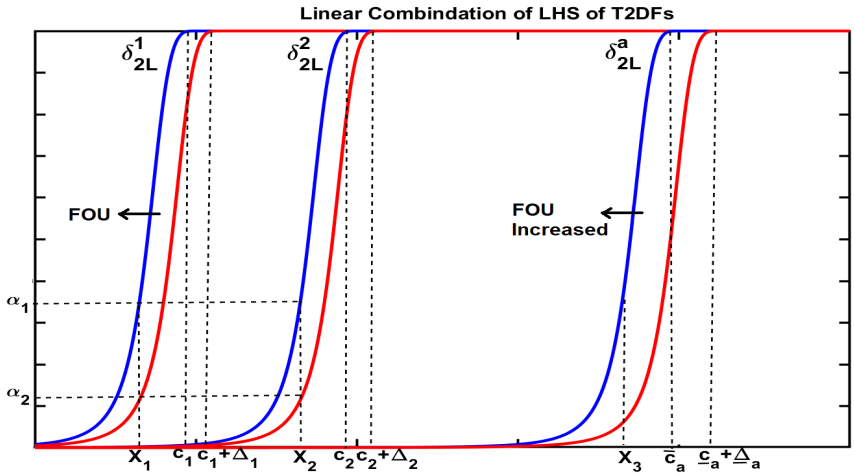

Figure 5.8: *Linear combination of LHS of the T2DFs.*

5.3 Linear Combination of T2DFs

Using fuzzy arithmetic operations, we show here that T2DFs are closed under linear combination (i.e. the linear combination of T2DFs is also a T2DF). Here, we examine the case of T2DFs with uncertain peak values. Consider n T2DFs $\delta_2^1, \delta_2^2, \ldots, \delta_2^n$ with the coordinates of the peak values $c_1, c_2, \ldots, c_n$, the uncertainties in the coordinates of peak values $\Delta_1, \Delta_2, \ldots, \Delta_n$ and tolerance values $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$, respectively.
Let us consider the LHS of two T2DFs i.e. δ_2^1 and δ_2^2 and two inputs x_1 and x_2 as shown in Fig. 5.8. c_1 and c_2 are the coordinates of the peak values of the UMF of T2DFs. Similarly $c_1 + \Delta_1$ and $c_2 + \Delta_2$ are the coordinates of peak value of LMFs. Δ_1, Δ_2 are the uncertainties in the coordinates of peak values of δ_2^1 and δ_2^2. Let α_1 and α_2 be the grades of UMFs and LMFs of LHS of δ_2^1 and δ_2^2 at the inputs x_1 and x_2, respectively. Then

$$\overline{\delta}_{2L}^1(x_1) = \frac{1}{1 + \frac{1-\nu}{\nu}\left|\frac{x_1 - c_1}{\varepsilon_1}\right|^{\lambda}} = \alpha_1, \tag{5.3}$$

$$\underline{\delta}_{2L}^1(x_1) = \frac{1}{1 + \frac{1-\nu}{\nu}\left|\frac{x_1 - (c_1+\Delta_1)}{\varepsilon_1}\right|^{\lambda}} = \alpha_2, \tag{5.4}$$

$$\overline{\delta}_{2L}^2(x_2) = \frac{1}{1 + \frac{1-\nu}{\nu}\left|\frac{x_2 - c_2}{\varepsilon_2}\right|^{\lambda}} = \alpha_1, \tag{5.5}$$

$$\underline{\delta}_{2L}^2(x_2) = \frac{1}{1 + \frac{1-\nu}{\nu}\left|\frac{x_2 - (c_2+\Delta_2)}{\varepsilon_2}\right|^{\lambda}} = \alpha_2. \tag{5.6}$$

Here we have neglected the term $\frac{1}{1+e^{(\lambda^*(x-c))}}$ in the LHS of the DF because we are only considering the interval $-\infty < x < c$ in which this term has a negligible influence. From Eq. (5.3) and Eq. (5.5), x_1 and x_2 are given by

$$\overline{w}_1 x_1 = \overline{w}_1 \left(\left(\frac{\nu}{1-\nu} \frac{1-\alpha_1}{\alpha_1} \right)^{\frac{1}{\lambda}} \varepsilon_1 + c_1 \right), \tag{5.7}$$

$$\overline{w}_2 x_2 = \overline{w}_2 \left(\left(\frac{\nu}{1-\nu} \frac{1-\alpha_1}{\alpha_1} \right)^{\frac{1}{\lambda}} \varepsilon_2 + c_2 \right). \tag{5.8}$$

Here $\overline{w}_1$ and $\overline{w}_2$ are the weights. Now let $\overline{w}_1 x_1 + \overline{w}_2 x_2 = x$. Adding Eq. (5.7) and Eq. (5.8), we get

$$x = \left(\frac{\nu}{1-\nu} \frac{1-\alpha_1}{\alpha_1} \right)^{\frac{1}{\lambda}} \left((\overline{w}_1 \varepsilon_1 + \overline{w}_2 \varepsilon_2) + (\overline{w}_1 c_1 + \overline{w}_2 c_2) \right). \tag{5.9}$$

From this, we can write

$$\overline{\delta}^a_{2L}(x) = \frac{1}{1 + \frac{1-\nu}{\nu} \left| \frac{x - (\overline{w}_1 c_1 + \overline{w}_2 c_2)}{\overline{w}_1 \varepsilon_1 + \overline{w}_2 \varepsilon_2} \right|^{\lambda}} = \alpha_1. \tag{5.10}$$

Here $\overline{\delta}^a_{2L}(x)$ is the UMF of the LHS of the aggregated T2DF i.e. $\delta^a_{2L}(x)$. The result can be generalized for n inputs $x_1, x_2, \ldots, x_n$ and n T2DFs $\delta^1_2, \delta^2_2, \ldots, \delta^n_2$

$$\overline{\delta}^a_{2L}(x) = \frac{1}{1 + \frac{1-\nu}{\nu} \left| \frac{x - \sum_1^n \overline{w}_i c_i}{\sum_1^n \overline{w}_i \varepsilon_i} \right|^{\lambda}}. \tag{5.11}$$

By following a similar procedure, the LMF of the LHS of the aggregated T2DF i.e. $\delta^a_{2L}(x)$ can be calculated and it is given as

$$\underline{\delta}^a_{2L}(x) = \frac{1}{1 + \frac{1-\nu}{\nu} \left| \frac{x - \left(\sum_1^n \underline{w}_i c_i + \sum_1^n \underline{w}_i \Delta_i \right)}{\sum_1^n \underline{w}_i \varepsilon_i} \right|^{\lambda}}, \tag{5.12}$$

where w_i is the weight and Δ_i is the uncertainty in the c value of the LHS of the ith DF.

In the same way, the following equations can be derived for the UMF and LMF of the

RHS of the aggregated T2DF (i.e $\delta^a_{2R}(x)$):

$$\overline{\delta}^a_{2R}(x) = \frac{1}{1+\frac{1-\nu}{\nu}\left|\frac{x-\sum_1^n \overline{w}_i c_i}{\sum_1^n \overline{w}_i \varepsilon_i}\right|^{\lambda}}, \tag{5.13}$$

$$\underline{\delta}^a_{2R}(x) = \frac{1}{1+\frac{1-\nu}{\nu}\left|\frac{x-\left(\sum_1^n \underline{w}_i c_i - \sum_1^n \underline{w}_i \Delta_i\right)}{\sum_1^n \underline{w}_i \varepsilon_i}\right|^{\lambda}}. \tag{5.14}$$

Let

$$\begin{aligned} \underline{c}_a &= \sum_{i=1}^n \underline{w}_i \underline{c}_i, \quad \underline{\varepsilon}_a = \sum_{i=1}^n \underline{w}_i \underline{\varepsilon}_i, \quad \underline{\Delta}_a = \sum_{i=1}^n \underline{w}_i \underline{\Delta}_i, \\ \overline{c}_a &= \sum_{i=1}^n \overline{w}_i \overline{c}_i, \quad \overline{\varepsilon}_a = \sum_{i=1}^n \overline{w}_i \overline{\varepsilon}_i. \end{aligned} \tag{5.15}$$

where $\underline{w}_i$ and $\overline{w}_i$ are the weights of the LMF and UMF of the ith T2DF, respectively. It can be shown that the LHS and RHS of the aggregated T2DF (i.e. $\delta^a_{2L}(x),\ \delta^a_{2R}(x)$) can be combined by applying the Dombi conjunctive operator. As a result, we get the UMF and LMF of the aggregated T2DF.
Applying the Dombi conjunctive operator on Eq. (5.11) and Eq. (5.13) results in

$$\overline{\delta}^a_{2}(x) = \frac{1}{1+\frac{1-\nu}{\nu}\left|\frac{x-\overline{c}_a}{\overline{\varepsilon}_a}\right|^{\lambda}}, \tag{5.16}$$

$$\underline{\delta}^a_{2}(x) = \frac{1}{1+\frac{1-\nu}{\nu}\left|\frac{x-(\underline{c}_a+\underline{\Delta}_a)}{\underline{\varepsilon}_a}\right|^{\lambda}+\left|\frac{x-(\underline{c}_a-\underline{\Delta}_a)}{\underline{\varepsilon}_a}\right|^{\lambda}}. \tag{5.17}$$

$$\overline{\delta}^a_{2}(x) = \frac{1}{1+\left(\frac{1-\nu}{\nu}\right)\left|\frac{x-(\overline{c}_a)}{\overline{\varepsilon}_a}\right|^{\lambda}}, \tag{5.18}$$

$$\underline{\delta}^a_{2}(x) = \frac{1}{1+\left(\frac{1-\nu}{\nu}\right)\left|\frac{x-\left(\underline{c}_a+\sum_1^n \underline{w}_i\Delta_i\right)}{(\underline{\varepsilon}_a)}\right|^{\lambda}+\left|\frac{x-\left(\underline{c}_a-\sum_1^n \underline{w}_i\Delta_i\right)}{(\underline{\varepsilon}_a)}\right|^{\lambda}}. \tag{5.19}$$

Here $\overline{\delta}^a_2(x)$ is the UMF and $\underline{\delta}^a_2(x)$ is the LMF of the aggregated membership function. As $\overline{\delta}^a_2(x)$ and $\underline{\delta}^a_2(x)$ are T2DFs, the linear combination of n T2DFs is also a T2DF.

Remark. *1. The UMF and LMF of the aggregated T2DF that has uncertain boundary*

values are

$$\overline{\delta}_2^a(x) = \frac{1}{1 + \left(\frac{1-\nu}{\nu}\right)\left|\frac{x-(\overline{c}_a)}{\overline{\varepsilon}_a}\right|^{\lambda}}. \tag{5.20}$$

and

$$\underline{\delta}_2^a(x) = \frac{1}{1 + 2\left(\frac{1-\nu}{\nu}\right)\left|\frac{x-\left(\underline{c}_a+\Delta\sum_1^n \underline{w}_i\right)}{\left(\underline{\varepsilon}_a-\sum_1^n \underline{w}_i\Delta_i\right)}\right|^{\lambda}} \tag{5.21}$$

2. *The UMF and LMF of the aggregated T2DF having an uncertain fuzziness measure are*

$$\overline{\delta}_2^a(x) = \frac{1}{1 + \left(\frac{1-\nu}{\nu}\right)\left|\frac{x-(\overline{c}_a)}{\overline{\varepsilon}_a}\right|^{\lambda}} \tag{5.22}$$

and

$$\underline{\delta}_2^a(x) = \frac{1}{1 + \left(\frac{1-\nu}{\nu}\right)\left(\sum_1^n \left|\frac{x-\left(\underline{c}_a+\sum_1^n \underline{w}_i\Delta_i\right)}{(\underline{\varepsilon}_a)}\right|^{\lambda-\Delta_i}\right)}. \tag{5.23}$$

5.4 Interval Type2 Fuzzy Sytem

Our design methodology is build on our previous work (Chapter 3). A multi-input single output (MISO) system is described by the following rules

$$\text{If } x_1 \text{ is } A_1^i \text{ and } ... \text{ and } x_n \text{ is } A_n^i \text{ then } y \text{ is } B^i \text{ ,} \tag{5.24}$$

where x_i is the ith input linguistic variable, A_i is the ith input fuzzy subset, B_i is the ith output fuzzy subset and y is the output of the system. Here, $i = 1, ..., l$ is the number of fuzzy rules. The part of the fuzzy rule before *then* is called the antecedent part and the part after it is called the consequent part. Uncertainty in the system is handled (modeled and minimized) by defining interval type-2 fuzzy sets for $A_1, A_2, ..., A_n$ and B. These type-2 sets will be represented using T2DFs. The inference mechanism will minimize this uncertainty and map the input-output space to generate a crisp output. In our approach, the fuzzy rules are evaluated by handling the antecedent and consequent parts separately.

5.4.1 The antecedent part

The antecedent part of the ith fuzzy rule is

$$\mathcal{L}(\delta_1(x_1)^i, \delta_2(x_2)^i, \ldots, \delta_n(x_n)^i) = \hat{w}_i(x), \tag{5.25}$$

where $\mathcal{L}$ is the fuzzy logical expression and $\hat{w}_i(x)$ is the rule applicability interval. $\hat{w}_i(x)$ contains two values corresponding to the upper and lower membership grades in T2DFs. For a specific input values $\underline{x}^*$, Eq. (5.25) can be evaluated using GDO 2.39 and this results in an interval $[\underline{\hat{w}}_i(\underline{x}^*) \;\; \overline{\hat{w}}_i(\underline{x}^*)]$

$$\mathcal{L}(\underline{\delta}^i_{1(x_1^*)}, \underline{\delta}^i_{2(x_2^*)}, \ldots, \underline{\delta}^i_{n(x_n^*)}) = \underline{\hat{w}}_i(\underline{x}^*),$$
$$\mathcal{L}(\overline{\delta}^i_{1(x_1^*)}, \overline{\delta}^i_{2(x_2^*)}, \ldots, \overline{\delta}^i_{n(x_n^*)}) = \overline{\hat{w}}_i(\underline{x}^*),$$

where $\underline{\delta}^i_n(x)$ is the LMF of the nth T2DF and $\overline{\delta}^i_{n(x_n)}$ is the UMF of the nth T2DF. $\underline{\hat{w}}_i(\underline{x}^*)$ is called the upper strength and $\overline{\hat{w}}_i(\underline{x}^*)$ is called the lower strength of the ith rule. We normalize these strengths (to compare the rules) to get the upper firing strengths $\overline{w}_i(x^*)$ and lower firing strengths $\underline{w}_i(x^*)$. The firing strengths give the probability of the rule. The lower and upper firing strengths of the ith rule are

$$\underline{w}_i(\underline{x}^*) = \frac{\underline{\hat{w}}_i(\underline{x}^*)}{\sum_{i=1}^{l} \underline{\hat{w}}_i(\underline{x}^*)}, \qquad \overline{w}_i(\underline{x}^*) = \frac{\overline{\hat{w}}_i(\underline{x}^*)}{\sum_{i=1}^{l} \overline{\hat{w}}_i(\underline{x}^*)}, \tag{5.26}$$

$$\text{where} \qquad \sum_{i=1}^{l} \underline{w}_i(\underline{x}^*) = 1, \qquad \sum_{i=1}^{l} \overline{w}_i(\underline{x}^*) = 1.$$

5.4.2 The consequent part

This part of the rule is a type-2 fuzzy set represented by a single T2DF. The upper and lower firing strengths of each rule (calculated from the antecedent part) are multiplied by the UMF and LMF of the consequent T2DF. As a result the fuzzy output obtained from each rule evaluation is also a T2DF. By combining all the rules, we can generate an LMF and UMF of the aggregated T2DF.
If $\underline{w}_1(\underline{x}^*), \underline{w}_2(\underline{x}^*), \ldots, \underline{w}_l(\underline{x}^*)$ are the lower firing strengths and $\underline{\delta}_{1o}(x), \underline{\delta}_{2o}(x), \ldots, \underline{\delta}_{lo}(x)$ are the l LMFs of the type-2 consequents, then the LMF of the aggregated output $(\underline{\delta}_a(x))$ of the l fuzzy rule is given by Eq. (5.19), where

$$\underline{c}_a = \sum_{i=1}^{l} \underline{w}_i(\underline{x}^*)\underline{c}_i, \qquad \underline{\varepsilon}_a = \sum_{i=1}^{l} \underline{w}_i(\underline{x}^*)\underline{\varepsilon}_i. \tag{5.27}$$

$\underline{c}_i$ and $\underline{\varepsilon}_i$ are the parameters of the LMF of the ith consequent T2DF. Similarly, the UMF of the aggregated output T2DF $(\overline{\delta}_a(x))$ has the following form given by Eq.

(5.18), where

$$\overline{c}_a = \sum_{i=1}^{l} \overline{w}_i(\overline{x}^*)\overline{c}_i, \qquad \overline{\varepsilon}_a = \sum_{i=1}^{l} \overline{w}_i(\overline{x}^*)\overline{\varepsilon}_i. \tag{5.28}$$

Here $\overline{c}_i$ and $\overline{\varepsilon}_i$ are the parameters of the UMF of the ith T2DF consequent. Now a crisp output can be generated as

$$c_{crisp} = \frac{\underline{c}_a + \overline{c}_a}{2}. \tag{5.29}$$

5.4.3 Algorithm

The whole procedure is summarized in Algorithm 4.

Algorithm 4 Algorithm for Type-2 FIS using the T2DF

Step 1: Transform the inputs into the $[0,1]$ interval.
Step 2: Define the T2DFs for the input and output linguistic variables.
Step 3: Fuzzify the crisp inputs using Eq. (2.45).
Step 4: Construct the rule base from the knowledge base using Eq. (5.24).
Step 5: Calculate the upper and lower strengths of each rule using Eq. (5.25) by choosing the appropriate fuzzy conjunctive/disjunctive operators.
Step 6: Calculate the l upper and l lower firing strengths using Eq. (5.26).
Step 7: Calculate the parameters $(\overline{c}_a, \overline{\varepsilon}_a, \underline{c}_a, \underline{\varepsilon}_a)$ of the aggregated output T2DF by using Eq. (5.27) and Eq. (5.28).
Step 8: Generate the UMF of the aggregated output T2DF using Eq. (5.18) and LMF of the aggregated output T2DF using Eq. (5.19).
Step 9: Get the crisp output signal u using Eq. (5.29).

5.5 Experiments, Results and Discussion

The effectiveness of the proposed type2 FIS will be shown by designing an altitude control system for a quadcopter (Parrot mini-drone).

5.5.1 Parrot Mini-Drone Mambo

The flight simulation model of the Mambo quadcopter is available in Matlab Simulink [57]. The model consists of: 1) An environment model; 2) An airframe model; 3) Sensors; 4) A flight control system. The airframe model describes the six degree of freedom (6DOF) dynamical model of the quadcopter structure (shown in Fig. 5.9) . It is characterised by the angular speed ω, motor torques τ, upward forces

f, three translational components (x, y, z) and three rotational components (ϕ, θ, ψ). The system model is described by

$$\dot{X} = f(X, U) + W,$$

where

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} = \begin{bmatrix} \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \\ x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \end{bmatrix}; \quad U = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ \Omega_r \end{bmatrix} = \begin{bmatrix} b\left(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2\right) \\ b\left(-\omega_2^2 + \omega_4^2\right) \\ b\left(\omega_1^2 - \omega_3^2\right) \\ d\left(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2\right) \\ -\omega_1 + \omega_2 - \omega_3 + \omega_4 \end{bmatrix}; \quad W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix}.$$

Here X is the state vector consisting of three translational and three rotational components, W is the additive noise affecting all these states of the quadcopter and U is the input to the system. U_1 is the total thrust and it governs the altitude z of the quadcopter. U_2, U_3, U_4 control the roll, pitch and yaw rotations and ω_r represents the overall residual angular speed. The state equations of the system are:

$$\dot{X} = \begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \\ \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \\ \dot{z} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \phi \\ \dot{\theta}\dot{\psi}\frac{I_{yy}-I_{zz}}{I_{xx}} + \dot{\theta}\frac{J_r}{I_{xx}}\Omega_r + \frac{L_a}{I_{xx}}U_2 \\ \dot{\theta} \\ \dot{\phi}\dot{\psi}\frac{I_{zz}-I_{xx}}{I_{yy}} - \dot{\phi}\frac{J_r}{I_{yy}}\Omega_r + \frac{L_a}{I_{yy}}U_3 \\ \dot{\psi} \\ \dot{\theta}\dot{\phi}\frac{I_{xx}-I_{yy}}{I_{zz}} + \frac{1}{I_{zz}}U_4 \\ \dot{x} \\ (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)\frac{U_1}{m} \\ \dot{y} \\ (\cos\phi\sin\theta\sin\psi + \sin\phi\cos\psi)\frac{U_1}{m} \\ \dot{z} \\ g - (\cos\phi\cos\theta)\frac{U_1}{m} \end{bmatrix} + \begin{bmatrix} 0 \\ w_1 \\ 0 \\ w_2 \\ 0 \\ w_3 \\ 0 \\ w_4 \\ 0 \\ w_5 \\ 0 \\ w_6 \end{bmatrix}.$$

Here, we have designed a type2 fuzzy controller which regulates the altitude z of the quadcopter by generating an appropriate total thrust U_1.

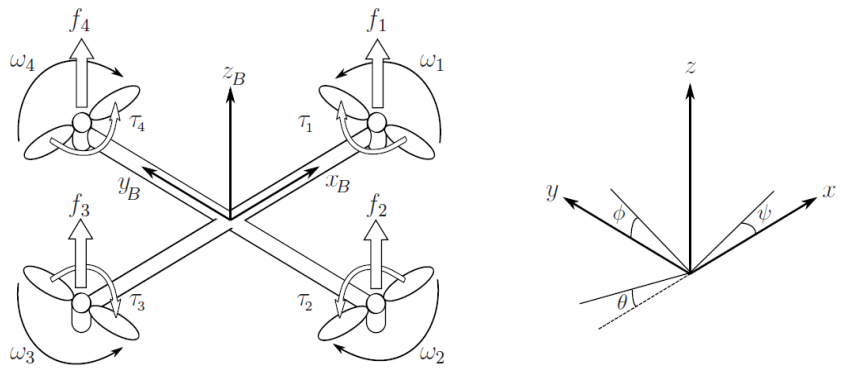

Figure 5.9: *The body and inertial frames of the Quadcopter structure [76].*

5.5.2 Control scenario: Altitude Control

A type2 fuzzy controller based on the proposed technique is designed to control the altitude of the quadcopter in a situation where quadcopter initiates the takeoff operation and reaches an altitude of 1m. It then increases its altitude to 2m and maintains this altitude for some time. The quadcopter then returns to an altitude of 1m. The type2 controller generates the required total thrust U_1 for all these events. The altitude z and rate of change of altitude $\dot{z}$ are the two inputs of the type2 controller. The T2DFs of z input and control output U_1 are shown in Fig. 5.13. The fuzzy rules of the controller are shown in Table 5.1. The upper and lower control surfaces of the controller are shown in Fig. 5.10. The controller output surface, used to control the quadcopter, is shown in Fig. 5.11. The reference signal commands the quadcopter to increase the altitude to 2m at $10th$ second and return to an altitude of 1m at $20th$ second. The reference signal and the altitude of the quadcopter have been plotted in Fig. 5.12. For comparison purposes, the altitude of the quadcopter is also controlled using a tuned PD controller. The response of the PD controller is also shown in Fig. 5.12.

Altitude Rate / Altitude Error	PL	NL	Don't Care
PL	-	-	PL
NL	-	-	NL
OK	P	N	NC

Table 5.1: *The rule base for the Quadcopter Altitude controller. PL (Positive Large), NL (Negative Large), NC (No Change)*

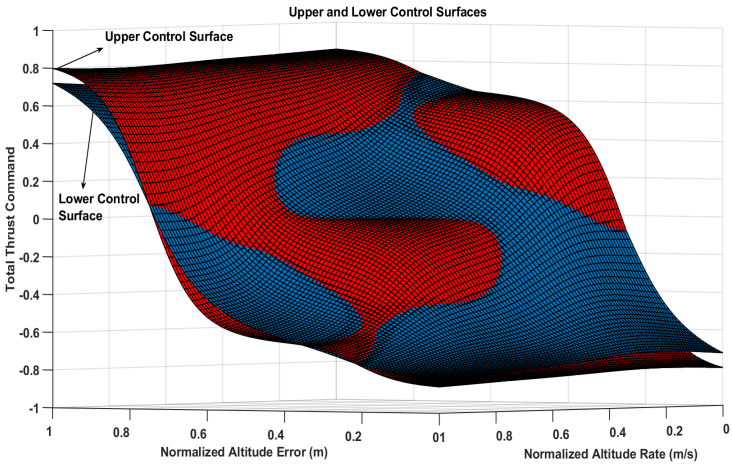

Figure 5.10: *The upper (red) and lower (blue) control surfaces.*

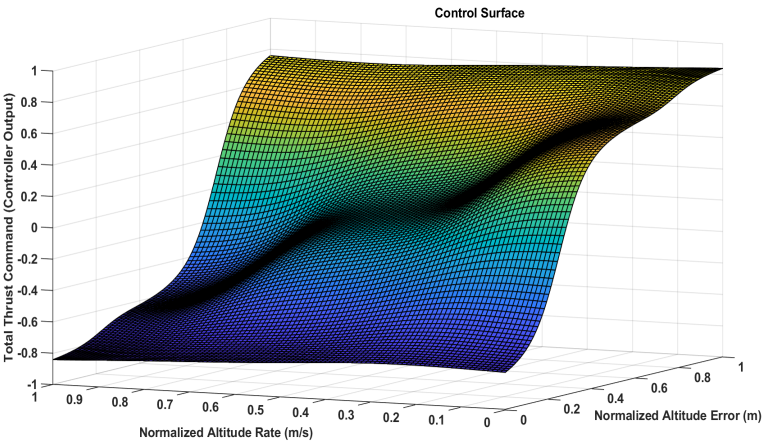

Figure 5.11: *The control surface used to generate controller output.*

5.5.3 Altitude control in the presence of measurement noise

Now we will consider a situation where the altitude measured by the sonar is corrupted by an additive white measurement noise. The quadcopter is ordered to take off and reach an altitude of 1m, maintain this height for 20 seconds and then increase its altitude to 2m. A fuzzy controller based on T2DFs was designed to generate the required thrust signal u_1 during this entire operation. The altitude error and rate of change of altitude are the two inputs of the controller and u_1 is the output. The controller knowledge base is shown in Table 5.2, and the T2DFs of the antecedent and consequent are shown in Fig. 5.17. The upper and lower control surfaces generated using the T2DFs are shown in Fig. 5.14. The control surface used to generate the thrust u_1 is shown in Fig. 5.15. The altitude of the quadcopter and the command

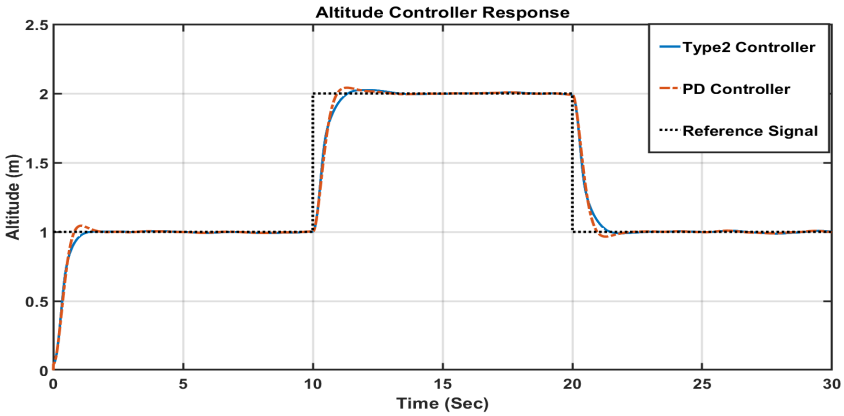

Figure 5.12: *Altitude Response of the proposed Type2 controller and PD controller.*

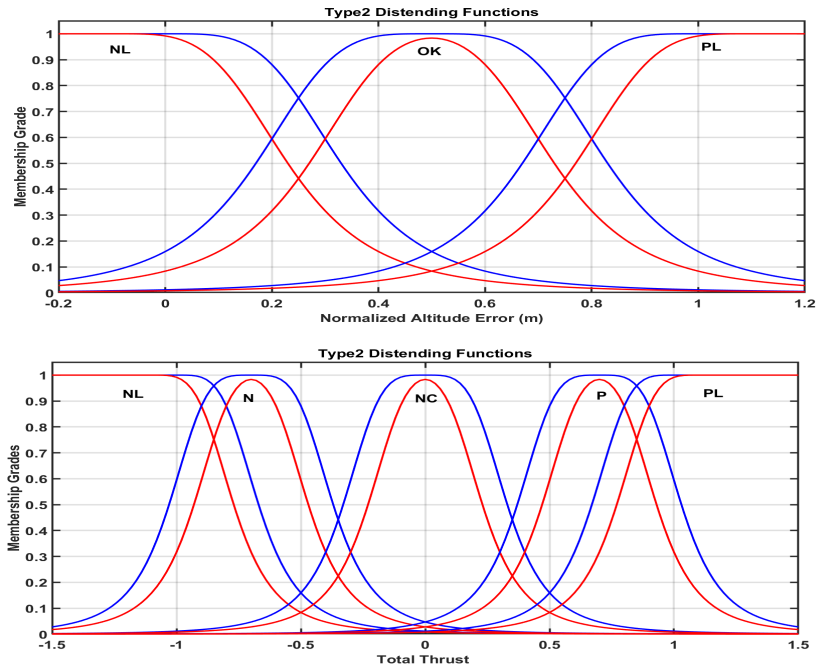

Figure 5.13: *T2DFs for the Altitude Error and Total Thrust.*

signal sent during the simulation have been plotted in Fig. 5.12. For comparison purposes, the altitude response generated using a Mamdani (type-1) fuzzy controller has also been plotted in Fig. 5.16. The top part of Fig. 5.16 shows the altitude response of both controllers in the absence of external noise. The bottom part of Fig. 5.16 shows the response in the presence of a large measurement noise (15 dB SNR).

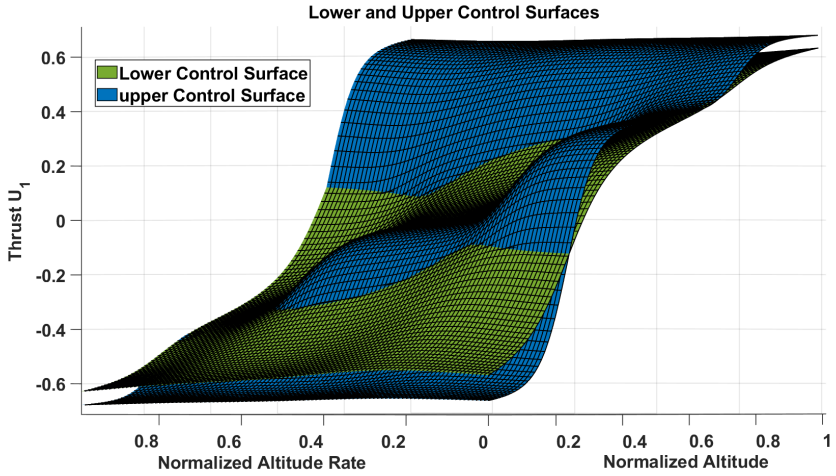

Figure 5.14: *The upper (blue) and lower (green) control surfaces.*

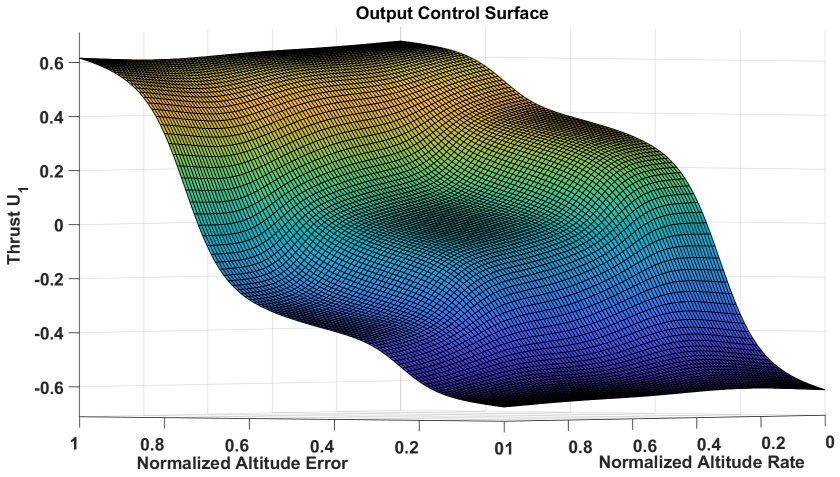

Figure 5.15: *The control surface used to generate the controller output.*

Rule No.	Altitude Error	Rate of change of Altitude	Thrust
1	Positive	-	Positive
2	Negative	-	Negative
3	Nominal	-	Minor
4	Nominal	Positive	Negative Small
5	Nominal	Negative	Positive Small

Table 5.2: *The rule base for the T2DF-based controller.*

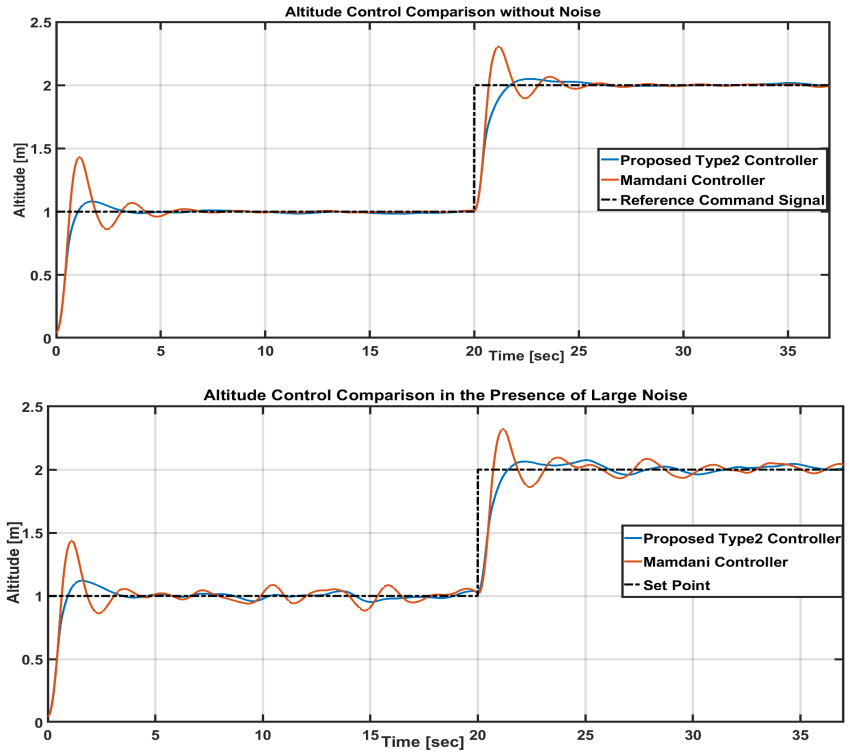

Figure 5.16: *An altitude control comparison of the proposed Type2 and Mamdani controllers. a) Without noise (top). b) With large measurement noise (bottom).*

5.5.4 Discussion

T2DFs with uncertain peak values are used in these simulation studies to generate a control signal for the altitude control. The appropriate values of $\lambda, \varepsilon, \nu$, and Δ have been selected and kept fixed for the T2DFs of the antecedents and consequents. With a few rules, a very smooth control surface is generated. A few closed form expressions are used (mentioned in Algorithm 4), and the computation cost is very low compared to the conventional iterative procedures for interval type2 fuzzy controllers. In addition to handling the uncertainty, the result is much better than the PD controller, as shown in Fig. 5.12. So the proposed design is simple, computationally fast, produces better results and it can handle uncertainties by using T2DFs.
In the absence of external noise, the response of the proposed controller is comparable with the conventional Mamdani controller. However in the presence of large measurement noise (15 dB SNR), the Mamdani controller produces small oscillations in the quadcopter altitude. In contrast, the proposed controller is more robust to this measurement noise and it produces a smooth altitude response compared to that for the the Mamdani controller. Therefore the proposed T2DF-based controller is better able to overcome the problem of external noise signals.

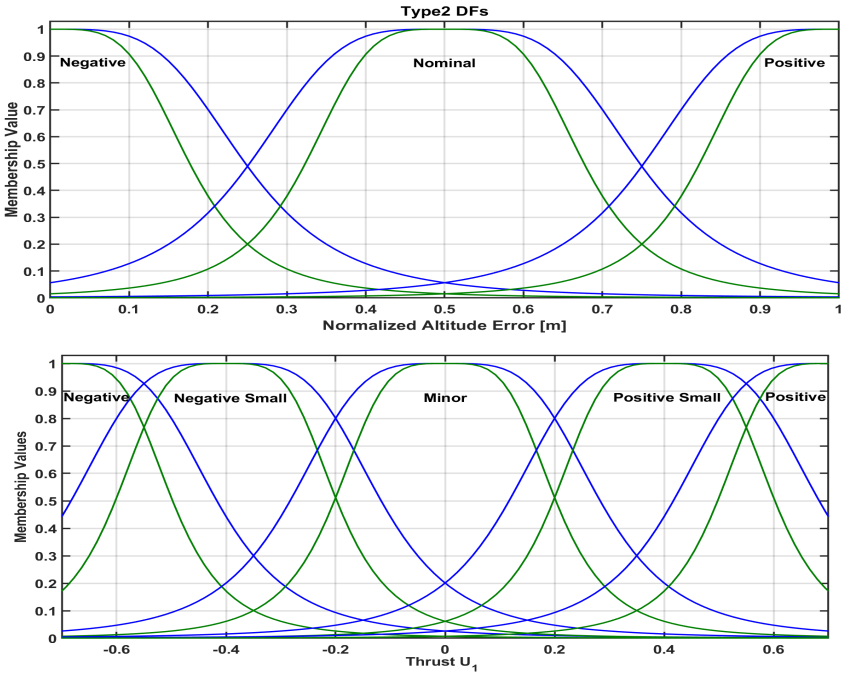

Figure 5.17: *T2DFs for Altitude Error and Thrust* u_1.

5.6 Conclusion

A new interval type-2 fuzzy FIS is proposed. This FIS is based on new type-2 membership function i.e. T2DF. The T2DF can be generated in a variety of ways by adding uncertainty to its parameters. Using T2DFs, uncertainties and noises can be handled easily. The proposed inference mechanism is based on the fuzzy arithmetic and uncertainty calculations. There are no type reduction, implication and aggregation steps. Therefore, the computational complexity is low and the proposed FIS can be implemented online. The limitation of the design is the assumption that the experts knowledge is available in the form of if-else rules which is not true in some cases. In such cases, the data-driven design of the proposed FIS will be presented in the next chapter.

Chapter 6

Data-Driven Arithmetic-Based Interval Type2 Fuzzy System

Fuzzy type2 modeling techniques are increasingly being used to model uncertain dynamical systems. However, some challenges arise when applying the existing techniques. A large number of rules are required to completely cover the whole input space. A large of parameters associated with type2 membership functions have to be determined and this leads to increased computation time and resources. The identified fuzzy model is usually difficult to interpret due to the large number of rules. Designing a fuzzy type2 controller using these models is also a computationally expensive task. To overcome these limitations, here in this chapter we propose a procedure to identify the fuzzy type2 model directly from the data. This model is called the Distending Function-based Fuzzy Inference System (DFIS). This model consists of rules and Type2 Distending Function (T2DF). First, a few key rules are identified from the data and later more rules are added until the error is less than the threshold. The proposed procedure is used to model the altitude controller of a quadcopter. The T2DF model performance is compared with various fuzzy models. Also, a simplified procedure based on the rules is presented to design a computationally low-cost interval type2 controller. The effectiveness of the controller is shown by regulating the height of a quadcopter in the presence of noisy sensory data. The performance of this controller is compared with type1 and type2 fuzzy controllers. Lastly, the proposed type2 controller was implemented on a Parrot Mambo quadcopter to demonstrate its real-time performance.
In the next couple of sections we will describe procedures for combining T2DFs and constructing T2DF in higher dimension.

6.1 Combining Interval T2DFs

Various T2DFs belonging to the same fuzzy variable can be combined to form a single T2DF. The support of the resultant T2DF will be approximately the same as the combined support of the individual T2DFs. The UMF of the T2DF consists of the LHS and RHS (the same is true for the LMF). The LHS and RHS are given by [35]:

$$\bar{\delta}_L^2(x-c) = \frac{1}{1+\frac{1-\nu}{\nu}\left|\frac{x-c}{\varepsilon}\right|^{\lambda}\frac{1}{1+e^{(\lambda^*(x-c))}}}, \tag{6.1}$$

$$\bar{\delta}_R^2(x-c) = \frac{1}{1+\frac{1-\nu}{\nu}\left|\frac{x-c}{\varepsilon}\right|^{\lambda}\frac{1}{1+e^{(-\lambda^*(x-c))}}}. \tag{6.2}$$

The LHS and RHS of the UMF and LMF can be combined using the Dombi conjunctive operator to get a single T2DF. Consider two T2DFs δ_1^2 and δ_2^2. The LHS of δ_1^2 and RHS of δ_2^2 can be combined using the Dombi conjunctive operator [30]. This produces a resultant T2DF δ_{result}^2, as shown in Fig. 6.1. Combining various T2DFs helps to reduce the number of fuzzy rules. This leads to a decrease in the computational complexity of the identified fuzzy model.

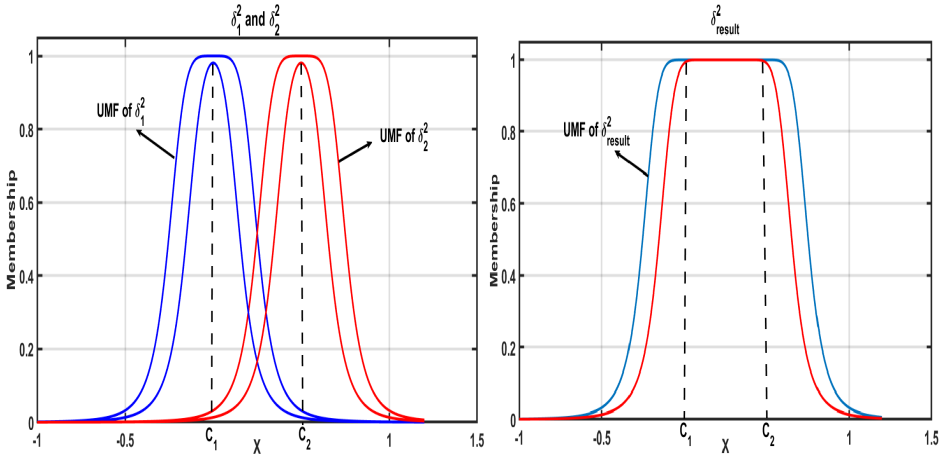

Figure 6.1: *Combining two T2DF (δ_1^2 and δ_2^2) to get a single T2DF (δ_{result}^2)*

6.1.1 The T2DF in a higher dimension

Consider n different T2DFs in n different dimensions given by $\delta_{1(\epsilon_1,\nu_1)}^{2(\lambda_1)}(x_1-c_1)$, $\delta_{2(\epsilon_2,\nu_2)}^{2(\lambda_2)}(x_2-c_2)\ldots,\delta_{n(\epsilon_n,\nu_n)}^{2(\lambda_n)}(x_n-c_n)$. If we apply the Dombi conjunctive operator on these n T2DFs, then the result will also be a T2DF $\delta^2(x_1,x_2,\ldots,x_n)$ in n dimensions (shown

in Fig. 6.2). And

$$\bar{\delta}^2(x_1, x_2, \ldots, x_n) = \frac{1}{1 + \sum_{i=1}^{n} \frac{1-\nu_i}{\nu_i} \left| \frac{x_i - c_i}{\epsilon_i} \right|^{\lambda_i}}, \tag{6.3}$$

$$\underline{\delta}^2(x_1, x_2, \ldots, x_n) = \frac{1}{1 + \sum_{i=1}^{n} \frac{1-\nu_i}{\nu_i} \left(\left| \frac{x_i - (c_i - \Delta)}{\epsilon_i} \right|^{\lambda_i} \sigma_{Li} + \left| \frac{x_i - (c_i + \Delta)}{\epsilon_i} \right|^{\lambda_i} \sigma_{Ri} \right)}, \tag{6.4}$$

$$\text{where } \sigma_{Li} = \frac{1}{1 + e^{-\lambda_i (x_i - (c_i - \Delta))}} \quad \text{and} \quad \sigma_{Ri} = \frac{1}{1 + e^{\lambda_i (x_i - (c_i + \Delta))}}.$$

Here Δ is the upper bound on the uncertainty in the c value, $\bar{\delta}^2(x_1, x_2, \ldots, x_n)$ is the UMF and $\underline{\delta}^2(x_1, x_2, \ldots, x_n)$ is the LMF of $\delta^2(x_1, x_2, \ldots, x_n)$.
Consider n T2DFs $\delta^{2(\lambda_1)}_{1(\epsilon_1,\nu_1)}(x_1 - c_1)$, $\delta^{2(\lambda_2)}_{2(\epsilon_2,\nu_2)}(x_2 - c_2) \ldots, \delta^{2(\lambda_n)}_{n(\epsilon_n,\nu_n)}(x_n - c_n)$ in n different dimensions $(x_1, x_2, \ldots, x_n)$. If we apply the Dombi conjunctive operator on these n T2DFs, it will produce a single n dimensional T2DF $\delta^2(x_1, x_2, \ldots, x_n)$.

Proof. Consider two T2DFs $\delta^2_1(x_1)$ and $\delta^2_2(x_2)$ in x_1 and x_2 (two) dimensions, respectively. The UMF and LMF of $\delta^2_1(x_1)$ are

$$\bar{\delta_1}^2(x_1) = \frac{1}{1 + \frac{1-\nu_1}{\nu_1} \left| \frac{x_1 - c_1}{\epsilon_1} \right|^{\lambda_1}}, \tag{6.5}$$

$$\underline{\delta_1}^2(x_1) = \frac{1}{1 + \frac{1-\nu_1}{\nu_1} \left(\left| \frac{x_1 - (c_1 - \Delta)}{\epsilon_1} \right|^{\lambda_1} \sigma_{L1} + \left| \frac{x_1 - (c_1 + \Delta)}{\epsilon_1} \right|^{\lambda_1} \sigma_{R1} \right)}, \tag{6.6}$$

$$\text{where } \sigma_{L1} = \frac{1}{1 + e^{-\lambda_1 (x_1 - (c_1 - \Delta))}} \quad \text{and} \quad \sigma_{R1} = \frac{1}{1 + e^{\lambda_1 (x_1 - (c_1 + \Delta))}}.$$

Similarly, the UMF and LMF of $\delta^2_2(x_2)$ are given by:

$$\bar{\delta_2}^2(x_2) = \frac{1}{1 + \frac{1-\nu_2}{\nu_2} \left| \frac{x_2 - c_2}{\epsilon_2} \right|^{\lambda_2}}, \tag{6.7}$$

$$\underline{\delta_2}^2(x_2) = \frac{1}{1 + \frac{1-\nu_2}{\nu_2} \left(\left| \frac{x_2 - (c_2 - \Delta)}{\epsilon_2} \right|^{\lambda_2} \sigma_{L2} + \left| \frac{x_2 - (c_2 + \Delta)}{\epsilon_2} \right|^{\lambda_2} \sigma_{R2} \right)}, \tag{6.8}$$

$$\text{where } \sigma_{L2} = \frac{1}{1 + e^{-\lambda_2 (x_2 - (c_2 - \Delta))}} \quad \text{and} \quad \sigma_{R2} = \frac{1}{1 + e^{\lambda_2 (x_2 - (c_2 + \Delta))}}.$$

$\delta_1^2(x_1)$ and $\delta_2^2(x_2)$ can be combined using the Dombi conjunctive operator. It will give a single 2-dimensional T2DF $\delta^2(x_1, x_2)$.
Applying the Dombi conjunctive operator on UMFs of $\delta_1^2(x_1)$ and $\delta_2^2(x_2)$ (see Eq. (6.5) and Eq. (6.7)),

$$\bar{\delta}^2(x_1, x_2) = \frac{1}{1 + \frac{1-\bar{\delta_1}^2}{\bar{\delta_1}^2} + \frac{1-\bar{\delta_2}^2}{\bar{\delta_2}^2}},$$

and this gives

$$\bar{\delta}^2(x_1, x_2) = \frac{1}{1 + \frac{1-\nu_1}{\nu_1}\left|\frac{x_1-c_1}{\epsilon_1}\right|^{\lambda_1} + \frac{1-\nu_2}{\nu_2}\left|\frac{x_2-c_2}{\epsilon_2}\right|^{\lambda_2}}, \tag{6.9}$$

where $\bar{\delta}^2(x_1, x_2)$ is the UMF of $\delta^2(x_1, x_2)$. In the same way, the LMFs of $\delta_1^2(x_1)$ and $\delta_2^2(x_2)$ can be combined using the Dombi conjunctive operator. It will generate the LMF of $\delta^2(x_1, x_2)$. Applying the Dombi conjunctive operator on Eq. (6.6) and Eq. (6.8),

$$\underline{\delta}^2(x_1, x_2) = \frac{1}{1 + \frac{1-\underline{\delta_1}^2}{\underline{\delta_1}^2} + \frac{1-\underline{\delta_2}^2}{\underline{\delta_2}^2}}.$$

Upon simplification, we get

$$\underline{\delta}^2(x_1, x_2) = \frac{1}{1 + \sum_{i=1}^{2} \frac{1-\nu_i}{\nu_i}\left(\left|\frac{x_i-(c_i-\Delta)}{\epsilon_i}\right|^{\lambda_i}\sigma_{Li} + \left|\frac{x_i-(c_i+\Delta)}{\epsilon_i}\right|^{\lambda_i}\sigma_{Ri}\right)}. \tag{6.10}$$

The results can be generalized to n T2DFs in n dimensions. It will produce a single n dimensional T2DF $\delta^2(x_1, x_2, \ldots, x_n)$. The UMF and LMF of this n-dimensional T2DF are given by:

$$\bar{\delta}^2(x_1, x_2, \ldots, x_n) = \frac{1}{1 + \sum_{i=1}^{n} \frac{1-\nu_i}{\nu_i}\left|\frac{x_i-c_i}{\epsilon_i}\right|^{\lambda_i}} \text{ and} \tag{6.11}$$

$$\underline{\delta}^2(x_1, x_2, \ldots, x_n) = \frac{1}{1 + \sum_{i=1}^{n} \frac{1-\nu_i}{\nu_i}\left(\left|\frac{x_i-(c_i-\Delta)}{\epsilon_i}\right|^{\lambda_i}\sigma_{Li} + \left|\frac{x_i-(c_i+\Delta)}{\epsilon_i}\right|^{\lambda_i}\sigma_{Ri}\right)}, \tag{6.12}$$

where

$$\sigma_{Li} = \frac{1}{1 + e^{-\lambda_i(x_i-(c_i-\Delta))}} \qquad \text{and} \qquad \sigma_{Ri} = \frac{1}{1 + e^{\lambda_i(x_i-(c_i+\Delta))}}.$$

□

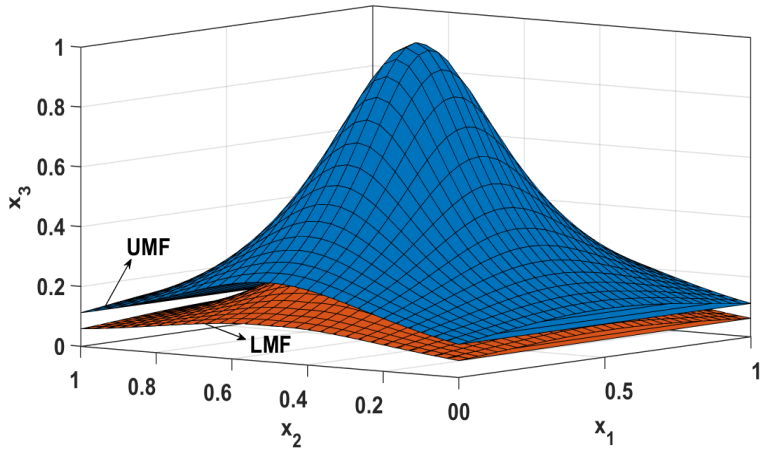

Figure 6.2: *The T2DF in third (x_3) dimension*

Fig. 6.3 shows a block diagram of the proposed design approach. The design consists of four main steps. The first three steps extract the type2 fuzzy model (called the T2DF model) from the process input and output data. The T2DF model consists of rules and type2 Distending Functions (T2DFs). The fourth step is the construction of the fuzzy type2 controller.
In the data preprocessing step, the inputs are normalized by transforming values to the $[0, 1]$ interval. This is needed to put all the inputs on the same scale and also the T2DFs have a significant effect on this interval. In the second step, several key rules are extracted from the data. T2DFs are defined for the each input in these rules.These rules and T2DFs jointly form the T2DF model. In the third step, the computational complexity of T2DF model is reduced by combining some of these rules. In the final step, an arithmetic-based type2 controller is designed using the T2DF model. And because T2DFs are used in this design, the type2 controller is robust in the presence of noise. The proposed design approach is explained in the next section.

6.2 Data-driven Type2 Fuzzy Modeling

Here, we assume that expert knowledge in the form of linguistic rules is not available. However the input and output data of the process are available. Using this data, we will drive a T2DF model composed of rules and T2DFs. Let us now assume that the

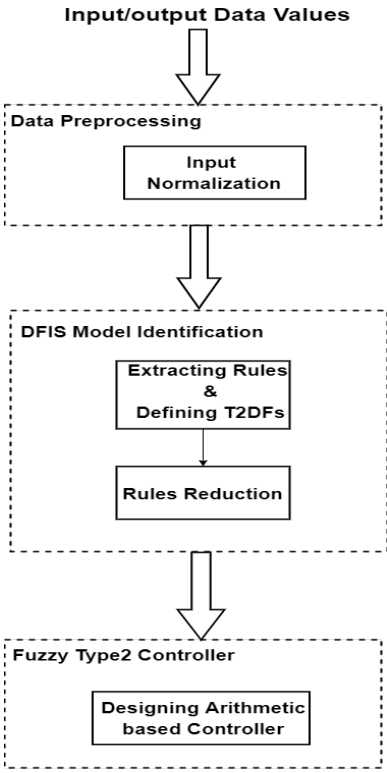

Figure 6.3: *Block Diagram showing the proposed design approach*

input and output databases have the following form:

$$U = \begin{bmatrix} a_1^1 & a_2^1 & \dots & a_n^1 \\ a_1^2 & a_2^2 & \dots & a_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ a_1^l & a_2^l & \dots & a_n^l \end{bmatrix}, \qquad V = \begin{bmatrix} b^1 \\ b^2 \\ \vdots \\ b^l \end{bmatrix}, \tag{6.13}$$

where U and V contains the l data points of each input and output variable. Here, $a_1, a_2, \dots, a_n$ are the data points belonging to the input fuzzy subsets $U_1, U_2, \dots, U_n$, respectively, and b_1 is included in the output fuzzy subset V. Each column of the U matrix corresponds to a unique feature (input variables) of the process. Therefore the U matrix forms an n dimensional input feature space. Each column of the training matrix U is normalized by transforming it to the $[0, 1]$ interval. As a result, the feature values are comparable on the same scale.

In our approach the fuzzy rules will be based on sample values in the U and V matrices. Therefore, a few rows from the data base matrix U are selected. These can be selected randomly, but from a practical point of view it is beneficial to choose those rows that contain the extremum (around 0 and 1) and average (around 0.5) values of the input variables. These rows and the corresponding elements in the V matrix

are used to construct the rule base. It is called the boundary-value rule base (R_b) because it mostly contains those values of the inputs that lie on the boundary of the input space.
In our procedure, two different surfaces are constructed. These are called the estimated and the fuzzy surfaces. The estimated surface is constructed directly from the database (Eq. (6.13). For missing data values, linear interpolation is used. The estimated surface is denoted by G. The fuzzy surface is generated from R_b and it is denoted by G^*. These two surface are then used to create a third surface called the error surface E. Next, we will describe the procedure used to construct the surfaces G^* and E using R_b .

6.2.1 Construction of Fuzzy Surface G^* and Error Surface E

Each selected row from the database matrix U corresponds to a single rule. It is a row vector and it consists of unique values of all the input variables (features). We will construct T2DFs for all input variables. The parameter c (peak value coordinate) of the T2DF is given and it is equal to the value of the corresponding input variable. A value of λ can be chosen between 1 to ∞, but for practical applications $\lambda = 3$ serves as a good initial value. The value of ε depends upon the number of rules in R_b ($\varepsilon = \frac{1}{\text{no. of rules in } R_b}$) and it ensures that the whole input space is covered. The input variables are usually measured using the feedback sensors. The Δ value of each sensor depends on the tolerance intervals of the corresponding sensor. All the Δ values are transformed into the $[0, 1]$ interval to make these compatible with the values of the input variables. T2DFs have a long tail. Consequently each T2DF influences the other existing T2DFs. The ν value of each T2DFs will be calculated based on the principle of minimum influence on all the other T2DFs. This influence can never be zero, but it can be decreased by a factor k. For less influence, a large value of k should be chosen. However from a practical point of view, a value of 10 is sufficient. It means that the influence will decrease 10 fold. The influence of the ith T2DF at the peak value of the jth T2DF is given by

$$\frac{1}{1 + \frac{1-\nu}{\nu}\left(|\frac{x_{i1}-x_{j1}}{\epsilon}|^{\lambda} + \cdots + |\frac{x_{in}-x_{jn}}{\epsilon}|^{\lambda}\right)} = \frac{1}{k}, \tag{6.14}$$

where $x_{i1}, \ldots, x_{in}$ are the n coordinates of the peak value of the ith T2DF and $x_{j1}, \ldots, x_{jn}$ are the coordinates of the peak value of the jth T2DF. This formula can be used for single dimensional T2DFs as well as n dimensional T2DFs. Then the required value of ν can be calculated using

$$\nu = \frac{1}{1 + \frac{k-1}{d}}, \tag{6.15}$$

where $d = \left(|\frac{x_{i1}-x_{j1}}{\epsilon}|^{\lambda} + \cdots + |\frac{x_{in}-x_{jn}}{\epsilon}|^{\lambda}\right)$. Obviously it is a distance measure if $\varepsilon = 1$ and $\lambda = 2$.
In the antecedent part of the ith rule, there are n T2DFs corresponding to n input variables. Each rule is evaluated using the Dombi conjunctive/disjunctive operator. By applying the Dombi conjunctive/disjunctive operator over the n input T2DFs, we get a single T2DF. This is called the output T2DF. The UMF and LMF of this output T2DF are given by Eq. (6.3) and Eq. (6.4), respectively. Here, l output T2DFs will be generated from the l rules. All these output T2DFs are superimposed in the input space to generate a fuzzy surface G^*.
An error surface E is defined as the difference between the estimated surface G and the fuzzy surface G^*. That is,

$$E(x_1, \ldots, x_n) = G(x_1, \ldots, x_n) - G^*(x_1, \ldots, x_n). \tag{6.16}$$

6.2.2 Extending the rule base

We shall decrease the magnitude of E below a chosen threshold τ_E ($E < \tau_E$). This is achieved by an iterative procedure of adding new rules to R_b. To add a new fuzzy rule, the coordinates of the maximum value on E are located. The corresponding row in the database containing these coordinates is selected. This row is then added to R_b as a new rule. This rule is evaluated to generate an output T2DF. The ν value of this output T2DF is then calculated using Eq. (6.15). This T2DF is superimposed in G^*. This will modify the surface G^* in such a way that the magnitude of the maximum error on the surface E at these coordinates will decrease. This process is repeated in an iterative manner until the error surface E is within the tolerance limit τ_E. For a very small value of τ_E, a large number of rules has to be extracted from the training data and vice versa. Therefore a compromise has to be made between the value of the threshold τ_E and the number of rules in R_b. It should be added that extracting the type2 fuzzy model from the data is based on the DF. Therefore, we call this type2 model the DF-based fuzzy inference system (T2DF). The whole procedure for extracting the T2DF from the data is summarized in Algorithm 5.
If the number of rules in R_b is large, then some of the rules can be merged to reduce the computational complexity. This is achieved using a reduction procedure.

6.2.3 Reducing the rule base

Here, we describe a heuristic approach used to decrease the number of rules in R_b. Rules reduction will lead to a lower computational cost and better interpretability. Various output T2DFs which are close to each other in the input space can be combined to get a single T2DF (as shown in Fig. 6.1). The procedure is explained below. One of the input variables is selected and we call it a principal feature. In a control

system, the error measure of the control variable is usually chosen as the principle feature. To ensure that the rule reduction procedure does not change the fuzzy surface significantly, the input space is divided into two half spaces. The space where the principle feature value is less than 0.5 lies in the first half and the rest of the space lies in the second second half. Within each half, the output T2DFs are segregated into different groups. If the Euclidean distance between the peak value coordinates of various output T2DFs is less than a predefined distance D, then these T2DFs are placed in the same group, where for each half:

$$D = \frac{\text{Sum of Euclidean distances b/w peak value coordinates of T2DFs}}{\text{Total no. of T2DFs in the same half}}. \quad (6.17)$$

Each output T2DF is obtained by applying a unique rule in R_b. The output T2DFs in the same group are combined together to produce a single T2DF. Consequently the rules associated with all these output T2DFs are eliminated and replaced by a single new rule. Therefore the number of rules in R_b decreases. Now it is called a reduced rule base R_r. Using R_r, a new fuzzy surface is constructed and it is denoted by G_r^*. Then a reduced error surface (E_r) is obtained using

$$E_r(x_1, \ldots, x_n) = G(x_1, \ldots, x_n) - G_r^*(x_1, \ldots, x_n). \quad (6.18)$$

This procedure is performed in an iterative way as long as $E_r(x_1, \ldots, x_n)$ is within a chosen threshold τ_R. If the threshold value τ_R is high then a large number of rules can be eliminated to get a much simpler and interpretable model. However, the accuracy of such an identified fuzzy model decreases. So the τ_R value should be chosen based on a compromise between model accuracy and interpretablility. A high τ_R value leads to better interpretability but less accuracy and vice versa.
For an accurate T2DF model, the magnitude of the error surface should be a minimum. However an error surface with magnitude very close to zero is not desirable as this will lead to poor generalization (overfitting) of the T2DF model. The value of the threshold τ_R is application-dependent and its optimum value may be different for different applications. However based on our experimental observations, the optimum value of the threshold ($\tau_{R_{op}}$) can be determined using the heuristic

$$\tau_{R_{op}} = \frac{Peak\ value\ of\ the\ output}{10} \quad (6.19)$$

A value of threshold τ_R less than $\tau_{R_{op}}$ will lead to a large number of rules. This is similar to overfitting in the learning paradigms. The T2DF model will accurately fit the training data but it will have poor generalization. This type of T2DF model will be less interpretable and it performs badly on unseen data. A value of τ_R greater than $\tau_{R_{op}}$ will produce a T2DF model with a smaller number of rules. This type of T2DF

model will be more interpretable and have a low computational complexity. However in this case, the accuracy of the T2DF model will be poor. Thus an optimum value of the threshold $\tau_{R_{op}}$ will lead to an optimum number of rules and it will produce a T2DF model with higher accuracy, better interpretability and good generalization capability.
The whole procedure for reducing the number of rules in the T2DF model is summarized in Algorithm 6.

6.2.4 The arithmetic-based fuzzy type2 controller design using the DFIS model

The extracted T2DF model (R_r plus T2DFs) can be used to design an arithmetic-based interval type-2 fuzzy controller. The procedure outlined in section 5.4 can be followed for this purpose.

6.2.5 Algorithms

The procedure for extracting the T2DF model is summarized in Algorithm 5. The

Algorithm 5 Algorithm for Extracting Type-2 Fuzzy Model (DFIS) from the Data

Step 1: Get the input and output databases of the process (Eq. (6.13)).
Step 2: Transform each column of U into the $[0, 1]$ interval.
Step 3: Construct the estimated surface G.
Step 4: Generate a boundary value rule base R_b from the input and output databases.
Step 5: Assign a T2DF to each input point in the antecedent part of the rule base R_b. Choose $k = 10$ and calculate ν using Eq. (6.14) and Eq. (6.15).
Step 6: Calculate the output T2DF for each rule using Eq. (6.3) and Eq. (6.4).
Step 7: Generate fuzzy surfaces G^* from all the output T2DFs.
Step 8: Construct the error surface E using Eq. (6.16). If E is within the threshold τ_E, then stop.
Step 9: Find the maximum value coordinates on E. Add a new rule in R_b corresponding to these coordinates in the databases. Go to Step 6.

number of rules in the rule base R_b can be reduced using Algorithm 6.

6.3 Benchmark System, Simulations Results, Hardware implementation and discussion

The effectiveness of the proposed technique is demonstrated by modeling the altitude control system of a quadcopter (Parrot mini-drone). We have assumed that the

Algorithm 6 Algorithm for Reducing the Number of Rules in DFIS

Step 1: Divide the input space into two parts based on the values of the principle feature.
Step 2: Segregate the output T2DFs in various groups based on the Euclidean distance D (Eq. (6.17)).
Step 3: Combine the T2DFs in each group to generate a single T2DF.
Step 4: Replace the rules associated with the combined T2DFs with a single rule.
Step 5: Construct the reduced error surface E_r using Eq. (6.18).
Step 6: If E_r is within the threshold τ_R, then go to step 1 else stop.

data of the (healthy) altitude controller is available. Using this training data, the proposed T2DF model is identified. This model consists of fuzzy rules and interval T2DFs. Additionally ANFIS type1 and ANFIS type2 fuzzy models are also identified from the same data. The performance of T2DF model is compared with ANFIS-based fuzzy models. Afterwards three fuzzy controllers are designed using these three fuzzy models. The performance of these controllers is compared in a situation where the altitude of the quadcopter is regulated in the presence of noisy sensor measurements. Later on the designed type2 controller will also be deployed on the mini-drone hardware to check the real-time performance.

6.3.1 The Quadcopter Model

A Parrot mini-drone Mambo was used in this study. The Matlab Simulink aerospace block set provides the simulation model of this quadcopter [57]. Section 5.5.1 describes the model in detail.

6.3.2 Extracting the data-driven type2 fuzzy model

Here in these simulations, we seek to model the altitude controller of the quadcopter. A training dataset which contains the samples of input and output of the altitude controller is a requirement for Algorithm 5. The requirement was satisfied by controlling the altitude of quadcopter using the PD controller in Matlab. The dataset containing the inputs and output of the PD controller was created. Later Algorithm 5 was used to generate the T2DF model of the altitude controller using this input and output dataset. The threshold τ_E was set to 0.15. Algorithm 5 then extracted 26 rules from dataset, and these formed the rule base R_b.
The number of rules in R_b was reduced using Algorithm 6. The algorithm reduced the number of rule to 17 by merging a few rules. This is called the reduced rule base R_r. A few T2DFs (3 out of 17) of each input are shown in Fig. 6.5. R_r and all the T2DFs in it, collectively form a reduced (simplified) T2DF model. The surface of the

T2DF model is shown in Fig. 6.4.
For comparison purposes, various fuzzy models were tested. ANFIS type1 model was extracted from the same input and output dataset using MATLAB Fuzzy logic toolbox. The dataset was then divided into train and test sets $(2:1)$. The model was trained for 100 epochs and 36 rules were generated. It is a Sugeno-based fuzzy model. Fig. 6.6 shows the surface plot of the ANFIS type1 Sugeno model. The fuzzy logic toolbox can be used to convert the type1 fuzzy model to a type2 fuzzy model. Therefore an ANFIS type2 model was also generated. This type2 model uses the type2 bell-shaped membership functions (see Fig. 6.7). Table 6.1 summarizes the key differences among the T2DF and various other fuzzy models.

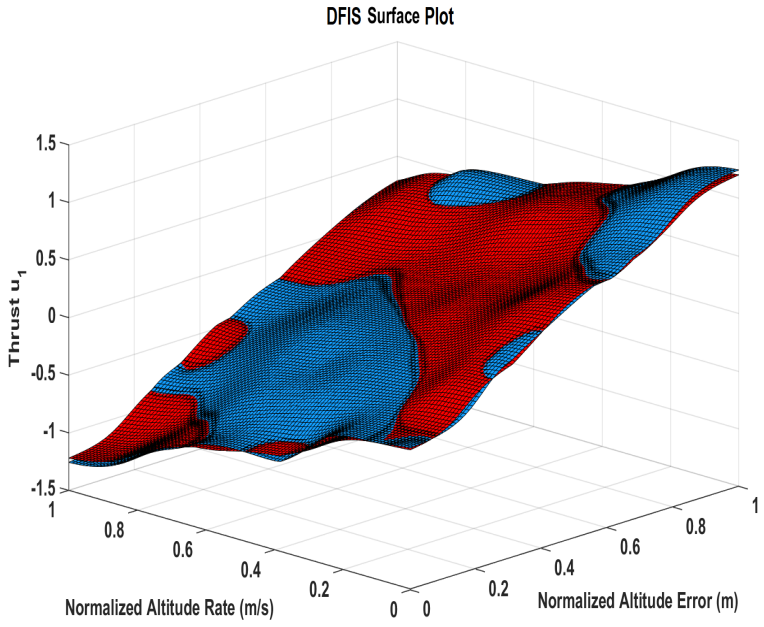

Figure 6.4: *The surface plot of the DFIS model. The upper surface is in blue and the lower surface is in red.*

6.3.3 Designing the arithmetic based type2 altitude controller

The objective is to control the altitude z by generating an appropriate total thrust u_1. The thrust u_1 depends on the height (sonar) measurements and rate of change of the height of the quadcopter. Using Algorithm 4, an arithmetic-based controller was designed using R_r. Fig. 6.8 shows the surface plot of the arithmetic based controller. The ANFIS fuzzy models (type1 and type2) were converted to a fuzzy controller using the Simulink fuzzy logic toolbox. These controllers were based on Sugeno inference logic. The three controllers (arithmetic-based, ANFIS type1, ANFIS type2)

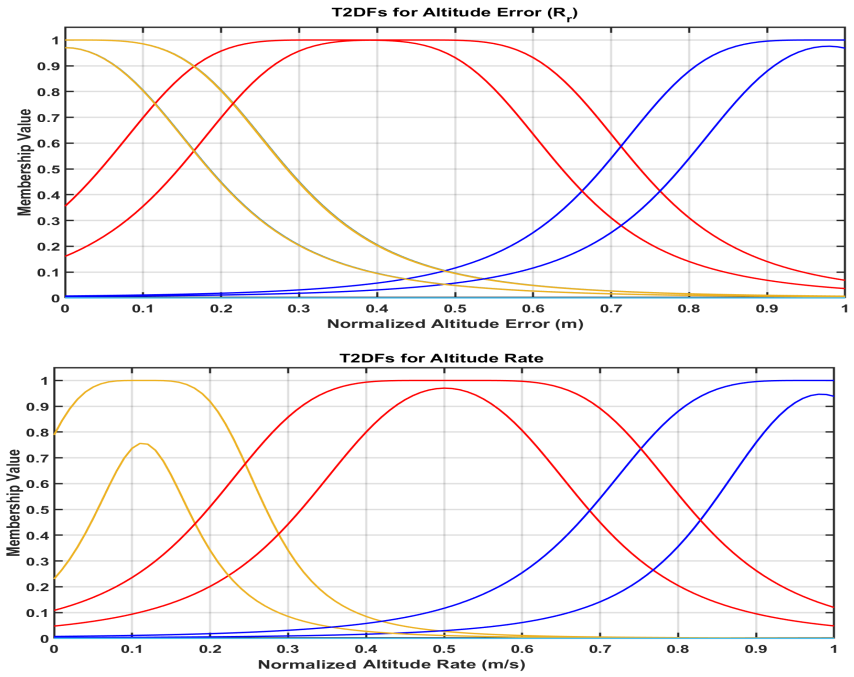

Figure 6.5: *Three T2DFs of each normalized input (DFIS model).*

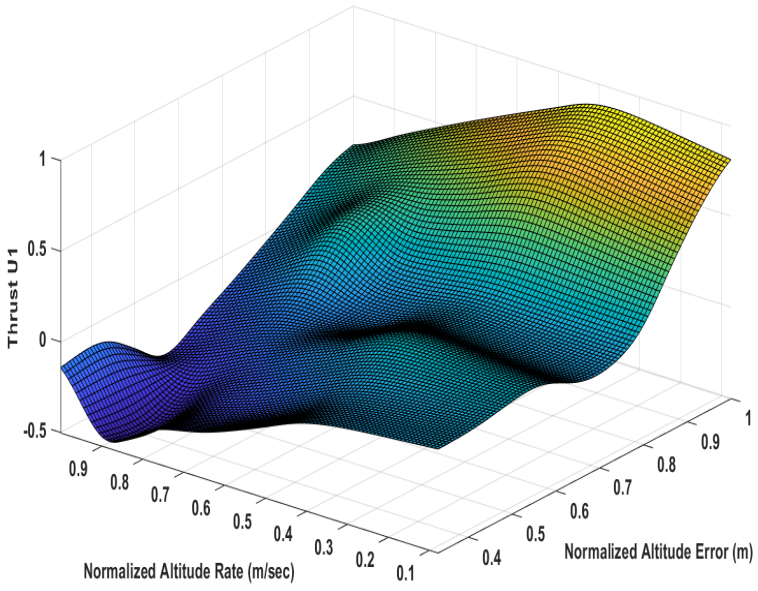

Figure 6.6: *The surface plot of the ANFIS type1 model.*

were used to regulate the altitude of the quadcopter in MATLAB Simulink. White noise was added to the altitude measurements by the sonar sensor. The quadcopter

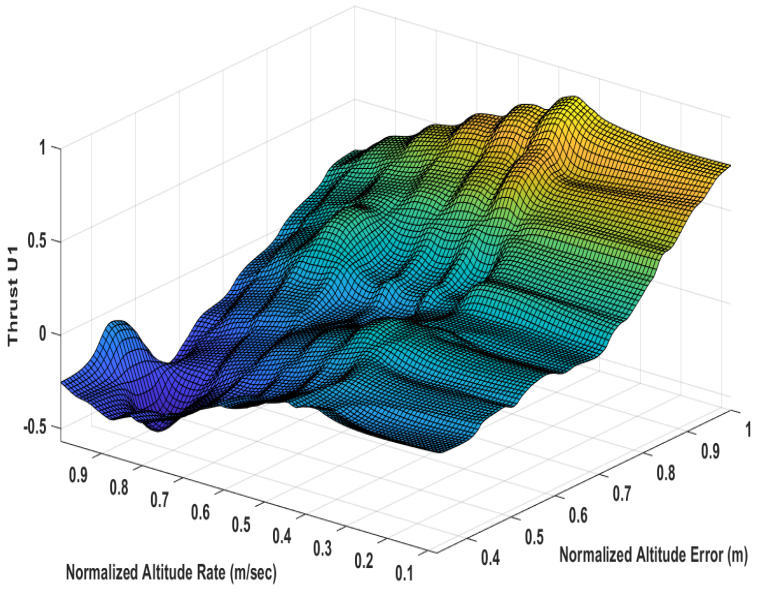

Figure 6.7: *The surface plot of the ANFIS type2 model.*

S. No.	Model	Number of rules	Number of tunable parameters	Membership functions used
1	ANFIS Type1	36	144	Bell shape
2	ANFIS Type2	36	168	Type2 Bell shape
3	IT2FNN [67]	72	336	T2GMF
4	MIT2FC [67]	36	168	T2GMF
5	Proposed T2DF model	17	34	T2DF

Table 6.1: *A performance comparison the proposed DFIS model with previously proposed models.*

was programmed to take off and reach an altitude of 0.7m, then rise to an altitude of 1m and finally descend to 0.7m. Fig. 6.9 shows the altitude response of the controlled quadcopter during this simulation study.

6.3.4 Hardware Implementation

MATLAB provides a support package for parrot quadcopter drones (Mambo FPV and Bebop2). It connects with the quadcopter hardware via Bluetooth/WiFi and it can send control commands. Matlab Simulink includes the simulation model of the quadcopter. The model contains the algorithm for the flight control system (FCS). This algorithm implements roll, pitch, yaw and altitude controllers. The support package

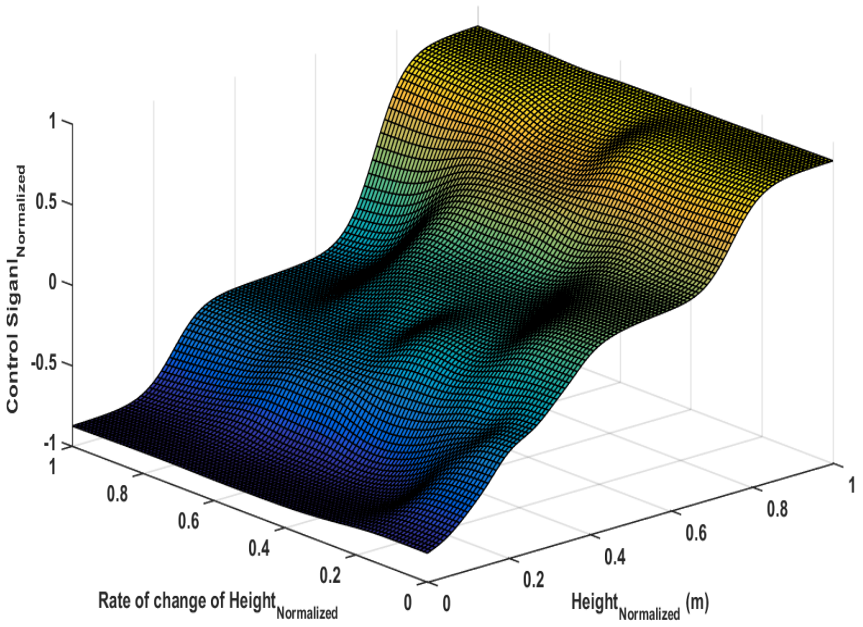

Figure 6.8: *The control surface of the arithmetic-based controller.*

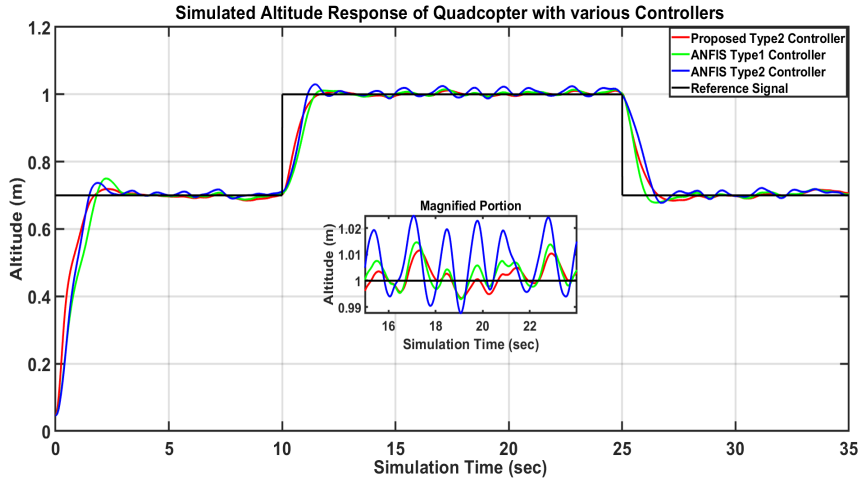

Figure 6.9: *The simulated altitude response of the quadcopter with various controllers (in Matlab Simulink). The measurements got from the altitude sensor (sonar) were corrupted with white noise.*

generates the C code of FCS and deploys it in the quadcopter. This algorithm can access the on-board sensors such as the accelerometer, gyroscopes, camera and sonar. The flight data values (altitude, images, etc) are saved in on-board storage and they can be retrieved from the quadcopter at the end of the flight. The altitude controller in the FCS was replaced with the proposed arithmetic-based type2 controller. The

C code of the FCS was generated and deployed in the quadcopter via Bluetooth. A fixed flight trajectory corresponding to the simulation scenario was also hard coded. The uncertainty in the controller input (altitude) data was created by adding white noise to the sonar measurements. The noisy data was the input to the controller in the FCS. The drone flight was tested in a protected (indoor) environment. The data recorded during the flight was retrieved at the end. Fig. 6.10 shows the altitude measurements recorded during this test flight. This tells us that the proposed controller successfully followed the desired trajectory, even in the presence of noisy sensor data.

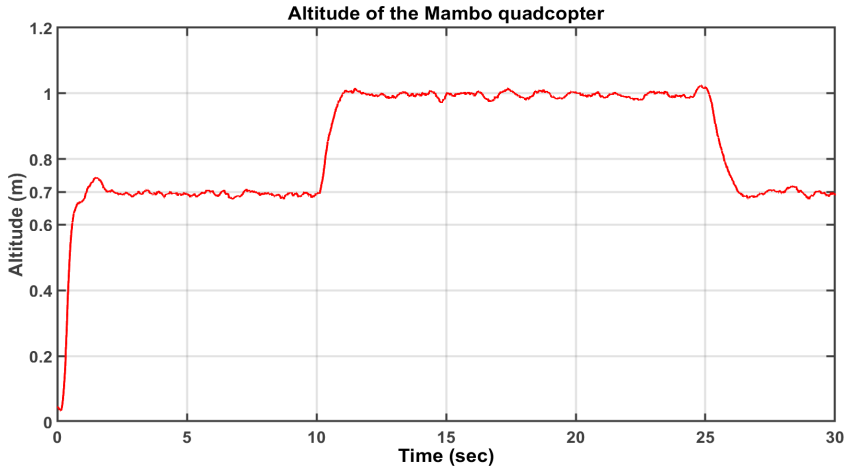

Figure 6.10: *The actual trajectory traversed by the Mambo quadcopter. (Flight data: https://github.com/Abrarlaghari/Mambo-Quadcopter-Simulink.git)*

6.3.5 Results and discussion

From the results obtained, it is evident that the T2DF model can be extracted directly from the process data. The T2DF model is composed of T2DFs and rules based on the Dombi operator. As shown in Fig. 6.4, the proposed model has a smooth surface and it covers the whole input space. In these simulations the λ and ϵ parameters were kept fixed, the c parameter was determined directly from the data and ν was calculated using Eq. (6.15). This reduces the number of parameter calculations per T2DF. In Fig. 6.6, it can be seen that ANFIS type1 model also has a smooth surface except for the boundary areas. On the boundaries there are a few bumps. The ANFIS type2 model (Fig. 6.7) has a less smooth surface compared to those for the T2DF and ANFIS type1 models. Boundary bumps in ANFIS models are due to the fact that only a few data points were available to accurately model these areas. However these bumps are non-existent in the T2DF model because of the long tails of T2DFs. These

tails tend to approximate a smooth transition in the areas where there are only a few data points or even no data points.
The performance of the T2DF model was evaluated by comparing it with the output of the ANFIS type1 and ANFIS type2 fuzzy models. These are two well-established and widely accepted fuzzy modeling approaches. As shown in Table 6.1, the T2DF model used a smaller number of rules and tunable parameters than the ANFIS type1 and ANFIS type2 models did. Fig. 6.4, Fig. 6.6 and Fig. 6.7 show the surface generated by these three models. It is evident that the T2DF model has much smoother surface transitions. Furthermore the qualitative aspects of these models were compared by designing the fuzzy controller. The controllers are based on the rules and membership functions in these models. These controllers were then used to control the altitude of the quadcopter in the presence of noisy sensor data. The altitude control responses of these controllers are shown in Fig. 6.9 for comparison purposes. The performance of the ANFIS type2 controller is worse than the ANFIS type1 and proposed arithmetic-based controllers. The proposed controller is better than the ANFIS type1 controller as it is more robust to the variations in the noisy sensor measurements. In addition to the simulations, the hardware implementations of the proposed controller showed very promising results for the real-time control applications. The controller performed the altitude control function in the presence of uncertain (noisy) measurement data (Fig. 6.10). This demonstrates the effectiveness of the proposed arithmetic based type2 controller.
In ANFIS type1 and type2 controllers, the choice of membership functions and number of rules are hyper-parameters. One has to decide in advance on the type of membership functions (triangular, Gaussian, trapezoidal etc.) and number of categories and rules. This hyper-parameter selection is usually based on experience or data insights. However, in our proposed T2DF model, the number of rules and T2DF are determined automatically by the algorithm based on the threshold τ_R. A high value of τ_R results in a small number of rules, more interpretable but least accurate T2DF model. A low value of τ_R produces a large number of rules and the resulting T2DF model is less interpretable and it also has the worst generalization capability.
As mentioned in the introduction section, most of the developed methods extract the fuzzy rule and tune the parameters using the meta-heuristics. However, these algorithms can get stuck in a local optimum and there is no guarantee that these will find the global optimum solution. Also the local optimum solution provided by these algorithms depends on the initial values of the tunable parameter. These tunable parameters are usually randomly initialized and then a heuristic is applied to get the optimum value. The process is repeated with several random initial values and the best values are chosen at the end via a comparison of the results achieved with each random initialization. The algorithm proposed here uses a different approach to tune the parameters of T2DF model. Values of a few parameters are fixed and they

are never changed during the training phase. For the remaining tunable parameters, there are closed-form expressions (Eq. 6.14 and Eq. 6.15) which can precisely determine the values of these parameters. Rules are added iteratively and the parameters in these rules are determined using the expressions. The rule addition process continues until the error is within the bounds of threshold τ_R. As no heuristic is employed here, the values determined for the tunable parameters are near-optimum.

6.4 Summary

In this chapter, we presented the solutions to some of the limitations associated with the existing fuzzy type2 modeling and control techniques. A procedure was proposed to identify the type2 model directly from the data, which we called the T2DF model. This model consists of rules and Type2 Distending Functions (T2DFs). The whole input space is covered using a few rules. T2DFs can model various types of uncertainties using its parameters. A rule reduction procedure is also proposed. It combines the T2DFs in the close vicinity and it significantly reduces the number of rules. The T2DF model was compared with ANFIS type1 and ANFIS type2 fuzzy models. The T2DF model produced a smooth surface with comparatively fewer number of rules and tunable parameters. Furthermore, a procedure was proposed to design an arithmetic-based interval type2 fuzzy controller using the rules. As the controller design does not include the implication and type reduction steps, This greatly reduces the computational complexity and paves the way for the real-time implementation of the proposed design. The effectiveness of the whole procedure was demonstrated by designing an altitude controller for the Parrot Mambo quadcopter. Simulations were carried out in Matlab Simulink to compare the proposed controller with ANFIS-based controllers. The proposed controller performed better and regulated the altitude of the quadcopter even in the presence of noisy (uncertain) sensor measurements. This robustness to noisy data is due to the use of T2DFs. The designed controller was then deployed and tested in the flight control system on the quadcopter hardware. Real-time hardware implementations produced the same results as those obtained in the simulations. Because of its low computational complexity and design simplicity, the controller is suitable for real-time control applications.

The proposed T2DF model has a few limitations. It can produce an interpretable and computationally less expensive model only if the number of features (inputs) is small. For a large number of features, a fuzzy surface has to be constructed in a higher dimension and a large number of complex rules is required to correctly capture the inter-feature correlations. In this case, the T2DF model will still be accurate but it will be less interpretable. Therefore, the proposed methodology is only applicable to application domains where the number of features is small such as control systems design and financial modeling. For other domains, where the number of features is

large (computer vision, natural language processing), the T2DF will produce a very complex model with low interpretability.

Chapter 7

Rule-based Neural Networks

Deep Neural Networks (DNNs) are currently one of the most important research areas of Artificial Intelligence (AI). Various types of DNNs have been proposed to solve practical problems in many fields. However all these different types of DNNs have poor interpretations and are black box solutions. Expert systems are also a key area of AI that are based on rules. They have a lot of applications and their results are interpretable. In this chapter we seek to combine the advantages of these two areas and it is a step towards interpretable neural networks. Here, we present a new type of learning paradigm called Rule-Based Neural Network (RBNN). RBNNs can be trained to learn various regression and classification tasks. It has relatively few trainable parameters. It is robust to (input) feature noise and it provides a good prediction accuracy even in the presence of large feature noise. The learning capacity of the RBNN can be enhanced by increasing the number of neurons, number of rules and number of hidden layers. It is interpretable and trained rules can be extracted, which show the relationship between the input features and outputs. The performance of the RBNN on a skewed dataset is comparatively better than that for the DNN and it produces higher F1 scores for the minority (smaller) classes. The effectiveness of RBNNs is also demonstrated by learning real world regression and classification tasks.

7.1 Introduction

DNNs and FISs seem to complement each other. It is the natural to expect that a combination of these two approaches might have the advantages of both. On the one hand, FISs can benefit from the computational learning procedures of the DNNs. The parameters of the rule (sometimes the whole rule) can be learnt directly from the data. Because of this the need for the expert knowledge is no longer essential. On the other hand, the DNN can take advantages of benefits offered by an FIS. The incorporation of fuzzy logic with a DNN leads to the development of deep fuzzy neural

networks (DFNNs). Here we try to generalize the concept of DFNNs by presenting a Rule-Based Neural Network (RBNN). The aim is to combine the advantages of a DNN and FIS. Having a similar architecture to a DNN, it has an input layer, hidden layers and an output layer. The input layer has the normalization functionality. A new type of normalization technique is used and it is called the Ordered Normalization (ON). ON is useful when the training data has an asymmetric distribution. Hidden and output layers contain Rule-Based Neuron (RBN). Each RBN has a built-in Fuzzy Inference System (FIS). An arithmetic-based FIS is used because it has good interpretation and low computational complexity [36]. This FIS is based on parametric rules. The parameter values of these rules are tuned during the training phase. Stochastic Gradient Descent (SGD), Batch Gradient Descent (BGD) and Levenberg-Marquadt (LM) optimization methods are used in the parameter update of back propagation. The output of each RBN is calculated by evaluating the rules using arithmetic operations. Here, we propose the training algorithms of an RBNN for regression and classification tasks. The number of neurons, number of hidden layers and number of rules inside each neurons are hyper-parameters of the RBNN. Compared to the DNN, the RBNN is robust to noise i.e. the prediction accuracy is not affected by noisy features (inputs). After the training phase, the prediction results of the RBNN can be interpreted using simple if-else rules, hence the RBNN is not a black box model. The number of RBNN parameters is quite small (only several hundred). We have tried to maintain a similarity between a DNN and an RBNN. Here, we trained the RBNN using batch gradient descent, stochastic gradient descent and Levenberg-Marquardt optimization methods. Other different variants of gradient-based optimization can also be used to train an RBNN. The weighted Dombi operator (WDO) [33] is a continuously valued logical operator, defined as:

$$O_D^{(\alpha)}(x) = \frac{1}{1 + \left(\sum_{i=1}^{n} w_i \left(\frac{1-x_i}{x_i}\right)^{\alpha}\right)^{\frac{1}{\alpha}}}, \tag{7.1}$$

where w_i is the weight of the input x_i. Here, α determines the nature of the logical operation. If $\alpha > 0$, the operator is conjunctive and if $\alpha < 0$ then the operator is disjunctive. Hence we can perform conjunction and disjunction operations between various inputs using a single operator. This is the main reason for using WDO in the parametric rules.

7.2 Proposed Network Structure

The proposed network structures for regression and classification tasks are shown in Fig. 7.1 and Fig. 7.2. The structures are similar to the multilayer perceptron model (MLP), but there are no weights on the connections. There are no bias terms. Each

node is a rule-based neuron (RBN) and it is fully connected. The feature values are normalized by the input layer. The hidden layer RBNs perform the arithmetic-based inferences and generate a single numerical output. In the case of regression tasks, the output layer produces a single value. In classification, the values generated by the output layer are passed through a softmax function to get the probability values of all the classes. Next, we describe in detail the feed-forward and feedback calculations for these RBNNs.

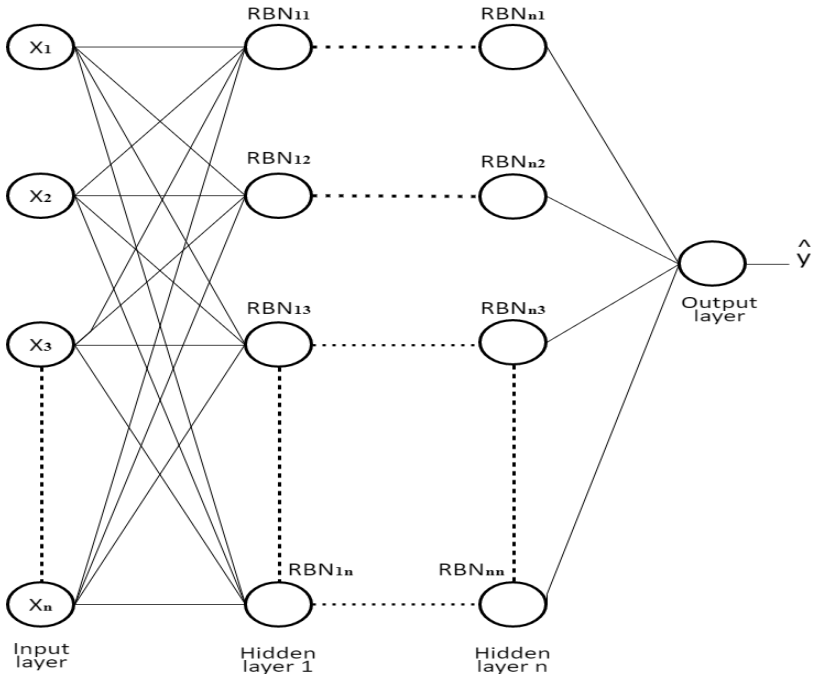

Figure 7.1: *RBNN structure for regression tasks.*

7.2.1 Feed-forward Calculations

The input layer performs the normalization function by transforming the feature values into the $[0,1]$ interval. Any existing normalization technique can be applied. If the features values are not distributed uniformly then a special type of normalization called Ordered Normalization (ON) can be used. Algorithm 7 describes the procedure used to generate the ordered normalized values of the input features. Table 7.1 shows a simple example of normalization using ON. Let $x_1, x_2, \ldots, x_n$ be the normalized feature values generated by the input layer. The number of rules in each RBN of the hidden and output layer is fixed and it is a hyper-parameter. Let us now consider the feed-forward calculations in a single RBN unit and it can be generalized to all the RBNs in the network. Each RBN contains L rules and each rule has the following

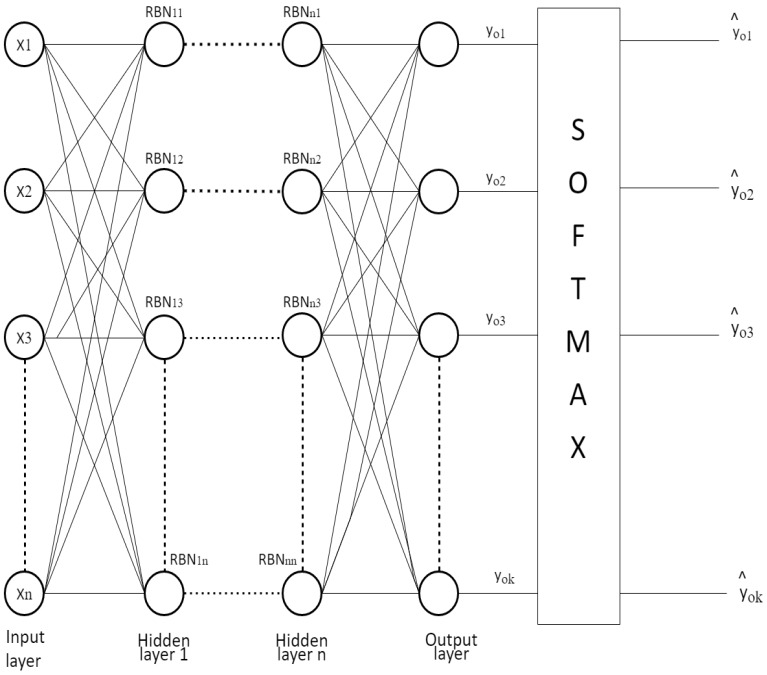

Figure 7.2: *The RBNN structure for classification tasks.*

S.No	Values	Normalized Values
1	10	.1
2	50	.2
3	70	.3
4	81	.4
5	85	.5
6	85	.5
7	85	.5
8	90	.8
9	95	.9
10	100	1

Table 7.1: *Ordered Normalization (here N=10)*

form:

$$\text{if } x_1 \text{ is } f_{j1} \text{ and } \ldots\ldots \text{ and } x_n \text{ is } f_{jn} \quad \text{then } y_j \text{ is } o_j. \tag{7.2}$$

The part between if and $then$ is called the antecedent and the rest of it is called the consequent. Here, $x_1, \ldots, x_n$ are the inputs and $f_{j1}, \ldots, f_{jn}$ are the n distending functions in the jth rule. y_j is the consequent of the jth rule and its value is o_j. A

Algorithm 7 Algorithm for Ordered Normalization

Step 1: Denote the non-normalized and normalized data columns by k and L, respectively.
Step 2: Sort the values in column k and count the total number of rows (N).
Step 3: Initialize a temporarily variable *incr* $= \frac{1}{N}$.
Step 4: Assign $incr$ to the first row in L.
Step 5: For row n ($n = 2, \ldots, N$) in L:

if $k(n+1) = k(n)$
then
$L(n+1) = L(n)$
incr $=$ *incr* $+ \frac{1}{N}$
else if $k(n+1) \neq k(n)$
then
$L(n+1) = L(n) +$ *incr*
incr $= \frac{1}{N}$

Step 7: L contains the ordered normalized values.

unique f_{ji} is associated with the input x_j in the jth rule. If there are L rules in an RBN unit, then there are L DFs for each input ($x_1, \ldots, x_n$). This implies that in a single RBN unit there is a total of nL DFs. Each rule is evaluated and it results in a single numerical value. This is the rule strength and it is given by

$$u_j = \frac{1}{1 + \left(\sum_{i=1}^{n} w_{ji} \left(\frac{1 - f_{ji}(x_i)}{f_{ji}(x_i)} \right)^{\alpha} \right)^{\frac{1}{\alpha}}}, \tag{7.3}$$

where $i = 1, \ldots, n$ is the number of inputs and $j = 1, \ldots, L$ is the number of the rules. Here,

$$f_{ji}(x_i) = \frac{1}{1 + \frac{1-\nu_{ji}}{\nu_{ji}} \left| \frac{x_i - c_{ji}}{\varepsilon_{ji}} \right|^{\lambda_{ji}}}.$$

Putting this into Eq. (7.3) gives

$$u_j = \frac{1}{1 + \left(\sum_{i=1}^{n} w_{ji} \left(\frac{1-\nu_{ji}}{\nu_{ji}} \left| \frac{x_i - c_{ji}}{\varepsilon_{ji}} \right|^{\lambda_{ji}} \right)^{\alpha} \right)^{\frac{1}{\alpha}}}. \tag{7.4}$$

L rules are evaluated using Eq. (7.4) to get L rule strengths $(u_1, u_2, \ldots, u_L)$. Then the output of the RBN unit is a single numerical value calculated using

$$y = \frac{u_1 o_1 + \cdots + u_L o_L}{u_1 + \cdots + u_L},$$
$$y = \frac{\sum_{j=1}^{L} u_j o_j}{\sum_{j=1}^{L} u_j}, \tag{7.5}$$

where o_j is the consequent value in the jth rule.
Using Eq. (7.4) and Eq. (7.5), the outputs of all the RBN units in the specific layer are calculated. Next, these values are propagated as input values to the next hidden layer. These feed-forward calculations are performed for all the layers. In the case of regression RBNN, the output value $\hat{y}$ is the predicted value of the desired variable y. In the case of classification RBNN, the k output values $y_{o1}, \ldots, y_{ok}$ are passed through a SOFTMAX layer and we get the probability values $(\hat{y}_{o1}, \ldots, \hat{y}_{ok})$ of the k classes using the expression

$$\hat{y}_{oi} = \frac{e^{y_{oi}}}{\sum_{i=1}^{k} e^{y_{oi}}}. \tag{7.6}$$

7.2.2 Feedback calculations

Now we describe in detail the gradient-based method for tuning the parameters. Here, we will concentrate only on the regression RBNNs. In the case of classification networks, most of the calculations are the same with a few minor changes. These changes are described at the end of this section.
The squared loss function is

$$J = \frac{1}{2n} \sum_{k=1}^{n} \left(\hat{y_k} - y_k\right)^2, \tag{7.7}$$

where $\hat{y_k}$ is the predicted output and y_k is the label in the database. Here n denotes the number of the training samples. The gradient of J with respect to the predicted output $\hat{y}$ is calculated via

$$\frac{\partial J}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} \left(\frac{1}{2n} \sum_{k=1}^{n} \left(\hat{y_k} - y_k\right)^2 \right),$$
$$= \frac{1}{n} \sum_{k=1}^{n} \left(\hat{y_k} - y_k\right). \tag{7.8}$$

The gradient J with respect to the consequent parameter o_j in the jth rule can be calculated via the chain rule

$$\frac{\partial J}{\partial o_j} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_j}. \tag{7.9}$$

Using Eq. (7.5),

$$\frac{\partial \hat{y}}{\partial o_j} = \frac{u_j}{\sum_{j=1}^{L} u_j},$$

where $j = 1, \ldots, L$ is the rule number in RBN. From Eq. (7.9),

$$\frac{\partial J}{\partial o_j} = \left(\frac{1}{n} \sum_{k=1}^{n} (\hat{y_k} - y_k) \right) \left(\frac{u_j}{\sum_{j=1}^{L} u_j} \right). \tag{7.10}$$

In a similar way, the gradient of J with respect to the jth rule strength u_j can be calculated via the chain rule

$$\frac{\partial J}{\partial u_j} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u_j},$$

where

$$\frac{\partial J}{\partial u_j} = \frac{\left(\sum_{j=1}^{L} u_j \right) o_j - \left(\sum_{j=1}^{L} u_j o_j \right)}{\left(\sum_{j=1}^{L} u_j \right)^2}. \tag{7.11}$$

For the ith distending function in the jth rule, the gradient of J is given by

$$\frac{\partial J}{\partial f_{ji}} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u_j} \frac{\partial u_j}{\partial f_{ji}},$$

and from Eq. (7.3)

$$\frac{\partial u_j}{\partial f_{ji}} = \frac{\gamma_j^{\frac{1}{\alpha_j}-1} w_{ji} \beta_{ji}^{\alpha_j}}{\left(\gamma_j^{\frac{1}{\alpha_j}} + 1 \right)^2 f_{ji} (1 - f_{ji})}, \tag{7.12}$$

where

$$\beta_{ji} = \frac{1 - f_{ji}}{f_{ji}}, \qquad \gamma_j = \sum_{i=1}^{n} w_{ji} (\beta_{ji})^{\alpha_j}.$$

In the same way, the gradient of J with respect to the weights of the Dombi operator is given by

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u_j} \frac{\partial u_j}{\partial w_{ji}},$$

and using Eq. (7.3)

$$\frac{\partial u_j}{\partial w_{ji}} = \frac{\gamma_j^{\frac{1-\alpha_j}{\alpha_j}} w_{ji} \beta_{ji}^{\alpha_j \frac{-1}{\alpha_j}}}{\left(\gamma_j^{\frac{1}{\alpha_j}} + 1\right)^2}. \tag{7.13}$$

Lastly, the gradients of J with respect to parameters of the DF can also be calculated via the chain rule

$$\begin{aligned}
\frac{\partial J}{\partial \varepsilon_{ji}} &= \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u_j} \frac{\partial u_j}{\partial f_{ji}} \frac{\partial f_{ji}}{\partial \varepsilon_{ji}} \\
\frac{\partial J}{\partial c_{ji}} &= \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u_j} \frac{\partial u_j}{\partial f_{ji}} \frac{\partial f_{ji}}{\partial c_{ji}} \\
\frac{\partial J}{\partial x_{ji}} &= \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u_j} \frac{\partial u_j}{\partial f_{ji}} \frac{\partial f_{ji}}{\partial x_i}.
\end{aligned}$$

These gradient calculations are required to update the DF parameters, where

$$\frac{\partial f_{ji}}{\partial \varepsilon_{ji}} = \frac{\lambda}{\varepsilon_{ji}} \left(1 - f_{ji}\right) f_{ji}, \tag{7.14}$$

$$\frac{\partial f_{ji}}{\partial x_{ji}} = \frac{-\lambda}{x_i - c_{ji}} \left(1 - f_{ji}\right) f_{ji}, \tag{7.15}$$

$$\frac{\partial f_{ji}}{\partial \nu_{ji}} = \frac{1}{\nu_{ji}\left(1 - \nu_{ji}\right)} \left(1 - f_{ji}\right) f_{ji}, \tag{7.16}$$

$$\frac{\partial f_{ji}}{\partial c_{ji}} = \frac{\lambda}{x_i - c_{ji}} \left(1 - f_{ji}\right) f_{ji}. \tag{7.17}$$

Once the gradients of the output RBN are available, the gradients for the hidden layers can be derived using the chain rule and back propagating the $\frac{\partial J}{\partial x}$ term. This term is the loss function for the hidden layer which are connected directly to the output RBN. This method can be extended for the feedback calculations throughout the network.

There are a few minor changes in the classification RBNN. The loss function is the

categorical cross entropy loss J given by

$$J = - \sum_{k=1}^{no.of classes} y_{ok} \log(\hat{y_{ok}}), \tag{7.18}$$

where y_{ok} is the label vector obtained from the database and $\hat{yok}$ is the class probabilities vector obtained from the softmax layer

$$\hat{y_{ok}} = \frac{e^{y_{ok}}}{\sum_{k=1}^{no.of classes} e^{y_{ok}}}. \tag{7.19}$$

Also for the classification network,

$$\frac{\partial J}{\partial y_{ok}} = \hat{y_{ok}} - y_{ok}. \tag{7.20}$$

The rest of the calculations are same as in a regression network and equations (7.10) to (7.17) can be used to calculate the gradients.

7.3 Training

The training process is similar to the training of the classical MLP, with some minor differences. A few metaparameters have to be chosen before starting the training process. These metaparameters are: 1) The number of hidden layers; 2) The number of RBN units in each hidden layer; 3) The number of rules in each RBN unit; 4) The learning rate; and 5) The values of some fixed parameters of the DF.
For interpretability and low computational complexity, it is recommended that one start the training using a small number of hidden layers (preferably one), a small number of RBN units in each layer (preferably two) and a few rules (preferably two) in each RBN unit. If the desired accuracy is not achieved after training, then increase the number of rules up to 6. In the next step increase the number of RBNs up to 10. If this is still insufficient then increase the number of hidden layers. As the learning capacity of an RBN is high, a large number of tasks can be learnt with good accuracy using only a single hidden layer with 3 RBN units and each having 2 to 5 rules. There are 4 parameters of the DF, namely $\nu, \varepsilon, \lambda$ and c. Usually λ and ε are fixed ($\lambda = 2$, $\varepsilon < 0.3$). The other two parameters (ν and c) are tunable and they are varied during the optimization process. The α parameter of the WDO determines the nature of the rule (conjunctive/disjunctive), and it is either 1 or -1. If we set $\alpha = 1$, then the rules have a good interpretation. Stochastic Gradient Descent (SGD), Batch Gradient Descent (BGD) and Levenberg-Marquadt (LM) optimization methods can be used for gradient updates. For SGD, a good initial value for the learning rate (lr) is 0.01.

A much smaller value initial value of lr can be chosen for BGD. In the case of LM optimization, the parameters are updated using the expression

$$P_{i+1} = P_i - \left(\left((\frac{\partial J}{\partial P_i})^2 + \mu \right)^{-1} \frac{\partial J}{\partial P_i} \right). \tag{7.21}$$

Here μ is the damping factor and its values are adaptive. Initially a large value of μ is chosen. Later its value is gradually decreased, as the value of the loss function decreases during the training phase. P_i is the updated value of tunable parameters at the ith training iteration. The tunable parameters in RBN are: 1) The parameters of the DF (ν and c); 2) The parameters of the WDO (weights); and 3) The consequent parameter (o). The tunable parameters are randomly initialized after setting proper values for the metaparameters.
Next the network is trained by passing a batch of training examples. Feed-forward calculations are performed and finally the loss J is calculated. The tunable parameters are updated by the optimization method. The process is repeated until we reach the desired number of epochs.

7.4 Algorithm

The whole training procedure is summarized in Algorithm 8 below.

Algorithm 8 Algorithm for Training of the RBNN

Step 1: Choose proper starting values for the metaparameters.
Step 2: Initialize the tunable parameters with random initial values.
Step 3: Select an optimization method with an appropriate learning rate.
Step 4: Feed the training data to the network in batches.
Step 5: Normalize the input feature vectors using Algorithm 7.
Step 6: Perform the feed-forward calculation using equations (7.3) to (7.6).
Step 7: Calculate the network loss using Eq. (7.7) or Eq. (7.18).
Step 8: Back propagate the loss and calculate the gradients using equations (7.8) to (7.20).
Step 9: Update the tunable parameters using the chosen optimization method and the calculated gradients.
Step 10: Repeat steps 5 to 9 until the required number of epochs are reached.
Step 11: Test the trained model using testing data and evaluate its performance.

7.5 Experiments, Results and Discussions

Next, the learning capability and performance of the proposed RBNN are demonstrated using various machine learning tasks. We will start with a simple XOR function and afterwards 3 regression and 4 classification tasks are presented.

7.5.1 Learning a XOR function

Approximating a XOR function is a classical learning problem. It is well known that a single neuron alone cannot approximate a XOR function. However, a single RBN unit can learn the XOR function. Here, the RBN unit is trained under two different parameter settings. Both cases successfully learnt the XOR function with a very low mean square error (MSE). The metaparameters in these configurations are shown in Table. 7.2. The RBN with a smaller number of rules produced a relatively higher value of the Mean Square Error (MSE). The loss curves for these two cases are shown in Fig. 7.3.

Config. No	Optimization	Rules	DF fixed Params	MSE
1	SGD (lr=.01)	6	$\lambda = 2, \varepsilon = 0.25$	9.71×10^{-15}
2	SGD (lr=.05)	4	$\lambda = 2, \varepsilon = 0.25,$ $\nu = 0.3$	1.251×10^{-6}

Table 7.2: *Metaparameter values of the two configurations of a single RBN unit for learning the XOR function.*

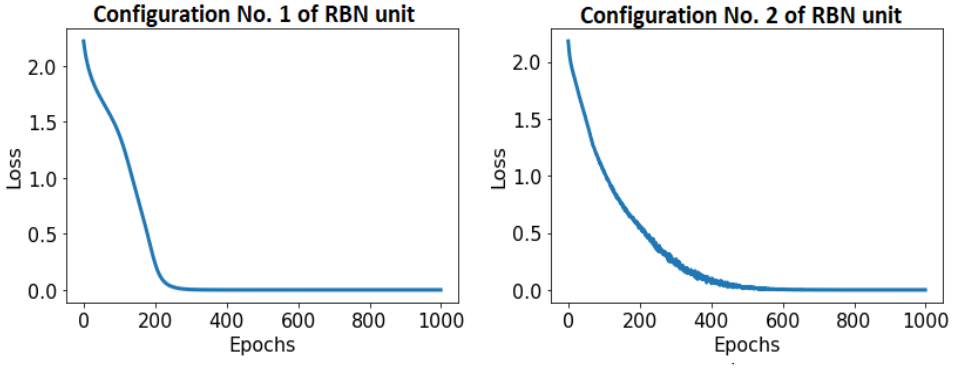

Figure 7.3: *Loss curves for learning the XOR function.*

7.5.2 California housing dataset

California housing is a real-world open source dataset available on StatLib repository [15]. It was compiled from the records of the U.S. census of 1990. Each row of the data contains the information about a single block group. The block group usually has a population between 6000 and 30000. The dataset consists of 9 features as shown in Fig. 7.4. The $median\_house\_value$ is the feature to be predicted. The dataset is not clean so preprocessing was done to remove the missing values and outliers. The total number of records in the cleaned dataset was 19369. The dataset was divided into three subsets for cross validation i.e. a training set with 14000 records, validation set with 3000 and test set with the remaining 2369 records. Fig. 7.5 shows the heat map indicating the correlations between the features. Categorical features were converted to numerical ones using the standard Python Pandas library. Various RBNN regression models were trained using different optimization methods to predict the housing values. Table 7.3 summarizes various configurations and the results obtained. Mean square error (MSE) and the R^2 coefficient were used as metrics to compare the performance of the various configurations. The R^2 coefficient determines how well the trained model fits the test data. It is clear from Table 7.3 that the RBNN (Configuration No. 2) trained using the SGD optimization and Ordered Normalization achieved the highest R^2 coefficeint. Fig. 7.6 shows the loss function (Eq. 7.7) plots for the RBNNs trained using the SGD and LM optimization methods. It is evident from these plots that if SGD is used for optimization, then the RBNNs can be trained faster. In this case, the trainable parameters settle around their final values within a few training epochs. Fig. 7.7 shows a comparison between the normalized housing price predicted by a trained RBNN (Configuration No. 3) and the actual housing price for the first 150 instances of the test dataset. It is clear that the RBNN predictions followed the actual housing price very closely and captured the pricing trend accurately.

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

Figure 7.4: *The header of the California housing dataset.*

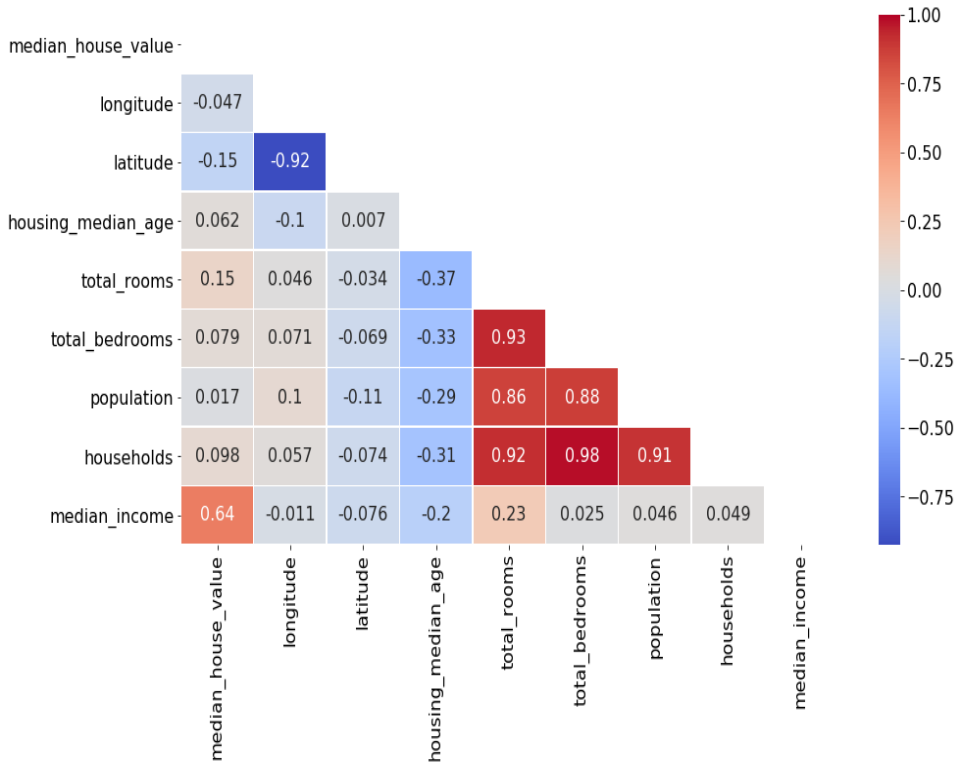

Figure 7.5: *A heat map of the features correlation of the California housing dataset.*

Config. No.	Optimizer	Normalization	No. of hidden RBN units	Rules per RBN	DF fixed Params	MSE	R^2 Coefficient
1	SGD (lr=.01)	MinMax Standard	2	3	$\lambda = 2 \quad \varepsilon = 0.2$	0.0153	0.626
2	SGD (lr=.01)	Ordered Normalization	2	3	$\lambda = 2 \quad \varepsilon = 0.25$	0.0274	0.676
3	SGD (lr=.01)	MinMax Standard	2	3	$\lambda = 2 \quad \varepsilon = 0.3 \; \nu = 0.3$	0.0166	0.596
4	LM ($\mu = 2000$)	MinMax Standard	3	3	$\lambda = 2 \quad \varepsilon = 0.3$	0.0185	0.5755
5	LM ($\mu = 2000$)	Ordered Normalization	3	3	$\lambda = 2 \quad \varepsilon = 0.25$	0.0298	0.642

Table 7.3: *Configurations and the performance of various RBNNs for the California housing dataset.*

7.5.3 Quadcopter Altitude Control

The model of an altitude controller of quadcopter can be learnt as a regression task. The simulation model is available in Matlab Simulink [20]. The model contains a PID

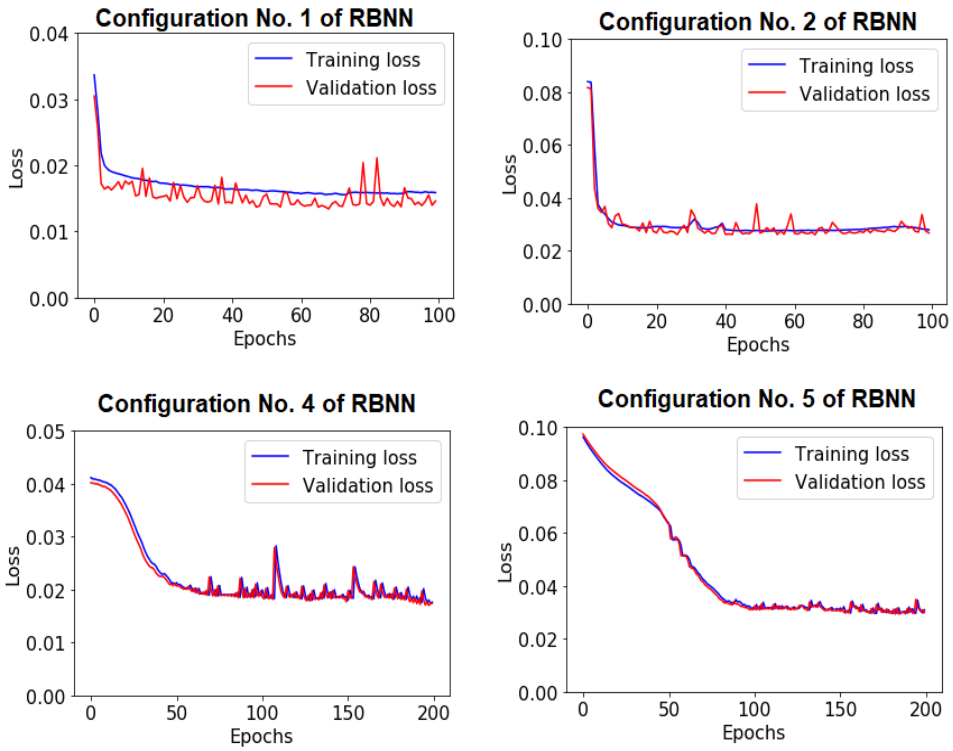

Figure 7.6: *Training and Validation loss curves of various RBNN models for the California housing price prediction.*

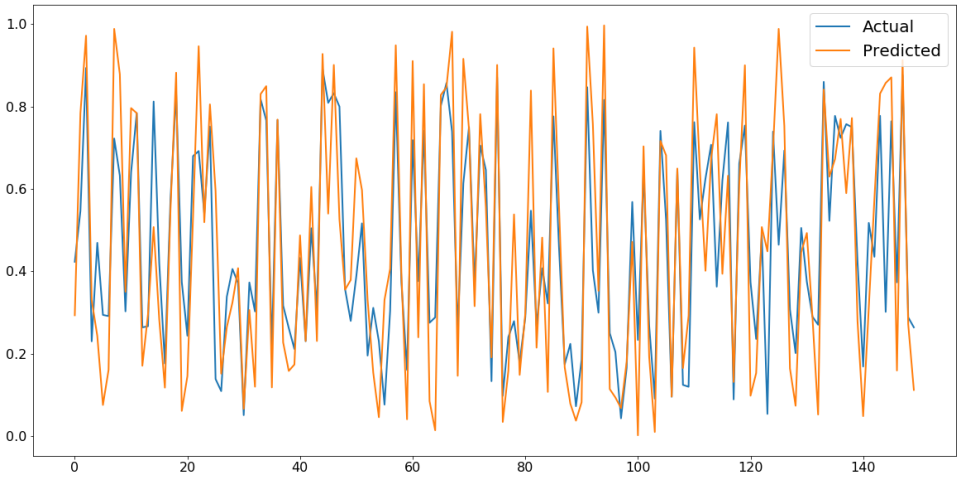

Figure 7.7: *A comparison of normalized housing prices predicted by the RBNN model (Configuration No. 3) and actual prices for 150 instances of the test dataset.*

controller which generates the altitude commands for the rotors to reach the desired altitude. The difference between the actual altitude of the quadcopter and the desired altitude is called the error signal. The derivative of this error signal is calculated

to produce an error rate signal. These two signals (error, error rate) are the inputs to the controller and the output is the control signal. The inputs are treated as features vectors and the control function is learnt using the RBNN. The dataset is generated for a scenario in which the quadcopter takes off and maintains different altitudes before landing. The dataset is biased as the transient (moving upward/downward) data is much less compared to the steady state (constant altitude). The dataset was augmented by adding more transient data. Table 7.4 shows the performance of various trained RBNN models, while Fig. 7.8 shows the training loss and validation loss curves for some of these models.

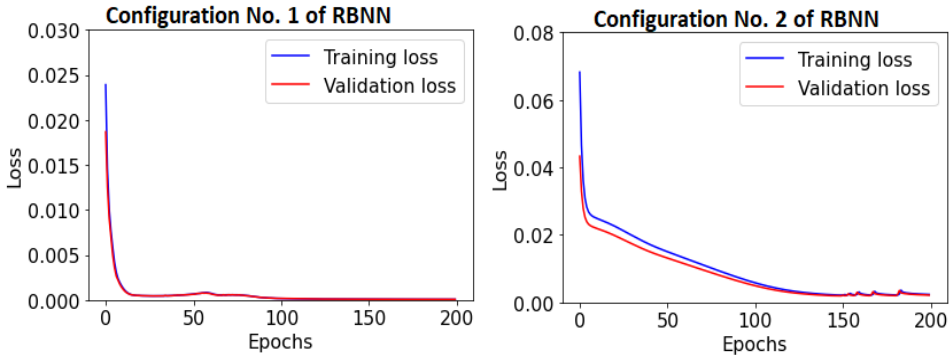

Figure 7.8: *Loss curves of quadcopter altitutde control RBNN models.*

Config. No.	Optimizer	Normalization	No. of hidden RBN units	Rules per RBN	DF fixed Params	MSE	R^2 Coefficient
1	SGD (lr=.001)	MinMax Standard	2	3	$\lambda = 2 \quad \varepsilon = 0.15$	4.81×10^{-5}	0.9982
2	BGD (lr=.0001)	MinMax Standard	4	6	$\lambda = 2 \quad \varepsilon = 0.15$	0.0022	0.9139
3	SGD (lr=.005)	MinMax Standard	2	2	$\lambda = 2 \quad \varepsilon = 0.15, \nu = 0.3$	7.89×10^{-5}	0.9972
4	LM (μ=2000)	Ordered Normalization	3	3	$\lambda = 2 \quad \varepsilon = 0.25$	0.0123	0.8491

Table 7.4: *Configurations and the performance of various RBNN models for the quadcopter altitude regression task.*

Model interpretations

We can examine one of the trained RBNN models in Table 7.4 and interpret its results. The structure of the RBNN in Configuration No. 3 is shown in Fig. 7.9. It has 1 hidden

layer with 2 RBN units. Each unit has 2 rules. After training, the DFs in RBN N_1 are shown in Fig. 7.10. Using the values of trained parameters o, the two rules learnt

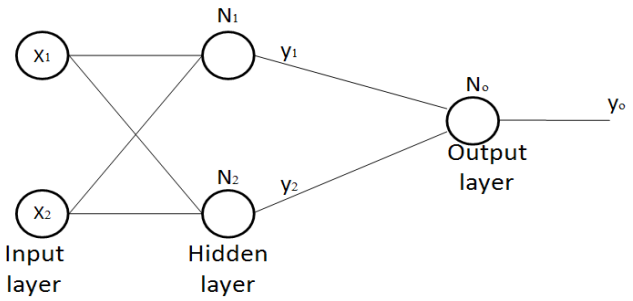

Figure 7.9: *Structure of the RBNN in Configuration No.*3 *of Table 7.4.*

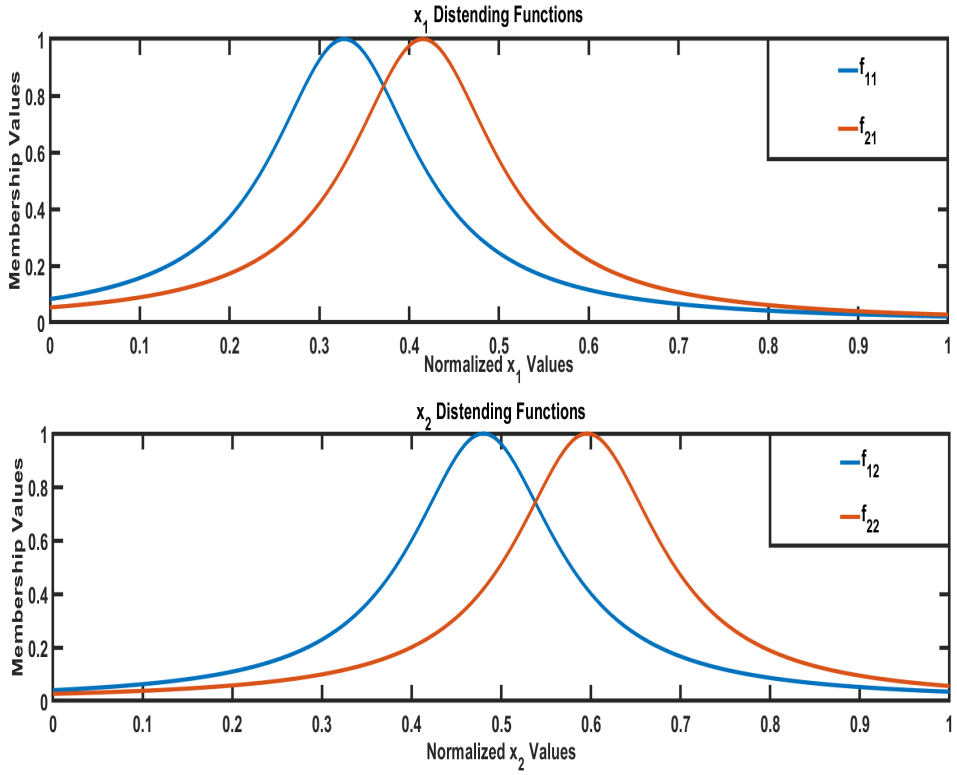

Figure 7.10: *Distending functions learnt by N*1.

by N_1 are

$$\begin{aligned} &\text{Rule } 1\text{: if } x_1 \text{ is } f_{11} \text{ and } x_2 \text{ is } f_{12} \text{ then } y_1 \text{ is } 0.840. \\ &\text{Rule } 2\text{: if } x_1 \text{ is } f_{21} \text{ and } x_2 \text{ is } f_{22} \text{ then } y_1 \text{ is } 0.187. \end{aligned} \tag{7.22}$$

The output y_1 is given by

$$y_1 = \frac{u_1(0.84) + u_2(.187)}{u_1 + u_2}$$

If the numerical values of input x_1 and x_2 are given, then u_1 and u_2 can be calculated (Eq. (7.3)) and we get y_1. In a similar way, the DFs of N_2 can be plotted as shown in Fig. 7.11. The rules learnt by N_2 are

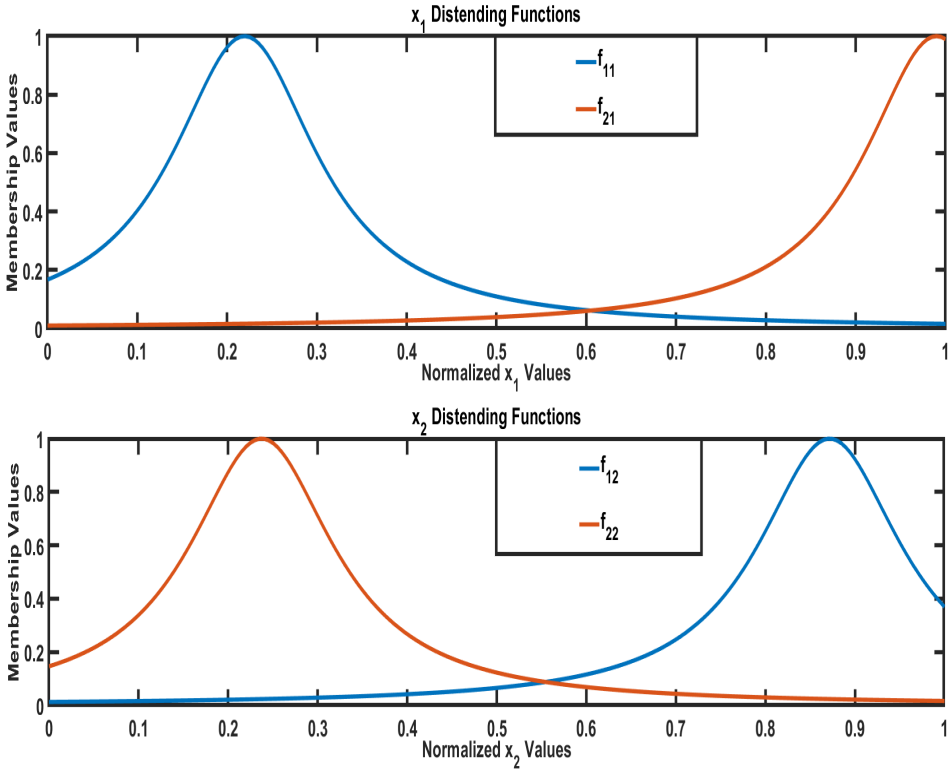

Figure 7.11: *Distending functions learnt by N2.*

Rule 1: if x_1 is f_{11} and x_2 is f_{12} then y_2 is 0.999.
Rule 2: if x_1 is f_{21} and x_2 is f_{22} then y_2 is 0.322. (7.23)

and the output y_2 is

$$y_2 = \frac{u_1(0.999) + u_2(0.322)}{u_1 + u_2}$$

Now the outputs y_1 and y_2 are treated as metafeatures and are inputs to N_o. The DFs of N_o are shown in Fig. 7.12 and the rules based on metafeatures are

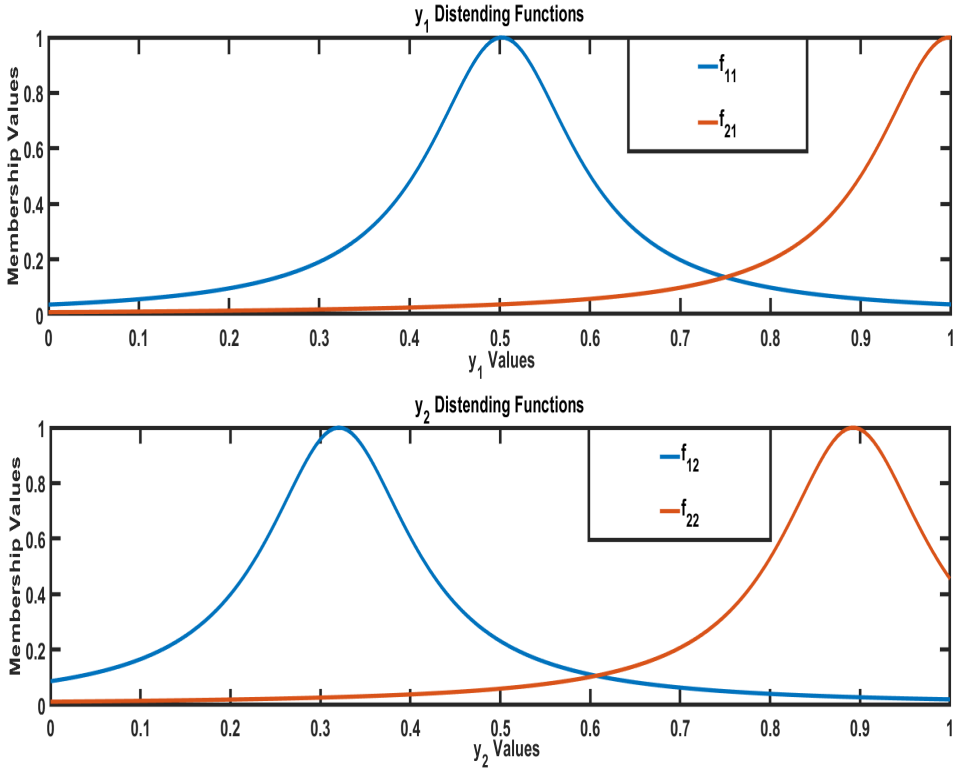

Figure 7.12: *Distending functions learnt by No.*

$$\begin{aligned} &\text{Rule } 1\text{: if } y_1 \text{ is } f_{11} \text{ and } y_2 \text{ is } f_{12} \text{ then } y_o \text{ is } 0.999. \\ &\text{Rule } 2\text{: if } y_1 \text{ is } f_{21} \text{ and } y_2 \text{ is } f_{22} \text{ then } y_o \text{ is } -.361, \end{aligned} \tag{7.24}$$

and the final output of the network is

$$y_o = \frac{u_1(0.999) + u_2(-.361)}{u_1 + u_2}$$

The rules in Eq. (7.22), Eq. (7.23) and Eq. (7.24) collectively provide an interpretation of the results. For various values of x_1 and x_2, the output y_o is calculated and the decision surface can be obtained as shown in Fig. 7.13. This gives a visual insight into the trained RBNN model.

7.5.4 Census income classification dataset

The Census income dataset (also known as Adult dataset) was extracted from 1994 US State Census Bureau data. It is available on the UCI ML repository [66]. It has 14 features which contain numerical, ordinal and and categorical data types. The features are age, workclass, final weight, education, education number of years, marital status, occupation, relationship, race, sex, capital gain, capital loss, hours per

week and native country. Based on income per year, each sample belongs to one of the two classes i.e. $< 50k$ and $> 50k$. Most of the samples (approximately 75%) belong to the $< 50k$ class and it is called the majority class. $> 50k$ is called the minority class. It is a large dataset with 46443 samples. The dataset is divided into three subsets for cross validation i.e. training set with 32000 samples, a validation set with 7000 and test set with the remaining 7443 samples. An RBNN is trained to correctly identify the classes from the feature vectors. As the dataset is skewed toward the majority class, the classification accuracy is not an appropriate metrics for judging the performance of a classifier. We will compare the performance of various classifier based on the $F1$ score for the minority class. Table 7.5 summarizes the performance and metaparameter values for various RBNN configurations. It can be seen that the RBNN trained using SGD with a learning rate of $.005$ achieved the highest accuracy and $F1$ scores among all the various configurations. It is worth mentioning that the RBNN trained using BGD was not able to correctly classify even a single sample from $< 50k$ class and this is evident from the $F1$ score for $< 50k$ class. Although it showed an accuracy of 75% on test dataset, this is due to the fact that the dataset was skewed towards the $> 50k$ class (75% samples). A fully connected DNN (layers detailed in Fig. 7.14) was trained for comparison purposes. It achieved an accuracy of 83.63% on test dataset and an $F1$ score for the $< 50k$ class. The performance of the relatively small RBNN (Config. No. 2) is comparatively better

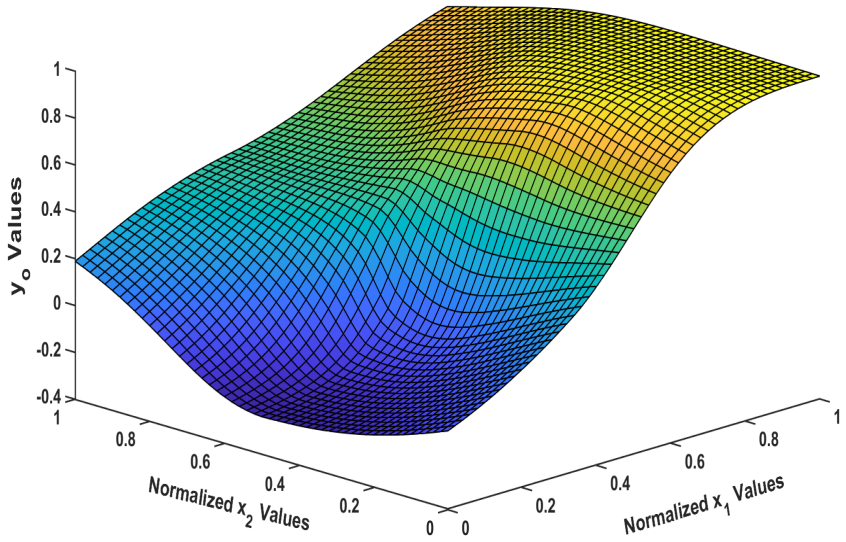

Figure 7.13: *A surface plot that allows us to interpret the relationship between the inputs and output of the RBNN for the quadcopter altitude control task.*

Config. No.	Optimizer	No. of hidden layers	No. of hidden RBN units	Rules per RBN	DF fixed Params	Accuracy on test dataset	$F1$ score for $> 50k$ (minority) class
1	SGD (lr=.01)	1	3	5	$\lambda = 2,\ \varepsilon = 0.25$	84.33 %	61 %
2	SGD (lr=.005)	1	3	2	$\lambda = 2,\ \varepsilon = 0.25,$ $\nu = 0.3$	85%	66%
3	BGD (lr=.005)	1	3	3	$\lambda = 2,\ \varepsilon = 0.25$	75 %	0 %
4	LM (μ = 300)	2	3 +3 = 6	3	$\lambda = 2,\ \varepsilon = 0.15$	83 %	58 %

Table 7.5: *Configurations and performance of various RBNNs for the census income classification dataset.*

than the DNN. This RBNN has only a few trainable parameters (202) compared with the DNN (4802). Fig. 7.16 shows the loss curves for various RBNN configurations and the trained DNN. The loss curve of RBNN trained using the LM optimization converges slowly to the final value. The loss curves of all other RBNN configurations and DNN converges within a few epochs and therefore their training process is much faster. Confusion matrices (an error matrix used to describe the performance of a classification network) of the RBNN (Config. No. 1) and DNN are shown in Fig. 7.15.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 32)                448
_________________________________________________________________
dense_2 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_3 (Dense)              (None, 32)                1056
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                2112
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 130
=================================================================
Total params: 4,802
Trainable params: 4,802
Non-trainable params: 0

Accuracy of the training dataset 84.109375
Accuracy of the test dataset 83.63563079403467
```

Figure 7.14: *Layers configuration of the DNN. The DNN achieved an accuracy of 83.63 % on the census income test dataset.*

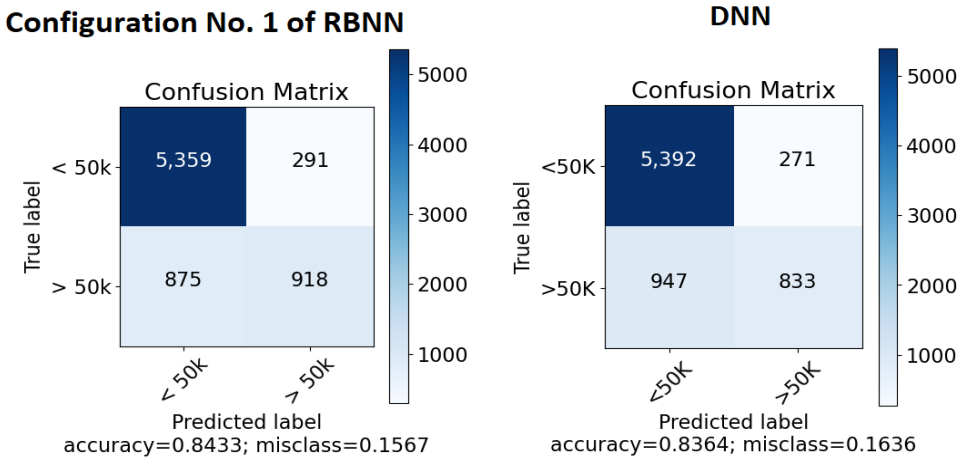

Figure 7.15: *Confusion matrices of Configuration No. 1 of a RBNN (left) and DNN (right) for the census income classification dataset.*

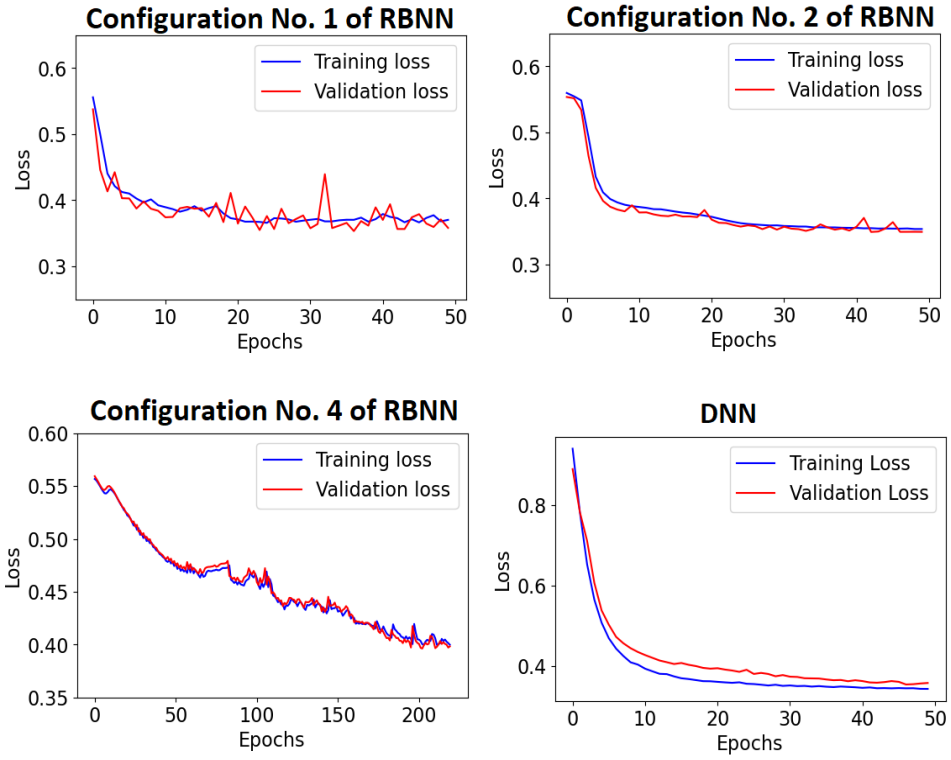

Figure 7.16: *Loss curves of various RBNNs and DNN trained on the census income classification dataset.*

Impact of feature noise

To find the classification accuracy for noisy features, various amounts of Gaussian noise were added to the test data. Table 7.6 shows the performance reduction of the

RBNN and DNN in the presence of feature noise. It is evident that the classification performance of the DNN has been degraded by noise. As a comparison, the RBNN accuracy also decreases but it is still better than the DNN in the presence of feature noise. Among various RBNNs, those which are trained using SGD are more robust to noise. As the dataset is skewed towards the majority class, the $F1$ score of the minority ($< 50k$) class is the crucial metric for judging the performance of various classifiers. It is clear from Table 7.6 that when the noise magnitude increased by two folds, the $F1$ score significantly fell (16% decreased) in the case of the DNN. In comparison, the $F1$ score fell by 7% for the RBNN. Fig. 7.17 shows the confusion matrices of the RBNN and DNN for a noisy feature test set.

S. No.	Noise Magnitude	DNN Test Dataset Accuracy	RBNN (Config. no. 1) Test Dataset Accuracy	DNN $F1$ score for $> 50k$ (minority) class	RBNN $F1$ score for $> 50k$ (minority) class
1	$\left[-\frac{\sigma}{2}, \frac{\sigma}{2}\right]$	80.79 %	82.57 %	52 %	59 %
2	$[-\sigma, \sigma]$	75.31 %	78.45 %	44 %	55 %

Table 7.6: *Effect of a noisy test set on the prediction accuracy of the DNN and RBNN on census income classification dataset. σ is the standard deviation of the feature vector.*

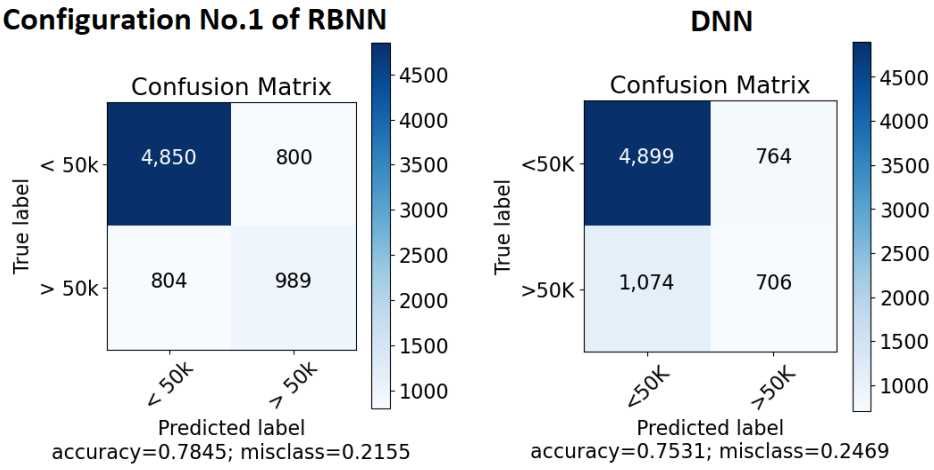

Figure 7.17: *Confusion Matrices of Configuration No. 1 of the RBNN (left) and DNN (right) in the presence of feature noise in the interval $[-\sigma, \sigma]$ (census income classification dataset).*

7.5.5 Occupancy detection dataset

The occupancy detection dataset is available on the UCI machine learning repository [16]. The dataset provides accurate information regarding the occupancy of an of-

fice room. It has 7 features and two output classes i.e. empty and occupied. It is an unbalanced dataset with 71% empty and 29% occupied rooms. The features include date-time, Temperature (Celsius), Humidity, Light (Lux), CO2 (ppm), Humidity Ratio (Derived using the temperature and humidity) and Occupancy. The features correlation heatmap is shown in Fig. 7.18. It shows that the occupancy is highly correlated with the light intensity in the room. The database comprises 20560 records. It was divided into three parts for cross validation i.e. a training set (13000), validation set (2500) and test set (5060). Various configurations of RBNNs were trained to correctly predict the room occupancy status. Table 7.7 summarizes the various configurations along with their classification accuracy scores. It is clear that the RBNN Configuration No. 2 achieved the highest accuracy on the test dataset and $F1$ score for the $occupied$ class. A fully connected DNN was also trained for this task. Its layers, structure and parameters are listed in Fig. 7.20. The accuracy and $F1$ scores achieved by this DNN on the test set are 98.91% and 98% respectively. The DNN was trained for 100 epochs with a batch size of 500. Fig. 7.19 shows the confusion matrices of this DNN and RBNN on the test dataset. Configuration No. 2 of RBNN achieved the same accuracy as the DNN did but with far fewer parameters. This RBNN had only 262 trainable parameters compared with the DNN which had 7810. Fig. 7.21 shows the training and validation loss curves of the DNN and various RBNNs. it is clear from these curves that the DNN and RBNNs (using the SGD and the BGD) can be trained with a few epochs. However, training the RBNN using the LM optimization is slow and requires a comparatively large number of epochs.

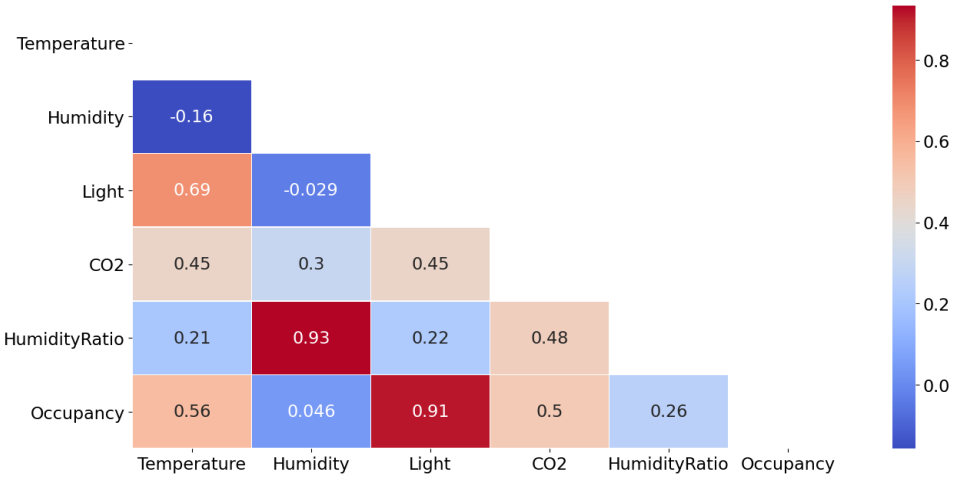

Figure 7.18: *Heat map of the occupancy detection dataset features.*

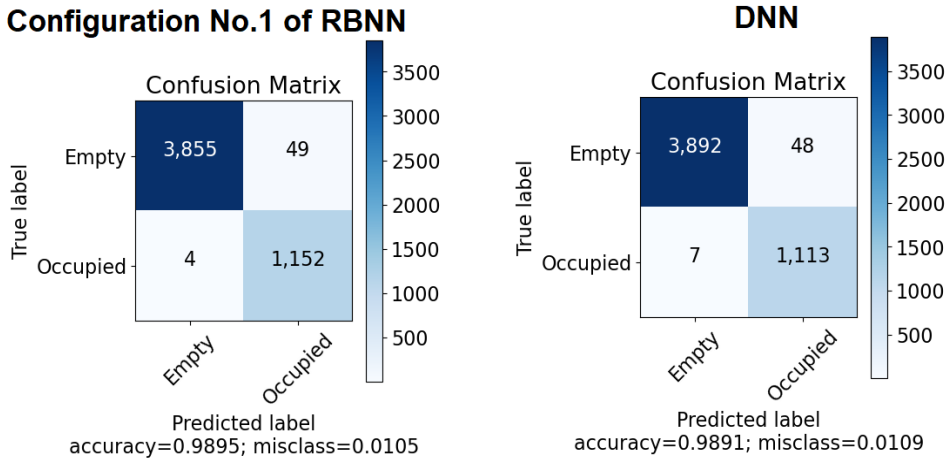

Figure 7.19: *Confusion matrices of Configuration No.* 1 *of the RBNN (left) and the DNN (right) for the occupancy detection dataset.*

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 64)                448
_________________________________________________________________
dense_2 (Dense)              (None, 64)                4160
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_4 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 66
=================================================================
Total params: 7,810
Trainable params: 7,810
Non-trainable params: 0

Accuracy of the training dataset 99.07692307692308
Accuracy of the test dataset 98.91233201581027
```

Figure 7.20: *Layers configuration of the DNN trained on the occupancy detection dataset.*

Impact of feature noise

Random Gaussian noise was added to the test dataset with various maximum magnitudes. The amount of the added noises was dependent on the standard deviation of the feature vectors. The noise robustness of the trained DNN and RBNNs (in Table 7.7) was investigated by making predictions on the noisy dataset. Table 7.8 shows the performance degradation in the classification accuracy for these trained networks. The best robustness was obtained when the RBNN was trained using the SGD. As the dataset is skewed, the $F1$ score for the minority ($occupied$) class is a more significant metric than classification accuracy in determining the performance of the

Config. No.	Optimizer	No. of hidden RBN units	Rules per RBN	DF fixed Parameters	Total trainable parameters	Accuracy on test dataset	$F1$ score for *occupied* class
1	SGD (lr=.001)	3	5	$\lambda = 2,\ \nu = 0.3$	430	98.95 %	98 %
2	SGD (lr=.001)	3	5	$\lambda = 2,\ \varepsilon = 0.2,$ $\nu = 0.3$	295	98.97 %	98 %
3	BGD (lr=.0001)	3	3	$\lambda = 2$	339	98.65 %	97 %
4	LM ($\mu = 1000$)	3	3	$\lambda = 2$	339	98.24 %	96 %

Table 7.7: *Configurations and performance of various RBNNs for the occupancy detection classification task. All the configurations have one hidden layer.*

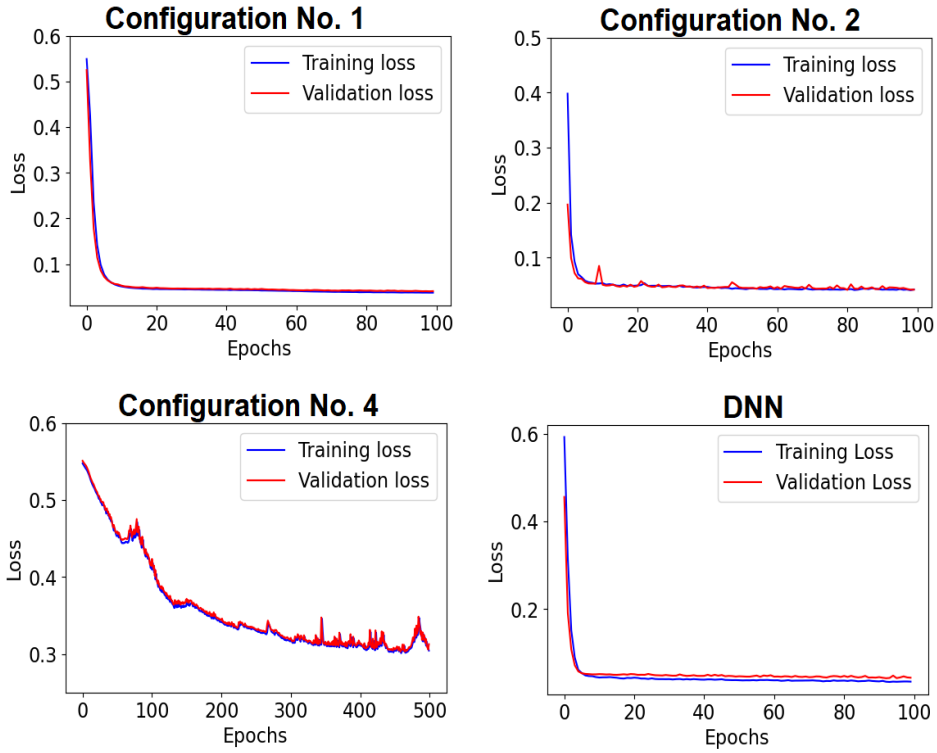

Figure 7.21: *Loss curves of the RBNNs and DNN for the occupancy detection classification task.*

training algorithm. In the case of a noisy test dataset, the DNN prediction accuracy and $F1$ scores are less compared with the RBNN's. Fig. 7.22 shows the confusion matrices of the DNN and RBNN obtained using the noise-corrupted dataset. From Table 7.8, it is clear that a three folds increase in the noise reduced the $F1$ score of

the DNN by 18%. However, the $F1$ score for *occupied* decreased by 10% in the case of RBNN. This shows that the RBNN is comparatively more robust to feature noise.

S. No.	Noise Magnitude	DNN Test Dataset Accuracy	RBNN (Config. no. 2) Test Dataset Accuracy	DNN $F1$ score for the *occupied* class	RBNN $F1$ score for the *occupied* class
1	$\left[-\frac{\sigma}{3}, \frac{\sigma}{3}\right]$	98.69 %	98.79 %	97 %	97 %
2	$\left[-\frac{\sigma}{2}, \frac{\sigma}{2}\right]$	97.35 %	98.03 %	94 %	96 %
3	$[-\sigma, \sigma]$	90.98 %	94.42 %	80 %	88 %

Table 7.8: *The effect of the noisy features test dataset on the prediction accuracy and $F1$ scores of the DNN and RBNN (occupancy detection dataset). σ is the standard deviation of the feature vector.*

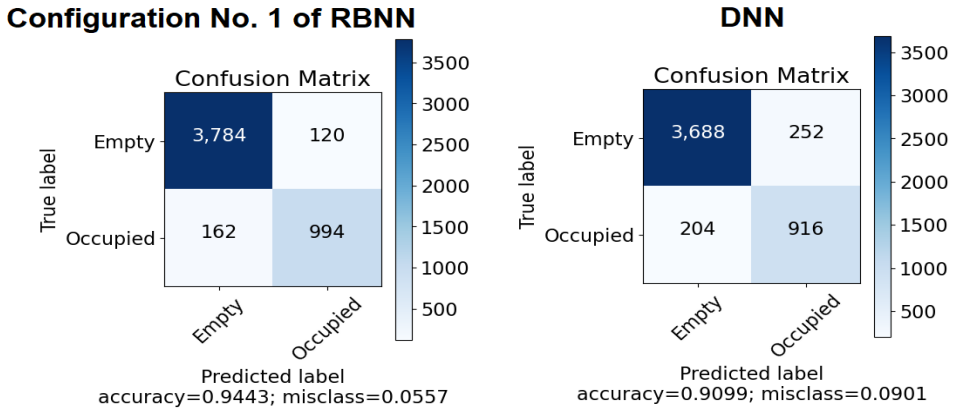

Figure 7.22: *Confusion matrices for Configuration No. 1 of the RBNN (left) and DNN (right) for a noise corrupted test dataset (occupancy detection dataset).*

7.5.6 Discussion

From the findings, it is evident that RBNNs can solve various real-life regression and classification problems. All these tasks were solved using a few hidden layers (usually 1 or 2), with a few neurons (from 3 to 6) and a minimum number of rules (2 to 5). As shown in Table 7.7, the total number of trainable parameters in the RBNN is small compared with the DNN. Most of the problems presented and solved in the simulation section include real-world datasets. Therefore, in general the RBNN can be used to solve real-world problems with high accuracy.

The experiments results indicate that most configurations of the RBNN achieve a higher accuracy on the test dataset compared with that for the DNN. Similar to DNNs, the parameters of RBNNs can be updated using back propagation in combination

with SGD, BGD and LM optimization methods. In tables 7.3 to 7.7, it can be seen that RBNNs give a higher accuracy on the test dataset when trained using SG. In Fig. 7.16 and Fig. 7.21, it is clear that training using the LM optimization method is much slower compared with SGD and BGD. From Table 7.4 and Table 7.6, it may be inferred that better results are obtained if Ordered Normalization is used for input normalization.
RBNN is more robust against feature noise and it gives a reasonably high accuracy even when the input features are corrupted by noise. This can be observed from the accuracy columns of Table 7.5 and Table 7.7. In comparison, the DNN performance on the test dataset is adversely affected and its accuracy falls if the feature noise increases. Those RBNN configurations that are trained using the SGD are more robust to feature noise and give a higher prediction accuracy on noisy test sets.
For skewed datasets, RBNN produces a higher $F1$ score for the smaller classes as shown in Table 7.5 and Table 7.7. Furthermore, if the test dataset is corrupted by feature noise then RBNN still maintains these $F1$ scores for the smaller classes. This is evident from the results shown in Table 7.6 and Table 7.8. In comparison, the DNN has a lower $F1$ score if the feature noise increases. This shows that the RBNN is more robust to noise if the dataset is skewed. This unique feature of the RBNN makes it a better option for training on real-world sets, as most of those are unbalanced (skewed).
It is interesting to note that the trained RBNN models are easy to interpret. DFs and trained rules can be extracted from the neurons. Using these rules, it is easy to interpret the function/behavior of each neuron and layer. A surface can be plotted between the input features and output of the RBNN. Therefore we can understand the relationship between the various features and the output. The number of the trainable parameters can be further reduced by optimizing only the c parameter of the DF (the other three parameters now remain fixed). Consequently, there is a negligible decrease in the accuracy. And we can say that there is a compromise between interpretablility and the learning capacity. By increasing the number of neurons, layers and number of rules, the learning capacity of the RBNN increases. However, the interpretability of a large RBNN becomes increasingly difficult.

7.6 Summary

In this chapter, we presented a new type of learning network called a rule-based neural network (RBNN). It consists of rule-based neurons (RBNs), arranged in layers and the layers are stacked together to form a fully connected network. The input layer normalizes the input feature values. We introduced a new type of normalization called Ordered Normalization (ON) and it is helpful in achieving higher training accuracy. Each RBN consists of parametric rules and the parameters of these rules

are updated using the Stochastic Gradient Descent (SGD), Batch Gradient Descent (BGD) or Levenberg-Marquadt (LM) optimization method. The rules are evaluated using arithmetic calculations (in the feed-forward step). We presented the algorithm for the training of the RBNN for solving regression and classification problems. The number of trainable parameters are few compared with a DNN. We got high prediction accuracy scores for the regression and classification tasks. The RBNN is robust to noisy feature and provides an excellent accuracy even in the case of a noise-corrupted test dataset. The RBNN is a good choice for solving problems with skewed datasets by producing high $F1$ scores for the smaller classes. In contrast to the DNN, RBNN-trained models are more interpretable. The rules learnt by each RBN can be extracted and the output of the RBN is then evaluated using simple arithmetic calculations. The relationship between the input features and the RBNN output can be determined and visualized. However, the RBNN can only be applied to learning tasks with a small or moderate number of feature sets. Because if the number of inputs increases then the rules become more complex. Although the accuracy remains high, the interpretation of the trained RBNN model becomes increasingly difficult. The work can be extended by incorporating continuous logic and more versatile rules to develop various interpretable AI models.

Bibliography

[1] Davood Nazari Maryam Abadi and Mohammad Hassan Khooban. Design of optimal mamdani-type fuzzy controller for nonholonomic wheeled mobile robots. *Journal of King Saud University-Engineering Sciences*, 27(1):92–100, 2015.

[2] Muhammad Akram and Wieslaw A Dudek. Regular bipolar fuzzy graphs. *Neural Computing and Applications*, 21(1):197–205, 2012.

[3] Muhammad Akram and Anam Luqman. Certain concepts of bipolar fuzzy directed hypergraphs. *Mathematics*, 5(1):17, 2017.

[4] Muhammad Akram and Saira Nawaz. Operations on soft graphs. *Fuzzy information and Engineering*, 7(4):423–449, 2015.

[5] Plamen P Angelov and Dimitar P Filev. An approach to online identification of takagi-sugeno fuzzy models. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):484–498, 2004.

[6] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[7] K Atanassov. Intuitionistic fuzzy sets. *Fuzzy Sets and Systems*, 20(1):87–96, 1986.

[8] Krassimir T Atanassov. *On intuitionistic fuzzy sets theory*, volume 283. Springer, 2012.

[9] George S Atsalakis, Ioanna G Atsalaki, Fotios Pasiouras, and Constantin Zopounidis. Bitcoin price forecasting with neuro-fuzzy techniques. *European Journal of Operational Research*, 276(2):770–780, 2019.

[10] Richard Bellman and Magnus Giertz. On the analytic formalism of the theory of fuzzy sets. *Information sciences*, 5:149–156, 1973.

[11] Richard Bellman, Robert Kalaba, and Lotfi A Zadeh. Abstraction and pattern classification. *Journal of Mathematical Analysis and Applications*, 13(1):1–7, 1966.

[12] Richard E Bellman and Lotfi A Zadeh. Decision-making in a fuzzy environment. *Management science*, 17(4):B–141, 1970.

[13] Richard E Bellman and Lotfi A Zadeh. Local and fuzzy logics. In *Modern uses of multiple-valued logic*, pages 103–165. Springer, 1977.

[14] Prabir Bhattacharya. Some remarks on fuzzy graphs. *Pattern recognition letters*, 6(5):297–302, 1987.

[15] Nehal Birla. Vehicle dataset from cardekho. `https://www.kaggle.com/nehalbirla/vehicle-dataset-from-cardekho`. Accessed 2020-12-21.

[16] Luis Candanedo. Occupancy detection dataset. `http://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+`. Accessed 2021-06-27.

[17] Shu Cao, Neville W Rees, and Gang Feng. Analysis and design of fuzzy control systems using dynamic fuzzy-state space models. *IEEE Transactions on Fuzzy Systems*, 7(2):192–200, 1999.

[18] Rudolf Carnap. Foundations of logic and mathematics. *Bull. Amer. Math. Soc*, 45:821–822, 1939.

[19] Oscar Castillo and Patricia Melin. A review on interval type-2 fuzzy logic applications in intelligent control. *Information Sciences*, 279:615–631, 2014.

[20] Matlab Help Center. Matlab quadcopter project. `https://www.mathworks.com/help/aeroblks/quadcopter-project.html`. Accessed 2020-12-22.

[21] Na Chen, Zeshui Xu, and Meimei Xia. Interval-valued hesitant preference relations and their applications to group decision making. *Knowledge-based systems*, 37:528–540, 2013.

[22] Siu Kai Choy, Shu Yan Lam, Kwok Wai Yu, Wing Yan Lee, and King Tai Leung. Fuzzy model-based clustering and its application in image segmentation. *Pattern Recognition*, 68:141–157, 2017.

[23] Rangan Das, Sagnik Sen, and Ujjwal Maulik. A survey on fuzzy deep neural networks. *ACM Computing Surveys (CSUR)*, 53(3):1–25, 2020.

[24] Supriya Kumar De, Ranjit Biswas, and Akhil Ranjan Roy. An application of intuitionistic fuzzy sets in medical diagnosis. *Fuzzy sets and Systems*, 117(2):209–213, 2001.

[25] André de Souza Mendes. Vehicle lateral dynamics. `https://www.mathworks.com/matlabcentral/fileexchange/58683-vehicle-dynamics-lateral?s_tid=FX_rc2_behav`.

[26] Yue Deng, Zhiquan Ren, Youyong Kong, Feng Bao, and Qionghai Dai. A hierarchical fused fuzzy deep neural network for data classification. *IEEE Transactions on Fuzzy Systems*, 25(4):1006–1012, 2016.

[27] Samuel Dodge and Lina Karam. Understanding how image quality affects deep neural networks. In *2016 eighth international conference on quality of multimedia experience (QoMEX)*, pages 1–6. IEEE, 2016.

[28] József Dombi. A general class of fuzzy operators, the demorgan class of fuzzy operators and fuzziness measures induced by fuzzy operators. *Fuzzy sets and systems*, 8(2):149–163, 1982.

[29] József Dombi. A general class of fuzzy operators, the demorgan class of fuzzy operators and fuzziness measures induced by fuzzy operators. *Fuzzy sets and systems*, 8(2):149–163, 1982.

[30] József Dombi. A general class of fuzzy operators, the demorgan class of fuzzy operators and fuzziness measures induced by fuzzy operators. *Fuzzy sets and systems*, 8(2):149–163, 1982.

[31] József Dombi. Towards a general class of operators for fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 16(2):477–484, 2008.

[32] József Dombi. Towards a general class of operators for fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 16(2):477–484, 2008.

[33] József Dombi. *The Generalized Dombi Operator Family and the Multiplicative Utility Function*, pages 115–131. Springer Berlin Heidelberg, 2009.

[34] Jozsef Dombi. Demorgan systems with an infinitely many negations in the strict monotone operator case. *Information Sciences*, 181(8):1440–1453, 2011.

[35] József Dombi and Abrar Hussain. Interval type-2 fuzzy control using distending function. In *Fuzzy Systems and Data Mining V: Proceedings of FSDM 2019*, pages 705–714. IOS Press, 2019.

[36] József Dombi and Abrar Hussain. A new approach to fuzzy control using the distending function. *Journal of Process Control*, 86:16–29, 2020.

[37] Jozsef Dombi and Tamas Szepe. Arithmetic-based fuzzy control. *Iranian Journal of Fuzzy Systems*, 14(4):51–66, 2017.

[38] Leandro dos Santos Coelho and Bruno Meirelles Herrera. Fuzzy identification based on a chaotic particle swarm optimization approach applied to a nonlinear yo-yo motion system. *IEEE Transactions on Industrial Electronics*, 54(6):3234–3245, 2007.

[39] Dimiter Driankov and Rainer Palm. *Advances in fuzzy control*, volume 16. Physica, 2013.

[40] Didier Dubois and Henri Prade. Operations on fuzzy numbers. *International Journal of systems science*, 9(6):613–626, 1978.

[41] Didier Dubois and Henri Prade. New results about properties and semantics of fuzzy set-theoretic operators. In *Fuzzy sets*, pages 59–75. Springer, 1980.

[42] Didier Dubois and Henri Prade. The mean value of a fuzzy number. *Fuzzy sets and systems*, 24(3):279–300, 1987.

[43] Didier Dubois and Henri Prade. Rough fuzzy sets and fuzzy rough sets. *International Journal of General System*, 17(2-3):191–209, 1990.

[44] Didier J Dubois. *Fuzzy sets and systems: theory and applications*, volume 144. Academic press, 1980.

[45] Chaimae El Hatri and Jaouad Boumhidi. Fuzzy deep learning based urban traffic incident detection. *Cognitive Systems Research*, 50:206–213, 2018.

[46] János C Fodor. Contrapositive symmetry of fuzzy implications. *Fuzzy Sets and Systems*, 69(2):141–156, 1995.

[47] Maurice J Frank. On the simultaneous associativity off (x, y) andx+ y- f (x, y). *Aequationes mathematicae*, 19(1):194–226, 1979.

[48] A Nagoor Gani and K Radha. On regular fuzzy graphs. 2008.

[49] A Nagoor Gani and K Radha. The degree of a vertex in some fuzzy graphs. *Int. J. Algorithms Comput. Math*, 2(3):107–116, 2009.

[50] Qing Gao, Xiao Zeng, Gang Feng, Yong Wang, and Jianbin Qiu. Ts fuzzy model-based approximation and controller design for general nonlinear systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1143–1154, 2012.

[51] Fernando Gaxiola, Patricia Melin, Fevrier Valdez, and Oscar Castillo. Interval type-2 fuzzy weight adjustment for backpropagation neural networks with application in time series prediction. *Information Sciences*, 260:1–14, 2014.

[52] Joseph A Goguen. The logic of inexact concepts. *Synthese*, pages 325–373, 1969.

[53] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[54] Xiaowei Gu. Multilayer ensemble evolving fuzzy inference system. *IEEE Transactions on Fuzzy Systems*, 29(8):2425–2431, 2021.

[55] Hami Hagras. Type-2 flcs: A new generation of fuzzy controllers. *IEEE Computational Intelligence Magazine*, 2(1):30–43, 2007.

[56] Horst Hamacher. *Über logische Aggregationen nicht-binär explizierter Entscheidungskriterien: Ein axiomat. Beitr. zur normativen Entscheidungstheorie*. Fischer, 1978.

[57] Mathworks Matlab hardware team. Parrot Drone Support from MATLAB, howpublished = "`https://www.mathworks.com/hardware-support/parrot-drone-matlab.html`", note = "accessed: 2020-03-11".

[58] John Hedengren. Simulink cstr simulation and control. `https://www.mathworks.com/matlabcentral/fileexchange/48018-simulink-cstr-simulation-and-control`. Accessed: 2019-09-15.

[59] Hans Hellendoorn and Dimiter Driankov. *Fuzzy model identification: selected approaches*. Springer Science & Business Media, 2012.

[60] Masahiro Inuiguchi. Two generalizations of rough sets and their fundamental properties. In *Proceedings of 6th Workshop on Uncertainty Processing*, pages 24–27, 2003.

[61] Nilesh N Karnik and Jerry M Mendel. Type-2 fuzzy logic systems: type-reduction. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, volume 2, pages 2046–2051. Ieee, 1998.

[62] Nilesh N Karnik and Jerry M Mendel. Operations on type-2 fuzzy sets. *Fuzzy sets and systems*, 122(2):327–348, 2001.

[63] Arnold Kaufmann. Introduction à la théorie des sous-ensembles flous à l'usage des ingénieurs (fuzzy sets theory). 1975.

[64] George J Klir and Tina A Folger. *Fuzzy sets, uncertainty, and information*. Prentice-Hall, Inc., 1987.

[65] LászlóT Kóczy and Kaoru Hirota. Ordering, distance and closeness of fuzzy sets. *Fuzzy sets and systems*, 59(3):281–293, 1993.

[66] Ronny Kohavi and Barry Becker. Census income dataset. `https://archive.ics.uci.edu/ml/datasets/adult`. Accessed 2021-05-30.

[67] Tien-Loc Le, Nguyen Vu Quynh, Ngo Kim Long, and Sung Kyung Hong. Multilayer interval type-2 fuzzy controller design for quadcopter unmanned aerial vehicles using jaya algorithm. *IEEE Access*, 8:181246–181257, 2020.

[68] Chaoshun Li, Jianzhong Zhou, Li Chang, Zhengjun Huang, and Yongchuan Zhang. T–s fuzzy model identification based on a novel hyperplane-shaped membership function. *IEEE Transactions on Fuzzy Systems*, 25(5):1364–1370, 2017.

[69] Mingxiao Li, Feng Lu, Hengcai Zhang, and Jie Chen. Predicting future locations of moving objects with deep fuzzy-lstm networks. *Transportmetrica A: Transport Science*, 16(1):119–136, 2020.

[70] Yongming Li, Shuai Sui, and Shaocheng Tong. Adaptive fuzzy control design for stochastic nonlinear switched systems with arbitrary switchings and unmodeled dynamics. *IEEE transactions on cybernetics*, 47(2):403–414, 2016.

[71] Qilian Liang and Jerry M Mendel. Interval type-2 fuzzy logic systems: theory and design. *IEEE Transactions on Fuzzy systems*, 8(5):535–550, 2000.

[72] Tsau Young Lin, Yiyu Y Yao, and Lotfi A Zadeh. *Data mining, rough sets and granular computing*, volume 95. Physica, 2013.

[73] Guilong Liu. Axiomatic systems for rough sets and fuzzy rough sets. *International Journal of Approximate Reasoning*, 48(3):857–867, 2008.

[74] Jan Lukasiewicz. Philosophical remarks on many-valued systems of propositional logic. *Jan Lukasiewicz Selected Works*, 1930.

[75] Jan Lukasiewicz and J Slupecki. *Select Works*. North-Holland Publishing Company, 1970.

[76] Teppo Luukkonen. Modelling and control of quadcopter. *Independent research project in applied mathematics, Espoo*, 22, 2011.

[77] Ebrahim H Mamdani and Sedrak Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International journal of man-machine studies*, 7(1):1–13, 1975.

[78] Ebrahim H Mamdani and Sedrak Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International journal of man-machine studies*, 7(1):1–13, 1975.

[79] Patricia Melin and Oscar Castillo. Intelligent control of non-linear dynamic plants using type-2 fuzzy logic and neural networks. In *2002 Annual Meeting of the North American Fuzzy Information Processing Society Proceedings. NAFIPS-FLINT 2002 (Cat. No. 02TH8622)*, pages 22–27. IEEE, 2002.

[80] Patricia Melin, Oscar Castillo, Janusz Kacprzyk, Marek Reformat, and William Melek. *Fuzzy logic in intelligent system design: Theory and applications*, volume 648. Springer, 2017.

[81] Jerry M Mendel. Computing with words: Zadeh, turing, popper and occam. *IEEE computational intelligence magazine*, 2(4):10–17, 2007.

[82] Jerry M Mendel and RI Bob John. Type-2 fuzzy sets made simple. *IEEE Transactions on fuzzy systems*, 10(2):117–127, 2002.

[83] JM Mendel. Uncertain rule-based fuzzy logic systems: Introduction and new. *directions. Ed. USA: Prentice Hall*, pages 25–200, 2000.

[84] Radko Mesiar. Triangular-norm-based addition of fuzzy intervals. *Fuzzy Sets and Systems*, 91(2):231–237, 1997.

[85] Masaharu Mizumoto and Kokichi Tanaka. Fuzzy sets and type 2 under algebraic product and algebraic sum. *Fuzzy Sets and Systems*, 5(3):277–290, 1981.

[86] Nehad N Morsi and Mohammed Mostafa Yakout. Axiomatics for fuzzy rough sets. *Fuzzy sets and Systems*, 100(1-3):327–342, 1998.

[87] Daniele Mundici. Bookmaking over infinite-valued events. *International journal of approximate reasoning*, 43(3):223–240, 2006.

[88] David F Nettleton, Albert Orriols-Puig, and Albert Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, 33(4):275–306, 2010.

[89] Adam Niewiadomski. A type-2 fuzzy approach to linguistic summarization of data. *IEEE Transactions on Fuzzy Systems*, 16(1):198–212, 2008.

[90] SV Ovchinnikov. General negations in fuzzy set theory. *Journal of Mathematical Analysis and Applications*, 92(1):234–239, 1983.

[91] Turhan Ozen and Jonathan Mark Garibaldi. Investigating adaptation in type-2 fuzzy logic systems applied to umbilical acid-base assessment. In *European Symposium on Intelligent Technologies, Hybrid Systems and Their Implementation on Smart Adaptive Systems*, pages 289–294. Citeseer, 2003.

[92] Hans Pacejka. *Tire and vehicle dynamics*. Elsevier, 2005.

[93] Madhumangal Pal, Sovan Samanta, and Ganesh Ghorai. *Modern trends in fuzzy graph theory*. Springer, 2020.

[94] Zdzisław Pawlak. Rough sets. *International journal of computer & information sciences*, 11(5):341–356, 1982.

[95] Patrice Perny. *Modélisation, agrégation et exploitation de préférences floues dans une problématique du rangement: bases axiomatiques, procédures et logiciels*. PhD thesis, Paris 9, 1992.

[96] Andrzej Piegat. *Fuzzy modeling and control*, volume 69. Physica, 2013.

[97] Radu-Emil Precup and Hans Hellendoorn. A survey on industrial applications of fuzzy control. *Computers in Industry*, 62(3):213–226, 2011.

[98] Gang Qian, Hai Wang, and Xiangqian Feng. Generalized hesitant fuzzy sets and their application in decision support system. *Knowledge-Based Systems*, 37:357–365, 2013.

[99] Anna Maria Radzikowska and Etienne E Kerre. A comparative study of fuzzy rough sets. *Fuzzy sets and systems*, 126(2):137–155, 2002.

[100] N. M. Raharja, Iswanto, O. Wahyunggoro, and A. I. Cahyadi. Altitude control for quadrotor with mamdani fuzzy model. In *2015 International Conference on Science in Information Technology (ICSITech)*, pages 309–314, Oct 2015.

[101] Hans Reichenbach. *The theory of probability*. Univ of California Press, 1971.

[102] Rosa M Rodriguez, Luis Martinez, and Francisco Herrera. Hesitant fuzzy linguistic term sets for decision making. *IEEE Transactions on fuzzy systems*, 20(1):109–119, 2011.

[103] Azriel Rosenfeld. Fuzzy graphs. In *Fuzzy sets and their applications to cognitive and decision processes*, pages 77–95. Elsevier, 1975.

[104] Enrique H Ruspini. A new approach to clustering. *Information and control*, 15(1):22–32, 1969.

[105] Daniel G Schwartz. The case for an interval-based representation of linguistic truth. *Fuzzy Sets and Systems*, 17(2):153–165, 1985.

[106] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[107] M Sugeno. Fuzzy measures and fuzzy integrals, a survey, fuzzy automata and decision processes (mm gupta, gn saridis and br gaines, eds.). *Am-sterdam: North-Holland*, pages 89–102, 1977.

[108] Roman W Swiniarski and Andrzej Skowron. Rough set methods in feature selection and recognition. *Pattern recognition letters*, 24(6):833–849, 2003.

[109] Eulalia Szmidt and Janusz Kacprzyk. Intuitionistic fuzzy sets in some medical applications. In *International conference on computational intelligence*, pages 148–151. Springer, 2001.

[110] Eulalia Szmidt and Janusz Kacprzyk. Medical diagnostic reasoning using a similarity measure for intuitionistic fuzzy sets. *Note on IFS*, 10(4):61–69, 2004.

[111] Pooneh R Tabrizi, Awais Mansoor, Juan J Cerrolaza, James Jago, and Marius George Linguraru. Automatic kidney segmentation in 3d pediatric ultrasound images using deep neural networks and weighted fuzzy active shape model. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 1170–1173. IEEE, 2018.

[112] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(1):116–132, Jan 1985.

[113] Vicenç Torra. Hesitant fuzzy sets. *International Journal of Intelligent Systems*, 25(6):529–539, 2010.

[114] Shun-Hung Tsai and Yu-Wen Chen. A novel identification method for takagi–sugeno fuzzy model. *Fuzzy Sets and Systems*, 338:117–135, 2018.

[115] IB Türkşen. Fuzzy normal forms. *Fuzzy Sets and Systems*, 69(3):319–346, 1995.

[116] Navneet Walia, Harsukhpreet Singh, and Anurag Sharma. Anfis: Adaptive neuro-fuzzy inference system-a survey. *International Journal of Computer Applications*, 123(13), 2015.

[117] N. Wang, H. R. Karimi, H. Li, and S. Su. Accurate trajectory tracking of disturbed surface vehicles: A finite-time control approach. *IEEE/ASME Transactions on Mechatronics*, 24(3):1064–1074, June 2019.

[118] N. Wang, S. Su, X. Pan, X. Yu, and G. Xie. Yaw-guided trajectory tracking control of an asymmetric underactuated surface vehicle. *IEEE Transactions on Industrial Informatics*, 15(6):3502–3513, June 2019.

[119] N. Wang, G. Xie, X. Pan, and S. Su. Full-state regulation control of asymmetric underactuated surface vehicles. *IEEE Transactions on Industrial Electronics*, 66(11):8741–8750, Nov 2019.

[120] Ning Wang and Hamid Reza Karimi. Successive waypoints tracking of an underactuated surface vehicle. *IEEE Transactions on Industrial Informatics (2019)*, 2019. Early Access.

[121] Ning Wang, Shun Su, Jianchuan Yin, Zhongjiu Zheng, and Meng Joo Er. Global asymptotic model-free trajectory-independent tracking control of an uncertain marine vehicle: An adaptive universe-based fuzzy control approach. *IEEE Transactions on Fuzzy Systems*, 26(3):1613–1625, 2017.

[122] Ning Wang, Jing Sun, and Meng Joo Er. Tracking-error-based universal adaptive fuzzy control for output tracking of nonlinear systems with completely unknown dynamics. *IEEE Transactions on Fuzzy Systems*, 26(2):869–883, 2017.

[123] Weiqiong Wang and Xiaolong Xin. Distance measure between intuitionistic fuzzy sets. *Pattern recognition letters*, 26(13):2063–2069, 2005.

[124] Siegfried Weber. A general concept of fuzzy connectives, negations and implications based on t-norms and t-conorms. *Fuzzy sets and systems*, 11(1-3):115–134, 1983.

[125] Dongrui Wu. An interval type-2 fuzzy logic system cannot be implemented by traditional type-1 fuzzy logic system. WORLD CONFERENCE ON SOFT COMPUTING, 2011.

[126] Dongrui Wu and Woei Wan Tan. Genetic learning and performance evaluation of interval type-2 fuzzy logic controllers. *Engineering Applications of Artificial Intelligence*, 19(8):829–841, 2006.

[127] Hongwei Wu and Jerry M Mendel. Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems. *IEEE Transactions on fuzzy systems*, 10(5):622–639, 2002.

[128] Xianbo Xiang, Caoyang Yu, Lionel Lapierre, Jialei Zhang, and Qin Zhang. Survey on fuzzy-logic-based guidance and control of marine surface vehicles and underwater vehicles. *International Journal of Fuzzy Systems*, 20(2):572–586, 2018.

[129] Ronald R Yager. On the measure of fuzziness and negation part i: membership in the unit interval. 1979.

[130] Ronald R Yager. An approach to inference in approximate reasoning. *International Journal of Man-Machine Studies*, 13(3):323–338, 1980.

[131] Ronald R Yager. On the measure of fuzziness and negation. ii. lattices. *Information and control*, 44(3):236–260, 1980.

[132] Ronald R Yager and Lotfi A Zadeh. *An introduction to fuzzy logic applications in intelligent systems*, volume 165. Springer Science & Business Media, 2012.

[133] Dejian Yu. Triangular hesitant fuzzy set and its application to teaching quality evaluation. *Journal of information &computational science*, 10(7):1925–1934, 2013.

[134] Lixin Yu and Yan-Qing Zhang. Evolutionary fuzzy neural networks for hybrid financial prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(2):244–249, 2005.

[135] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[136] Lotfi A Zadeh. Fuzzy sets and systems in system theory, ed. *J. Fox (brooklyn, NY: Polytechnic Press, 29–39*, 1965.

[137] Lotfi A Zadeh. Fuzzy sets, information and control, 8: 338-353. *MathSciNet zbMATH*, 1965.

[138] Lotfi A Zadeh. " fuzzy algorithms", information and control, vol. 12. 1968.

[139] Lotfi A Zadeh. Probability measures of fuzzy events. *Journal of mathematical analysis and applications*, 23(2):421–427, 1968.

[140] Lotfi A Zadeh. *Toward a theory of fuzzy systems*, volume 1432. National Aeronautics and Space Administration, 1969.

[141] Lotfi A Zadeh. Similarity relations and fuzzy orderings. *Information sciences*, 3(2):177–200, 1971.

[142] Lotfi A Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on systems, Man, and Cybernetics*, (1):28–44, 1973.

[143] Lotfi A Zadeh. The concept of a linguistic variable and its application to approximate reasoning–I. *Information sciences*, 8(3):199–249, 1975.

[144] Lotfi A Zadeh. The concept of a linguistic variable and its application to approximate reasoning—i. *Information sciences*, 8(3):199–249, 1975.

[145] Lotfi A Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems*, 1(1):3–28, 1978.

[146] Lotfi A Zadeh. Soft computing and fuzzy logic. In *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi a Zadeh*, pages 796–804. World Scientific, 1996.

[147] Lotfi A Zadeh. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy sets and systems*, 90(2):111–127, 1997.

[148] Lotfi A Zadeh. Fuzzy logic= computing with words. In *Computing with Words in Information/Intelligent Systems 1*, pages 3–23. Springer, 1999.

[149] Lotfi A Zadeh. A new direction in system analysis: From computation with measurements to computation with perceptions. In *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, pages 10–11. Springer, 1999.

[150] Ridong Zhang and Jili Tao. A nonlinear fuzzy neural network modeling approach using an improved genetic algorithm. *IEEE Transactions on Industrial Electronics*, 65(7):5882–5892, 2018.

[151] Yu-Jun Zheng, Sheng-Yong Chen, Yu Xue, and Jin-Yun Xue. A pythagorean-type fuzzy deep denoising autoencoder for industrial accident early warning. *IEEE Transactions on Fuzzy Systems*, 25(6):1561–1575, 2017.

[152] Huan Zhou, Jianqiang Wang, and Hongyu Zhang. Multi-criteria decision-making approaches based on distance measures for linguistic hesitant fuzzy sets. *Journal of the operational research Society*, 69(5):661–675, 2018.

[153] Bin Zhu, Zeshui Xu, and Meimei Xia. Dual hesitant fuzzy sets. *Journal of Applied Mathematics*, 2012, 2012.

[154] Wojciech Ziarko. Variable precision rough set model. *Journal of computer and system sciences*, 46(1):39–59, 1993.

Summary

In this dissertation, fuzzy membership functions and fuzzy operators are applied in the domain of machine learning. It consists of five major parts. A brief description of these parts along with the contents is described below:

Arithmetic-based Type1 Fuzzy Inference System

A novel technique for the design of an arithmetic-based type1 Fuzzy Inference System (FIS) is proposed, which is based on a new type of parametric membership function called the Distending Function (DF). The DF is a continuous and differentiable function (it is analytical). The Generalized Dombi operator (GDO)is used to calculate the firing strengths of the rules. The operator system and the DF are consistent with each other. The design process is simplified by handling the antecedent and consequent parts separately. Aggregation is performed using fuzzy arithmetic operations, more precisely using a linear combination of the DFs. Also, defuzzification is just a single step calculation. The technique is simple, computationally efficient and overcomes some of the drawbacks of the existing established techniques. Based on this new arithmetic-based type1 FIS and the gradient-based optimization method, a hybrid adaptive type2 FIS is also presented. An arithmetic-based type1 FIS is 20 to 50 times faster than the conventional FISs.
In Chapter 3, the design process of arithmetic-based type1 FIS is elaborated upon. Section 3.1 describes the fuzzy arithmetic and it is proved there that the DF is closed under linear combination. Section 3.2 describes the step-by-step procedure for developing the FIS using the available expert rules. Algorithm 1 is developed for a summarization and quick implementation. Section 3.3 presents an adaptive design for the FIS and the procedure is summarized in Algorithm 2. Section 3.4 shows the effectiveness of the proposed approach by controlling the level of a water tank system and tank temperature of a continuously stirred reactor system. Lastly, Section 3.5 concludes with a short discussion.

Data-Driven Arithmetic-Based Type1 Fuzzy System

Defining a proper rule system for a fuzzy inference system (FIS) is a difficult task. This is due to the unavailability of experts or incomplete knowledge of the process. Fortunately, rules can be extracted from the training data of the system. This allows us to design a data-driven FIS. Data-driven FIS techniques suffer from the flat structure problem i.e. the number of fuzzy rules grows exponentially as the input dimension increases. The consequence of this is greater complexity and poorer interpretability of the fuzzy rules. We presented a solution to the above-mentioned problem by proposing a novel data-driven arithmetic-based type1 FIS.
In Chapter 4, the procedure for designing data-driven type1 FIS is presented. Section 4.1 describes how DFs in various dimensions can be combined to get a single resultant DF in a higher dimension. Section 4.3 presents the steps to develop a type1 FIS using the data of the system. Algorithm 3 summarizes the these steps. Section 4.3 also proves the efficiency of the proposed data-driven FIS using two benchmark systems. Section 4.4 provides some concluding remarks.

Arithmetic-based Interval Type2 Fuzzy Inference System

To counter the effect of noise and uncertainties, a procedure was proposed to design an arithmetic-based interval type2 FIS. An interval Type2 Distending Function (T2DF) was developed for this purpose. Various types of T2DFs can be generated by adding uncertainty to its parameters. These T2DFs can represent and handle different types of uncertainties. A type2 FIS was designed using the fuzzy arithmetic operations. The design process did not include the implication and the type reduction steps, hence it was computationally efficient. The expressions used were in closed form, and this made it suitable for the on-line implementation. The proposed design is simple, intuitive and handles uncertainties. The T2DF can also handle the uncertainties that are generated because of measurement noise.
In Chapter 5, the procedure for designing arithmetic-based interval type2 FIS is presented. Section 5.2 describes three types of T2DFs. Section 5.3 proves that the T2DF is closed is under linear combination. Section 5.4 describes the step-by-step procedure for developing an arithmetic-based interval type2 FIS using the fuzzy rules. Algorithm 4 is presented for a quick implementation. Section 5.5 demonstrates the effectiveness of the proposed approach by designing an arithmetic-based interval type2 fuzzy controller for regulating the altitude of a quadcopter in the presence of a noisy feedback signal. Lastly, Section 5.6 concludes the discussion.

Data-Driven Arithmetic-Based Interval Type2 Fuzzy System

Fuzzy type2 modeling techniques are increasingly being used to model uncertain dynamical systems. However, some challenges arise when applying the existing techniques. A large number of rules are required to completely cover the whole input space. A large of parameters associated with type2 membership functions have to be determined and this leads to increased computation time and resources. The identified fuzzy model is usually difficult to interpret due to the large number of rules. Designing a fuzzy type2 controller using these models is also a computationally expensive task. To overcome these limitations, we proposed a procedure to identify the fuzzy type2 model directly from the data. This model is called the Distending Function-based Fuzzy Inference System (DFIS). This model consists of rules and T2DFs.
In Chapter 6, the design of data-driven type2 FIS is presented. Section 6.1 outlines the procedure for combining various T2DFs to get a single T2DF. This concept helps to reduce the number of rules. Section 6.1.1 provides a rigorous proof for extending the concept of a T2DF in higher dimensions. Section 6.2 describes the data-driven fuzzy type2 modeling in some detail. This section also outlines the procedure for reducing the number of fuzzy rules. Algorithms 5 and 6 summarize the procedures. Then Section 6.3 presents the benchmark simulations and online implementation results. Section 6.4 summarizes the discussion.

Rule-based Neural Networks

Deep Neural Networks (DNNs) are currently one of the most important research areas of Artificial Intelligence (AI). Various types of DNNs have been proposed to solve practical problems in many fields. However all these different types of DNNs have poor interpretations and are black box solutions. Expert systems are also a key area of AI that are based on rules. They have a lot of applications and their results are interpretable. We sought to combine the advantages of these two areas and it is a step towards having interpretable neural networks. We presented a new type of learning paradigm called Rule-Based Neural Networks (RBNNs). RBNNs can be trained to learn various regression and classification tasks.
In Chapter 7, RBNNs are elaborated upon. Section 7.2 describes the RBNN structures for solving regression and classification tasks. Feed-forward and feedback calculations are also formulated. Section 7.3 explains the training procedure of the RBNN. Algorithm 7 summarizes these training steps. Section 7.5 demonstrates the learning capacity of the RBNN. Various regression and classification tasks are solved and the

performance is compared with that for a DNN. Section 7.6 provides some concluding remarks and recommendations for the future.

Contributions of the thesis

A detailed discussion of the **Arithmetic-based Type1 Fuzzy Inference System** can be found in Chapter 3. The contributions are briefly summarized below:

I/1. A new type of parametric membership function called the Distending Function (DF) is introduced. With a few rules, it can cover the whole input space. It has three parameters and each has a semantic meaning. It has two types, namely the symmetric and asymmetric DF and both can be utilized for developing an FIS.

I/2. The DF is analytical i.e. higher derivatives exist at each point. This property is used in optimization procedures to tune the parameters of the DF.

I/3. The Generalized Dombi Operator (GDO) is used for evaluating the antecedent part of the rule. The GDO and the DF are consistent with each other.

I/4. Our approach does not involve the implication step. Instead the activation strength of each rule is multiplied by the consequent DF to get the fuzzy output of each rule.

I/5. The consequent of each rule is a DF. Aggregation is carried out using the weighted arithmetic mean of these consequent DFs of all the rules. A linear combination is closed for DFs and so the result of an aggregation is also a DF.

I/6. Defuzzification in this case is only a single-step calculation (finding the point that has the highest value of the aggregated DF).

I/7. Using our proposed approach, we designed an adaptive FIS. It consists of tuning the DF parameters using gradient descent optimization. The adaptive FIS can handle the changing process dynamics.

A nice discussion of the **Data-driven arithmetic-based fuzzy type1 system** can be found in Chapter 4. The contributions can be briefly summarized as follows:

II/1. DFs were used to cover the input space entirely. A DF has a long tail and it is defined on [-∞, ∞]. The grade of membership always has a non-zero value and the whole input space can be covered. If we apply the Dombi operator

on the DFs in the input space, the result is also a DF in the output (higher dimensional) space. So the whole area of the output is covered using a few rules.

II/2. The DF has three parameters. Usually two parameters can be kept constant and one parameter is used for tuning. The latter increases/decreases the influence area of the DF function. Also, a single step calculation is required to calculate this parameter. Because of this, due to the smaller number of identified rules and the deterministic nature of single parameter, the computation complexity of the quantitative part is negligible.

II/3. The interpretability of the model increases due to the significant decrease in the number of fuzzy rules.

A technical discussion of the **Arithmetic-based Interval Type2 Fuzzy Inference System** can be found in Chapter 5. The contributions are briefly summarized below:

III/1. A type2 extension of the DF called the Type2 Distending Function (T2DF) is proposed. Different types of uncertainties can be expressed by associating it with the parameters of the T2DF. It can effectively represent most of the forms of uncertainties used in type-2 fuzzy systems.

III/2. The fuzzy arithmetics approach is also utilized here for designing a type2 fuzzy logic controller. So it has no type reduction step and it does not require any iterative algorithms. It is simple, computationally fast and suitable for on-line implementations.

III/3. Most of the parameters of the T2DF are fixed. Usually just the parameter associated with the uncertainty is varied. We can say that the number of parameters is the same as that for a type1 FLS. Therefore the optimization process is easy to perform.

A detailed discussion of the **Data-Driven Arithmetic-Based Interval Type2 Fuzzy System** can be found in Chapter 6. The contributions are briefly summarized like so:

IV/1. We used the interval Type-2 Distending Function (T2DF). With a few rules, it can overcome the flat structure issue associated with the existing fuzzy modeling techniques.

IV/2. Our approach (called the DFIS) identifies a few important fuzzy rules from the data. We have also developed a rule reduction algorithm that can further reduce the number of identified rules. It results in an interpretable model.

IV/3. The type reduction, implication and aggregation steps are not involved. Also only a few parameters are varied during the design process. Therefore the type2 FIS can be implemented online.

A detailed discussion of the **Rule-based Neural Networks** can be found in Chapter 7. The contributions are briefly summarized below:

V/1. An RBNN can be trained to solve various real-world regression and classification tasks. The RBNN has a similar architecture to a DNN, but it has relatively few trainable parameters.

V/2. The input layer in an RBNN has normalization functionality. We proposed a new type of normalization technique and it is called the Ordered Normalization (ON). The ON is especially useful when the training data has an asymmetric distribution.

V/3. The training of an RBNN is similar to that of a DNN. Stochastic gradient (SG), batch gradient descent (BGD) and Levenberg-Marquadt (LM) optimization methods are used in the parameter update of back propagation. Other variants of gradient-based optimization can also be used to train an RBNN.

V/4. The results of an RBNN are interpretable and hence it is not a black box model. Hidden and output layers in the RBNN contain Rule-Based Neurons (RBNs). Each RBN has a built-in fuzzy inference system. After the training phase, the prediction results of the RBNN can be interpreted using simple if-then-else rules.

V/5. RBNN is robust to (input) feature noise and compared to a DNN, it produces a higher prediction accuracy even in the presence of large feature noise.

V/6. The performance of the RBNN on a skewed dataset is comparatively better than that of the DNN and it produces higher F1 scores for the minority (smaller) classes.

Összefoglalás

Az értekezés megmutatja hogyan lehet használni a halmazhoztartozási függvényt és az operátorokat a gyakorlatban a fuzzy control területén. A dolgozat 5 fő részből áll, új eljárásokat ismertet.

1. Egyes típusú (Type-1) aritmetikai alapú fuzzy következtetési rendszer

Egyes típusú aritmetikai fuzzy következtetési rendszer fejlesztésének megvalósítása új módon történt. Az eljárás az ún. paraméteres felfújó (distending) halmazhoztartozási függvény alkalmazásával került kifejlesztésre. A felfújó (distending) függvények folytonosak és differenciálhatóak (analítikus függvény) csak néhány paraméterrel rendelkeznek, segítségükkel az input tér néhány szabállyal lefedhető. Az eljárás során általánosított Dombi operátort használtuk a szabályok alkalmazási erősségének meghatározására (firing strength). Megmutattuk, hogy az algoritmus egyszerűsíthető, ha a feltételt (antecedent) és a következményt (consequent) külön kezeljük. Az aggregációt aritmetikai műveletek alkalmazásával hajtottuk végre, pontosabban meghatároztuk a felfújó (distending) függvények lineáris kombinációját. Így a defuzzifikációs eljárás egy lépéses számítással megkapható. A kidolgozott technológia egyszerű, számítási szempontból hatékony és a jelenleg alkalmazott eljárások hátrányait kiküszöböli. Megmutattuk, hogy az aritmetikai alapú egyes típusú (Type-1) fuzzy következtetési rendszer (FIS) és a gradiens alapú optimalizálási módszer alkalmazásával egy kettes tipusú (Type-2) következtetési rendszert is kezelni tudunk. Az aritmetikai alapú egyes tipusú (Type-1) következtetési rendszer 20-50-szer gyorsabb, mint a hagyományos eljárások.

2. Adatvezérelt aritmetikai alapú egyes típusú (Type-1) fuzzy rendszer

A fuzzy következtetési rendszer szabályait nem egyszerű megadni. Ennek oka, hogy szakértők legtöbbször nem állnak rendelkezésre, illetve a folyamatok működéséről nincs teljes, pontos információnk. A szabályokat meg lehet alkotni a rendszer tanító adatainak segítségével és ezek alapján lehet az adatvezérelt fuzzy következtetési rendszert létrehozni. A klasszikus adatvezérelt technikák hátránya, hogy a fuzzy szabályok száma exponenciálisan nő az input dimenzóinak számával, így a szabályok nem interpretálhatók. Ezeket a hátrányokat a dolgozatban kiküszöböltük, megadtuk a különböző dimenziókban konstruált felfújó (distending) függvények aggregációját, aminek az eredménye egyetlen felfújó függvény. Az eljárásnak a hatékonyságát két benchmark (viszonyítási alap) felhasználásával bizonyítottuk.

3. Aritmetikai alapú kettes típusú (Type-2) következtetési rendszer

A zaj és a bizonytalanság kiküszöbölését a kettes típusú (Type-2) halmazhoztartozási függvények alkalmazásával lehet kezelni. Ennek elérése céljából a dolgozatban megalkottuk a kettes típusú (Type-2) felfújó (distending) függvényeket, amelyeket úgy generáltunk, hogy a paraméterek értékei helyett intervallumokat választottunk. Itt is a fuzzy aritmetikus műveleteket alkalmaztuk a létrejött felfújó (distending) függvényekre. Az eljárás nem alkalmaz implikációt és nem tartalmaz egyszerűsítést (reduction), így számítási szempontból hatékony. Mivel a végeremény zárt formulában adható meg, az eljárás online számításokra is alkalmazható.
Összefoglalva az eljárás egyszerű, számítása gyors, kezeli a bizonytalanságot és zajos környezetben is működőképes.

4. Adatvezérelt intervallum alapú kettes típusú (Type-2) fuzzy rendszer aritmetikai alapú következtetéssel

Az utóbbi időben egyre fontosabbá vált a bizonytalan dinamikus rendszerek leírása. Ennek egyik eszköze a fuzzy kettes típusú (Type-2) modellek alkalmazása. Sok alkalmazás esetében a jelenleg létező technikák nem megfelelőek, ugyanis a teljes input tér szabályokkal való lefedésére volna szükség. A szabályok száma nagyon nagy, és ezzel a kettes típusú halmazhoztartozási függvény paparmétereinek száma is jelentősen megnő. A felmerülő nehézségek leküzdéséhez egy új megoldást határoztunk

meg, ahol a kettes típusú (Type-2) modell direkt módon származtatható az adatokból. Ez a modell nem más, mint a felfújáson alapuló következtetési rendszer (distending function based on fuzzy inference system DFIS)

5. Szabály alapú neurális hálózatok

A mesterséges intelligencia legfontosabb kutatási területéhez tartoznak a mély neurális hálózatok. Ezeknek nagyon sok fajtája létezik, amik mind fekete doboz jellegű megoldások, ezért az eredmények nem interpretálhatók. A mesterséges intelligencia másik fontos kutatási területe a szakértői rendszerek. Itt is sok megoldás létezik, melyek interpretálhatók, de a megoldásuk nem hatékony. A két terület összekapcsolásának segítségével létrehoztuk a szabályalapú neurális hálózatokat. Az így kialakult struktúra tanítható mind regressziós, mind osztályozás alapú feladatokra.

A disszertáció tézisei

Az **első téziscsoportban** részletes bemutatása a 3. fejezetben fejezetben található. Egyes típusú (Type-1) aritmetikai alapú fuzzy következtetési rendszer.

I/1. Egy új típusú parametrikus halmazhoztartozási függvényt vezettünk be: felfújó függvény (distending function). Néhány szabály alkalmazásával az egész input tér lefedhető. A függvénynek csupán három paramétere van és mindegyik szemantikus jelentéssel bír. A felfújó (distending) függvények két lényeges csoportba foglalhatók: szimmetrikus, nem szimmetrikus. Mindkettő a fuzzy következtetési rendszerben alkalmazható.

I/2. A bevezetett felfújó (distending) függvények analítikusak, azaz magasabb deriváltjai is léteznek. Ez a tulajdonság az optimalizálás során előnyös, azaz az adott alkalmazásban az optimális paraméterek meghatározhatók.

I/3. A szabályrendszer feltétel része az általánosított Dombi operátorral megadott és a felfújó (distending) függvény és az operátor konzisztensek.

I/4. A következtetési rendszer nem alkalmaz implikációt. A szabály alkalmazásának erősségét a feltételrendszer határozza meg. A következményt szintén felfújó (distending) függvény adja meg.

I/5. A szabályok következményeinek aggregációját a következmények súlyozott átlaga adja, ami a következményben szereplő felfújó függvények lineáris kombinációja. Megmutattuk, hogy ez szintén egyetlen felfújó (distending) függvényt eredményez.

I/6. A defuzzifikáció ebben az esetben egy lépéses számítás, azaz az eredménnyel kapott felfújó (distending) függvény maximuma.

I/7. Tanuló algoritmus segítségével a felfújó (distending) függvények optimális paramétereit gradiens módszerrel meg lehet határozni. Az íly módon meghatározott algoritmus a változó dinamikájú folyamatokat is hatékonyan kezeli.

A **második téziscsoport** részletes bemutatás a 4. fejezetben található.
Adatvezérelt aritmetikai alapú egyes típusú (Type-1) fuzzy rendszer.

II/1. Kihasználjuk a felfújó (distending) függvény azon tulajdonságát, hogy az input teret teljesen lefedi, ugyanis hosszú farokkal rendelkezik (a mínusz végtelentől plusz végtelenig definiált) Ha a felfújó (distending) függvényeket a Dombi operátor változóiba helyettesítjük, akkor egy magasabb dimenziójú felfújó (distending) függvényeket kaphatunk. Az így kapott függvények a magasabb dimenziójú teret is teljesen lefedik. A tér néhány kulcsfontosságú pontjában elhelyezett felfújó (distending) függvények megadják a fuzzy szabályokat, amik a teljes magasabb dimenziójú teret kifeszítik.

II/2. A felfújó (distending) függvény három paraméterének beállítása helyett csupán egyetlen lehet változó, azaz két paramétert rögzítünk, a harmadik pedig az adaptív folyamat végrehajtásához szükséges. Ez a harmadik paraméter felelős az adott felfújó (distending) függvény kiterjedéséért, illetve befolyásolási területéért. Ez szintén egy egy dimenziós optimalizálás, aminek következményeként az eljárás hatékonysága nő.

II/3. A modell interpretálhatósága jelentősen megnőtt, mivel az eljárás során a szabályok száma is csökkent.

A **harmadik téziscsoport** részletes bemutatás a 5. fejezetben található.
Aritmetikai alapú kettes típusú (Type-2) következtetési rendszer.

III/1. A kettes típusú (Type-2) halmazhoztartozási függvény kiterjesztésének megfelelően létrehoztuk a kettes típusú (Type-2) felfújó (distending) függvényeket (T2DF)Az így létrehozott függvénnyel különböző típusú bizonytalanságokat lehet modellezni, mégpedig a felfújó (distending) függvény paramétereinek változtatásával. Így a klasszikus kettes típusú (Type-2) közelítéseknél a bizonytalanságokhoz több függvényt kell konstruálni. A felfújó (distending) függvény esetében ez csupán a paraméter változtatással elérhető.

III/2. A fuzzy aritmetikai közelítés alkalmazása a fuzzy logiikai szabályozást helyettesíti Itt sincs ún. “ reduction” lépés, ezért az algoritmus nem iteratív. A gyorsaság miatt online interpretációra is alkalmazható.

III/3. Az eljárás során a legtöbb paraméter rögzített, egyedül a bizonytalansághoz kapcsolódó változik. Ebben az esetben a parameterek száma ugyanaz, mint az előző eljárásban, ezért az optimalizálás egyszerűen végrehajtható.

A **negyedik téziscsoport** részletes bemutatása az 6. fejezetben található.
Adatvezérelt intervallum alapú kettes típusú (Type-2) fuzzy rendszer aritmetikai alapú következtetéssel.

IV/1. Kettes típusú (Type-2) intervallum felfújó (distending) függvényt határoztunk meg (T2DF). Néhány szabály segítségével ez jól alkalmazható a lapos (flat) struktúrákra, ami a létező fuzzy technikák esetén előfordul.

IV/2. A megközelítés során néhány fontos fuzzy szabály kinyerhető az adatbázisból. Kifejlesztettük egy szabályszám redukáló algoritmust, ami jelentősen csökkentheti a szabályok számát és növeli az interpretálhatóságot.

IV/3. Az eljárás nem tartalmaz implikációt, aggregációt és bizonyos redukciós lépéseket. Ebben az eljárásban is csak néhány parameter változhat. Az eljárás online alkalmazható.

Az **ötödik téziscsoport** részletes bemutatás a 7. fejezetben található.
Szabály alapú neurális hálózatok.

V/1. A szabály alapú neurális hálózatnak nagyon sok alkalmazása van a mind regressziós és az osztályozási feladatokra. Hasonló struktúrával rendelkeznek, mint a mély neurális hálózatok, azonban sokkal kevesebb paraméterük van.

V/2. A szabály alapú neurális hálózatok input rétegén levő adatainak normalizáltnak kell lenni. Bevezettünk egy új normalizálást, ami empirikus eloszlás alapú (rendezés alapú) normalizálásnak neveztük el. Megmutattuk, hogy ez a normalizálás különösen hatékony, ha az input adatok asszimetrikus eloszlásúak.

V/3. Az RBNN tanulása hasonló a mély neurális hálózatokéhoz. A tanulás során stochasztikus gradiens módszert és Lebenberg-Marquadt optimalizálót alkalmaztunk a paraméterek meghatározására. Az iteráció alapja a backpropagation eljárás. Megjegyezzük, hogy a gradiens alapú optimalizálás más variensei is használhatók. A kiértékelés során aritmetikus eljárást választottunk.

V/4. A szabály alapú neurális hálózatok interpretálhatók. Rejtett rétegek tartalmazzák a szabályokat. Minden szabály alapú neurális hálózat értelmezhető, mint egy következtetési rendszer. A tanulási fázis után az előrejelzés megfeleltethető egy egyszerű “ha/akkor” szabálynak.

V/5. Az eljárás robosztus és zajtűrő. Megmutattuk, hogy a klasszikus mélytanulási eljárásokhoz képest nagyobb pontosságot lehet elérni jelentős zaj esetén is. A szabály alapú neurális hálózatok torzított (ferde) adatbázison sokkal jobban teljesítenek, mint a mély neurális hálózatok és nagyobb pontszámmal rendelkeznek, mint kis osztályok esetén.

Publications

Journal publications

[1] József Dombi and **Abrar Hussain**. A new approach to fuzzy control using the distending function. *Journal of Process Control*, 86, 16-29, 2020.

[2] József Dombi and **Abrar Hussain**. Data-Driven Interval Type2 Fuzzy Inference System Based on the Type2 Distending Function. *Submitted in IEEE Transaction on Fuzzy Systems, Revision requested after the first round of review*, 2022.

Full papers in conference proceedings

[3] József Dombi and **Abrar Hussain**. Data-Driven Arithmetic Fuzzy Control Using the Distending Function. In *Proceedings of Human Interaction and Emerging Technologies (IHIET)*, Springer, 215-221, 2019.

[4] József Dombi and **Abrar Hussain**. Interval Type-2 Fuzzy Control Using Distending Function. In *5th International Conference on Fuzzy Systems and Data Mining (FSDM)*, IOS Press, 705-714, 2019.

[5] József Dombi and **Abrar Hussain**. Robust Fuzzy Control using the Type2 Distending Function. In *19th International Symposium on Computational Intelligence and Informatics (CINTI-MARCRo)*, IEEE, 125-130, 2019.

[6] József Dombi and **Abrar Hussain**. Robust Rule Based Neural Network Using Arithmetic Fuzzy Inference System. *Accepted for publication in Intelligent Systems Conference (IntelliSys), 1-2 September 2022 in Amsterdam, Netherlands*.

[7] József Dombi and **Abrar Hussain**. Distending Function-Based Data-Driven Type2 Fuzzy Inference System. *Accepted for publication in Future Technologies Conference (FTC), 20-21 October 2022 in Vancouver, Canada*.

Other related publications

[8] Muhammad Jamil Hussain, Ahmad Shaoor, Salman Baig, **Abrar Hussain** and Syed Atif Moqurrab. A hierarchical based ensemble classifier for behavioral malware detection using machine learning. *Accepted for publication in IEEE 19th International Bhurban Conference on Applied Sciences and Technology (IBCAST) -2022*

[9] **Abrar Hussain**, Abdul Qayyum Khan, Muhammad Abid. Robust fault detection using subspace aided data driven design. *Asian Journal of Control*, 18(2), 709-720, 2015.

[10] **A. Hussain**, A. Q. Khan, M. Abid, M. Tufail. On fault detection in coupled liquid three tank system using subspace aided data driven design. In *International Conference on Emerging Technologies*, IEEE, 1-6, 2012.

Acknowledgments

First of all, I would like to express my heartfelt gratitude to my supervisor, József Dombi, for his excellent guidance throughout my Ph.D. studies. I would also like to thank my colleagues and friends who have supported me with their constant presence and have helped me to achieve the results presented here, and for making the term of my studies a memorable and meaningful one.
I would like to dedicate this thesis to my late father, who had been a pillar of strength for me, and who had anxiously looked forward to my graduation. And I wish to express my gratefulness to my wife and family for consistently loving and supporting me, and without whom my Ph.D. thesis wouldn't materialize. Last, but not least, I would like to thank David P. Curley for reviewing and proofreading this thesis.