Rita Lovassy

Multilayer Perceptrons Based on Fuzzy Flip-Flops

Ph.D. Dissertation

Supervisor: László T. Kóczy
Jedlik Ányos Institute of Informatics, Electrical and Mechanical Engineering
Faculty of Engineering Sciences, Széchenyi István University Győr, Hungary

Interdisciplinary Doctoral School of Engineering
Faculty of Engineering Sciences, Széchenyi István University, Győr

Győr, 2010

Acknowledgements

First of all, I would like to thank to my supervisor László T. Kóczy, who introduced me to the world of research. Many thanks also for his valuable help in professional questions and encouragement throughout this project, furthermore for the introduction to numerous great researchers in the international research community in Computational Intelligence.

I would like to thank to my fellow Ph.D. student László Gál, who, stimulated and helped me during this research with his enthusiasm and professional experience. He gave me free run of his own developed programs in the subject of fuzzy flip-flop based neural networks optimization and training. Thanks must also go to János Botzheim for the supportive discussions during this research.

Finally, my sincere thanks to my parents and children for their support and patience during the last years.

Contents

Chapter 1

Introduction

1.1 Approximate Computational Intelligence Models: A Historical Survey

Fuzzy set theory, artificial neural networks, evolutionary algorithms and their hybrid combination have been the object of intense study and application, especially in the last decade. There are several different manners to combine the mentioned three main branches of Computational Intelligence (CI) which may differ essentially according to the approaches and the tasks. Many applications show an increased interest in using either two or all three of them in one system.

The real problem does not lie in a direct comparison between “new” and “traditional” methodologies, but rather in their field of applicability. Daily life is full with activities, such as making choices, performing decisions, handling timetables, etc. These daily routines possess a high degree of chaos and uncertainty, which might be developed with mental activities (fuzzy decision-making).

Artificial neural networks and fuzzy logic systems share common features and techniques in the context of approximate reasoning. The main idea is using the high flexibility of neural networks produced by learning, in order to tune the membership functions used in fuzzy control. The approximate reasoning capability, transparency and interpretability of fuzzy sets, the learning capabilities and the property of auto-adaptability of neural networks, furthermore the optimal structure approximation properties of evolutionary, especially bacterial algorithms were developed with the aim to deal with problems which were hard to solve using traditional techniques.

In the years 1990-92 papers by D. Dubois, M. Grabisch and H. Prade [19], B. Kosko [61], furthermore Wang and Mendel [111], [112] proved almost simultaneously that fuzzy systems are universal approximators. In 1997 E. P. Klement, L. T. Kóczy and B. Moser [54] argued that fuzzy systems can only be universal approximators in a rather restricted sense,

because of the limits set by computational complexity. The authors also exemplified the main approaches to realize the idea of controlling real world processes by means of linguistic variables.

In the field of artificial neural networks (connectionist models, parallel distributed processing systems and neuromorphic systems) mathematical function approximation using input-output data pairs from a set of examples is the object of study in different applications such as applied mathematics, and computer science. The paper of Hornik, Stinchcombe and White [46], established that standard multilayer feedforward networks with a single hidden layer constitute a class of universal approximators. They gave a general investigation of the capabilities and properties of multilayer feedforward networks, without any suggestion to the number of hidden units needed to achieve a given accuracy of approximation. The function approximation capability of multilayered neural networks was studied in detail by Ciuca [12], Cybenko [13], Funahashi [27], Hecht-Nielsen [40] and Ito [50]. They proved that any continuous function can be approximated by a three-layered feedforward network with hidden sigmoid units and one linear output unit. The use of four-layered (that have two sigmoid unit layers) neural network as universal approximators of continuous functions have been investigated by Funahashi [27], Girosi and Poggio [30] and Hecht-Nielsen [40]. Kurkova [63] studied also multilayer feedforward networks with sigmoid activation function approximation capabilities, analyzing also their computational complexity issues. Blum and Li showed [3] that four-layered feedforward networks with two hidden layers of semi linear units and with a single linear output unit are universal approximators. In [47] Hornik generalized the set of activation functions. He concluded that the neural networks approximation capability is very strongly dependent on the multilayer feedforward architecture, and less on the choice of the activation function. Furthermore, the number of hidden units exponentially depends on the dimension of the approximated function.

1.2 Aims of the Research

This research was designed to explore and tackle the interleaving of imprecision, computer science and a subset of evolutionary algorithm inspired by natural evolution. The aim of this study is essentially to show how fuzzy models, neural networks, and bacterial algorithms can be usefully merged and deployed to solve parameter optimization and function

approximation problems. The combination of these three techniques tries to minimize their weaknesses and to profit their advantages.

The defined new types of fuzzy flip-flop offer an attractive opportunity for designing generic hardware for fuzzy information processing. In analogy with the binary flip-flops which are the basic units in every synchronous sequential digital circuit, the fuzzy flip-flops can be reckoned as the basic functional blocks for developing dynamical fuzzy systems. The various novel types of fuzzy flip-flop have been proposed in a manner that covers possible combinations of fuzzy J-K and different types of fuzzy D flip-flop based on various fuzzy operations. The transfer characteristics of some of the studied fuzzy flip-flop types present quasi sigmoid behavior. This fact indicates a possible connection of this fuzzy unit with the artificial neuron the basic component of neural networks.

The concept of fuzzy flip-flop neuron is introduced. They are interlinking to form a complex fuzzy neural network. How these neurons are organized is itself a highly complex problem. For a given problem the best network is selected from the performance and complexity points of view. This study has been focused more or less on the neuro-computations direction. Fuzzy Flip-Flop based Neural Networks (FNNs) as a novel implementation possibility of multilayer perceptron neural networks are investigated and its learning algorithm was proposed, furthermore giving a possible starting point for researchers interested in a novel way of hybridization of fuzzy logic with artificial neural networks.

The Levenberg-Marquardt (LM) method [66], [78] and a special combination of LM algorithm with the Bacterial Evolutionary Algorithm (BEA) [85], the Bacterial Memetic Algorithm with Modified Operator Execution Order (BMAM) [28] have been applied to the novel model framework parameter optimization and training. The function approximation properties of various FNNs built up from different types of fuzzy flip-flop neuron have been investigated.

1.3 Thesis Structure

The dissertation consists of six chapters. After the Introduction and a historical survey, the rest of this thesis is structured as follows:

- Chapter 2. *Overview of Concepts, Approaches, and Related Main Results*

This chapter comprises background information related to fuzzy set theory, artificial neural networks, and evolutionary algorithms. It overviews the basic concepts and principles of fuzzy norms, triangular norms and fuzzy flip-flops, the neuron model, and neural networks architecture with some main types of neural networks and learning algorithms. Detailed descriptions of various kinds of evolutionary techniques are also given.

- Chapter 3. *Definition and Properties of Several New Fuzzy Flip-Flops*

First, interval valued fuzzy flip-flops are defined in the sense of Türkşen's Interval Valued Fuzzy Sets (IVFS) according to a new concept of minterm-maxterm IVFS (MIVFS). In this chapter the research introduces new definitions of different novel (reset and set, furthermore unified) types of fuzzy J-K flip-flop (F^3). The concept is extended on non-associative fuzzy operations. In this case a somewhat surprising result is proved, that comparing the equations of the next state of the set and reset type of a modified version of the Fodor fuzzy flip-flop the two formulas are equivalent. Equivalence does not hold for any other F^3 having been defined as far in the literature. Several new types of fuzzy flip-flop are defined, determining their basic definitions and properties, including also graphical illustrations, comparisons. Eventually, the research gives a comparison of the novel types of fuzzy flip-flop in a manner which cover all possible combinations of fuzzy J-K and two types of fuzzy D flip-flop based on various fuzzy operations. It is shown that broadly they may be classified into two groups, one presenting quasi s-shape transfer characteristics and the rest with non-sigmoid character.

- Chapter 4. *Multilayer Perceptrons Based on Fuzzy Flip-Flops*

The concept of fuzzy J-K and D flip-flop neuron is introduced. The second main topic here is the study of the function approximation capability of the novel Fuzzy Flip-Flop based Neural Network (FNN), as a new type of neural networks. A comparison of the effect of applying some well known t-norms in the investigation of the F^3 based neurons and the Multilayer Perceptrons (MLPs) based on them is presented. The proposed network is a structure consisting of the same types of F^3 whose training is much simpler than training the individual flip-flops. The used training method is the Levenberg-Marquardt algorithm. The effect of fuzzy operations parameters and fuzzy neurons numbers are studied. Illustrative examples are presented, in order to demonstrate the success of this work in terms of the function approximation capability of the proposed fuzzy combinational system. The FNNs for hardware implementation with general purpose (unknown application) are more suitable to avoid overfitting than customary (e.g. *tansig* based) neural networks.

- Chapter 5. *Parameter Optimization in the Fuzzy Neural Networks*

The application of several model identification algorithms for parameter optimizing in order to achieve as good as possible approximation features of the FNN is proposed. The research strove after the improvement of the function approximation capability of the proposed fuzzy neural networks by applying Levenberg-Marquardt method and a special kind of bacterial memetic algorithm, the Bacterial Memetic Algorithm with Modified Operator Execution Order. Parameter optimization methods have been developed, which provide good function approximation capability. By extensive simulations the most suitable types of F^3 neurons in FNNs are found.

- Chapter 6. *Conclusions and Further Work*

This chapter gives a summary of the results introduced in the dissertation chapter by chapter. Details of proposed future work in this area are presented.

Chapter 2

Overview of Concepts, Approaches, and Related Main Results

2.1 Fuzzy Set Theory

Since 1965, when L. A. Zadeh [120] proposed the theory of fuzzy set, the fuzzy logic and systems theory have gone through a long theoretical development and its applications to real life problems have been demonstrated through many applications. Thus, the role of fuzzy sets is not restricted to a specific field; it can be embedded to almost any area requiring human knowledge.

The fuzzy set theory describes the degree of belonging of an individual member, which can exist as any real value between 0 and 1, instead of the crisp 0 and 1 value. Zadeh also demonstrated how these fuzzy sets could be operated, and developed a framework for using all these structures. This kind of soft computing deals with imprecision and vagueness formulated in strict mathematical theory, in opposite to the crisp logic, where the accuracy, confidence and rigidity are far the best properties.

The notion of linguistic variables and values has been introduced by Zadeh in 1973 [121]. This new technique was able to model nonlinear systems, by approximating its operation with IF-THEN rules. In his seminal paper he pointed out the close connection between natural languages and fuzzy logic. Zadeh in [122] highlight the importance of Computing with Words within fuzzy logic. The basic idea was the close connection between words and computation with imprecise probabilities because they are generally described in a natural language. The linguistic variables and fuzzy IF-THEN rules are employed in almost all applications of fuzzy logic.

Nowadays, fuzzy logic is widely applied in many real-world applications, starting from fuzzy air-conditioners, fuzzy washing-machines, to complex control systems such as fuzzy robotics [49], traffic junction [10], model car parking [65], power system [1], fuzzy

based ABS [119] etc. Fuzzy systems are very useful when the desired system is with high complexity whose behaviors are not well understandable and in situations where an approximate, but fast solution is warranted [95].

The fuzzy system is defined by three main components [52], [53]:

- Fuzzy input and output variables defined by their fuzzy values;
- A set of fuzzy rules
- Fuzzy inference mechanism

Fuzzy rules deal with fuzzy values as, for example, “high”, “cold”, “very low”, etc. Those fuzzy concepts are usually represented by their membership functions. A membership function shows the extent to which a value from a domain (also called universe) is included in a fuzzy concept (see, e.g. Figures 2.1 - 2.3).

Fuzzy inference takes inputs, applies fuzzy rules and produces outputs. The inputs and outputs values of a fuzzy system can be either crisp or fuzzy values.

Case example: On a real-world application a simple fuzzy air-conditioning control system (FACS) illustrates the basic components of fuzzy logic [64]. The structural frameworks of fuzzy logic are the following modules:

- Fuzzy Identification Module

In this module the fuzzy variables (FV) are determinate and identified. For example, the air temperature (Temp) and relative humidity (RH) are given by the sets of fuzzy variables:

Temp = [Mild, Warm, Hot, Very Hot]

RH = [Dry, Moderate, Humid, Very Humid]

- Fuzzy Categorization Module

This module focuses on how to categorize the chosen FVs into different fuzzy sets. Using the above example, the fuzzy sets are with approximately ranges of values:

Mild = (17-25)˚C	Dry <=60%
Warm = (23-31)˚C	Moderate = 55-85%
Hot = (25-33)˚C	Humid = 65-95%
Very Hot = >28˚C	Very Humid = >75%

The boundaries fuzziness is one of the major characteristics of fuzzy sets.

- Fuzzy Modeling Module

The corresponding membership functions are defined in Figures 2.1 and 2.2:

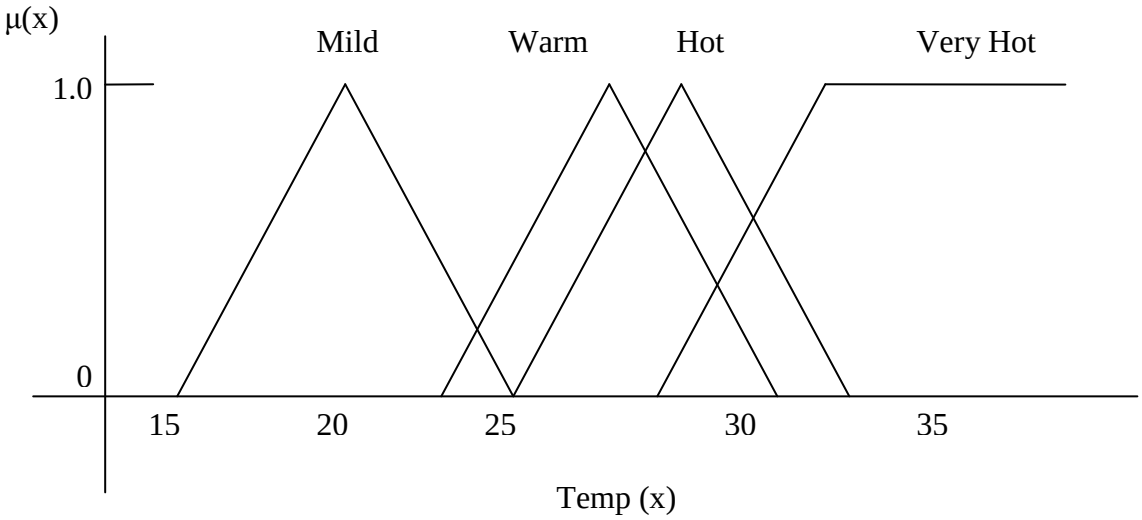


Figure 2.1 Fuzzy membership functions for air temperature

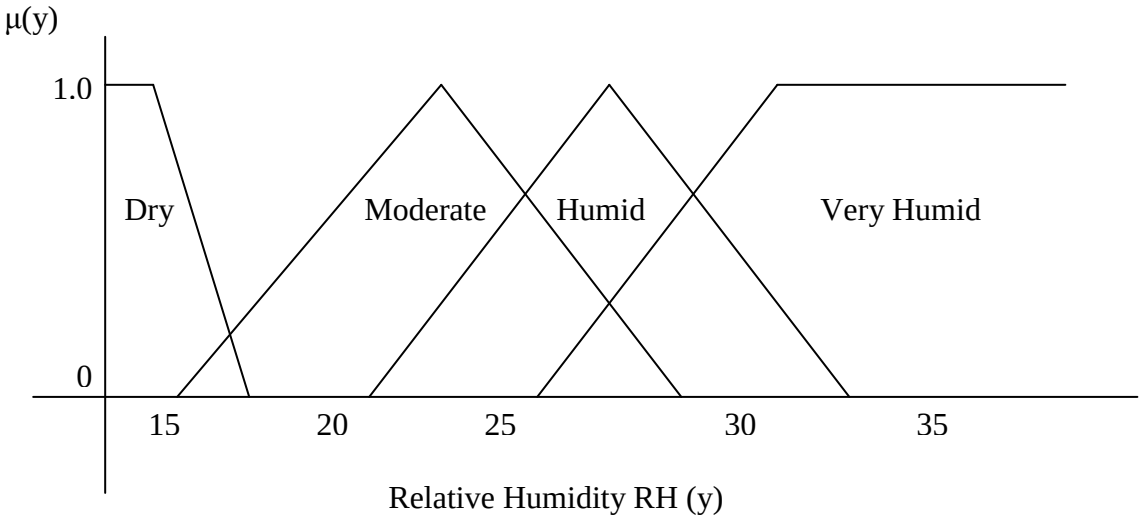


Figure 2.2 Fuzzy membership functions for relative humidity

Suppose that the power setting of the FACS has three different fuzzy states, given by: Power = [Low, Medium, High] with the membership function shown in Figure 2.3.

- Fuzzy Reasoning Module

This module consists of three main processes:

- Fuzzy rule construction process
- Fuzzy knowledge-base construction process
- Fuzzy inference process

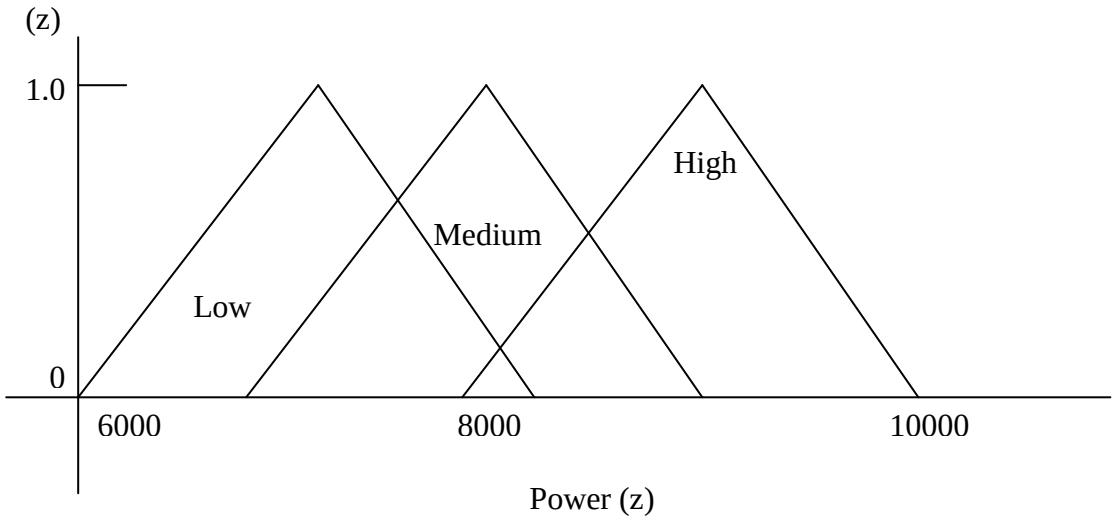


Figure 2.3 Fuzzy membership functions of fuzzy air-conditioning control system power

The power switch is controlled by the following three fuzzy rules:
Rule1: If Temp is “Very Hot” **AND** RH is “Very Humid” Then switch to “High”
Rule2: If Temp is “Hot” **AND** RH is “Humid” Then switch to “Medium”
Rule3: If Temp is “Warm” **OR** RH is “Moderate” Then switch to “Low”.

2.1.1 Basic Concepts

Fuzzy sets are always functions, which define a universal set *X* onto the unit interval [0, 1]. The fuzzy sets *A* and *B* are the functions μ_A and μ_B that carries *X* into [0, 1] interval. The functional mapping is given by

$$\mu_A(x): X \to [0,1] \tag{2.1}$$

According to the above notations, $\mu_A(x)$ is a value on the unit interval that measures the degree to which element *x* belongs to fuzzy set *A*.

2.1.2 Interval Valued Fuzzy Sets

The requirement for a precise membership function can be relaxed by allowing values $\mu_A(x)$ to be intervals of real numbers in [0, 1] rather than single numbers [56]. These fuzzy sets types are called Interval Valued Fuzzy Sets (IVFS). An IVFS is a fuzzy set whose

membership function is many-valued and forms an interval in the membership scale. They are defined by membership functions of the form

$$\mu_A(x): X \to E([0,1]) \tag{2.2}$$

where $E([0,1])$ represents a set of closed intervals in $[0,1]$.

An interval valued fuzzy set was first proposed by Sambuc [101] under the name Φ-fuzzy sets. Another way of generating interval valued fuzzy sets was noticed by Türkşen and Yao [108]. In classical logic connectives can be expressed in normal forms: conjunctions of disjunctions (conjunctive normal form, CNF) and in very similar way disjunctions of conjunctions (disjunctive normal form, DNF). The Disjunctive and Conjunctive Normal Forms play very special roles in classical Boolean logic. They represent those standard forms, which do not contain any redundancy in the sense that they cannot be further reduced by applying the idempotence law; however they consist of complete members containing all variables in question Thus, usually they can be simplified by merging and eliminating, this way obtaining the corresponding minimal forms. Thus, in Boolean logic the two normal forms are equivalent and can be transformed into each other using the De Morgan theorems. In fuzzy logic there are no standard forms in this sense as idempotence itself does usually not hold. Any repeated member would change the value of the expression.

Despite this latter fact several authors consider the fuzzy extensions of DNF and CNF as having special significance. Especially, Türkşen et al. in [110] examined the properties of the extended normal forms and came to the somewhat surprising and very convenient conclusion that in fuzzy logic the equivalents of the CNF and DNF forms represent the extremes of all possible expressions that correspond to forms being equivalent in binary logic. Thus he showed that for any fuzzy connective the value of every other form lies in the interval formed by the CNF and DNF values. In [109] this result was proven for logical operations with an arbitrary number of variables.

While the theory of the interval valued fuzzy sets is much more general and can be considered as a special case of L-fuzzy sets [31], Türkşen proposed the interval determined by the disjunctive and conjunctive normal forms as the interval associated with the value belonging to an expression obtained by the fuzzy extension of some classic binary concept. Indeed, any theoretically possible formulation of the same concept would result in a value lying within the interval thus proposed.

According to Türkşen's statement the DNF is always included in the corresponding CNF, i.e., $DNF(\cdot) \subseteq CNF(\cdot)$ where $(\cdot)$ represents a particular expression [109]. The fundamental result that every DNF is contained in its corresponding CNF is true for min-max operators and for algebraic triplets as well. Türkşen [109] proposed to define the interval-valued fuzzy set representing a Boolean expression as follows:

$$IVFS(\cdot) = \left[DNF(\cdot), CNF(\cdot) \right] \tag{2.3}$$

2.1.3 Fuzzy Operations

The three basic crisp (Boolean) logical operations (namely complementation-negation, intersection-conjunction and union-disjunction), well-defined on traditional sets can be generalized in many ways using fuzzy sets. Zadeh proposed operators for set complement, intersection and union. These operators have been referred to classic, standard or Zadeh-type ones. The generalized class of intersections was shown to satisfy the axiomatic properties of t-norms, while unions were proven to be t-conorms (s-norms).

The standard fuzzy operations for the fuzzy sets *A* and *B* defined on the universe *X* and for a given element *x* of the universe, are given in Eqs. (2.4) - (2.6) [95].

Complement $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$ (2.4)

Union $\mu_{A \cup B}(x) = \mu_A(x) \vee \mu_B(x) = \max\left[\mu_A(x), \mu_B(x) \right]$ (2.5)

Intersection $\mu_{A \cap B}(x) = \mu_A(x) \wedge \mu_B(x) = \min\left[\mu_A(x), \mu_B(x) \right]$ (2.6)

For each of the three set operations a large variety of formulas and even different classes of functions possessing appropriate axiomatic properties have subsequently been proposed.

The history of triangular norms started with the paper of Menger [79]. The author proposed the construction of metric spaces where probability distributions are used in order to describe the distance between two elements of the space in question [55]. Later it was proved that the triangular norms are mathematically identical with fuzzy operators. In fuzzy set theory the study of triangular operators has been going on for a long time. The triangular norms are based on the geometrical aspects of probability theory published in the early 60's by Schweizer and Sklar [102], [103], and Ling [68].

The fuzzy literature offers a large variety of triangular operators; researchers still propose again and again new fuzzy operations to be used in a given field. Obviously, the performance of fuzzy systems depends on the choice of different triangular operators. Despite the variety of available fuzzy set operators, however, the classic triplet of complement, intersection and union still bear particular significance, especially in the practical applications. The big challenge for fuzzy researchers is to fit the fuzzy sets into the context of applications. For instance, if the functions within a class are interpreted as performing union or intersection operations of various strengths, then the classical max union is found to be the strongest of these and the classical min intersection the weakest.

2.1.3.1 Fuzzy Negations

The complement (negation) of a fuzzy set *A* is specified by a unary operation on the unit interval:

$$c : [0,1] \to [0,1] \tag{2.7}$$

For each element *x* this function yields the membership grade of the same element in the complement of the original set thus

$$\mu_{\bar{A}}(x) = c\left[A(x)\right] \qquad \text{for all } x \in X. \tag{2.8}$$

The axiomatic skeleton for fuzzy complement satisfies two axiomatic requirements: the boundary conditions and monotonicity [57]. Examples of general fuzzy complements that satisfy only the axiomatic skeleton are the threshold-type complements defined by

$$c(a) = \begin{cases} 1 & \text{for } a \le t \\ 0 & \text{for } a > t \end{cases} \tag{2.9}$$

where $a \in [0,1]$ and $t \in [0,1)$; t is called the threshold of *c*. An example of a fuzzy complement that is continuous is the function

$$c(a) = \frac{1}{2}(1 + \cos \pi a) \tag{2.10}$$

One class of involutive ($c(c(a)) = a$, for any $a \in [0,1]$) fuzzy complements is the Sugeno class defined by

$$c_\lambda(a) = \frac{1-a}{1+\lambda a} \text{ where } \lambda \in (-1,\infty) \tag{2.11}$$

Another example of a class of involutive fuzzy complements is defined by

$$c_w(a) = \left(1 - a^w\right)^{1/w} \text{ where } w \in (0,\infty) \tag{2.12}$$

the Yager class of fuzzy complements.

For $\lambda = 0$ or $w = 1$, the functions given by equations (2.11) and (2.12) become the classical fuzzy complement (strong negation). Function *c* satisfies the boundary conditions, is a monotonic, continuous function and is involutive.

$$c(a) = 1 - a\,. \tag{2.13}$$

2.1.3.2 Fuzzy Intersections and Unions (t-norms and t-conorms)

Klement, Mesiar and Pap enumerate and give the basic definitions and properties of the most general fuzzy operations in [55] including also graphical illustrations and comparisons. A special class of operators and their respective characteristics were proposed also in [17], [20]. The intersection (conjunction) and union (disjunction) of two fuzzy sets *A* and *B* are specified in general by binary operations on the unit interval; i.e., functions of the form

$$i : [0,1] \times [0,1] \to [0,1] \quad \text{and} \tag{2.14}$$

$$u : [0,1] \times [0,1] \to [0,1]\,. \tag{2.15}$$

For each element x of the universal set X, these functions take the pair consisting of the element's membership grades in set A and in set B as their arguments and yield the membership grades of the elements in the set constituting the intersection and the union of A and B, resp. Thus,

$$\mu_{(A\wedge B)}(x) = i\left[\mu_A(x), \mu_B(x)\right] \qquad \text{and} \qquad (2.16)$$

$$\mu_{(A\vee B)}(x) = u\left[\mu_A(x), \mu_B(x)\right] \qquad (2.17)$$

for all $x \in X$. (See [56], [57].)

While these two operations have a completely dual axiomatic skeleton they can be defined independently from each other. The axiomatic skeleton for fuzzy set intersections and unions satisfy boundary conditions, they are commutative, monotonic, and associative.

Table 2.1 shows various popular t-norms and t-conorms. Each class contains parameterized as well as non-parameterized operators. All of them can be used as basic operators in the definition of the fundamental formulas of the fuzzy flip-flops introduced in the next section.

TABLE 2.1 SOME SELECTED T-NORMS AND T-CONORMS

Fuzzy operation	t-norm; $i(a, b)$	t-conorm; $u(a, b)$
Standard (min-max) [120]	$\min(a,b)$	$\max(a,b)$
Algebraic [55]	ab	$a+b-ab$
Drastic [55]	a when b=1, b when a=1, 0 otherwise	a when b=0, b when a=0, 1 otherwise
Łukasiewicz [55]	$\max(0, a+b-1)$	$\min(1, a+b)$
Yager [117]	$1-\min\left[1,((1-a)^w+(1-b)^w)^{1/w}\right]$	$\min\left[1,(a^w+b^w)^{1/w}\right]$
Dombi [17]	$\dfrac{1}{1+\left[(1/a-1)^\alpha+(1/b-1)^\alpha\right]^{1/\alpha}}$	$\dfrac{1}{1+\left[(1/a-1)^{-\alpha}+(1/b-1)^{-\alpha}\right]^{-1/\alpha}}$
Hamacher [37]	$\dfrac{ab}{\nu+(1-\nu)(a+b-ab)}$	$\dfrac{a+b-(2-\nu)ab}{1-(1-\nu)ab}$
Frank [26]	$\log_s\left[1+\dfrac{(s^a-1)(s^b-1)}{s-1}\right]$	$1-\log_s\left[1+\dfrac{(s^{1-a}-1)(s^{1-b}-1)}{s-1}\right]$
Dubois-Prade [18]	$\dfrac{ab}{\max(a,b,d)}$	$\dfrac{a+b-ab-\min(a,b,1-d)}{\max(1-a,1-b,d)}$
Schweizer-Sklar [103]	$\max(0, a^p+b^p-1)^{1/p}$	$1-\max(0,(1-a)^p+(1-b)^p-1)^{1/p}$

Parameters w, α, ν and s lie within the open interval $(0,\infty)$, $d \in [0,1]$, $s \neq 1$ and $p \neq 0$.

In the next, for simplicity the t-norm is denoted by i (intersection), and the t-conorm by u (union), where the subscripts refer to the initial of the type of the norm: e.g., in case of

algebraic norms: $i_A\ (a,b) = a\ i_A\ b$ and $u_A\ (a,b) = a\ u_A\ b$.

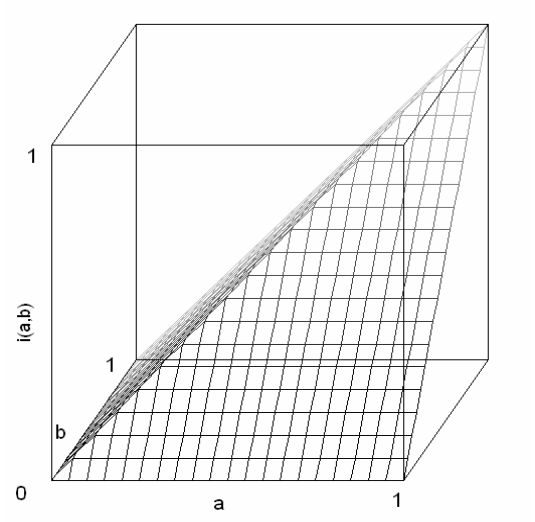
In particular cases when the parameters v and d are equal to 1:

$i_A = i_H = i_{DP}$ and $u_A = u_H = u_{DP}$,

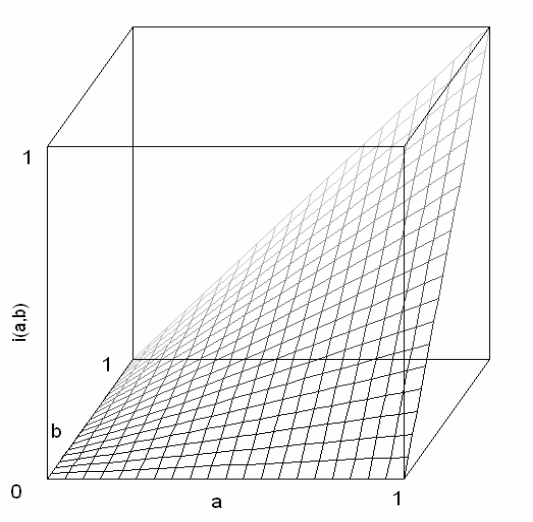
furthermore when $w = 1$ and $p = 1$:

$i_L = i_Y = i_{SS}$ and $u_L = u_Y = u_{SS}$.

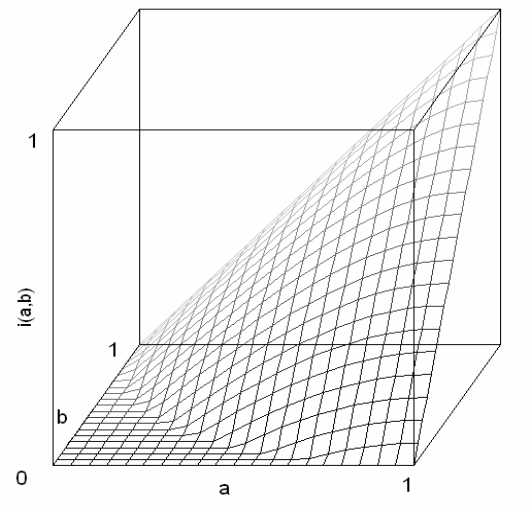
The Hamacher and Dubois-Prade norms include the algebraic ones, additionally; the Yager and Schweizer-Sklar norms include Łukasiewicz operations. For example, the character of standard, algebraic, Yager and Dombi t-norms and t-conorms for a selected parameter value ($w = 2$ and $\alpha = 2$) is illustrated in Figures 2.4 and 2.5.



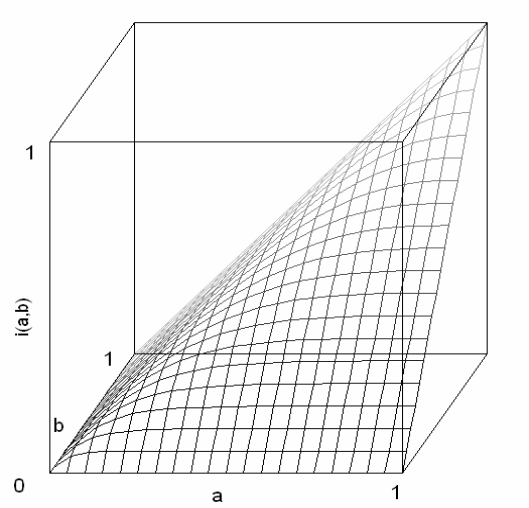
(a) Standard



(b) Algebraic

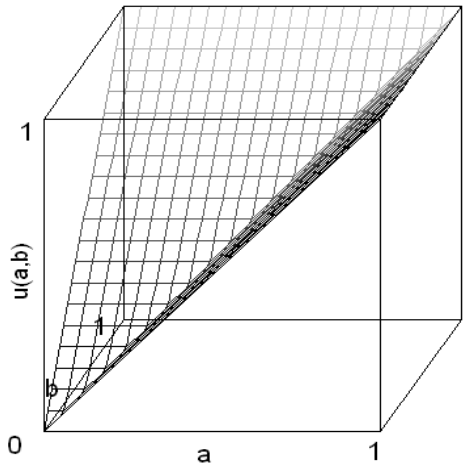


(c) Yager

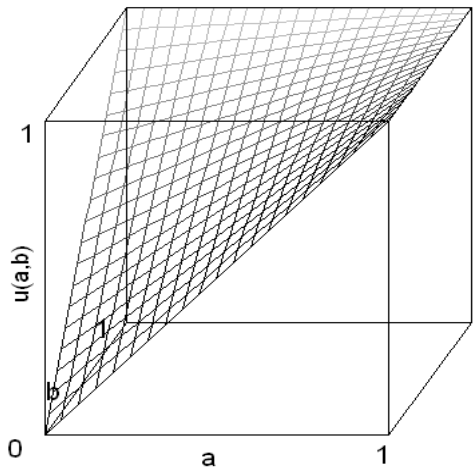


(d) Dombi

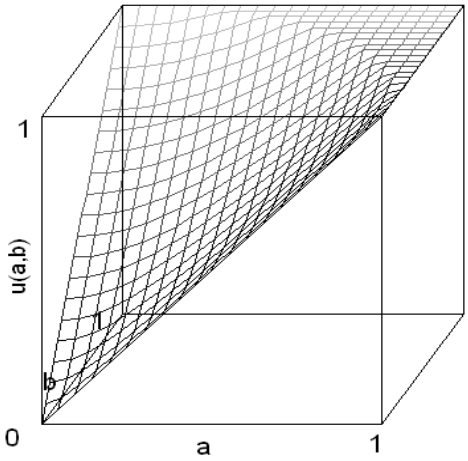
Figure 2.4 Graphs of some selected fuzzy t-norms



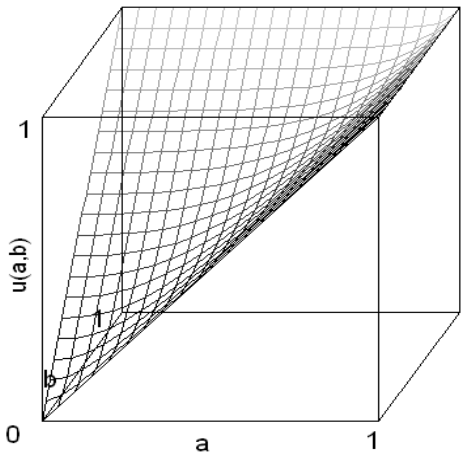
(a) Standard



(b) Algebraic



(c) Yager



(d) Dombi

Figure 2.5 Graphs of some selected fuzzy t-conorms

2.1.3.3 Non - Associative Operations

The motivation for non - associative fuzzy operations can be found in analyzing the behavior of the connectives in subjective probability or certainty calculation contexts [81]. If the degrees of certainty (subjective probabilities) are known, p (S_1) and p (S_2) in two statements S_1 and S_2, then possible values of $p(S_1 \wedge S_2)$ form an interval

$$p = \left[\max(p_1 + p_2 - 1, 0), \min(p_1, p_2) \right]. \tag{2.18}$$

As a numerical estimate, it is natural to use the midpoint of this interval:

$$p_1 \wedge p_2 = \frac{1}{2}\left[\max\left(p_1 + p_2 - 1, 0\right) + \min\left(p_1, p_2\right)\right]. \tag{2.19}$$

Similarly, for the union operation, can be taken the midpoint of the corresponding interval:

$$p_1 \vee p_2 = \frac{1}{2}\left[\max\left(p_1, p_2\right) + \min\left(p_1 + p_2, 1\right)\right]. \tag{2.20}$$

The natural interpretation of such midpoint based operations has the property [81]:

$$\left(p_1 \wedge p_2\right) \wedge p_3 \neq p_1 \wedge \left(p_2 \wedge p_3\right) \quad \text{and} \tag{2.21}$$

$$\left(p_1 \vee p_2\right) \vee p_3 \neq p_1 \vee \left(p_2 \vee p_3\right), \tag{2.22}$$

both operations are non - associative!

There are many possible motivations behind using non - associative norms, such as an interesting class of fuzzy flip-flops [24] or preference and ranking in multicriteria decision making (see [25]). Some very enlightening examples for real-life related decision situations where one of the connectives has partial features of its dual pair (an intersection with partial union properties or vice versa) have been suggested in [126].

2.1.4 Fuzzy Flip-Flops (F^3s)

One of the less deeply explored applications of fuzzy sets and logic is the extension of traditional Boolean logic based digital circuitry towards “fuzzy digital circuits”. Fuzzy gates represent no scientific challenge as they are nothing else but physical realizations of the operations themselves, except for more complex gates like XOR. It is much more interesting to examine the possibility of extension of the elementary sequential circuits, the flip-flops. The most general are the J-K and D flip-flops. The next sections review the results already available in the literature including also a brief summary of the concept of binary J-K and D flip-flops.

2.1.4.1 Binary J-K Flip-Flops

The flip-flop is an electronic circuit which has two stable states. It is capable to work as a memory. All types of traditional binary flip-flop circuit, such as the most general J-K flip-flop store a single bit of information. These elementary units are the basic components of every synchronous sequential digital circuit. The flip-flop behavior can be described by truth/characteristic table and the characteristic equation, which gives the next output in terms of the input control signals and the current output. The truth table of a J-K flip-flop can be seen in Table 2.2.

J	K	$Q(t)$	$Q(t+1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

TABLE 2.2 TRUTH TABLE OF BINARY J-K FLIP-FLOP

The next state $Q(t+1)$ of a J-K flip-flop is characterized as a function of both the present state $Q(t)$ and the two present inputs $J(t)$ and $K(t)$. In the next, *J*, *K* and *Q* will be used instead of $J(t)$, $K(t)$ and $Q(t)$, respectively, as simplified notations. The minterm expression (DNF) of $Q(t+1)$ can be written as:

$$Q(t+1) = J\overline{Q} + \overline{K}Q + J\overline{K} \tag{2.23}$$

or simplified as

$$Q(t+1) = J\overline{Q} + \overline{K}Q \tag{2.24}$$

The third (redundant) term in the equation (2.23) is necessary for the elimination of (static) race hazards [60], [62]. The redundant terms are often needed to assume race-free dynamic performance. The equation (2.23) is well-known as the characteristic equation of J-K flip-flops. On the other hand, the equivalent maxterm expression (CNF) can be given by

$$Q(t+1) = (J + \overline{K})(J + Q)(\overline{K} + \overline{Q}) \tag{2.25}$$

or simplified as

$$Q(t+1) = (J + Q)(\overline{K} + \overline{Q}) \tag{2.26}$$

Using symbolic notations the binary J-K flip-flop logical block diagram based on equation 2.23 is represented in Figure 2.6. In the feedback loop the one step delay is made by a clock pulse controlled switch and a memory unit, which could be a D flip-flop [60].

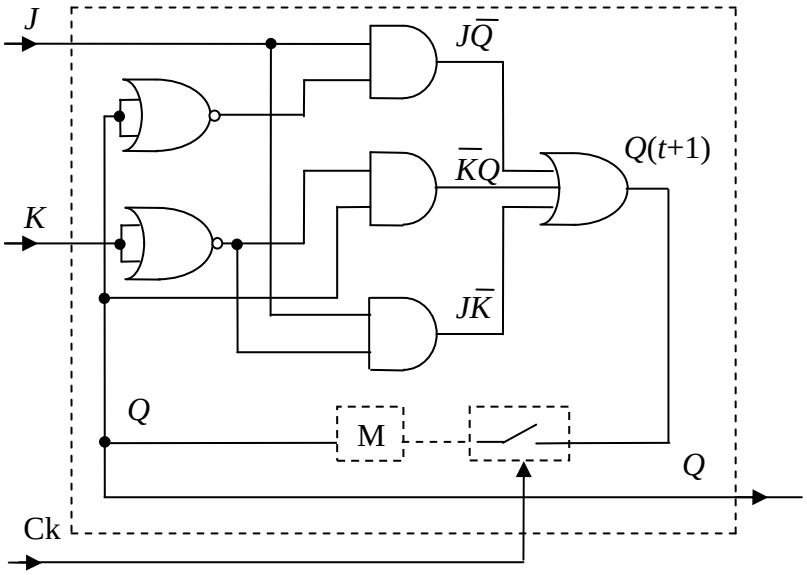


Figure 2.6 J-K flip-flop block diagram

The J-K flip-flop is one of the most widely used flip-flop because of its versatility.

2.1.4.2 Binary D Flip-Flops

The D flip-flop is a basic memory cell, a sequential electronic circuit which characteristics are summarized by the truth table (Table 2.3):

D	$Q(t)$	$Q(t+1)$
0	0	0
0	1	0
1	0	1
1	1	1

TABLE 2.3 TRUTH TABLE OF BINARY D FLIP-FLOP

The D flip-flop stores the value that is on the single data line and the most recent input D is remembered. It states that the next state $Q(t+1)$ of the output is equal to the value of input D in the present state. For the D flip-flop the characteristic equation is:

$$Q(t+1) = D \tag{2.27}$$

It is very easy to synthesize a D flip-flop from the J-K one by simply set J equal to the complement of K having an inverter before the K entrance.

The D flip-flop yields the simplest design procedure from all common flip-flops such as J-K, R-S, and T flip-flops. It is considered as the best choice where the area/pin count is important. The D flip-flop storage devices are very transistor efficient in VLSI technique. While in TTL and CMOS technology basically the J-K and D flip-flops are used, the PLA and PLD implementation needs only the D flip-flop. In this kind of implementation the J-K flip-flop should be used only by additionally circuitry. Static RAMs are built up also from flip-flops.

In a digital system the registers, which are used to store and/or shift data entered from external sources are generally used for temporary storage of the data. They are constructed by connecting a number of flip-flops in cascade. The storage and shift register is usually built up using D flip-flops. From both of J-K and D flip-flops are usually constructed different counter types another special type of sequential circuits.

2.1.4.3 Reset and Set Type Fuzzy J-K Flip-Flops

The four expressions, (2.23), (2.24), (2.25) and (2.26) are equivalent in Boolean logic. It is a rather obvious question, which of these four, or any other equivalent should be considered as the most proper fuzzy extension of the definitive equation of the very fundamental concept of fuzzy J-K flip-flop. Thus, there is no unambiguous way to introduce the concept of fuzzy J-K flip-flop. While normal forms are especially important for theoretical reasons minimal forms are equally important in the practice. This is why Hirota and Ozawa [41], [42] proposed two dual definitions of fuzzy flip-flops. They interpreted (2.24) as the definition for what they called “reset type fuzzy flip-flop”.

$$Q_R(t+1) = (J \wedge \overline{Q}) \vee (\overline{K} \wedge Q) \tag{2.28}$$

where the denotations for logic operations stand this time for Zadeh type fuzzy conjunction, disjunction and negation. The over bar denotes complement (e.g. $\overline{K} = 1 - K$), furthermore $\wedge$ and $\vee$ denote t-norm and t-conorm, $J, K, Q, Q(t+1) \in [0,1]$. J plays a role of a set input while K the reset one, as in case of binary flip-flops. In a similar way the definition of “set type fuzzy flip-flop” was obtained by re-interpreting (2.26) with fuzzy operations:

$$Q_S(t+1) = (J \vee Q) \wedge (\overline{K} \vee \overline{Q}) \tag{2.29}$$

As a matter of course, it is possible to substitute the Zadeh (standard) operations by any other reasonable fuzzy operation triplet (e.g. a De Morgan triplet), this way obtaining a multitude of various fuzzy flip-flop (F^3) pairs such as the algebraic F^3 introduced in [88] and [89].

Standard

Schweizer and Sklar [102] state that there is a one-to-one correspondence between t-norms and their dual t-conorms. Most definitions of F^3’s in the literature use such corresponding pairs. Often they also form a De Morgan triplet with standard complementation. Supposing that standard negation (2.13), conjunction (2.16), and

disjunction (2.17) are selected as fuzzy negation, t-norm, and t-conorm. (see Table 2.1). The expressions (2.28) and (2.29) are re-written as follows:

$$Q_R(t+1) = \max(\min(J, 1-Q), \min(1-K, Q)) \text{ and} \quad (2.30)$$

$$Q_S(t+1) = \min(\max(J, Q), \max(1-K, 1-Q)) . \quad (2.31)$$

The fuzzy flip-flop characterized by (2.30) is thus called standard reset type F^3, and the one given by (2.31) standard set type F^3.

Algebraic

In the case of standard complementation for fuzzy negation, algebraic product and algebraic sum for t-norm and t-conorm, respectively, the characteristic equations (2.28) and (2.29) can be rewritten as [88]

$$Q_R(t+1) = J + Q - 2JQ - KQ + JQ^2 + JQK - JQK^2 \quad (2.32)$$

$$Q_S(t+1) = J + Q - JQ - JKQ - KQ^2 + JKQ^2 \quad (2.33)$$

These interpretations of reset and set type F^3s inspired several authors to propose hardware implementation [88], which was done by using fuzzy gate circuits. The hardware implementation of algebraic norm based F^3 pointed out the methodology benefit resulting from the deep mathematical relation of fuzzy systems and neural networks.

Drastic

Let the intersection and union operations in equations 2.29 and 2.30 have drastic t-norm and t-conorm expressions labeled as i_{DR} and u_{DR}. Then, the drastic reset and set type fuzzy J-K flip-flop characteristic equations can be describing by its equations

$$Q_R(t+1) = \left(J \; i_{DR} \; (1-Q)\right) \; u_{DR} \; \left((1-K) \; i_{DR} \; Q\right) \quad (2.34)$$

$$Q_S(t+1) = \left(J \; u_{DR} \; Q\right) \; i_{DR} \; \left((1-K) \; u_{DR} \; (1-Q)\right) \quad (2.35)$$

Łukasiewicz

Based on Łukasiewicz fuzzy operations the Łukasiewicz reset and set type J-K flip-flops are defined as follows [16]

$$Q_R(t+1) = \min\left(\max(J-Q,0)+\max(Q-K,0),1\right) \tag{2.36}$$

$$Q_S(t+1) = \max\left(\min(J+Q,1)+\min(2-K-Q,1)-1,0\right) \tag{2.37}$$

Non - Associative

In [24] Fodor and Kóczy proposed a pair of non-associative operations for a new class of fuzzy flip-flops. Equations (2.28) and (2.29) can be extended by formulae

$$F_1(J,K,Q) = S_1\left(T_1\left(J,n(Q)\right),T_1\left(Q,n(K)\right)\right) \quad \text{and} \tag{2.38}$$

$$F_2(J,K,Q) = T_2\left(S_2\left(J,Q\right),S_2\left(n(K),n(Q)\right)\right) \tag{2.39}$$

where T_i are continuous t-norms, S_i are continuous t-conorms (i = 1, 2) and n is the strong negation (e.g. $n(Q) = 1-Q$). It was stated there that any F^3 satisfying:

P1: $F_i(0, 0, Q) = Q$,

P2: $F_i(0, 1, Q) = 0$,

P3: $F_i(1, 0, Q) = 1$,

P4: $F_i(1, 1, Q) = n(Q)$,

P5: $F_i(e, e, Q) = e$,

P6: $F_i(D, n(D), Q) = D$.

where $e = n(e)$ is the equilibrium belonging to n, and $D \in [0,1]$.

$F_1(J,K,Q)$ fulfils P6 iff there exist an automorphism ϕ of the unit interval such that

$$F_1(J,K,Q) = \phi^{-1}\left[\phi(J)(1-\phi(Q))+\phi(Q)(1-\phi(K))\right] \tag{2.40}$$

Similarly, for the ψ-transform

$$F_2(J,K,Q)=\psi^{-1}\left[\psi(J)(1-\psi(Q))+\psi(Q)\left(1-\psi(K)\right)\right] \tag{2.41}$$

Fodor and Kóczy [24] conclude that in this approach there is no need for associativity in modeling fuzzy conjunctions and disjunctions. They found other solutions of P6:

$$F_1(J,K,Q)=\frac{\min(J,1-Q)+\max(J-Q,0)+\min(Q,1-K)+\max(Q-K,0)}{2} \tag{2.42}$$

and

$$F_2(J,K,Q)=\frac{\max(J,Q)+\max(J+Q-1,0)+\max(1-Q,1-K)+\max(1-Q-K,0)}{2} \tag{2.43}$$

substitute T_i and S_i operations in the equations (2.38) and (2.39) by $T_1 = i_F$; $T_2 = i_L$; furthermore $S_1 = u_L$, and $S_2 = u_F$, where

$i_F=\frac{i_L+i_S}{2}$ and $u_F=\frac{u_L+u_S}{2}$.

The subscripts refer to the initial of the name of the operations - standard and Łukasiewicz - and that of the first author in case of the Fodor norms.

These equations were obtained by combining the standard (Zadeh) and the Łukasiewicz norms by the arithmetic mean in the inner part of the formula. The other parts use Łukasiewicz operations. Let the “Fodor-Kóczy type fuzzy flip-flop” be briefly denoted by F^4.

The extended formula for the set type F^4 (2.42) from [24] is however not the proper dual pair of (2.43), moreover, it is problematic in the sense of closeness for the unit interval. Thus, in section 3.3 is proposed a corrected definition for the set type formula and proved analytically the very special property of the reset and set formula being equivalent.

2.1.4.4 Unified Fuzzy J-K Flip-Flops

From a practical aspect it is confusing that reset and set type F^3s sometimes do have very different behavior. It was recognized that the reset and set equations cannot be easily used as element of memory module because of the asymmetrical nature. In the next the unified equation of reset and set type is presented as it was proposed in [89]. This equation

simultaneously involved both set and reset characteristics. Therefore, in order to extend the binary J-K flip-flop to a fuzzy flip-flop smoothly the newly defined function is applied

$$Q(t+1)=\begin{cases}\left(J\vee Q\right)\wedge\left(\left(1-K\right)\vee\left(1-Q\right)\right) \text{ if } \left(J\geq K\right)\\ \left(J\wedge\left(1-Q\right)\right)\vee\left(\left(1-K\right)\wedge Q\right) \text{ if } \left(J<K\right)\end{cases} \tag{2.44}$$

The unified formula of the fuzzy J-K flip-flop was expressed as follows:

$$Q(t+1)=(J\vee\overline{K})\wedge(J\vee Q)\wedge(\overline{K}\vee\overline{Q})\,. \tag{2.45}$$

The over bar denotes complement, furthermore $\wedge$ and $\vee$ denote t-norm and t-conorm. Equation (2.45) results also by substituting the binary operations in equation (2.25) with their corresponding fuzzy counterparts.

Standard

In his very first paper on fuzzy sets Zadeh proposed the standard (min-max) operators on fuzzy sets [120]. Using standard negation (2.13), and the standard norms (Table 2.1), equation (2.45) can be expressed as

$$\begin{aligned}&Q(t+1)=\left(J\ u_S\left(1-K\right)\right)\ i_S\ \left(J\ u_S\ Q\right)\ i_S\ \left(\left(1-K\right)\ u_S\left(1-Q\right)\right)= \\ &=\min(\max(J,(1-K)),\max(J,Q),\max((1-K),(1-Q)))\end{aligned} \tag{2.46}$$

the characteristic equation of the standard type fuzzy J-K flip-flop [41].

Algebraic

In the next, the unified equation (2.45) is a combination of reset and set type using the algebraic product and sum. The simplified form of the fundamental equation of the algebraic type fuzzy flip-flop [88] can be rewritten in the form

$$Q(t+1)=\left(J\ u_A\left(1-K\right)\right)\ i_A\ \left(J\ u_A\ Q\right)\ i_A\ \left(\left(1-K\right)\ u_A\left(1-Q\right)\right)=J+Q-JQ-KQ \tag{2.47}$$

This equation is considered the fundamental equation of the algebraic fuzzy flip-flop. It is remarkable how simple this combined equation is. In addition to its simplicity it represents a symmetrical, dual solution. In [41] Hirota and Ozawa proposed the hardware implementation of fuzzy J-K flip-flops base on algebraic norms using TTL logic in an analog configuration.

Drastic

Using the drastic norms (denoted by i_{DR} and u_{DR}) the maxterm form of the unified equation (2.45) can be defined as

$$Q(t+1) = \left(J \; u_{DR}\left(1-K\right)\right) \; i_{DR} \; \left(J \; u_{DR} \; Q\right) \; i_{DR} \; \left(\left(1-K\right) \; u_{DR}\left(1-Q\right)\right) \tag{2.48}$$

The drastic sum and product have a simple implement, having a narrow field of use due to their nontraditional discontinuities.

Łukasiewicz

Based on Łukasiewicz norms the corresponding unified $Q(t+1)$ definition is [93]

$$Q(t+1) = \left(J \; u_{L}\left(1-K\right)\right) \; i_{L} \; \left(J \; u_{L} \; Q\right) \; i_{L} \; \left(\left(1-K\right) \; u_{L}\left(1-Q\right)\right) = \tag{2.49}$$
$$= \max\left(0, \; -1+\max\left(0, \; -1+\min\left(1, \; 1+J-K\right)+\min\left(1, \; J+Q\right)\right)+\min\left(1, \; 2-K-Q\right)\right)$$

2.1.4.5 Choi's Fuzzy D Flip-Flops

As an alternative approach, Choi and Tipnis [11] proposed an equation which exhibits the characteristics of a kind of fuzzy D flip-flop, as follows:

$$Q(t+1) = D \wedge (D \vee Q) \wedge (\overline{Q} \vee D) \tag{2.50}$$

In the above mentioned paper the authors presented the design of the first D type fuzzy flip-flop using min-max type operation and standard negation that can be also used as a fuzzy memory cell. In the next this type of fuzzy D flip-flop is referred as fuzzy Choi D flip-flop.

2.2 Artificial Neural Networks

2.2.1 Introduction to Neural Networks

In this section a general description of neural networks is given, followed by their classification according to some important characteristics. To characterize a neural network it is necessary to specify the neurons employed, how they are interconnected, and the learning mechanism associated with the network.

Warren McCulloch and Walter Pitts proposed the first artificial neural network (ANN) model in 1943 [80], making a parallelism between Boolean algebra and nerve net behavior. In 1957 Rosenblatt developed the first neurocomputer, called perceptron. He proposed a learning rule for this first artificial neural network. His results were summarized in the very interesting book [94]. About the same time Widrow modeled learning from the point of view of minimizing the mean-square error (MSE) between the output of a different type of ANN processing element, the ADALINE – single layer neural network [35], [36] and the desired output vector over the set of patterns [114]. In the late 1980s as a promising approach to deal with some problems with satisfactory solutions were explored the neural networks, making ANNs one of the most widely used areas of research.

The ANN models have biological inspirations. The basic component of brain circuitry is a specialized cell called the neuron. A biological neuron may have as many as 10.000 different inputs and may send its output to many other neurons. Neurons are wired up in a 3-dimensional pattern. An interconnected neuron structure composed from a large number of neurons acting in parallel is capable of learning.

Function approximation, or regression analysis, classification, including pattern and sequence recognition, data processing and clustering are the main tasks to which ANNs are applied. Neural networks research covers different areas as character and face recognition [113], [115], [116], speech recognition and synthesis [48], robotics [49] and control systems [77], [92].

2.2.2 Basic Concepts, Models

2.2.2.1 The Neuron Model

The schematic structure of a neuron (Figure 2.7) can be interpreted as a model in which the weights modulate the input signals effect [64]. The argument, *y*, of the transfer function *f* depends on the bias *b* value and furthermore on the weighted inputs:

$y = w_{1,1}x_1 + w_{1,2}x_2 + ... + w_{1,n}x_n + b = w^T x + b$. The output signal is computed as

$$a = f(w^T x + b) = f(\sum_{i=1}^{n} w_i x_i) \tag{2.51}$$

where $w = (w_1, ..., w_n)^T \in \mathrm{R}^n$ is the weight vector.

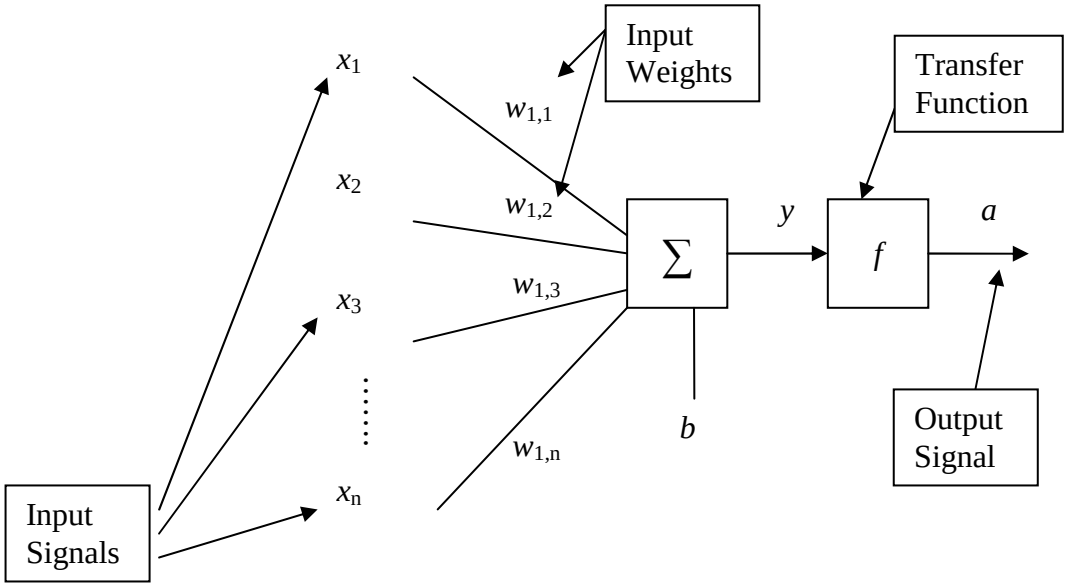


Figure 2.7 Schematic model of a neuron

The neuron output signal is computed as the weighted sum of the input signals transformed by the transfer function. The learning capability of an artificial neuron is achieved by adjusting the weights in accordance with a predefined learning algorithm.
The most common form of the transfer function used in the construction of neural networks is the following sigmoid activation function

$$f(t) = \frac{1}{1 + e^{-\sigma t}} \tag{2.52}$$

where σ is the steepness parameter, a positive constant, to control the curvature of the sigmoid curve. The sigmoid function natural way maps the values to the (0,1) interval and is easy differentiable. The first derivative satisfies the following equality

$$f'(t) = f(t)(1 - f(t)) \tag{2.53}$$

The strictly increasing s-shaped curve with a near-linear central response and saturating limits is showed in Figure 2.8.

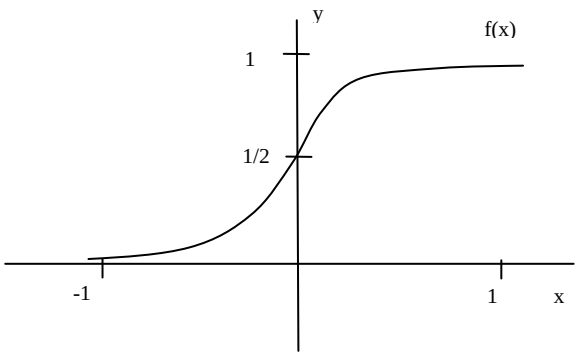


Figure 2.8 Typical sigmoid function

The sigmoid function $y = f(x)$ is defined as follows:

$$\frac{d^2 f(x)}{dx^2} \geq 0 \qquad \text{for } 0 \leq x \leq 1/2 \tag{2.54}$$

$$\frac{d^2 f(x)}{dx^2} \leq 0 \qquad \text{for } 1/2 \leq x \leq 1 \tag{2.55}$$

In the literature usually symmetrical and continuous differentiable activation functions are used. However for practical purposes not entirely symmetrical functions may be satisfactory and a few numbers of singularities (breakpoints) are often the case.

The neuron type can be classified into three groups from the point of view of where they are located [64]

- input neurons with one output, without signal processing role,
- output neurons forward the information toward the environment,
- hidden neurons with inputs and outputs connected exclusively with another neurons.

The nodes are organized into layers. Only neurons of the same type and connections are present. Analogously, the input, hidden and output layers can be distinguished. Typical ANNs often consist of intermediate layers known as hidden layers to facilitate the nonlinear computational capabilities of the networks model.

2.2.2.2 Main Types of Neural Networks

Artificial neural networks built from neurons are densely interconnected. The two main types of neural networks according to their connectivity are [118]:

- feedforward networks, which allow a flow of the signals from input to output. In this case exists a method which numbers all the nodes in the network. All the connections are from nodes with small numbers to nodes with larger numbers;
- recurrent networks if such a numbering method does not exist.

From another point of view the neural networks can be classified into:

- multilayer neural networks if in the network structure are hidden neurons;
- single layer neural networks, otherwise.

From the ANN system architecture point of view the perceptron is a typical single layer ANN. The most commonly used network architecture is the multilayer feedforward network. The network topology consists of the connection between neurons, the input and the output structure, which in general is represented by a diagram. Such a network has no direct connection between input and output layer and no connection within a layer. In general, more than three fully connected layers are used. The number of output units need not be equal to the number of input units and the number of hidden units per layer can be more or less than the input or output units.

2.2.2.3 Learning Paradigms [64]

During ANNs learning their weights can be modified such that a desired goal is reached. With respect to the learning mechanism, supervised, reinforcement and unsupervised learning can be mentioned [118]:

- Supervised neural networks operating with supervised learning and training strategies such as the Feedforward Backpropagation Network (FFBPN), Hopfield network,

Radial-Basis Function (RBF) network, etc. This is based on the direct comparison between the actual - and the target output of an ANN.

- Reinforcement learning, special case of supervised learning where the exact desired output is unknown.
- Unsupervised neural networks, which do not need any supervised learning and training strategies, including all kinds of self-organizing, self-clustering and learning networks such as Self-Organizing Map (SOM) or Adaptive Resonant Theory (ART) and so on. All this is based on the correlations among input data.

2.2.3 Multilayer Perceptrons

The MLP is a multilayer feedforward network, which usually organizes its units into several layers. The first layer is called input layer – the input signals are feedforwarded to the network while the last one is called output layer – the neurons forward the information to the outside word. The intermediate layers are called hidden layers [64].

The role of the network is to learn the association between input and output patterns or to find the structure of the input patterns. The learning process is achieved through the modification of the connection weights between units. The basic neural unit processes the input information into the output information using the computation and the transformation of the activation. A unit collects information provided by the external world (other units) to which it is connected with weighted connections. These weights multiply the input information.

The architecture (i.e. the pattern of connectivity) of the network along with the transfer function used by the neurons and the synaptic weights completely specifies the behavior of the network [64]. The following diagram (Figure 2.9) illustrates the architecture of an MLP network with three layers:

The w_{ij}^{h} denote the network weights between the input and hidden layer, and w_{jk}^{y} denote the network weights between the hidden and output layer. The total numbers of neurons in the input, hidden, and output layers are p, t, and m respectively. Normally, a sigmoid transfer function is adopted as activation function.

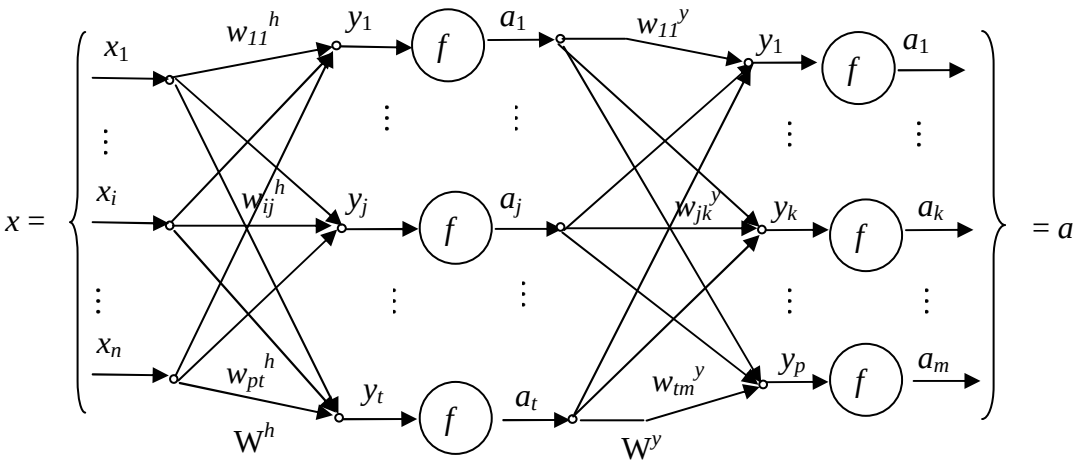


Figure 2.9 System architecture of a layered feedforward neural network

2.2.3.1 Training Multilayer Perceptrons with Gradient Type Algorithms

The MLPs training consist of three main processes [64]:

- the feedforward process of network training;
- the error evaluation process to calculate the errors between the calculated output values and the target output values;
- the backpropagation process of the errors for weight adjustments.

The training stops when a certain termination criterion is satisfied.

In designing and training a multilayer perceptron network the selection of the number of the hidden layer and the right decision of the neuron numbers in each hidden layer are the main issues. The next step is to find a global optimal solution that avoids local minima and it is also important to validate the neural network. Too few neurons lead to underfitting while too many neurons cause similarly undesired overfitting.

The Backpropagation Algorithm (BP)

The backpropagation training algorithm was first described by Rumelhart and McClelland in 1986 [98]; it was the first practical method for supervised learning neural networks. During training the network weights and biases are updated in the direction in which the performance function decreases most rapidly. The standard backpropagation is a gradient descent

algorithm, often converging very slowly in practical problems but having relatively modest memory requirements [2], [104].

The backpropagation algorithm requires that the activation function describing the neural network should be differentiable. If the activation function is sigmoid function, the network finds a local, not necessarily global error minimum, which in practice often works well. The network requires multiple invocations with different initial weights.

The Levenberg-Marquardt Algorithm (LM)

A more sophisticated technique for nonlinear function optimization is the Levenberg-Marquardt algorithm (LM) [66], [78] which is a second order gradient type training algorithm.

The Levenberg-Marquardt algorithm was designed to approach second order training speed without having to compute the Hessian matrix of the performance index at current values of the weights and biases [124].

If the error function is some kind of squared sum further the Jacobian matrix contains first derivatives of the network errors with respect to the weights and biases, then the Hessian matrix can be approximated as:

$$\mathbf{H} = \mathbf{J}^T \mathbf{J} \tag{2.56}$$

and the gradient can be computed as:

$$\underline{g} = \mathbf{J}^T \underline{e} \tag{2.57}$$

$\underline{e}$ is a vector of network errors. The Jacobian matrix determination requests less computation effort than the Hessian matrix. If $\underline{x}_k$ is the vector of current weights and biases, the parameter μ is a scalar controlling the behavior of the algorithm, then the new weight vector $\underline{x}_{k+1}$ can be adjusted as:

$$\underline{x}_{k+1} = \underline{x}_k - \left[\mathbf{J}^T \mathbf{J} + \mu \, \mathbf{I} \right]^{-1} \mathbf{J}^T \underline{e} \tag{2.58}$$

For $\mu = 0$, the algorithm follows Newton's method, when μ is high, this becomes gradient descent with small step size. The computation of the Jacobian matrix is described in [7].

This algorithm appears to be the fastest method for training moderate-sized feedforward neural networks. Usually it is used on single output networks with the sum squared error function and has memory requirements proportional to the number of weights in the network. The LM algorithm is best suited for function approximation problems where the network has fewer than one hundred weights and the approximation must be very accurate and in many cases it finds a solution even if it starts very far from the final minimum. The LM algorithm is one of the most popular training methods for feedfordward neural networks despite of its high memory requirements and high complexity [34].

2.3 Evolutionary Algorithms

One of the main paradigms of Computational Intelligence is Evolutionary Computing (EC). The EC algorithms are modeled after nature and cover genetic algorithms, genetic programming, evolutionary programming, evolutionary strategies, differential evolution and cultural evolution etc. These algorithms are introduced and different applications are presented in [22].

2.3.1 Genetic Algorithms (GA)

Genetic Algorithms represent the most widely used technique in the class of evolutionary computing. The original Genetic Algorithm (GA) was developed by Holland [44] and was based on the process of evolution of biological organisms (chromosomes). These optimization techniques simultaneously examine and manipulate a set of possible solutions. In genetic algorithms [83] the basic entity is called "chromosome". A chromosome with high fitness value will reproduce more intensively than one with low fitness. During natural selection the fittest biological entity survives. Genetic computations are aimed at finding the global maximum of a function of an arbitrary number of variables through performing a genetic - inclined search of space [4], [15], [38], [64] and [93].

The basic genetic algorithm consists of the following steps:

- Population initialization
- Fitness evaluation of each individual
- Rank the individuals in accordance with their fitness values - parent selection
- Crossover and mutation
- Evaluate the fitness of each individual
- Iterative execution on the new population until satisfactory performance is reached

However, GAs have the ability to converge to the global optimum but it is generally acceptable that convergence speed of GAs is quite slow, i.e. finding the global optimum with sufficient precision often takes a very long time [106]. Nevertheless, genetic algorithms are generally able to find reasonable “good” solutions too many problems with high complexity. The genetic algorithm typical flowchart can be seen in Figure 2.10.

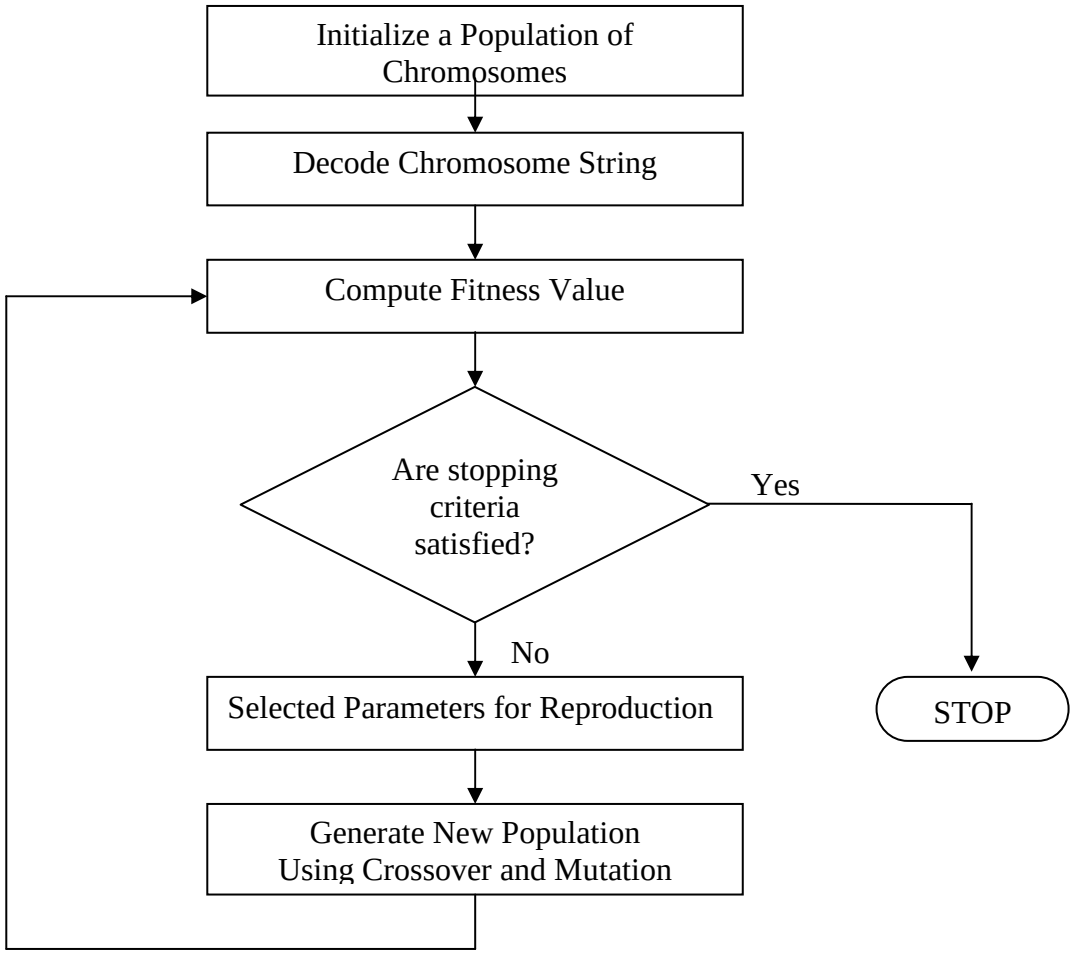


Figure 2.10 GA system flow diagram

2.3.1.1 Population Initialization

The initial phase of the algorithm refers to the way of coding the elements. The population is a collection of the chromosomes with the representation of a parameter set, usually represented as bit strings. In fact, the chromosome is a sequence of states. First, the population is initialized randomly. If the chromosome length is *m*, the possible number of different chromosome strings is 2^m. These individuals are evaluated using a specific fitness function [107].

2.3.1.2 Fitness Evaluation

The fitness value of a chromosome is computed with the help of the fitness evaluation function. The algorithm efficiency depends on the proper fitness function selection; in fact it is responsible for increasing the chromosome reproduction possibility [15]. The search is performed by modifying the binary string, increasing its fitness. The genetic algorithm uses the individuals with high fitness value to produce new generations of expectedly better solutions.

2.3.1.3 Parent Selection Scheme

The selection and reproduction capability of one string is in proportion with their fitness value [15]. Two parents are selected from each generation. The chromosomes with higher than medium fitness values are copied with a large probability while the chromosomes with weak fitness values are eliminated. A commonly adopted scheme a simple selection method for parent selection is the roulette-wheel parent selection technique. The wheel is split into sectors. The sector sizes are proportional with the respective fitness values. From this point of view even the strings with low fitness values may contain partly useful information. Even though their survival probabilities are low but their influence is still important.

2.3.1.4 Crossover and Mutation

The two main processes for reproduction in the GA are crossover and mutation [32]. Similarly like in the case of reproduction, in crossover technique there are many different mechanisms known from the literature. In general, in the crossover operation, a pair of parent

chromosomes is selected from the population and then between two randomly selected strings the exchange operation is applied to generate two new individuals. In this way, both new individuals will possess features from both parents.

In one-point crossover a random location is selected from the chromosome strings and chromosome elements beyond this crossover point are exchanged to form a new pair according to the crossover rate. Similarly, for two-point and uniform crossover multiple points are selected for crossover operations.

The mechanism of reproduction and crossover is surprising very simple, they consist of a random value generator, string copy, and the string part exchange. The crossover is a randomly selected process while the reproduction depends on the string fitness value.
During the mutation operation, a single chromosome is selected from the population, and only a particular element of this chromosome will be changed according to the mutation rate. Each member of the population might undergo a random change although with a small probability only.

The mutation operation is very useful in the information protection. With the help of mutation completely new bit strings can be created preventing the construction of uniform populations which are unable to reproduce, this way making the whole search space accessible.

Finally, after the crossover and mutation operations a new generation develops and the genetic cycle continues until a good solution is reached.

2.3.2 Pseudo-Bacterial Genetic Algorithm (PBGA)

The Pseudo-Bacterial Genetic Algorithm (PBGA) is a special kind of Genetic Algorithm [45]. Its core contains the bacterium which is able to carry a copy of a gene from a host cell and insert it into an infected cell. By this process, called bacterial mutation, the characteristics of a single bacterium can spread to the rest of the population, so this method mimics the process of microbial evolution [84]. A similar process to bacterial genetics is implemented in the PBGA. Nawa et.al. in [85] proposed this algorithm as a new approach combining a genetic algorithm with a local improvement mechanism inspired by a process in bacterial genetics, named bacterial operation or mutation.

Algorithm Description [29]

- Initial population creation

A generation of the initial population have been randomly created and evaluated, which consist of *N* chromosomes.

- Genetic Operations

a) Bacterial mutations are applied to each individual selected one by one. A number of copies (*M*) of the selected chromosome are created. They are called clones. The same randomly selected part or parts are mutated in each clone. After mutation all the clones and the original bacterium are evaluated by using an evaluation function. Next, the best individual among the *M* chromosomes is selected and the mutated part or parts is transferred to the other clones. This process mutation-evaluation-selection-replacement is repeated until each segment of the bacterium has been mutated and tested once. The best chromosome from the *M* individuals is selected to remain in the population and all other (*M*-1) clones are deleted. This process is applied to all chromosomes in the population. In fact, the bacterial mutation operation optimizes the chromosome of one bacterium.

b) Conventional genetic operations, the selection, reproduction and crossover are applied. The individuals with lower fitness value are eliminated, deleted and some randomly selected chromosomes from the remaining group are reproduced.

- Stop Condition

The bacterial mutation is repeated until a stop condition is satisfied (e.g. maximum number of generations is reached).

The Genetic Algorithm is efficient in improving local parts of chromosomes, while the PBGA when the chromosomes parts are in weak relationships with each other.

2.3.3 Bacterial Evolutionary Algorithm (BEA)

Nawa and Furuhashi had improved the PBGA [85], completed the bacterial mutation with a gene transfer operation which transfers information between different bacteria within the population. The proposed new method is called Bacterial Evolutionary Algorithm (BEA).

Algorithm Description

The first steps of BEA are similar to those of the PBGA mentioned before [29]. It starts with the generation of the initial population when a random bacterial population is created, then the bacterial mutation is applied to each bacterium one by one. New, randomly generated information is added to the bacteria using bacterial mutation. This cycle is repeated until all bacterium parts have been mutated and tested. The gene transfer operation is used instead of the selection - reproduction and crossover part of the PBGA. Finally, if the population satisfies a stop condition the algorithm ends, otherwise it returns to the bacterial mutation step. During gene transfer operation a recombination of genetic information between two bacteria is made. The main steps of the gene transfer operation are:

- The population is split into two equal parts: the superior and inferior half. The “better” individuals, with high fitness value belong to the superior part and the rest remained in the inferior one.
- One “source” chromosome is selected randomly from the superior half and another called “destination” from the inferior half.
- A randomly selected part (or by a predefined criteria) from the source chromosome is transferred to the destination chromosome.
- The above mentioned three steps are repeated for the numbers of the infections per generation times.

In conclusion, the bacterial mutation is working on one individual trying to optimize that; the gene transfer is applied to the whole bacterial population avoiding the local minima solutions. BEA has been successfully applied for example to rule learning [5] and feature selection [6].

2.3.4 Memetic Algorithm (MA)

The GAs is able to find the region of the global optimum but they are not well suited for fine-tuning structure close to optimal solutions [32]. Memetic Algorithms (MA) combine evolutionary and local search methods [82], they are also known as hybrid evolutionary algorithms because of the genetic local search.

The two main goals in the design of an optimization method are the *exploration* and *exploitation*, as discussed in [38] and [107]. While the exploration estimates the global optimum by searching through the entire search space, the exploitation makes searching effort

on its local neighborhood in order to produce a sufficiently accurate global optimum. The exploration has been made possible by using the randomness and the exploitation has lead to strict mathematical procedures. MA combines the exploration with the exploitation during the algorithm, obtaining an efficient optimum searcher [106].

Algorithm Description [38]

- Population Initialization
- Fitness value evaluation of each individual
- Rank the individuals in accordance with their fitness value

For each individual in the population redefine the individual through a local search

- Rank the individuals in accordance with their fitness value
- Select some individuals based on their ranking as candidates for crossover and mutation
- Evolve the population through crossover and mutation
- Evaluate the fitness of each individual
- Iterative execution on the new population until satisfactory performance is reached

2.3.5 Bacterial Memetic Algorithm (BMA)

The Bacterial Memetic Algorithm (BMA) is a recently developed technique [9]. This particular merger of evolutionary and gradient based (global and local search) algorithms is used rather successful for global optimization approaches, in particular by optimizing parameter values, and to improve function approximation performance. In bacterial algorithms the bacteria can transfer genes to other bacteria by bacterial mutation and gene transfer operations obtaining in this way suitable optimal solutions. The method interleaves the bacterial mutation (optimizes the rules in one bacterium) and the gene transfer operation (recombinants the genetic information between the chromosomes) with the Levenberg - Marquardt method to reach the local optimum.

Algorithm Description

The algorithm is based on the bacterial mutation, gene transfer, and the Levenberg-Marquardt method. The main steps of the algorithm are [29]:

- Population Initialization
- Bacterial mutation is applied for each individual
- The Levenberg-Marquardt method is applied for each individual
- Gene transfer operation is applied infection times per generation
- The procedure above is repeated from the bacterial mutation step until a certain stopping criterion is satisfied

Memetic algorithm has been successfully applied in engineering fields ranging from microarray gene selection [125], aerodynamic design [87], drug therapies design [86] to assignment problems [106].

2.3.6 Bacterial Memetic Algorithm with Modified Operator Execution Order (BMAM)

Hybrid evolutionary methods that combine genetic type algorithms with "classic" local search have been proposed to perform efficient global search. BMAM is an improved version of the BMA [28] which consist of bacterial mutation and the LM method for each individual and gene transfer operation for a partial population. The algorithms executes several LM cycles during the bacterial mutation after each mutation step saving some potential clones that could be lost otherwise.

Algorithm Description [29]

A. Modified mutation operation (applied for each individual):

- Each individual is selected one by one
- *M* clones are created from the selected individuals
- The same part or parts are selected randomly from the clones and mutated
- Some LM iterations are run after each bacterial mutation step
- The best clone is selected and transferred with its all parts to the other clones

- Choosing-mutation-LM-selection-transfer cycle is repeated until all the parts are mutated, improved and tested
- The best individual is remaining in the population all other clones are deleted
- This procedure is repeated until all the individuals are taking part in the modified bacterial mutation

B. Levenberg-Marquardt algorithm (applied for each individual)
C. Gene transfer operation (applied for a partial population)

The Levenberg-Marquardt algorithm in the optimization process often finds only the local minimum while the bacterial algorithm can avoid the local minima but finding a quasi-optimal result. The Bacterial Memetic Algorithm (BMA) resulted in better approximation properties in fuzzy modeling problems than the Bacterial Evolutionary Algorithm (BEA) (which outperformed the traditional Genetic Algorithms [8], [9]).
In [21] Eiben outlined the main problems in parameter calibration of evolutionary algorithms. He proposed a promising approach to calibrate the attributes defining (e.g., the parent selection method) and the parameter values (e.g., the mutation rate) which are chosen in an ad hoc manner.

2.4 Summary

This chapter has reveal the concepts, approaches and main results related to the research topics of the three main branches of Computational Intelligence, namely, fuzzy systems, neural networks and evolutionary computing. Among basic fuzzy concepts and comprehensive coverage of operations on fuzzy sets, fuzzy flip-flops have been paid extra attention as they represent the focal question of this Ph.D. dissertation. Some typical types of fuzzy flip-flop which will be used in comparison have been described in detail.

It is known that simple parametrical t-norms, furthermore the characteristic equations of fuzzy flip-flops based on these t-norms are uncomplicated for tuning and hardware realization. Rudas et al. [97] proposed the hardware implementation of a generation of parametric families of fuzzy connectives together with min-max, Łukasiewicz and drastic t-norms and t-conorms. In [123] Zavala et al. used FPGA technology to implement the above mentioned fuzzy t-norms into an 8 bit single circuit that allows operator selection.

In [43] Hirota and Pedrycz interpreted the unified equation of the fuzzy J-K flip-flop (2.45) as a collection of logic-oriented AND and OR neurons with binary (0-1) connections organized into a two layer topology with a single feedback loop. In the same paper the authors realized the hardware implementation of a general fuzzy flip-flop, set and reset type fuzzy J-K flip-flop based on standard and Łukasiewicz conjunctions. They developed the 8 bit fuzzy flip-flop design having the ability to create an *n*-bit logical fuzzy flip-flop.

Diamond et al. [16] consider the VLSI implementation of CMOS J-K fuzzy flip-flops, examining a general design environment for digital fuzzy implementation. Choi and Tipnis [11] present new designs of fuzzy NOT, OR and AND gates based on min-max operations.

Chapter 2 presents the neuron model, the architecture of the multilayer perceptrons, additionally the gradient type Backpropagation and Levenberg-Marquardt training algorithms.

The outlined evolutionary algorithms all belong to the Genetic Algorithm (GA) class and five subtypes, namely: Pseudo-Bacterial Genetic Algorithm (PBGA), Bacterial Evolutionary Algorithm (BEA), Memetic Algorithm (MA), Bacterial Memetic Algorithm (BMA) and Bacterial Memetic Algorithm with Modified Operator Execution Order (BMAM) are presented.

The GA is based on the process of evolution of biological organisms. The PBGA's core is the bacterial mutation. The BEA is supported by the gene transfer operation. Memetic Algorithms combine evolutionary and local search methods. The BMA combines BEA and Levenberg-Marquardt method. Finally, the BMAM exploits the Levenberg-Marquardt method more efficiently.

Chapter 3

Definitions and Properties of Several New Fuzzy Flip-Flops

The concept of fuzzy flip-flop was introduced in the middle of 1980's by Hirota (with his students) [41]. The Hirota Lab recognized the essential importance of the concept of a fuzzy extension of a sequential circuit and the notion of fuzzy memory. From this point of view they proposed alternatives for "fuzzifying" digital flip-flops. The starting elementary digital units were the binary J-K flip-flops. Their definitive equation was used both in the minimal disjunctive and conjunctive forms. As fuzzy connectives do not satisfy all Boolean axioms the fuzzy equivalents of these equations resulted in two non - equivalent definitions, "reset and set type" fuzzy flip-flops (F^3), using the concepts of fuzzy negation, t-norm and t-conorm operations. In [42] Hirota et al. recognized that the reset and set equations cannot be easily used as elements of memory module because of their asymmetrical nature. In their 1988's paper [89] Ozawa, Hirota and Kóczy proposed a unified form of the fuzzy J-K flip-flop characteristic equation involving the reset and set characteristics, based on min-max and algebraic norms. A few years later the hardware implementation of these fuzzy flip-flop circuits in discrete and continuous mode was presented by the Hirota Lab. The fuzzy flip-flop was proposed as basic unit in fuzzy register circuits. This chapter is organized as follows. Sections 3.1 and 3.2 define the concept of new fuzzy flip-flops based on Türkşen's Interval Valued Fuzzy Sets, furthermore the concept of reset and set type fuzzy J-K flip-flops based on Yager and Dombi norms. In the next part (section 3.3) it is proved analytically that in case of the modified Fodor fuzzy J-K flip-flop, the graphs of the reset and set types of flip-flop are identical which leads to the same flip-flop unit. In section 3.4 the concept of unified fuzzy J-K flip-flop based on seven popular norms is defined. In sections 3.5 and 3.6 the concept of two interpretations of fuzzy D flip-flop is defined. For all above mentioned new types of fuzzy flip-flop are determined their characteristic equations and are investigated their properties. It is shown that the F^3s may be classified into two groups: one presenting quasi s-shape transfer characteristics and the rest with non - sigmoid character.

3.1 Interval Valued Fuzzy J-K Flip-Flops

Let introduce the concept of Interval Valued Fuzzy Flip-Flops, which are originally for applications without any precise knowledge of the function that should be taken. In these cases it is convenient to represent the membership degree of each element to the fuzzy set by means of an interval. The binary J-K flip-flops Disjunctive Normal Form (DNF) and Conjunctive Normal Form (CNF) have been introduced in subsection 2.1.2. By applying the usual denotations for fuzzy negation, t-norm and t-conorm, the minterm expression (DNF) of $Q(t+1)$ (2.23) is re-written as follows:

$$Q_{DNF}(t+1) = \left(J \wedge \overline{Q}\right) \vee \left(\overline{K} \wedge Q\right) \vee \left(J \wedge \overline{K}\right) \tag{3.1}$$

further, in the same way, the maxterm expression (2.25) becomes

$$Q_{CNF}(t+1) = \left(J \vee \overline{K}\right) \wedge \left(J \vee Q\right) \wedge \left(\overline{K} \vee \overline{Q}\right) \tag{3.2}$$

Two more definitions of fuzzy flip-flops were obtained next to the reset (2.28) and set (2.29) ones (called “normalized reset type and set type” F^3-s) as (3.3), (3.4).

$$Q_{DMF}(t+1) = Q_R(t+1) = \left(J \wedge \overline{Q}\right) \vee \left(\overline{K} \wedge Q\right) \tag{3.3}$$

$$Q_{CMF}(t+1) = Q_S(t+1) = \left(J \vee Q\right) \wedge \left(\overline{K} \vee \overline{Q}\right) \tag{3.4}$$

It is questionable, of course, whether these new definitions play any more important role in the practice. Reset and set type behaviors are generally different and none of them possess the ˝nice˝ symmetrical behavior of the original J-K flip-flop. This is why a symmetrical F^3 was proposed earlier by combining the two minimal forms in the equilibrium point [88]. Only one exceptional combination of operations has been found so far where the two types completely coincided (see section 3.3).

To combine two different extensions of the original definition it might also choose a single representative point of each interval corresponding to the Interval Valued Fuzzy Sets (IVFS) obtained from the two normal forms. As the most obvious representative, the midpoint is proposed here.

In all earlier publications on F^3-s, however, minterm and maxterm expressions played important roles rather than the normal forms. Thus, a new interval limited by the minterm (DMF) and the maxterm (CMF) expressions, given in Eqs. (3.3) and (3.4) is proposed:

$$MIVFS(\cdot)=\left[DMF(\cdot),CMF(\cdot)\right] \tag{3.5}$$

where $(\cdot)$ represents a particular expression [109]. In order to obtain a single fuzzy flip-flop the midpoint of *MIVFS* is considered as the definition of the new type standard F^3.

3.1.1 Standard

The fuzzy extensions of DNF and CNF forms of fuzzy J-K flip-flops based on standard norms (standard t-norm is denoted by i_S, and the standard t-conorm by u_S), and standard complementation (2.13) are defined by

$$Q_{DNF}(t+1)=\left(J\ i_S\ (1-Q)\right)\ u_S\ \left((1-K)\ i_S\ Q\right)\ u_S\ \left(J\ i_S\ (1-K)\right)= \tag{3.6}$$
$$=\max\left(\min\left(J,(1-Q)\right),\min\left((1-K),Q\right),\min\left(J,(1-K)\right)\right)$$
$$Q_{CNF}(t+1)=\left(J\ u_S\ (1-K)\right)\ i_S\ \left(J\ u_S\ Q\right)\ i_S\ \left((1-K)\ u_S\ (1-Q)\right)= \tag{3.7}$$
$$=\min\left(\max\left(J,(1-K)\right),\max\left(J,Q\right),\max\left((1-K),(1-Q)\right)\right)$$

the corresponding reset and set interpretation

$$Q_{DMF}(t+1)=Q_R(t+1)=\left(J\ i_S\ (1-Q)\right)\ u_S\ \left((1-K)\ i_S\ Q\right)= \tag{3.8}$$
$$=\max\left(\min\left(J,(1-Q)\right),\min\left((1-K),Q\right)\right)$$
$$Q_{CMF}(t+1)=Q_S(t+1)=\left(J\ u_S\ Q\right)\ i_S\ \left((1-K)\ u_S\ (1-Q)\right) \tag{3.9}$$
$$=\min\left(\max\left(J,Q\right),\max\left((1-K),(1-Q)\right)\right)$$

Figure 3.1 depicts the DNF and CNF furthermore DMF (labeled as R) and CMF (labeled as S) values of standard fuzzy J-K flip-flops for various values of *K* and *J* = 0. The midpoints of the intervals (DNF+CNF)/2 denoted by IVFS and (DMF+CMF)/2 denoted by (R+S)/2 are showed.

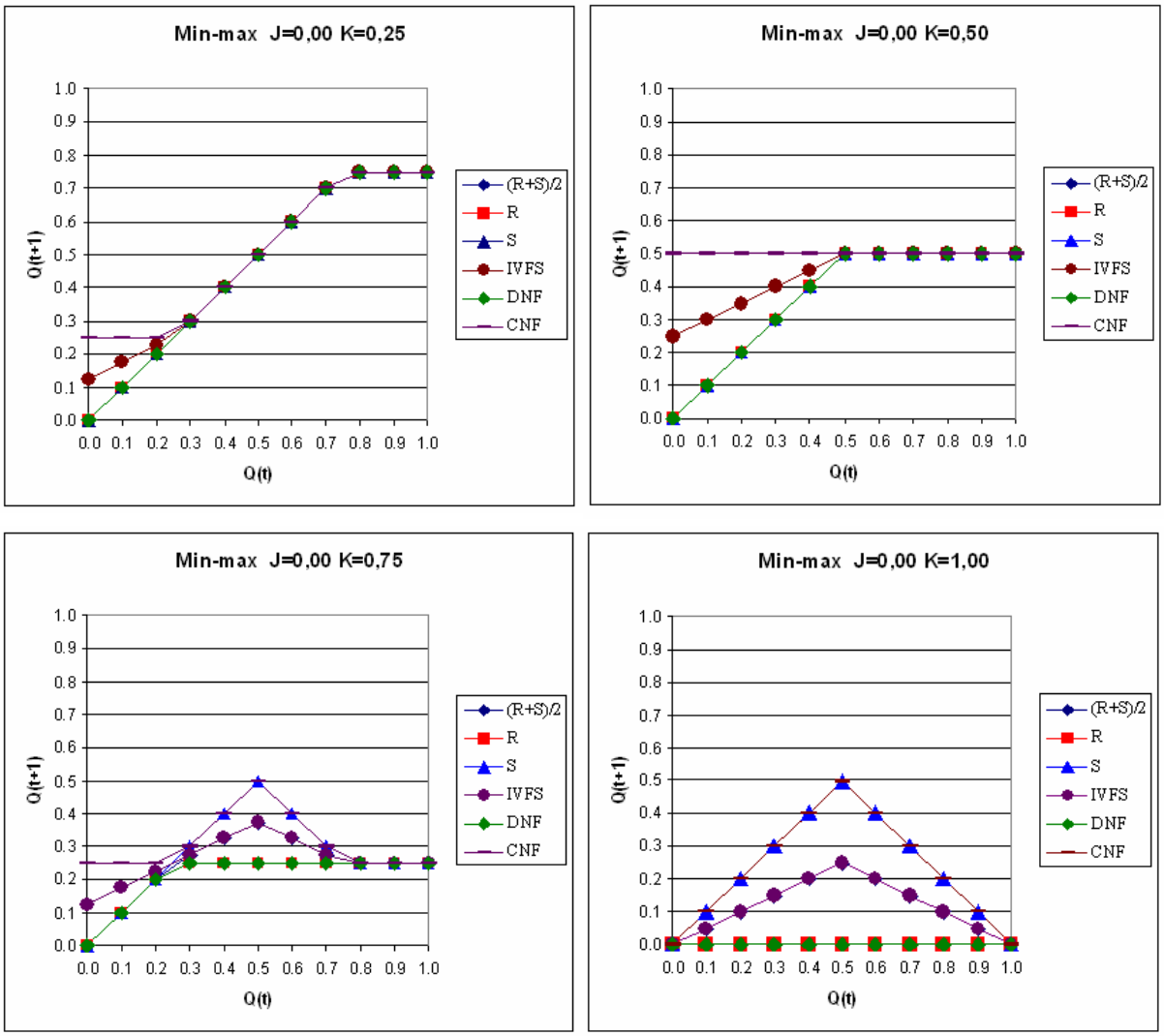


Figure 3.1
Interval valued new J-K F^3s based on min-max norms

3.1.2 Algebraic

In case of another special conjugate pair of t-norm and t-conorm with the standard complementation (2.13), namely the algebraic product and algebraic sum (Table 2.1) the expressions (3.1) and (3.2) are re-written as follows:

$$Q_{DNF}(t+1) = \left(J\ i_A\ (1-Q)\right)\ u_A\ \left((1-K)\ i_A\ Q\right)\ u_A\ \left(J\ i_A\ (1-K)\right) = \tag{3.10}$$

$$= J(1-K) + J(1-Q) + (1-K)Q - J(1-K)(1-Q)Q -$$
$$-J(1-K)\left(J(1-Q) + (1-K)Q - J(1-K)(1-Q)Q\right)$$

$$Q_{CNF}(t+1) = \left(J\ u_A\ (1-K)\right)\ i_A\ \left(J\ u_A\ Q\right)\ i_A\ \left((1-K)\ u_A(1-Q)\right) = \tag{3.11}$$

$$= \left(1+J-J(1-K)-K\right)\left(2-K-(1-K)(1-Q)-Q\right)\left(J+Q-JQ\right) \quad \text{and}$$

$$Q_{DMF}(t+1) = Q_R(t+1) = (J\ i_A\ (1-Q))\ u_A\ ((1-K)\ i_A\ Q) = \tag{3.12}$$

$$= J(1-Q) + (1-K)Q - J(1-K)(1-Q)Q$$

$$Q_{CMF}(t+1) = Q_S(t+1) = (J\ u_A\ Q)\ i_A\ ((1-K)\ u_A\ (1-Q)) = \tag{3.13}$$

$$= (2-K-(1-K)(1-Q)-Q)(J+Q-JQ)$$

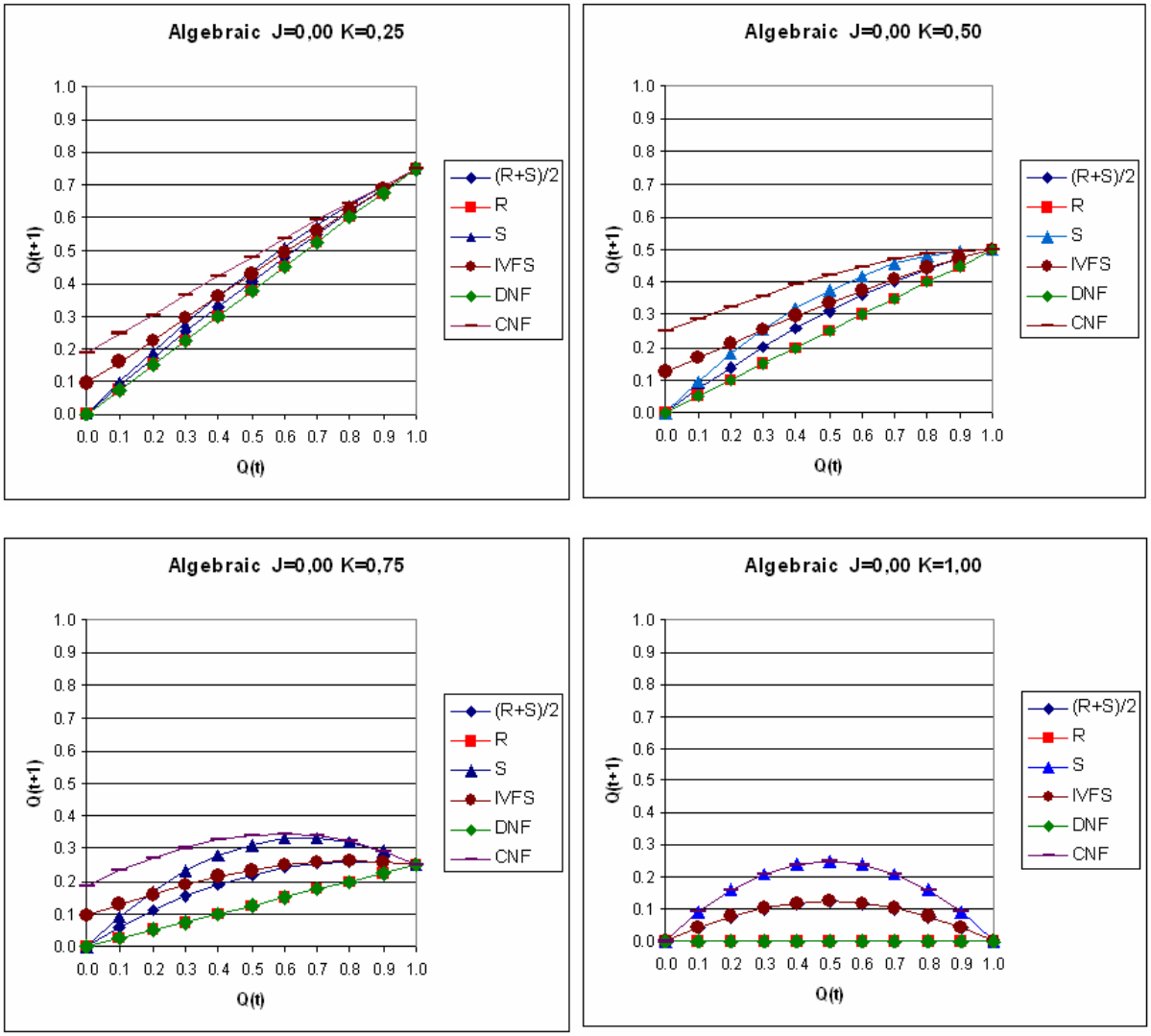


Figure 3.2
Interval valued new J-K F^3s based on algebraic norms

Figure 3.2 presents the graphs corresponding to algebraic type new interval valued F^3s. Studying the behavior of new interval valued fuzzy J-K flip-flops based on min-max and algebraic norms (Figures 3.1 and 3.2), it is remarked that in the case of $J = K$, the values of the next state $Q(t+1)$ are equal in every 4 focused case. If $Q(t) = 0$, the starting values of the piecewise linear characteristics are equal to the value of J. If $Q(t) = 1$ the endpoint values for

all 4 graphs are identical with the negated values of *K*. When the values of *J* and *K* are complementary the diagrams are symmetrical. The characteristics belonging to the intermediate cases show the obvious situation when the reset type curves always go below the set type ones. In these cases the graphs belonging to the DNF and CNF values represent the extreme cases. This, so called Türkşen interval is the largest when $J = 1$, $K = 0$, and "vice versa". The characteristics representing the (DNF+CNF)/2 and (DMF+CMF)/2 curves are naturally in the middle. *MIVFS* is the narrower interval given by the inequality $DNF(\cdot) \subseteq DMF(\cdot) \subseteq CMF(\cdot) \subseteq CNF(\cdot)$. Comparing the behavior of the algebraic interval valued fuzzy J-K flip-flop (Figure 3.2), with the standard one (Figure 3.1), it can be seen, that the algebraic operations produced smooth (differentiable) curves and surfaces with no breakpoints or lines at all. The border characteristics of the Türkşen interval are in this case also the graphs corresponding to DNF and CNF. This interval represents a larger surface than in the case discussed first. It is clearly demonstrated that Türkşen's statement discussed in subsection 2.1.2 holds. Completely new is the fact that in case of $J, K \in (0,1)$, the values of Q (t+1) for different values of $Q(t) \in [0,1]$ are not equal. The appearance of this interval in the real fuzzy cases for *J* and *K* introduces a new concept, a truly interval valued fuzzy flip-flop.

Generally, it may be remarked, that MIVFS reflects the typical behavior of fuzzy flip-flop more obviously in both cases and closes around the average behavior in a narrower stripe while the original IVFS opens up wider in the areas where real fuzziness occurs intensively. It remains an open question which of the two new average types F^3-s will be more applicable for practical purposes.

3.2 Reset and Set Type Fuzzy J-K Flip-Flops

The concept of reset and set type fuzzy J-K flip-flops based on Yager, and Dombi operations is defined. In the next their characteristic equations have been determinate and their properties have been investigated. Additionally is showed that the reset type's curves always go below the set ones. The negation used throughout the whole work is the strong negation (2.13).

3.2.1 Yager

Several classes of functions have been proposed whose individual members satisfy all the axiomatic requirements for the fuzzy union and neither, one, or both of the optional axioms. One of these classes of fuzzy unions is known as the Yager class. Substitute the Yager intersection and union formula given in Table 2.1 in equations (2.28) and (2.29) the solutions for novel Yager reset and set type fuzzy J-K flip-flop are:

$$Q_R(t+1) = \left(J \; i_Y \; (1-Q)\right) \; u_Y \; \left((1-K) \; i_Y \; Q\right) = \tag{3.14}$$

$$= \min\left(1, \left(\left(1-\min\left(1, \left((1-J)^w + Q^w\right)^{1/w}\right)\right)^w + \left(1-\min\left(1, \left(K^w + (1-Q)^w\right)^{1/w}\right)\right)^w\right)^{1/w}\right)$$

$$Q_S(t+1) = \left(J \; u_Y \; Q\right) \; i_Y \; \left((1-K) \; u_Y \; (1-Q)\right) = \tag{3.15}$$

$$= 1-\min\left(1, \left(\left(1-\min\left(1, \left(J^w + Q^w\right)^{1/w}\right)\right)^w + \left(1-\min\left(1, \left((1-K)^w + (1-Q)^w\right)^{1/w}\right)\right)^w\right)^{1/w}\right)$$

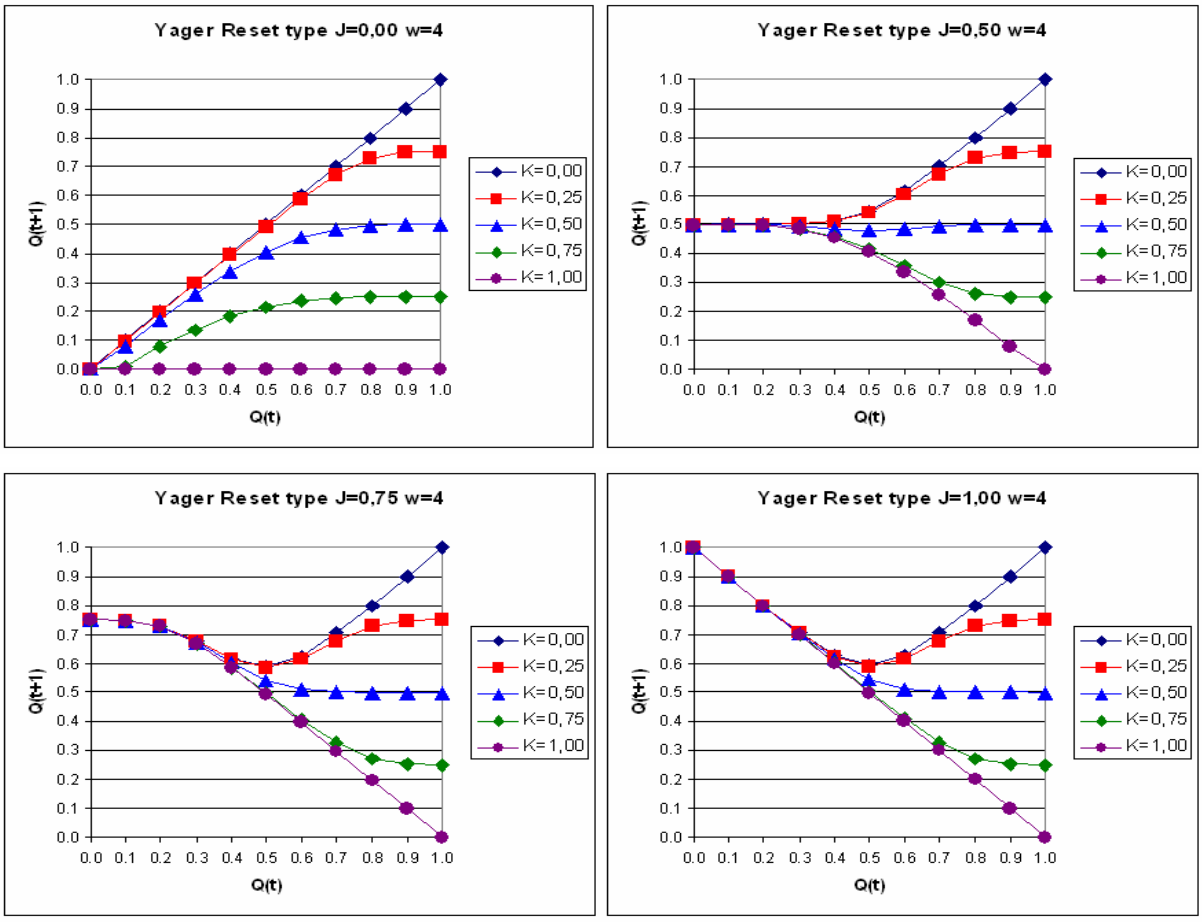


Figure 3.3
Reset type Yager J-K F[3]s

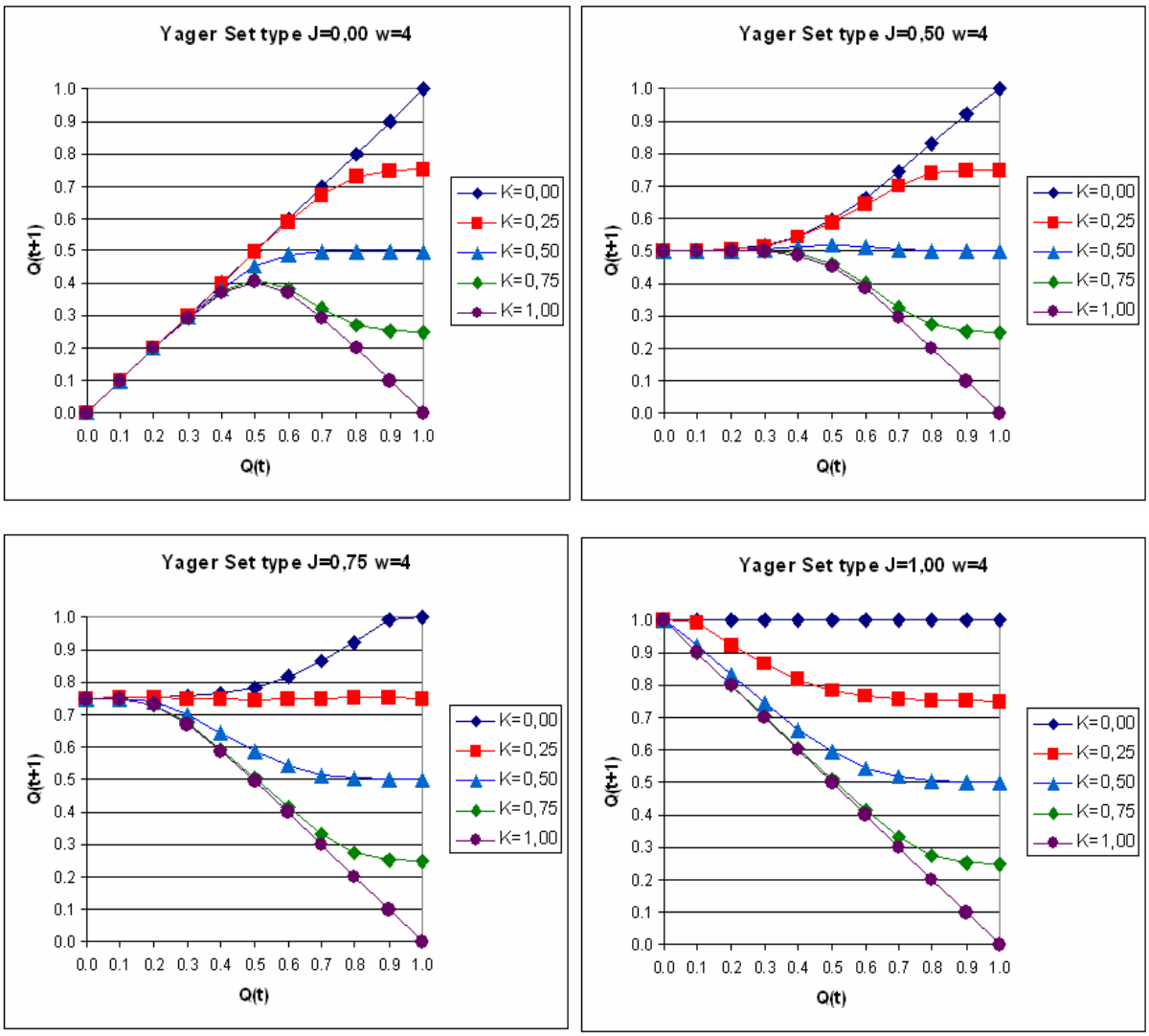


Figure 3.4
Set type Yager J-K F^3s

The main feature of these norms is that as parameter *w* varies among 0 and infinity the t-norms (and t-conorms) span the space of fuzzy operations between the drastic product (drastic sum) and the minimum (maximum) [56].

Figures 3.3 and 3.4 present the diagrams for Yager fuzzy J-K flip-flops for a typical case when the parameter of Yager norm is equal to four ($w = 4$) for various values of *J* and *K*. The real curves are here smooth.

3.2.2 Dombi

The novel Dombi reset and set type F^3 is defined as:

$$Q_R(t+1) = \left(J \; i_D \; (1-Q)\right) \; u_D \; \left((1-K) \; i_D \; Q\right) = \tag{3.16}$$

$$= \frac{1}{1+\left(\left(\left(\left(-1+\frac{1}{J}\right)^{\alpha}+\left(-1+\frac{1}{1-Q}\right)^{\alpha}\right)^{1/\alpha}\right)^{-\alpha}+\left(\left(\left(-1+\frac{1}{1-K}\right)^{\alpha}+\left(-1+\frac{1}{Q}\right)^{\alpha}\right)^{1/\alpha}\right)^{-\alpha}\right)^{-1/\alpha}}$$

$$Q_S(t+1) = \left(J\ u_D\ Q\right)\ i_D\ \left(\left(1-K\right)\ u_D\ \left(1-Q\right)\right) = \qquad (3.17)$$

$$= \frac{1}{1+\left(\left(\left(\left(-1+\frac{1}{J}\right)^{-\alpha}+\left(-1+\frac{1}{Q}\right)^{-\alpha}\right)^{-1/\alpha}\right)^{\alpha}+\left(\left(\left(-1+\frac{1}{1-K}\right)^{-\alpha}+\left(-1+\frac{1}{1-Q}\right)^{-\alpha}\right)^{-1/\alpha}\right)^{\alpha}\right)^{1/\alpha}}$$

Figures 3.5 and 3.6 depict typical values in the Dombi operator case when the Dombi parameter is α = 4. These curves are also smooth. In the expressions of Dombi fuzzy operations (Table 2.1) if a = 0 or b = 0 the value is understood in limit.

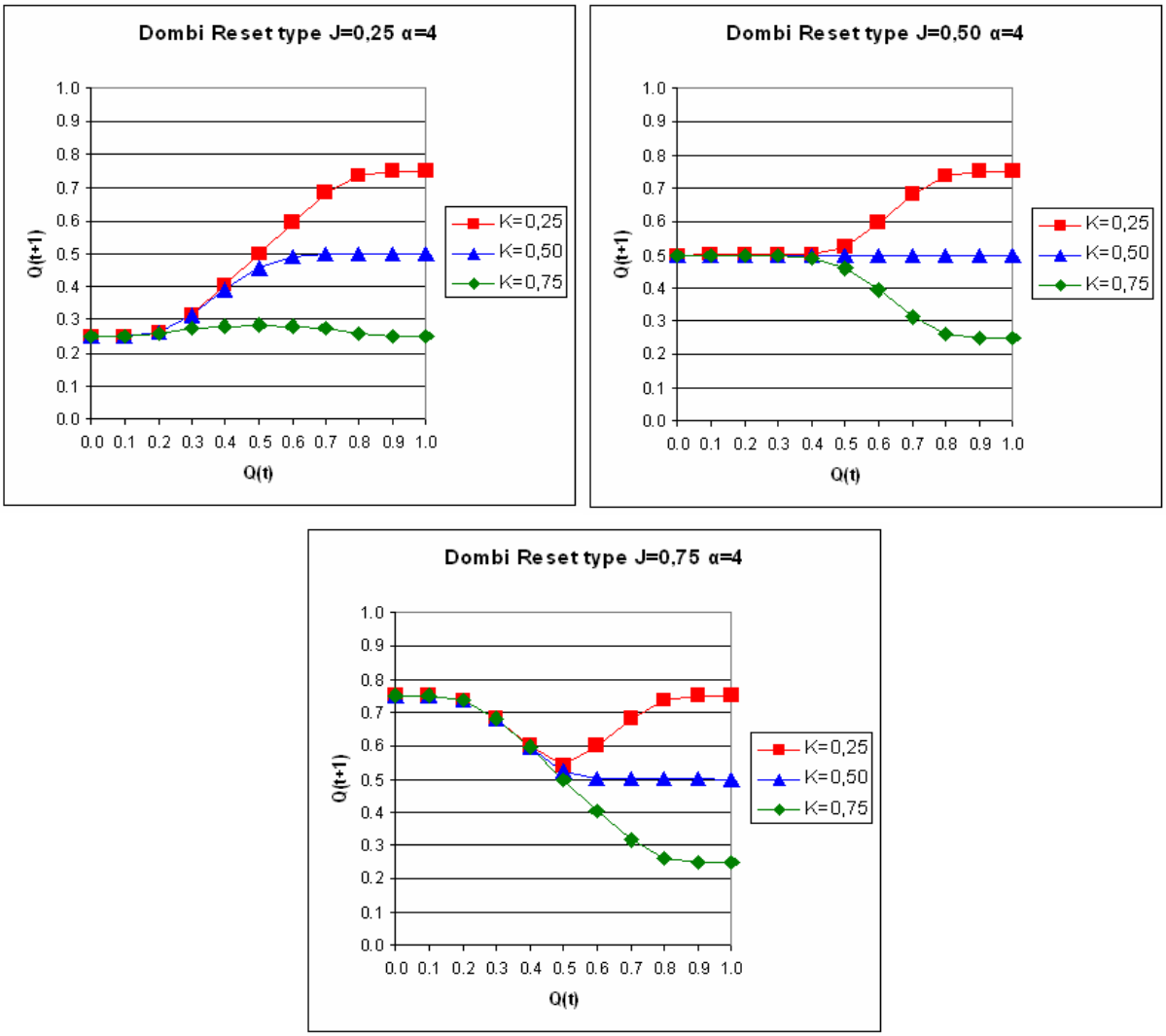


Figure 3.5
Reset type Dombi J-K F^3s

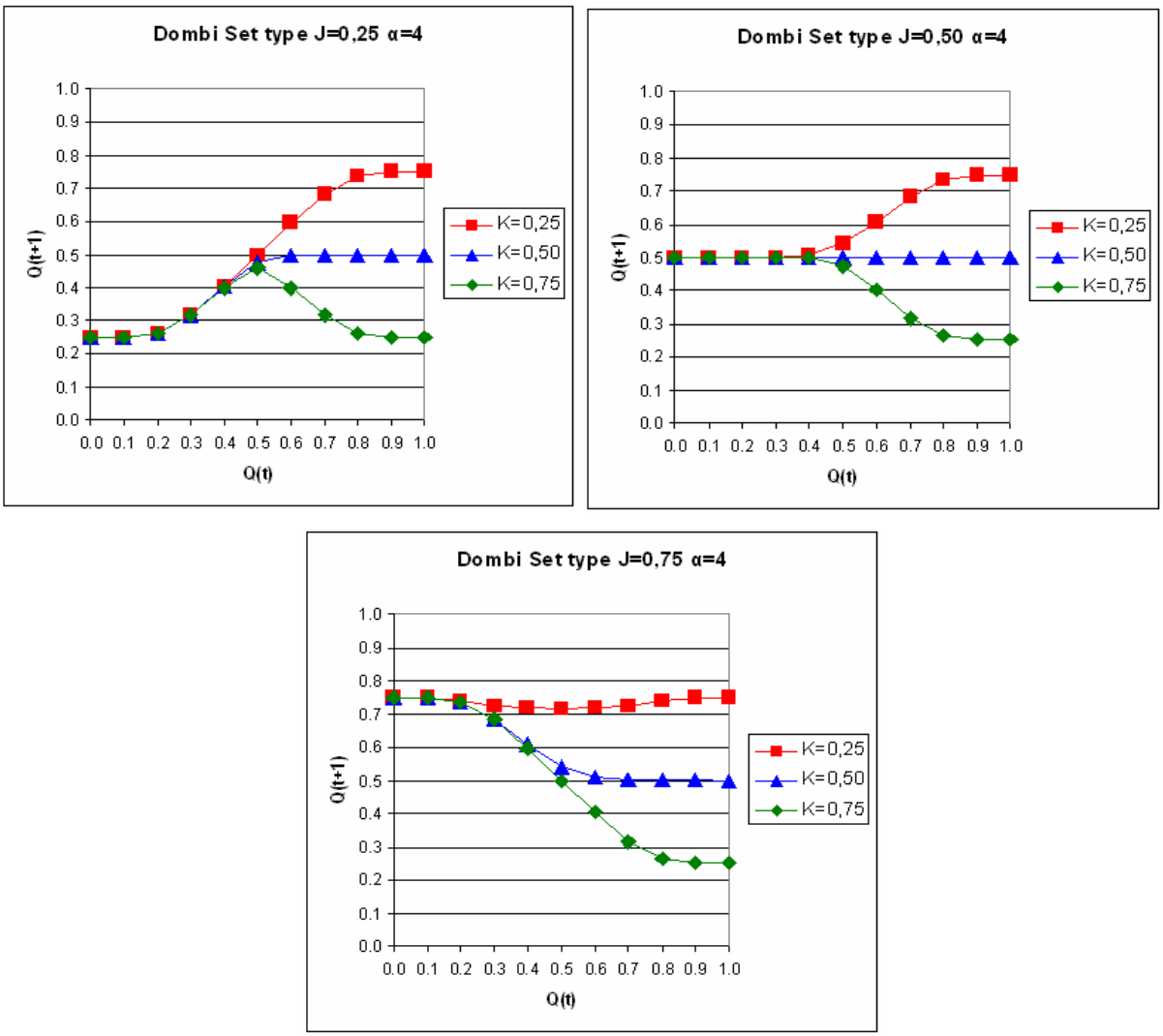


Figure 3.6
Set type Dombi J-K F^3s

It can be state, that evaluating the curves belonging to these two, reset and set next states (see Figures 3.3-3.6) clearly demonstrate the relation between $Q_R(t+1)$ and $Q_S(t+1)$:

$$Q_S(t+1) - Q_R(t+1) = \left[J(1-K) + K(1-J)\right]Q(1-Q) \geq 0$$

thus $Q_R(t+1) \leq Q_S(t+1)$ as the reset types curves always go below the set ones.

3.3 The Modified Non - Associative Fuzzy J-K Flip-Flop

In the first result published in the field of fuzzy flip-flops based on non - associative operations by Fodor and Kóczy [24], the proposed formula for the set type F^4 (2.43) is however not the proper dual pair of reset one (2.42), moreover, it is problematic in the sense of closeness for the unit interval. The “Fodor-Kóczy type fuzzy flip-flop” is briefly denoted

by F^4. Thus in the next a corrected definition for the set type formula is proposed by the author, as follows:

$$Q_S(t+1) = \frac{\max(J,Q) + \max(1-K,1-Q) - 1}{2} + \frac{\min(J+Q,1) + \min(2-Q-K,1) - 1}{2} \qquad (3.18)$$

Comparing this corrected form of the set type F^4 (3.18) with the reset type form (2.42), a surprising result was obtained. There is only one F^4 in this particular case as the two formulas are equivalent. This fact was strongly suggested by the simulation results obtained in [69], now is presented the exact proof as in [70]. The modified Fodor fuzzy J-K flip-flop introduced satisfies the very special property of the reset and set formula being equivalent:

$Q_R{}^{F^4} = Q_S{}^{F^4}$, thus there is only one (symmetrical, corrected) F^4.

Proof

It has to be proven that

$$Q_R(t+1) = Q_S(t+1) \qquad (3.19)$$

for every possible combination of J, K, Q.

Substitute (2.42) and (3.18) the following equivalent equation was obtained to prove:

$$\begin{aligned} &\min(J,1-Q) + \max(J-Q,0) + \min(Q,1-K) + \max(Q-K,0) = \\ &\max(J,Q) + \max(1-K,1-Q) + \min(J+Q,1) + \min(2-K-Q,1) - 2. \end{aligned} \qquad (3.20)$$

Any variable and their negated are symmetrical to the equilibrium e = 0.5. Consequently, for describing a case it is sufficient to tell which one of the ponated or negated version of each of the three variables is less or equal then e. The 8 main cases to be considered are as follows:

1) J, K, Q
2) $\overline{J}, \overline{K}, \overline{Q}$
3) $\overline{J}, K, Q$
4) $J, \overline{K}, Q$
5) $J, K, \overline{Q}$
6) $\overline{J}, \overline{K}, Q$
7) $\overline{J}, K, \overline{Q}$
8) $J, \overline{K}, \overline{Q}$

Each of these 8 cases has 3! = 6 sub-cases depending on the sequence of these three. For the 6 values J, K, Q, $\overline{J}$, $\overline{K}$, $\overline{Q}$, the total number of all possible combinations is theoretically 3! x 2^3 = 48. The 48 cases are not all essentially different. In the next table these combinations will be discussed so that some sub-cases can be merged in the sense that Q_R and Q_S are identical. The total number of essentially different sub-cases is 13. The first column of the table contains the serial number of the essentially different sub-case; the second column describes the inequality conditions applying for the given essential sub-case, while the third column gives the identical value of $2Q_R$ and $2Q_S$ in the given sub-case. The deduction of these results is omitted here for the sake of saving space.

TABLE 3.1 ESSENTIALLY DIFFERENT SUBCASES FOR THE F^4

Case#	Conditions	$2Q_R(t+1) = 2Q_S(t+1)$
1.	$J, K \le Q, \overline{Q}$	$J - K + 2Q$
2.	$J, \overline{K} \le Q, \overline{Q}$ $K, \overline{J} \le Q, \overline{Q}$	$1 + J - K$
3.	$\overline{J}, \overline{K} \le Q, \overline{Q}$	$2 + J - K - 2Q$
4.	$J, Q \le K, \overline{K}$	$J + Q$
5.	$J, \overline{Q} \le K, \overline{K}$	$1 + J - 2K + Q$
6.	$\overline{J}, Q \le K, \overline{K}$	$1 + J - Q$
7.	$\overline{J}, \overline{Q} \le K, \overline{K}$	$2 + J - 2K - Q$
8.	$K, Q \le J, \overline{J}$	$2J - K + Q$
9.	$K, \overline{Q} \le J, \overline{J}$	$1 - K + Q$
10.	$\overline{K}, Q \le J, \overline{J}$	$1 + 2J - K - Q$
11.	$\overline{K}, \overline{Q} \le J, \overline{J}$	$2 - K - Q$
12.	$Q \le J, K, \overline{J}, \overline{K}$	$2J$
13.	$\overline{Q} \le J, K, \overline{J}, \overline{K}$	$2 - 2K$

In all cases the third column contains a single expression, so $Q_R{}^{F^4}$ is always identical with $Q_S{}^{F^4}$. Q.e.d.

It has been shown that the modified F^4 proposed by the author in [69] is indeed a single F^3 with nice dual and symmetrical behavior. Figure 3.7 presents some diagrams illustrating the behavior of F^4 for some typical values of *J*, *K* and *Q*.

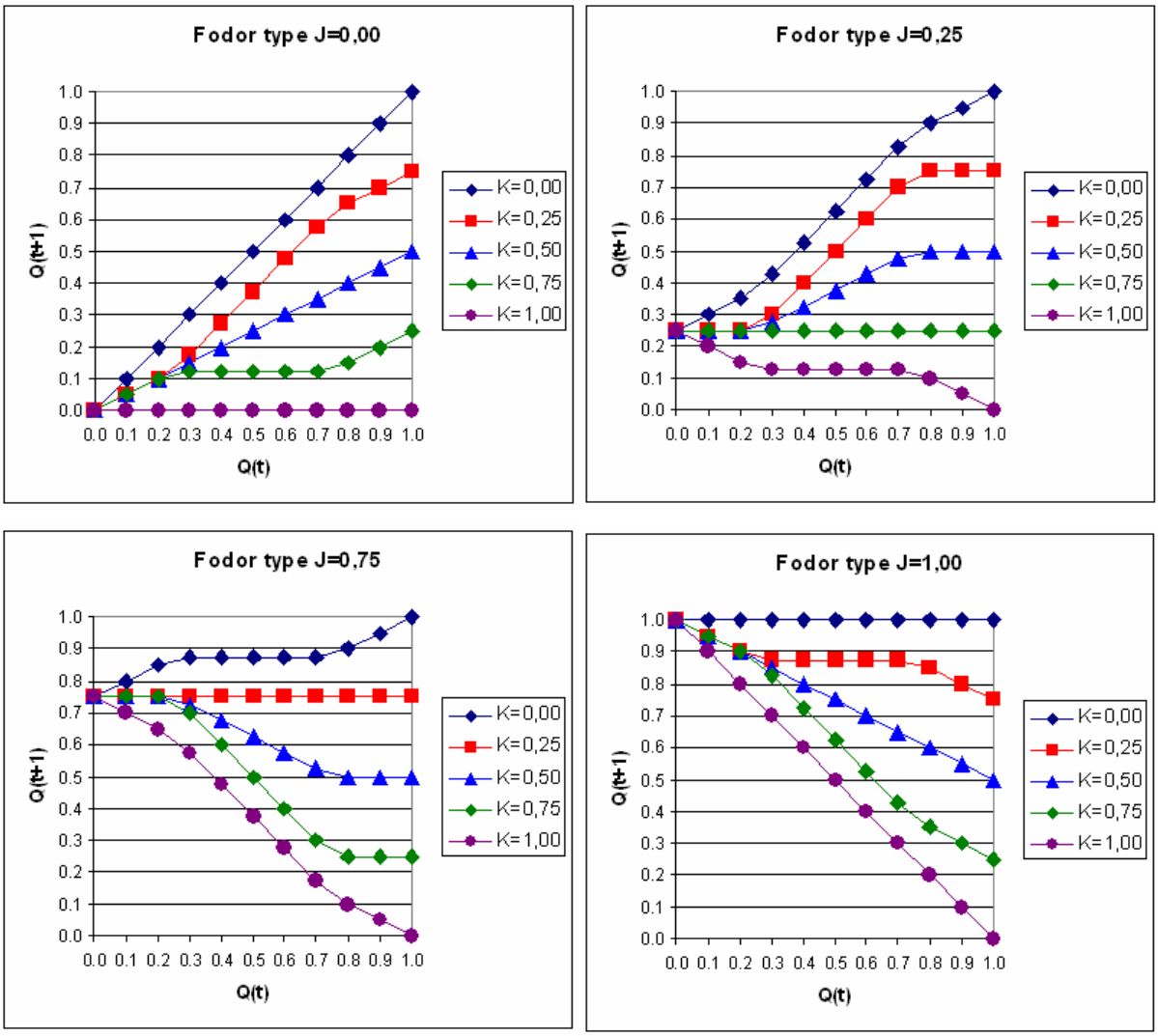


Figure 3.7
Fodor J-K F^3 for various values of *J* and *K*

3.4 Unified Fuzzy J-K Flip-Flops

From a practical aspect it is confusing that reset and set type F^3s sometimes do have very different behavior even though they are both supposed to be the extension of the same original binary circuit. Indeed, they are identical at the border lines when *J* and *K* are 0 or 1, thus these pairs are always justifiable generalizations and nevertheless they are disturbingly

non - dual. In order to eliminate this break of symmetry a combined reset-set type F^3 was proposed in [89], called the unified form of the fuzzy J-K flip-flop characteristic equation (equation 2.45). The new concept of unified fuzzy J-K flip-flops based on Yager, Dombi, Hamacher, Frank, Dubois-Prade and Schweizer-Sklar fuzzy operations is defined. In the subsections 3.4.1-3.4.7 the characteristic equation of all above mentioned new types of fuzzy flip-flop is determined. A set of eleven norms combined with the standard negation is analyzed in order to investigate, whether and to what degree they present more or less sigmoid $J \rightarrow Q(t+1)$ transfer characteristics in particular cases, when $K = 1 - Q$ (fuzzy flip-flop with feedback), for same selected values of Q. The characteristic equations of unified fuzzy J-K flip-flops based on standard and algebraic norms were introduced in subsection 2.1.4.4, equations (2.46), (2.47).

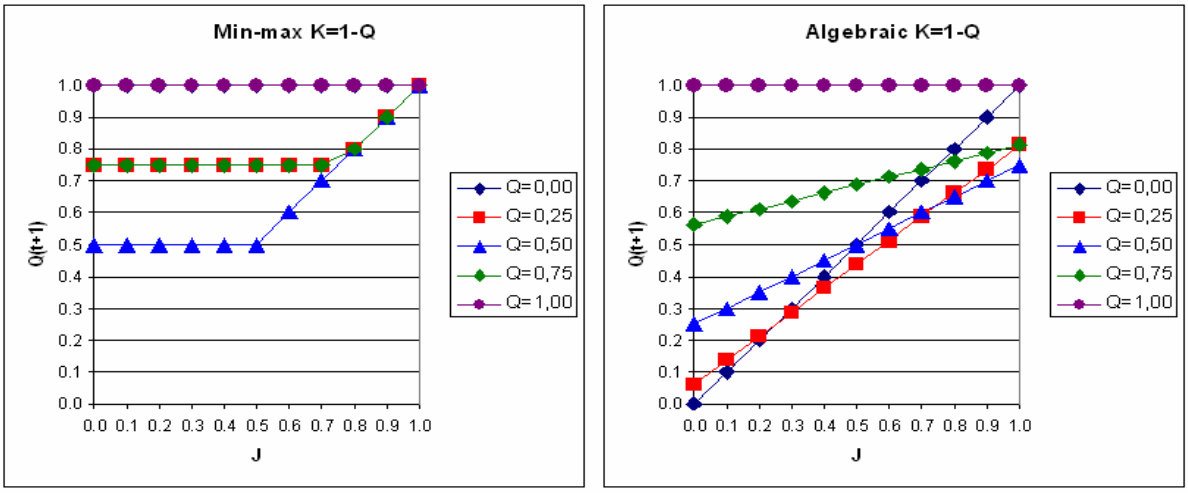


Figure 3.8
Unified standard and algebraic J-K F^3s

Figure 3.8 shows the non - sigmoid character of their transfer characteristics for some typical values of J and K, case of $K = 1\text{-}Q$. Min and max are often selected as t-norm/t-conorm pair. This choice is mainly due to the simplicity of the calculations. Figure 3.9 shows the character of next states of the defined drastic and Łukasiewicz type F^3s with feedback for various values of J and K. The characteristic curves of drastic type are piecewise linear with several breakpoints. For some selected Q values the Łukasiewicz type F^3s with feedback shows sigmoid character.

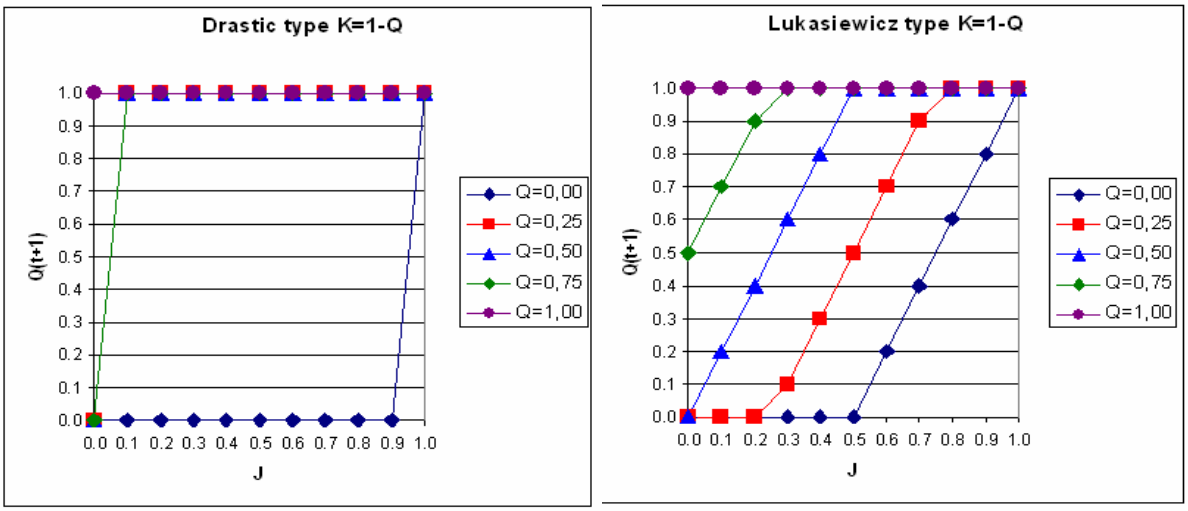


Figure 3.9
Unified drastic and Łukasiewicz J-K F[3]s

3.4.1 Yager

Using the parameterized families of Yager norms and the standard complementation the maxterm form in the unified equation (2.45) can be defined as

$$Q(t+1) = \left(J\ u_Y\left(1-K\right)\right)\ i_Y\ \left(J\ u_Y\ Q\right)\ i_Y\ \left(\left(1-K\right)\ u_Y\left(1-Q\right)\right) = \tag{3.21}$$

$$= 1 - \min\left(1, \left(\left(1 - \min\left(1, \left((1-K)^w + (1-Q)^w\right)^{1/w}\right)\right)^w + \min\left(1, \left(\left(1 - \min\left(1, \left(J^w + (1-K)^w\right)^{1/w}\right)\right)^w + M\right)^{1/w}\right)^w\right)^{1/w}\right)$$

$$M = \left(1 - \min\left(1, \left(J^w + Q^w\right)^{1/w}\right)\right)^w$$

This is the characteristic equations of the unified fuzzy J-K flip-flops based on Yager norms. Several values of the Yager parameter were considered in effort to tune the dissimilarity measure. On the one hand the equality $K = 1 - Q$, and on the other hand the Yager t-norm and t-conorm with different parameters w are used for typical values of Q. Figure 3.10 compares characteristics by a large number of various parameter values. For one selected parameter value ($w = 2$) by conducting extensive investigations it was found that the transfer characteristics have quasi sigmoid curvature (Figure 3.11).

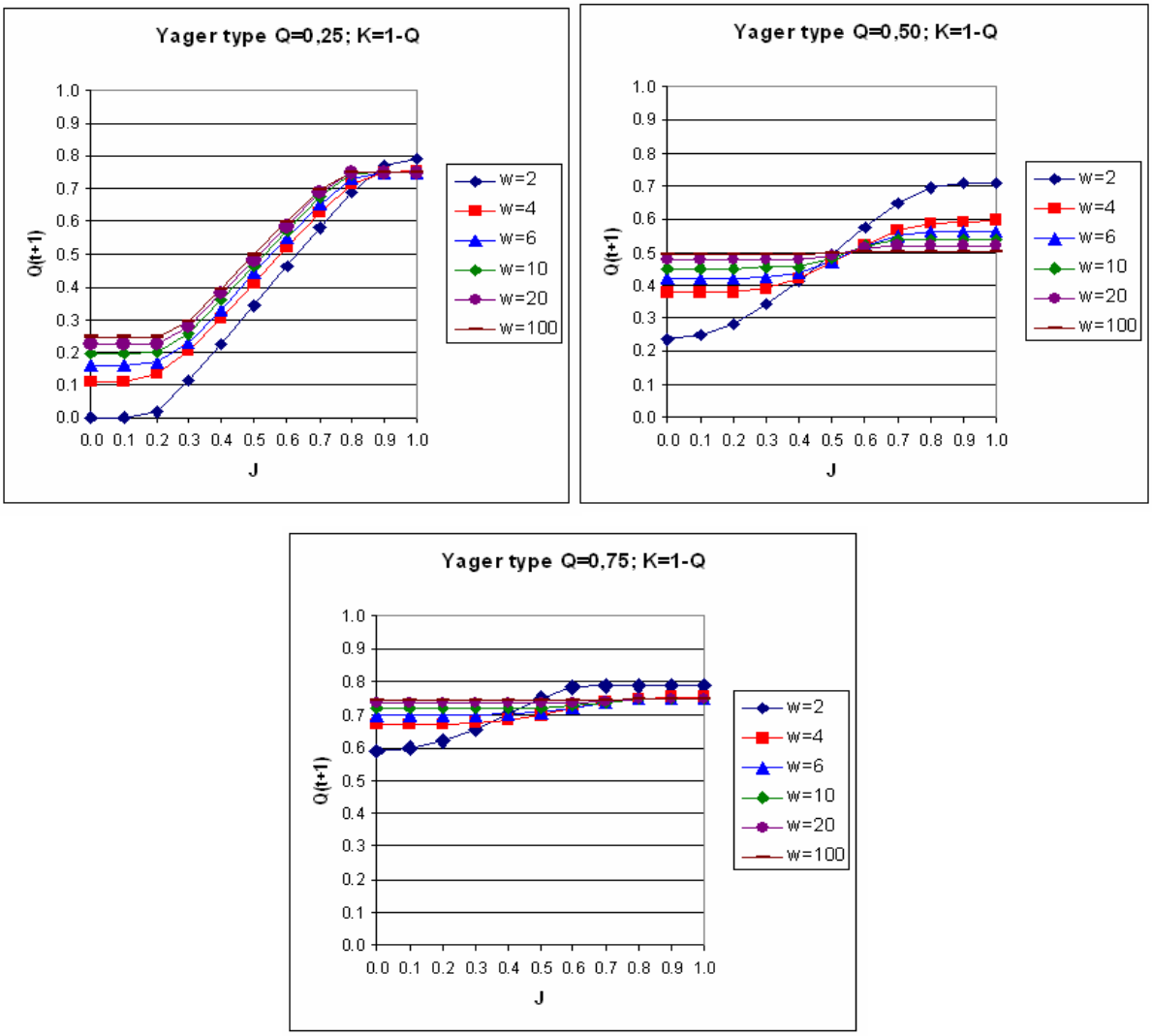


Figure 3.10
Unified Yager J-K F^3s for various values of parameter *w*

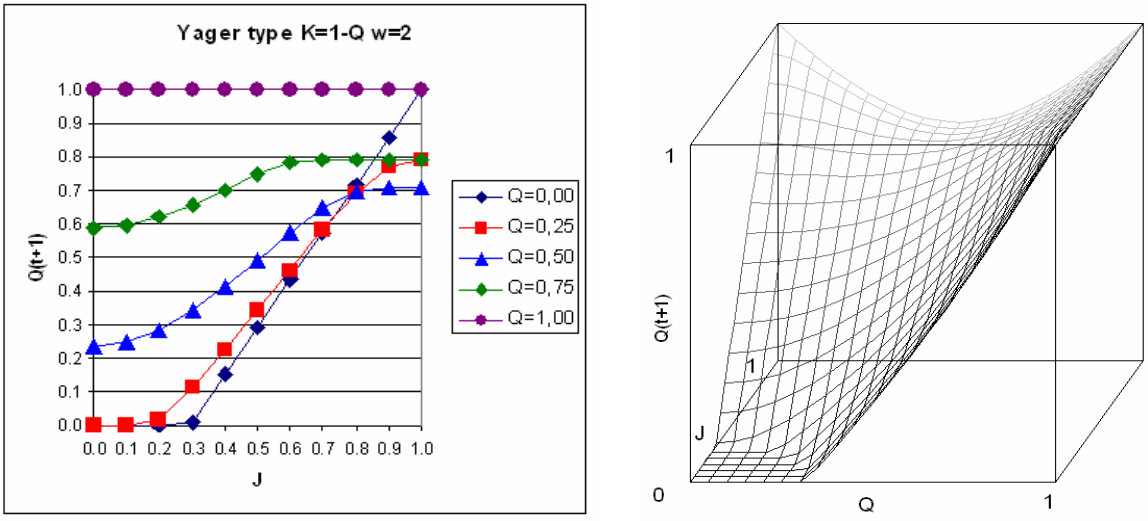


Figure 3.11
Unified Yager J-K F^3s, $w = 2$

3.4.2 Dombi

The characteristic equations of the unified fuzzy J-K flip-flops based on Dombi norms are defined as

$$Q(t+1) = \left(J\ u_D\ (1-K)\right)\ i_D\left(J\ u_D\ Q\right)\ i_D\ \left((1-K)\ u_D(1-Q)\right) = \tag{3.22}$$

$$= 1/\left(1+\left(\left(\left(\left(\left(-1+\frac{1}{J}\right)^{-\alpha}+\left(-1+\frac{1}{1-K}\right)^{-\alpha}\right)^{-1/\alpha}\right)^{\alpha}+\left(\left(\left(\left(-1+\frac{1}{J}\right)^{-\alpha}+\left(-1+\frac{1}{Q}\right)^{-\alpha}\right)^{-1/\alpha}\right)^{\alpha}\right)^{1/\alpha}\right)^{\alpha}+M\right)^{1/\alpha}\right)$$

$$M = \left(\left(\left(-1+\frac{1}{1-K}\right)^{-\alpha}+\left(-1+\frac{1}{1-Q}\right)^{-\alpha}\right)^{-1/\alpha}\right)^{\alpha}$$

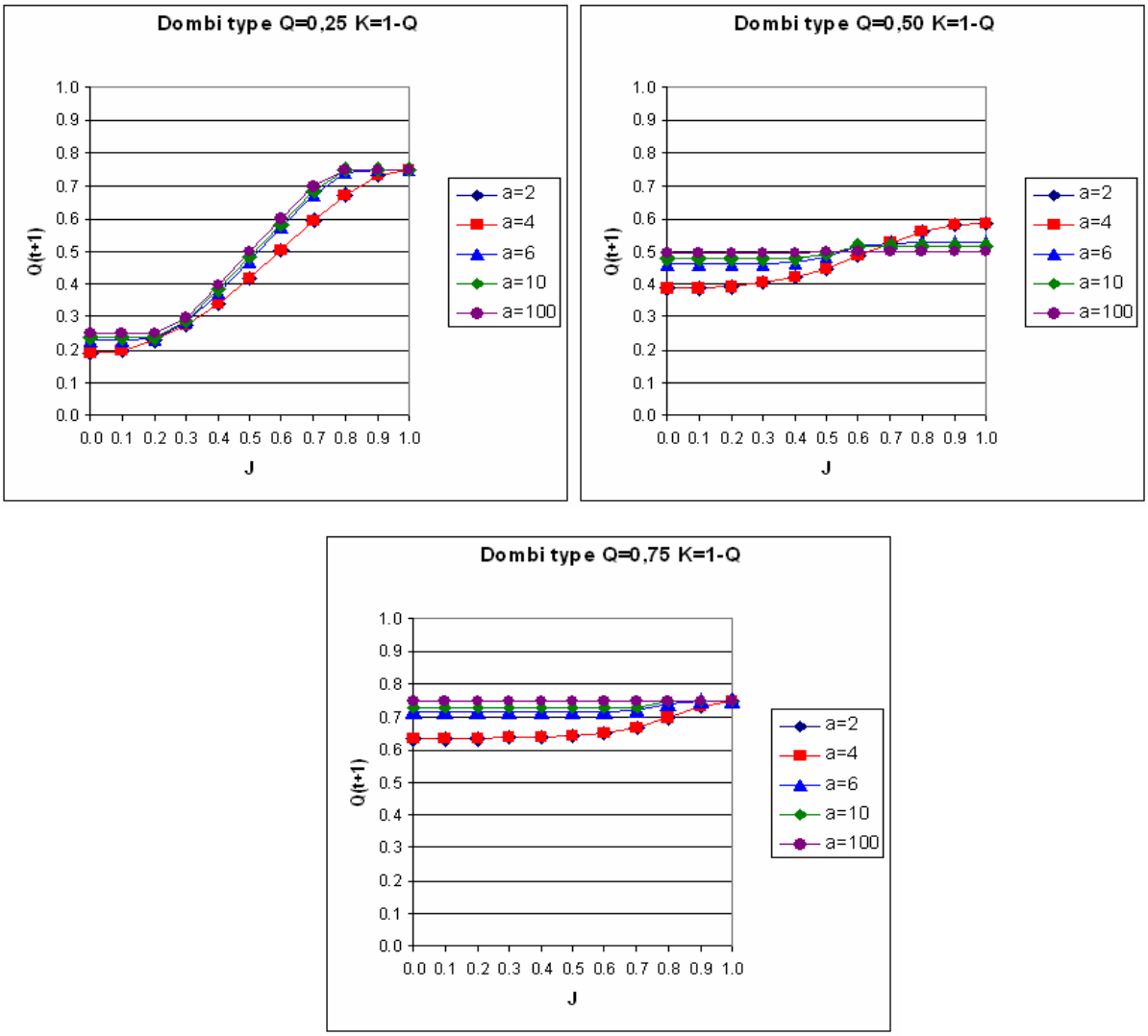


Figure 3.12
Unified Dombi J-K F[3]s for various values of parameter α

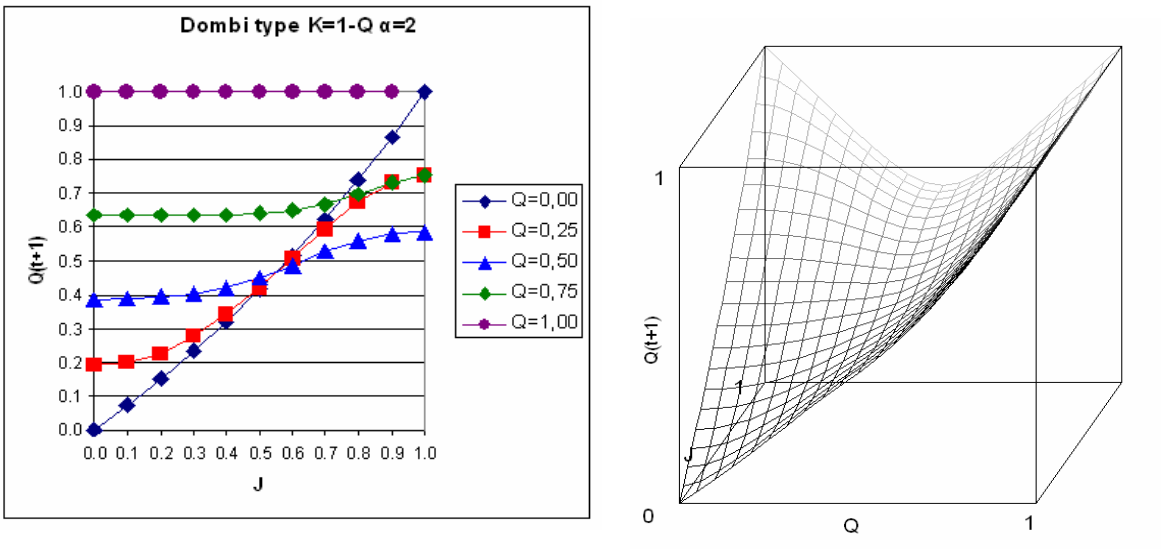


Figure 3.13
Unified Dombi J-K F[3]s, $\alpha = 2$

Figure 3.12 depicts characteristics by some selected parameter values. From a large number of various parameter values $\alpha = 2$ was selected. If $J = 0$, $K = 0$ or $Q = 0$ the values are obviously at the limit. The $J \to Q(t+1)$ transfer characteristics of fuzzy J-K flip-flops (case $K = 1 - Q$) based on Dombi norms show smooth quasi sigmoid curves (Figure 3.13). For the selected parameter and Q values the sections of the 3D surface are more or less sigmoid.

3.4.3 Hamacher

Based on an uncomplicated parametrical class of Hamacher norms the unified equation of the fuzzy J-K flip-flop is defined as

$$Q(t+1) = \left(J \; u_H \left(1-K\right)\right) \; i_H \left(J \; u_H \; Q\right) \; i_H \; \left(\left(1-K\right) \; u_H \left(1-Q\right)\right) = \tag{3.23}$$

$$= \left(\left(Q + J\left(1+Q\left(-2+\nu\right)\right)\right)\left(-1+K+J\left(1+K\left(-2+\nu\right)-\nu\right)\right)\left(Q+K\left(1+Q\left(-2+\nu\right)-\nu\right)+\nu-Q\nu\right)\right) /$$

$$\left(\left(1+J\left(-1+\nu\right)\right)\left(1+JQ\left(-1+\nu\right)\right)\left(Q\left(-1+\nu\right)-\nu\right)-K\left(-1+\nu\right)\left(\left(-1+Q\right)\left(1+Q\left(-2+\nu\right)-\nu\right)+M\right)+N\right)$$

$$M = J^2\left(Q+\nu+2Q^2\left(-1+\nu\right)\nu-2Q\nu^2\right)+J\left(1+Q^2\left(3-2\nu\right)-4\nu+Q\left(-5+6\nu\right)\right)$$

$$N = K^2\left(-1+\nu\right)\left(-1-3Q\left(-1+\nu\right)+\nu+Q^2\left(-3+2\nu\right)+J\left(3+Q^2\left(8-6\nu\right)-3\nu+Q\left(-9+8\nu\right)\right)+O\right)$$

$$O = J^2\left(-1+\nu-Q\left(-4+2\nu+\nu^2\right)+Q^2\left(-4+2\nu+\nu^2\right)\right)$$

The parameter of Hamacher families was optimized by comparing characteristics by a large number of various parameter values (Figure 3.14). For the quasi optimized parameter value $\nu = 10$ and $K = 1 - Q$ it was found that the transfer characteristics have quasi sigmoid $J \to Q(t+1)$ curvature (Figure 3.15).

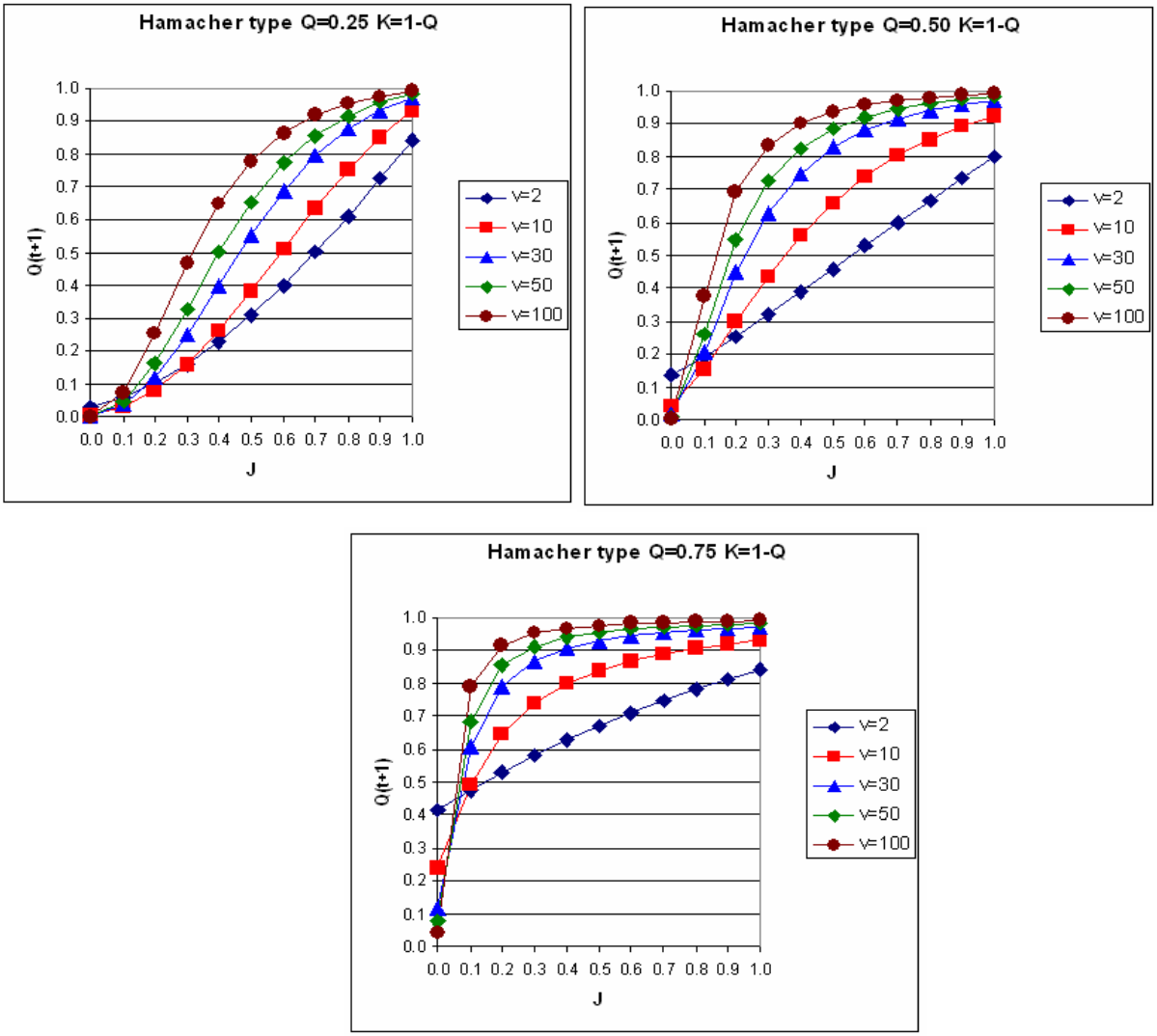


Figure 3.14
Unified Hamacher J-K F[3]s for various values of parameter ν

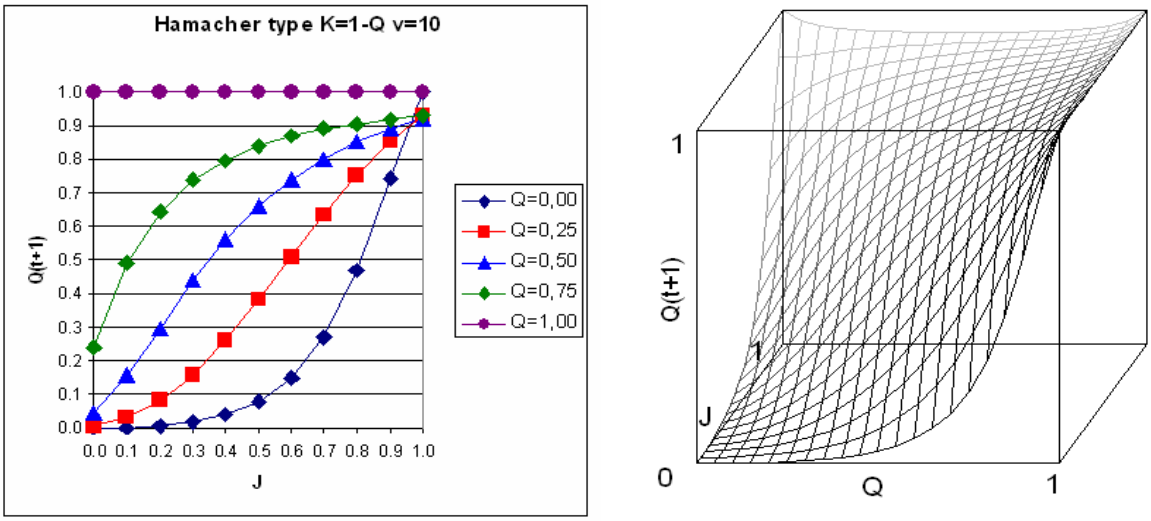


Figure 3.15
Unified Hamacher J-K F[3]s, $\nu = 10$

3.4.4 Frank

By substitute the family of Frank operations in the equation (2.45), the expression of the Frank type unified fuzzy J-K flip-flop is given by

$$Q(t+1) = \left(J\ u_F\left(1-K\right)\right)\ i_F\left(J\ u_F\ Q\right)\ i_F\ \left(\left(1-K\right)\ u_F\left(1-Q\right)\right) = \tag{3.24}$$

$$= \log_s\left(1+\left(-1+s/\left(1+\left(-1+s^{1-J}\right)\left(-1+s^{K}\right)/\left(-1+s\right)\right)\right)\cdot M\cdot N/\left(-1+s\right)^2\right)$$

$$M = \left(-1+s/\left(1+\left(-1+s^{1-J}\right)\left(-1+s^{1-Q}\right)/\left(-1+s\right)\right)\right)$$

$$N = \left(-1+s/\left(1+\left(-1+s^{K}\right)\left(-1+s^{Q}\right)/\left(-1+s\right)\right)\right)$$

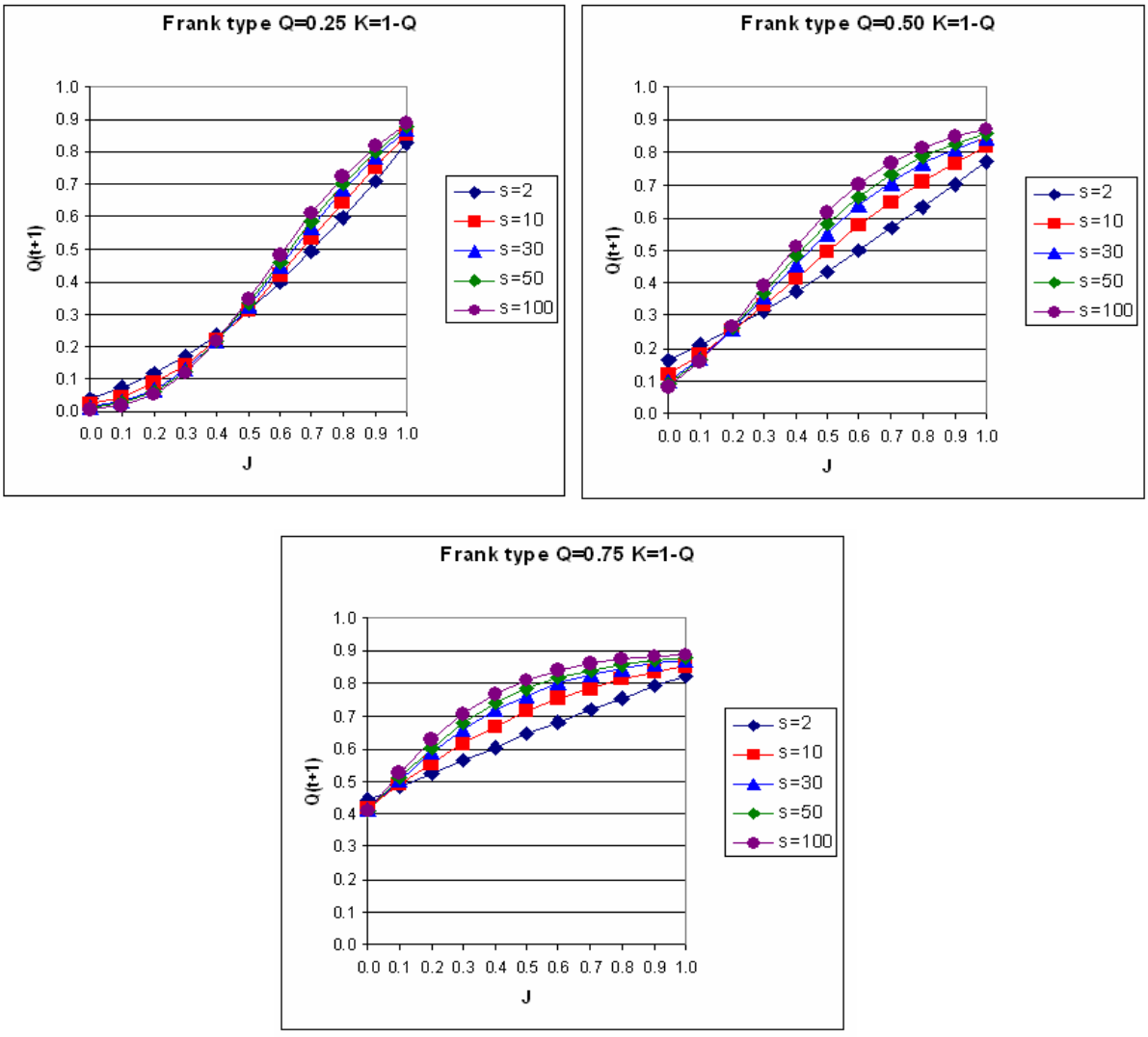


Figure 3.16
Unified Frank J-K F[3]s for various values of parameter *s*

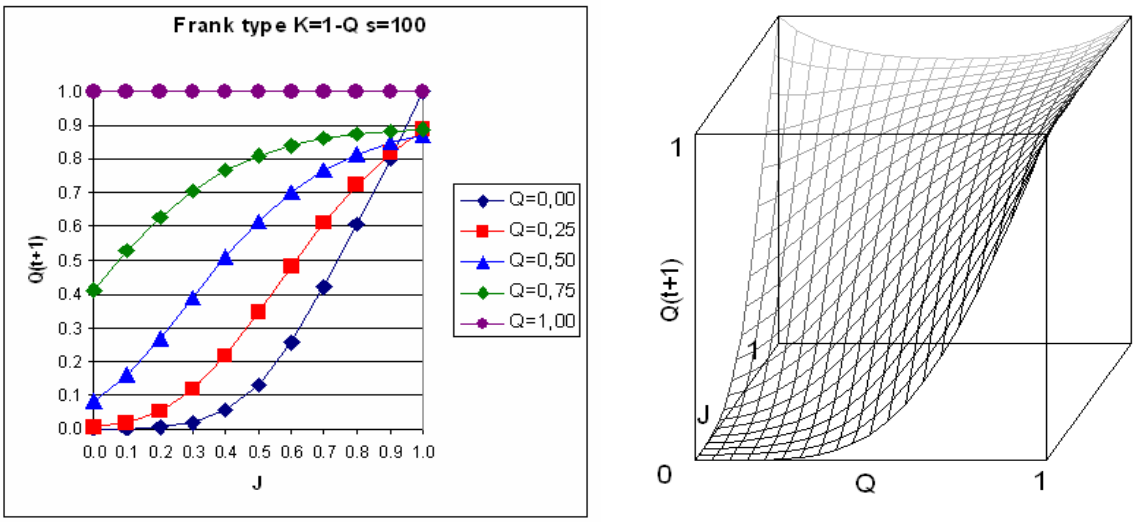


Figure 3.17
Unified Frank J-K F[3]s, s = 100

The quasi optimal selected parameter value, s = 100, (Figure 3.16) depicts the more or less s-shaped $J \to Q(t+1)$ characteristics for fuzzy J-K flip-flops with feedback based on Frank norms Figure 3.17.

3.4.5 Dubois-Prade

Using the parameterized families of Dubois-Prade norms the unified fuzzy J-K flip-flop is defined as

$$Q(t+1) = \left(J\ u_{DP}\left(1-K\right)\right)\ i_{DP}\left(J\ u_{DP}\ Q\right)\ i_{DP}\ \left(\left(1-K\right)\ u_{DP}\left(1-Q\right)\right) = \tag{3.25}$$

$$= \left(\left(1+J-J\left(1-K\right)-K-\min\left(1-d,J,1-K\right)\right)\left(J+Q-JQ-\min\left(1-d,J,Q\right)\right)\cdot M\right)/$$

$$\left(\max\left(d,1-J,K\right)\max\left(d,1-J,1-Q\right)\max\left(d,K,Q\right)\cdot N\cdot P\right)$$

$$M = \left(2-K-\left(1-K\right)\left(1-Q\right)-Q-\min\left(1-d,1-K,1-Q\right)\right)$$

$$N = \max\left(d,\left(1+J-J\left(1-K\right)-K-\min\left(1-d,J,1-K\right)\right)/\max\left(d,1-J,K\right),O\right)$$

$$O = \left(J+Q-JQ-\min\left(1-d,J,Q\right)/\max\left(d,1-J,1-Q\right)\right)$$

$$P = \max\left(d,\left(1+J-J\left(1-K\right)-K-\min\left(1-d,J,1-K\right)\right)\left(J+Q-JQ-\min\left(1-d,J,Q\right)\right)/T,W\right)$$

$$T = \max\left(d,1-J,K\right)\max\left(d,1-J,1-Q\right)\max\left(d,\left(1+J-J\left(1-K\right)-K-\min\left(1-d,J,1-K\right)\right)/V,Z\right)$$

$$V = \max\left(d,1-J,K\right)$$

$$Z = J+Q-JQ-\min\left(1-d,J,Q\right)/V$$

$$W = \left(2-K-\left(1-K\right)\left(1-Q\right)-Q-\min\left(1-d,1-K,1-Q\right)\right)/\max\left(d,K,Q\right)$$

3.4.6 Schweizer-Sklar

The next unified equation is defined by proposing the Schweizer-Sklar norms in the maxterm form of unified fuzzy J-K flip-flop formulae

$$Q(t+1) = \left(J\ u_{SS}\left(1-K\right)\right)\ i_{SS}\left(J\ u_{SS}\ Q\right)\ i_{SS}\ \left(\left(1-K\right)\ u_{SS}\left(1-Q\right)\right) = \tag{3.26}$$

$$= \max\left(0,-1+\left(1-\max\left(0,-1+K^{p}+Q^{p}\right)^{1/p}\right)^{p}+\left(\max\left(0,-1+\left(1-\max\left(0,-1+\left(1-J\right)^{p}+K^{p}\right)^{1/p}\right)^{p}+M\right)^{1/p}\right)^{p}\right)^{1/p}$$

$$M = \left(1-\max\left(0,-1+\left(1-J\right)^{p}+\left(1-Q\right)^{p}\right)^{1/p}\right)^{p}$$

Figure 3.18 depicts the behavior of the fuzzy J-K flip-flop (case K = 1 - Q) based on Dubois-Prade and Schweizer-Sklar norms for selected parameter values. There are obviously quasi sigmoid curves for same Q values in case of Dubois-Prade norms and lines in the surface with several breakpoints case of Schweizer-Sklar norms.

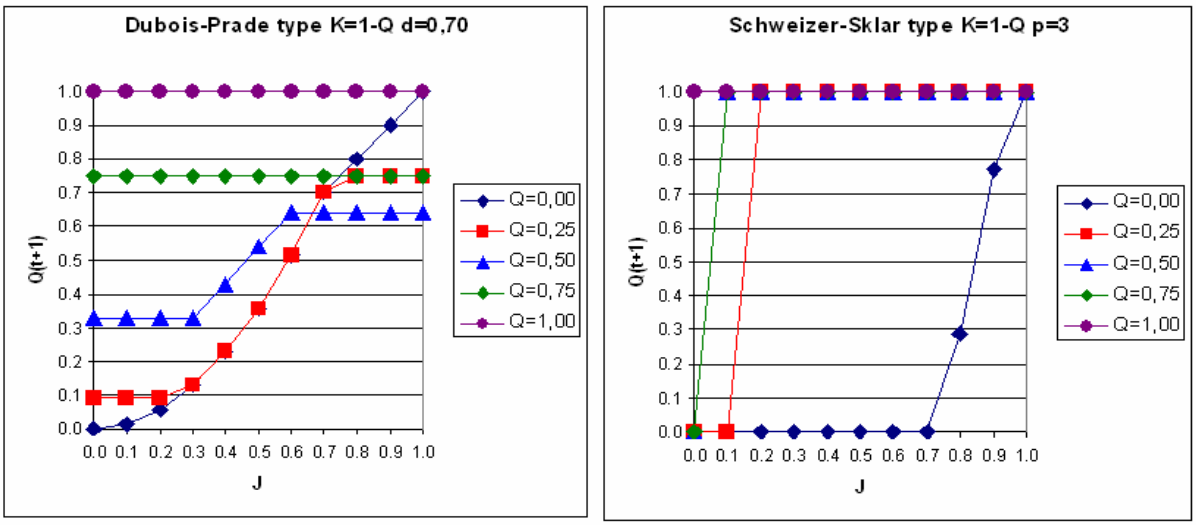


Figure 3.18
Unified Dubois-Prade and Schweizer-Sklar J-K F^3s

3.4.7 Fodor

Figure 3.19 depicts the almost linear transfer characteristics of the fuzzy J-K flip-flop with feedback based on Fodor norms.

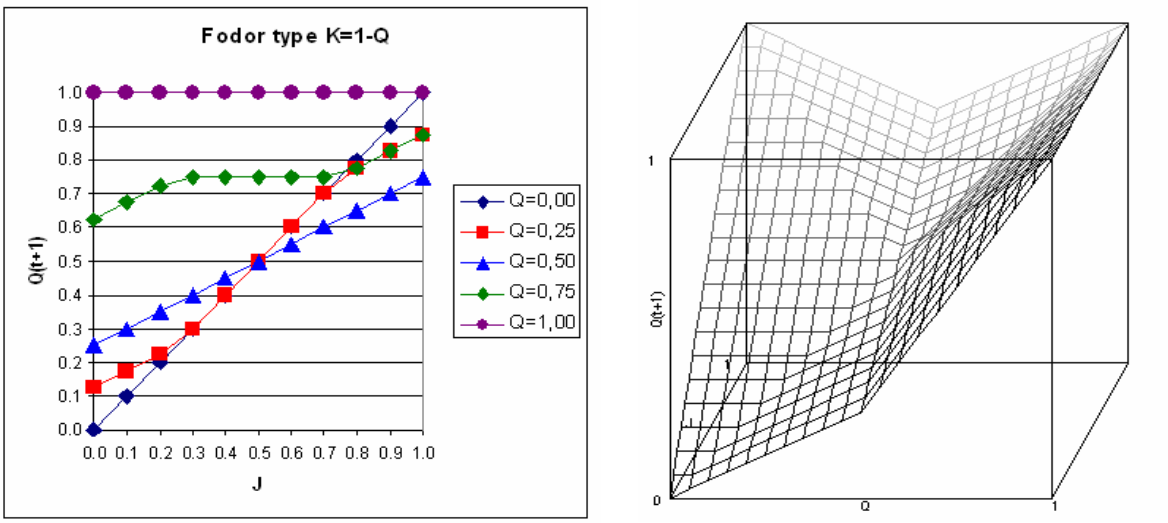


Figure 3.19
Fodor J-K F^3

In this subsection the concept of the unified fuzzy J-K flip-flops based on Yager, Dombi, Hamacher, Frank, Dubois-Prade and Schweizer-Sklar fuzzy operations has been defined. The new F^3s characteristic equations have been determined. It can be concluded that the $J \to Q(t+1)$ transfer characteristics of fuzzy J-K flip-flops with feedback based on Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade norms show quasi sigmoid character for some selected Q values and the rest with non - sigmoid shape. The quasi optimized fuzzy operations parameter values have been found by comparing a large number of various parameter values.

3.5 Fuzzy D Flip-Flops

The concept of a novel type of fuzzy flip-flop is defined. Connecting the inputs of the fuzzy J-K flip-flop in a particular way namely by applying an inverter in the connection of the input J to K, case of J = 1 - K, a fuzzy D flip-flop is obtained. Substitute $J = \overline{K}$ in equation (2.45) and let $D = J$, the fundamental equation of fuzzy D flip-flop is

$$Q(t+1) = (D \vee D) \wedge (D \vee Q) \wedge (D \vee \overline{Q}) \qquad (3.27)$$

where the over bar denotes complement (e.g. $\overline{Q} = 1 - Q$), furthermore $\wedge$ and $\vee$ denote t-norm and t-conorm, labeled as i and u in the next.

After determining the novel characteristic equation the various fuzzy D flip-flop behaviors are illustrated by the graphs belonging to the next states of fuzzy flip-flops for typical values of *Q*, *J* and *K*, based on eleven fuzzy operations namely standard, algebraic, drastic, Łukasiewicz, Yager, Dombi, Hamacher, Frank, Dubois-Prade, Schweizer-Sklar and Fodor. In order to test the $D \to Q(t+1)$ transfer characteristics behaviors several characteristics were performed.

3.5.1 Standard

Substitute the standard norms and the standard complementation (2.13) in (3.27), the standard type fuzzy D flip-flop characteristic equation is defined by

$$Q(t+1) = \left(D\ u_S\ D\right)\ i_S\ \left(D\ u_S\ Q\right)\ i_S\ \left(D\ u_S\ \left(1-Q\right)\right) = \tag{3.28}$$
$$= \min\left(D, \max\left(D, 1-Q\right), \max\left(D, Q\right)\right)$$

3.5.2 Algebraic

The unified equation of the next state of algebraic type fuzzy D flip-flop depending on the algebraic norms can be defined as

$$Q(t+1) = \left(D\ u_A\ D\right)\ i_A\ \left(D\ u_A\ Q\right)\ i_A\ \left(D\ u_A\ \left(1-Q\right)\right) = \tag{3.29}$$
$$= \left(2D - D^2\right)\left(1 + D - D\left(1-Q\right) - Q\right)\left(D + Q - DQ\right)$$

The transfer characteristics of fuzzy D flip-flop based on standard and algebraic operations are illustrated in Figure 3.20. The sections of the surfaces are linear with several breakpoints.

3.5.3 Drastic

In a similar way, using the drastic expressions for fuzzy set intersection and union the equation of the drastic type D F^3 is defined by

$$Q(t+1) = \left(D\ u_{DR}\ D\right)\ i_{DR}\ \left(D\ u_{DR}\ Q\right)\ i_{DR}\ \left(D\ u_{DR}\ \left(1-Q\right)\right) \tag{3.30}$$

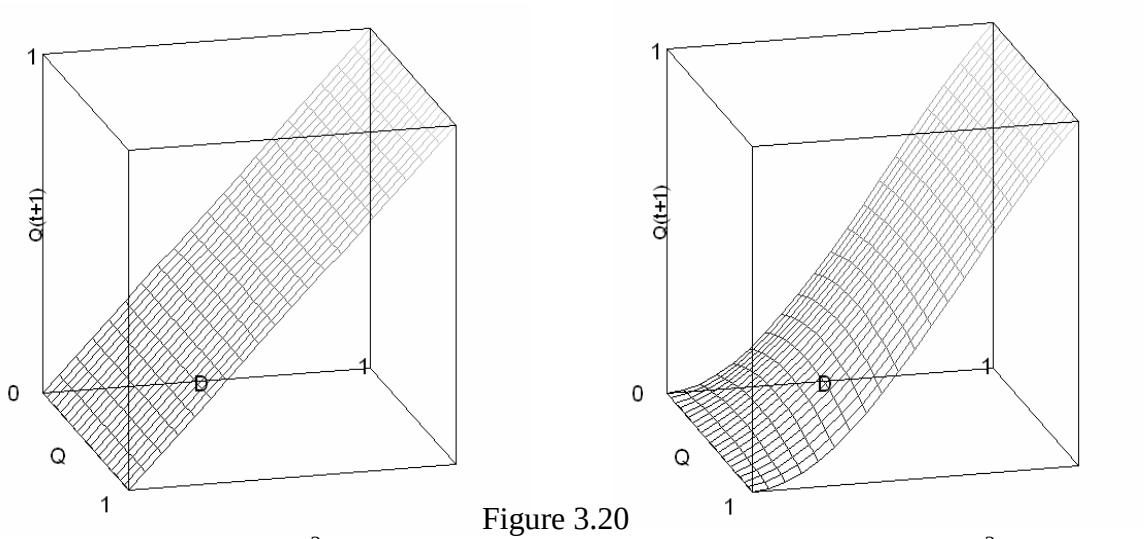


Figure 3.20

Standard D F^3 Algebraic D F^3

3.5.4 Łukasiewicz

Substitute the Łukasiewicz norms in (3.27) the D F^3 characteristic equation can be defined as

$$Q(t+1) = (D\ u_L\ D)\ i_L\ (D\ u_L\ Q)\ i_L\ (D\ u_L\ (1-Q)) = \tag{3.31}$$

$$= \max\left(0,\ -1+\max\left(0,\ -1+\min(1,\ 2D)+\min(1,\ D+Q)\right)+\min(1,\ 1+D-Q)\right)$$

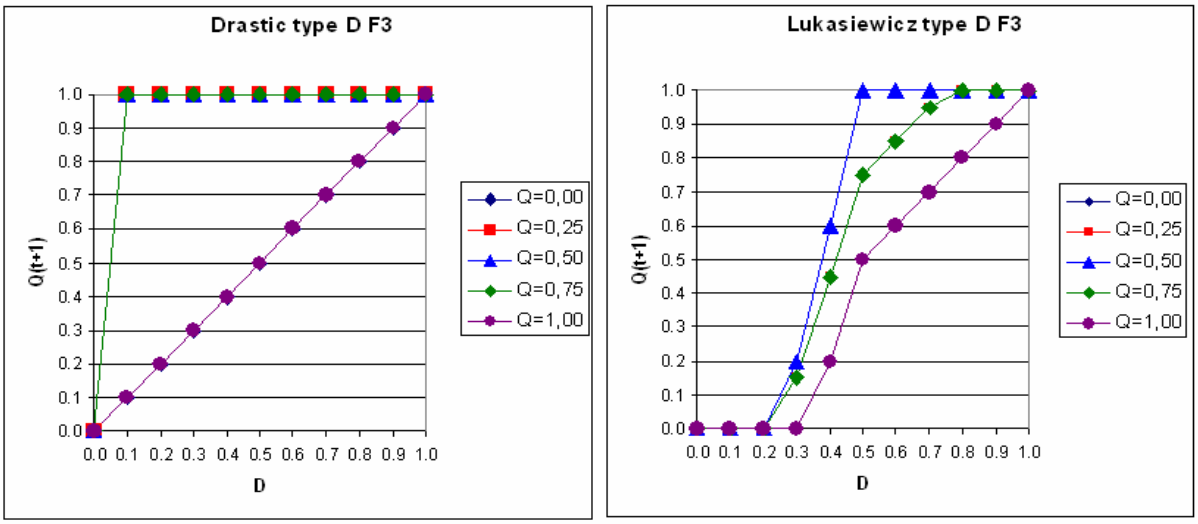


Figure 3.21

Drastic D F^3 Łukasiewicz D F^3

The behaviors of $D \to Q(t+1)$ transfer characteristics of drastic (linear with breakpoints) and Łukasiewicz (quasi sigmoid for same selected Q values) type D F^3s are illustrated in Figure 3.21.

3.5.5 Yager

By applying the Yager norms the maxterm form in the unified equation (3.27) can be defined as

$$Q(t+1) = (D\ u_Y\ D)\ i_Y\ (D\ u_Y\ Q)\ i_Y\ (D\ u_Y\ (1-Q)) = \tag{3.32}$$

$$= 1 - \min\left(1, \left(\left(1 - \min\left(1, \left(D^w + (1-Q)^w\right)^{1/w}\right)\right)^w + \min\left(1, \left(\left(1 - \min\left(1, 2^{1/w}\left(D^w\right)^{1/w}\right)\right)^w + M\right)^{1/w}\right)^w\right)^{1/w}\right)$$

$$M = \left(1 - \min\left(1, \left(D^w + Q^w\right)^{1/w}\right)\right)^w$$

The 2D figures and the section of the 3D surfaces show nice approximately sigmoid characteristics. This distinctiveness is very well illustrated when the value of the parameter w is equal to 2 (see Figure 3.22).

3.5.6 Dombi

The next state of the fuzzy D flip-flop equation based Dombi-class operators is defined as

$$Q(t+1) = (D\ u_D\ D)\ i_D\ (D\ u_D\ Q)\ i_D\ (D\ u_D\ (1-Q)) = \tag{3.33}$$

$$= 1/\left(1 + \left(\left(\left(\left(2^{-1/\alpha}\left(\left(-1+\frac{1}{D}\right)^{-\alpha}\right)^{-1/\alpha}\right)^{\alpha} + \left(\left(\left(-1+\frac{1}{D}\right)^{-\alpha} + \left(-1+\frac{1}{Q}\right)^{-\alpha}\right)^{-1/\alpha}\right)^{\alpha}\right)^{1/\alpha} + M\right)^{\alpha}\right)^{1/\alpha}\right)$$

$$M = \left(\left(\left(-1+\frac{1}{D}\right)^{-\alpha} + \left(-1+\frac{1}{1-Q}\right)^{-\alpha}\right)^{-1/\alpha}\right)^{\alpha}$$

It is very surprising that in this approach there are obviously lines in the surface, independently from the parameter values. Figure 3.23 shows the linear character in a particular case when the parameter value is $\alpha = 2$.

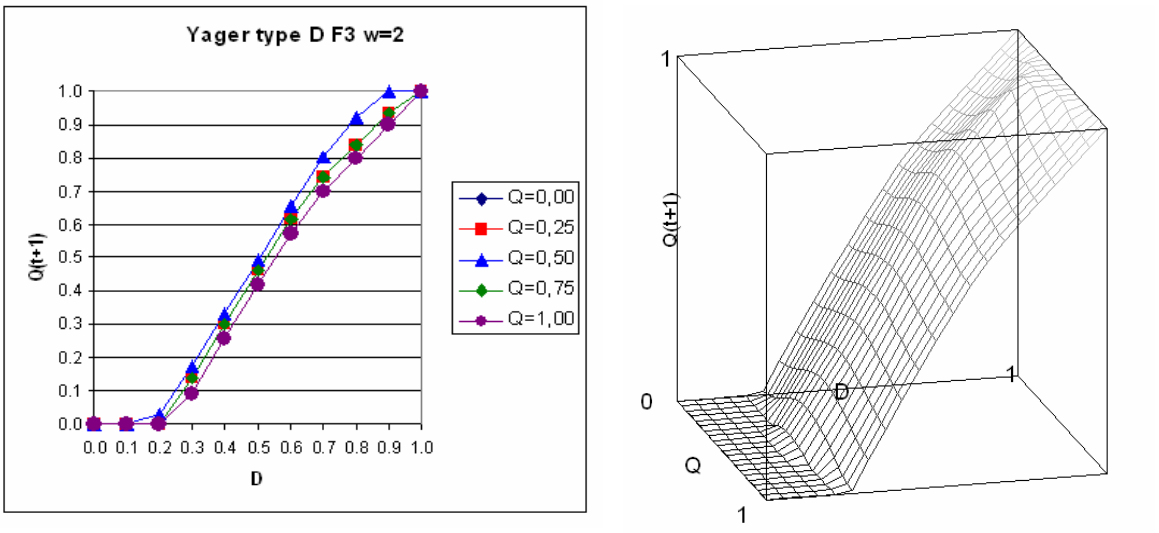


Figure 3.22
Yager D F^3s

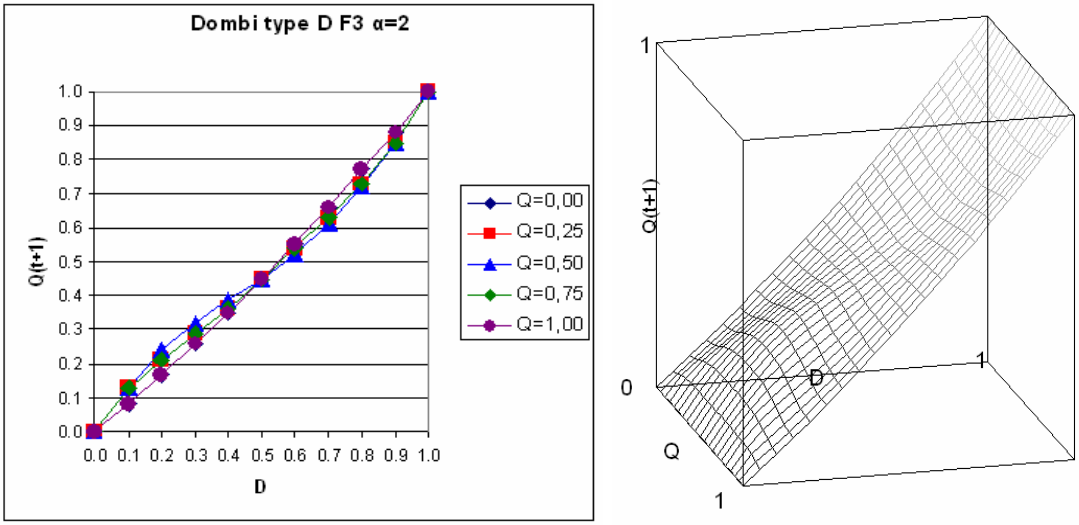


Figure 3.23
Dombi D F^3s

3.5.7 Hamacher

Based on Hamacher norms the next state of the D F^3 is

$$Q(t+1) = (D\ u_H\ D)\ i_H\ (D\ u_H\ Q)\ i_H\ (D\ u_H\ (1-Q)) = \tag{3.34}$$

$$= -\Big(D\big(Q+D\big(1+Q\big(-2+\nu\big)\big)\big)\Big)\Big(-1+Q+D\big(1+Q\big(-2+\nu\big)-\nu\big)\Big)\Big(2+D\big(-2+\nu\big)\big)\Big) /$$

$$\Big(Q\big(-1+\nu\big)^2 - Q^2\big(-1+\nu\big)^2 + \nu + D^3\big(-1+\nu\big)\big(-5+4\nu-4Q\big(-4+3\nu\big)+4Q^2\big(-4+3\nu\big)\big)+M\Big)$$

$$M = D\big(-1+\nu\big)\big(-3+Q\big(8-6\nu\big)+\nu+Q^2\big(-8+6\nu\big)\big) - D^2\big(-1+\nu\big)\big(-7+Q\big(19-14\nu\big)+N\big)$$

$$N = -D^4\big(-1+\nu\big)\big(-1+\nu-Q\big(-4+2\nu+\nu^2\big)\big)$$

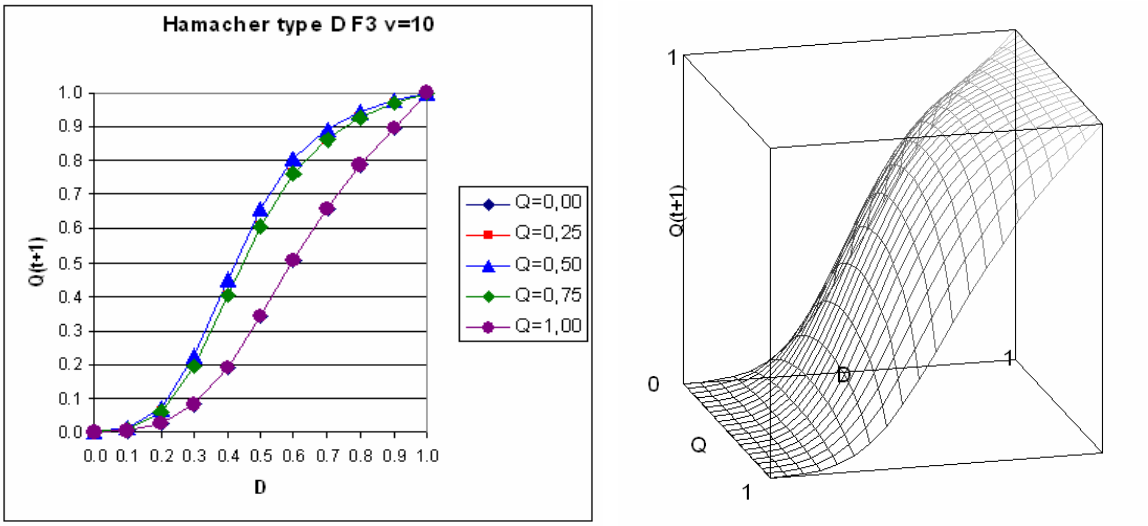


Figure 3.24
Hamacher D F^3s

3.5.8 Frank

Using the definition of Frank operators and standard complementation the Frank type D F^3 characteristic equation is

$$Q(t+1) = (D\ u_F\ D)\ i_F\ (D\ u_F\ Q)\ i_F\ (D\ u_F\ (1-Q)) = \tag{3.35}$$

$$= \log_s\left(1+\left(-1+s/\left(1+\left(-1+s^{1-D}\right)^2/(-1+s)\right)\right)\cdot M\cdot N/(-1+s)^2\right)$$

$$M = \left(-1+s/\left(1+\left(-1+s^{1-D}\right)\left(-1+s^{1-Q}\right)/(-1+s)\right)\right)$$

$$N = \left(-1+s/\left(1+\left(-1+s^{1-D}\right)\left(-1+s^{Q}\right)/(-1+s)\right)\right)$$

Applying the parameterized family of Hamacher and Frank norms for typical parameter values $\nu = 10$ and s = 100 (Figures 3.24 and 3.25), the sections of the $D \to Q(t+1)$ surface are more or less s-shaped.

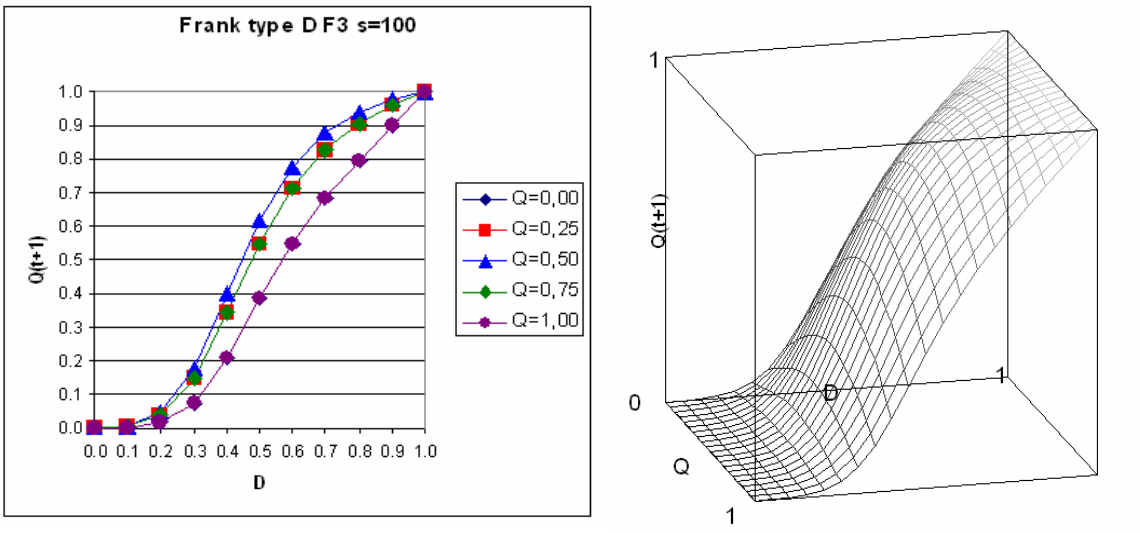


Figure 3.25
Frank D F^3s

3.5.9 Dubois-Prade

In a similar way, if the parameterized family of Dubois-Prade norms is used in the expression (3.27) for fuzzy set intersection and union the D F^3 equation will be

$$Q(t+1) = (D\ u_{DP}\ D)\ i_{DP}\ (D\ u_{DP}\ Q)\ i_{DP}\ (D\ u_{DP}\ (1-Q)) = \tag{3.36}$$

$$= \left(2D - D^2 - \min(1-d, D)\right)\left(1 + D - D(1-Q) - Q - \min(1-d, D, 1-Q)\right)$$

$$\left(D + Q - DQ - \min(1-d, D, Q)\right) / M$$

$$M = \max(d, 1-D)\max(d, 1-D, 1-Q)\max(d, 1-D, Q)\cdot N \cdot P$$

$$N = \max\left(d, \left(2D - D^2 - \min(1-d, D)\right) / \max(d, 1-D), O\right)$$

$$O = \left(D + Q - DQ - \min(1-d, D, Q)\right) / \max(d, 1-D, 1-Q)$$

$$P = \max\left(d, \left(1 + D - D(1-Q) - Q - \min(1-d, D, 1-Q)\right) / \max(d, 1-D, Q), T\right)$$

$$T = \left(2D - D^2 - \min(1-d, D)\left(D + Q - DQ - \min(1-d, D, Q)\right)\right) / V$$

$$V = \max(d, 1-D)\max(d, 1-D, 1-Q)\max(d, Z, W)$$

$$Z = \left(2D - D^2 - \min(1-d, D)\right) / \max(d, 1-D)$$

$$W = \left(D + Q - DQ - \min(1-d, D, Q)\right) / \max(d, 1-D, 1-Q)$$

3.5.10 Schweizer-Sklar

Schweizer-Sklar norms combined with the standard negation and applying in the expression of fuzzy D flip-flop is defined as

$$Q(t+1) = \left(D\ u_{SS}\ D\right)\ i_{SS}\ \left(D\ u_{SS}\ Q\right)\ i_{SS}\ \left(D\ u_{SS}\ \left(1-Q\right)\right) = \tag{3.37}$$

$$= \max\left(0, -1+\left(1-\max\left(0, -1+\left(1-D\right)^{p}+Q^{p}\right)^{1/p}\right)^{p}+M\right)^{1/p}$$

$$M = \left(\max\left(0, -1+\left(1-\max\left(0, -1+2\left(1-D\right)^{p}\right)^{1/p}\right)^{p}+N\right)^{1/p}\right)^{p}$$

$$N = \left(1-\max\left(0, -1+\left(1-D\right)^{p}+\left(1-Q\right)^{p}\right)^{1/p}\right)^{p}$$

The transfer characteristics curvature in case of Dubois-Prade are quasi sigmoid (d = 0.7), furthermore are linear, case of Schweizer-Sklar norms (p = 3) (Figure 3.26).

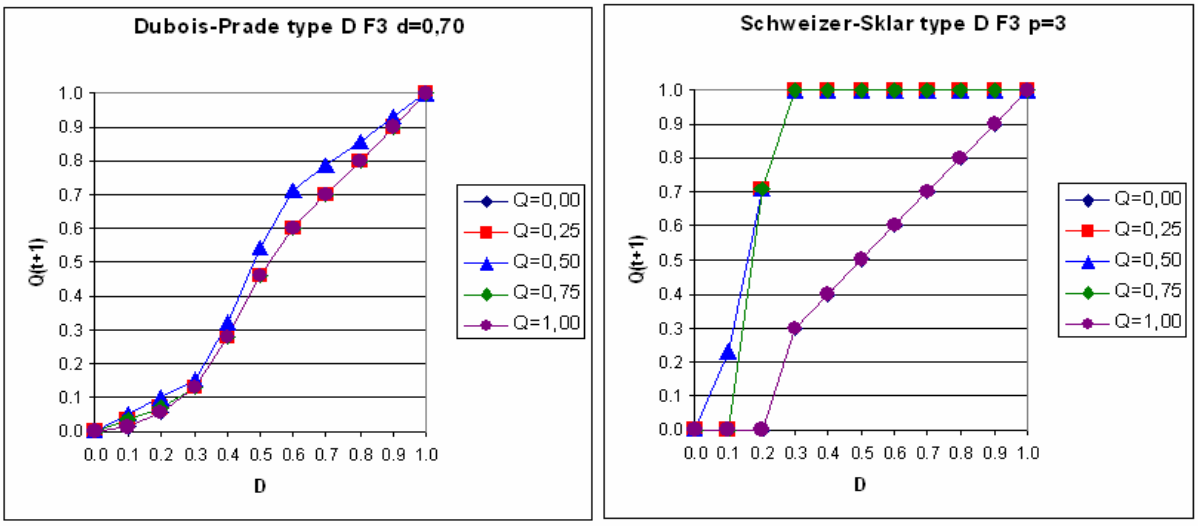


Figure 3.26

Dubois-Prade D F^3s Schweizer-Sklar D F^3s

3.5.11 Fodor

Finally, the expression of the next state Fodor type D F^3 and piecewise linear transfer characteristics (Figure 3.27) are given

$$Q(t+1) = (D\ u_F\ D)\ i_F\ (D\ u_F\ Q)\ i_F\ (D\ u_F\ (1-Q)) = \tag{3.38}$$

$$= \frac{1}{2}\left(\max\left(0, \frac{1}{2}\left(-2 + \max\left(0, \frac{1}{2}\left(-2 + D + \max(D,Q) + \min(1,2D) + \min(1, D+Q)\right)\right) + M\right)\right) + N\right)$$

$$M = \max(D, 1-Q) + \min(1, 1+D-Q) + \min\left(\frac{1}{2}\left(D + \min(1,2D)\right), \frac{1}{2}\left(\max(D,Q) + \min(1, D+Q)\right)\right)$$

$$N = \min\left(\frac{1}{2}\left(\max(D, 1-Q) + \min(1, 1+D-Q)\right), \frac{1}{2}O\right)$$

$$O = \left(\max\left(0, \frac{1}{2}\left(-2 + D + \max(D,Q) + \min(1,2D) + \min(1, D+Q)\right)\right) + P\right)$$

$$P = \min\left(\frac{1}{2}\left(D + \min(1,2D)\right), \frac{1}{2}\left(\max(D,Q) + \min(1, D+Q)\right)\right)$$

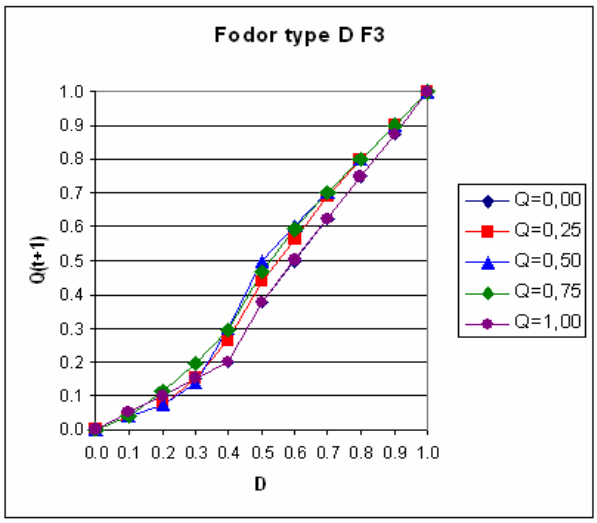


Figure 3.27
Fodor D F^3

In this subsection the concept of new fuzzy D flip-flops based on standard, algebraic, drastic, Łukasiewicz, Yager, Dombi, Hamacher, Frank, Dubois-Prade, Schweizer-Sklar and Fodor fuzzy operations has been defined. In all above mentioned sub cases the characteristic equations have been determined. Figures 3.21 - 3.27 show the behavior of fuzzy D flip-flops based on the above mentioned popular fuzzy operations. For well selected parameter (i.e. w = 2, v = 10, s = 100 and d = 0.7) and Q values, the $D \to Q(t+1)$ characteristics of D F^3s based on Łukasiewicz, Yager, Hamacher, Frank and Dubois-Prade norms present quasi sigmoid behavior for some selected Q values while all other F^3s investigated have non - sigmoid character. From the neural networks perspective (regarding the ability to use the learning and adaptation mechanisms used with classic neuron models), suitable t-norms may be deployable for defining fuzzy neurons.

3.6 Choi Type Fuzzy D Flip-Flops

This section defines the concept of Choi type fuzzy D flip-flops starting from the characteristic equation of a fuzzy D flip-flop (2.50) proposed by Choi and Tipnis [11]. The F^3s characteristic equation based on algebraic, drastic, Łukasiewicz, Yager, Dombi, Hamacher, Frank, Dubois-Prade, Schweizer-Sklar and Fodor fuzzy operations combined with the standard complementation are determined. Figures 3.28 - 3.34 depict the behavior by the graphs belonging to the next states of different types of fuzzy Choi D flip-flop for various typical values of *Q*, *J* and *K*.

3.6.1 Algebraic

The definition of fuzzy Choi type D flip-flop based on algebraic norms is

$$Q(t+1) = (D)\ i_A\ (D\ u_A\ Q)\ i_A\ ((1-Q)\ u_A\ D) = \tag{3.39}$$
$$= D(1+D-D(1-Q)-Q)(D+Q-DQ)$$

The section of the 3D surface corresponding to the original Choi type fuzzy D flip-flop based on standard norms (equation 2.50), furthermore the algebraic one are more or less linear (Figure 3.28).

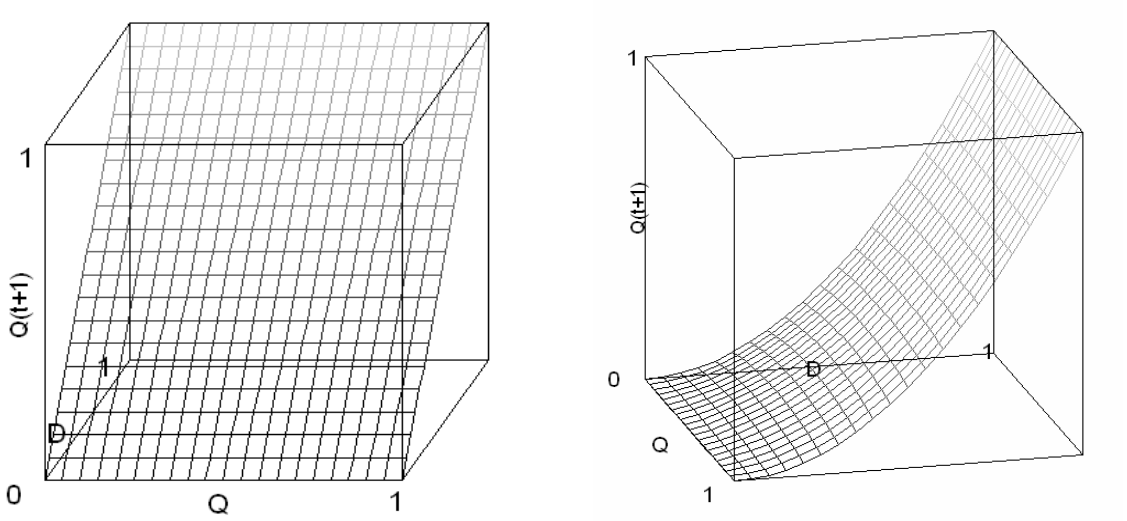


Figure 3.28

Standard Choi D F^3 Algebraic Choi D F^3

3.6.2 Drastic

In a similar way, using the drastic expressions for fuzzy set intersection and union the equation of the drastic Choi type D F^3 is defined by

$$Q(t+1) = (D)\ i_{DR}\ (D\ u_{DR}\ Q)\ i_{DR}\ ((1-Q)\ u_{DR}\ D) \tag{3.40}$$

3.6.3 Łukasiewicz

By substitute the Łukasiewicz norms in the Choi type D F^3 the characteristic equation can be defined as

$$Q(t+1) = (D)\ i_L\ (D\ u_L\ Q)\ i_L\ ((1-Q)\ u_L\ D) = \tag{3.41}$$

$$= \max\left(0,\ -1+\max\left(0,\ -1+D+\min\left(1,\ D+Q\right)\right)+\min\left(1,\ 1+D-Q\right)\right)$$

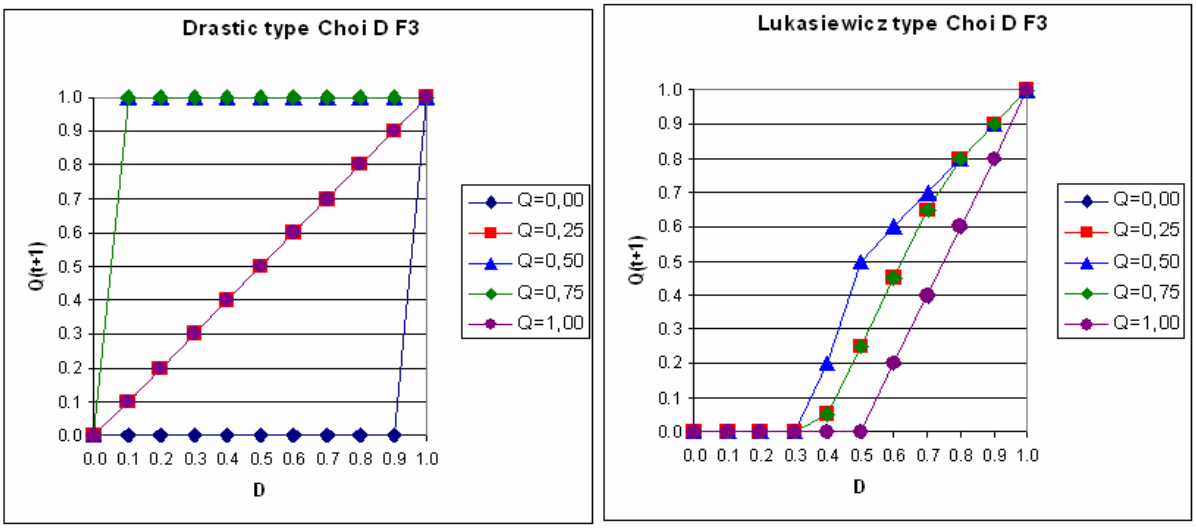


Figure 3.29

Drastic Choi D F^3 Łukasiewicz Choi D F^3

Figure 3.29 depicts the transfer characteristics of drastic and Łukasiewicz Choi D F^3s.

3.6.4 Yager

The next state of the Choi D flip-flop equation based on Yager-class operators is defined as

$$Q(t+1) = (D)\ i_Y\ (D\ u_Y\ Q)\ i_Y\ ((1-Q)\ u_Y\ D) = \tag{3.42}$$

$$= 1 - \min\left(1, \left(\left(1 - \min\left(1, \left(D^w + (1-Q)^w\right)^{1/w}\right)\right)^w + \min\left(1, \left((1-D)^w + M\right)^{1/w}\right)^w\right)^{1/w}\right)$$

$$M = \left(1 - \min\left(1, \left(D^w + Q^w\right)^{1/w}\right)\right)^w$$

The sections of the surface (Figure 3.30) are quasi sigmoid. Comparing the figures belonging to the two types of fuzzy D flip-flop based on Yager norms (Figure 3.22), it can be seen that for the same value of *Q*, the sharpness clearly differs.

3.6.5 Dombi

By applying the Dombi norms the maxterm form of the unified equation (2.50) can be defined as

$$Q(t+1) = (D)\ i_D\ (D\ u_D\ Q)\ i_D\ ((1-Q)\ u_D\ D) = \tag{3.43}$$

$$= 1/\left(1 + \left(\left(\left(\left(-1+\frac{1}{D}\right)^{\alpha} + \left(\left(\left(-1+\frac{1}{D}\right)^{-\alpha} + \left(-1+\frac{1}{Q}\right)^{-\alpha}\right)^{-1/\alpha}\right)^{\alpha}\right)^{1/\alpha}\right)^{\alpha} + M\right)^{1/\alpha}\right)$$

$$M = \left(\left(\left(-1+\frac{1}{D}\right)^{-\alpha} + \left(-1+\frac{1}{1-Q}\right)^{-\alpha}\right)^{-1/\alpha}\right)^{\alpha}$$

The extensive investigations show that the transfer characteristics of the Choi type fuzzy D flip-flop are linear with several breakpoints (Figure 3.31).

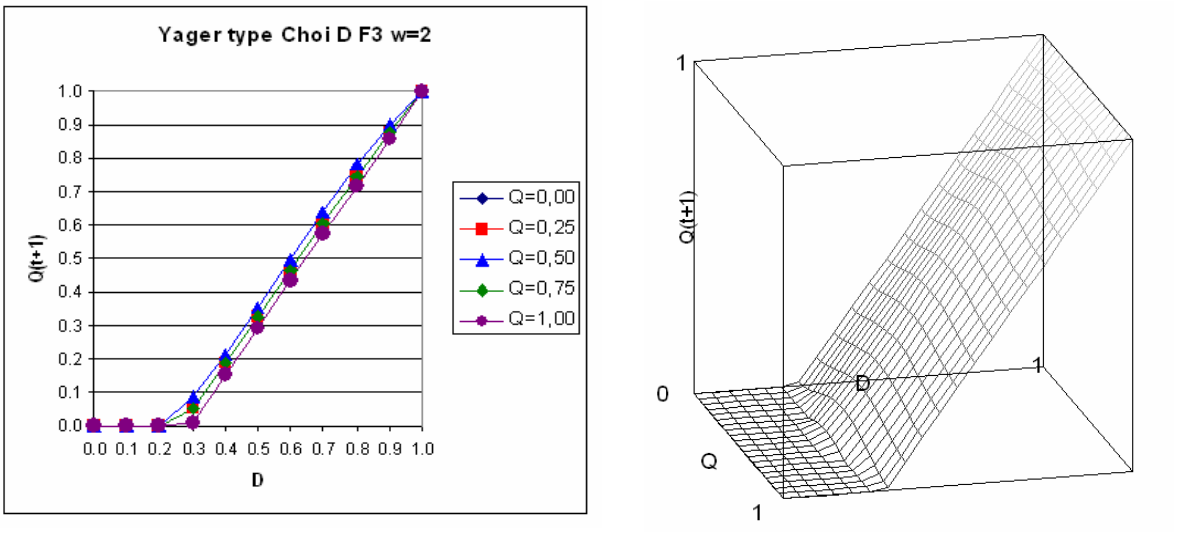


Figure 3.30
Yager Choi D F^3s

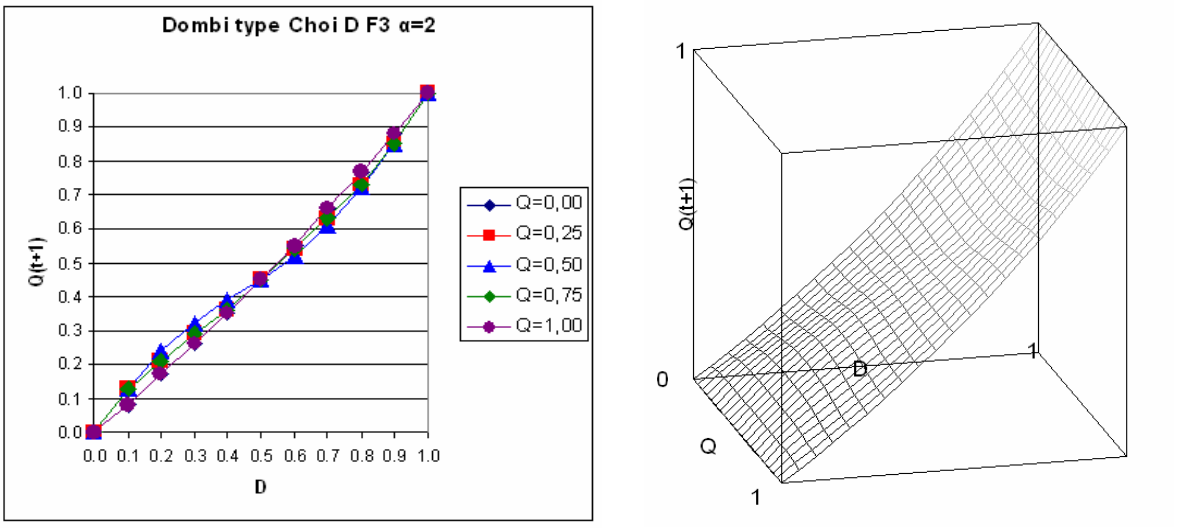


Figure 3.31
Dombi Choi D F^3s

3.6.6 Hamacher

The characteristic equation of the Choi D F^3 based on Hamacher families is

$$Q(t+1) = (D)\ i_H\ (D\ u_H\ Q)\ i_H\ ((1-Q)\ u_H\ D) = \tag{3.44}$$
$$= -D\big(Q + D(1+Q(-2+\nu))\big)\big(-1+Q+D(1+Q(-2+\nu)-\nu)\big) /$$
$$Q(-1+\nu)^2 - Q^2(-1+\nu)^2 + \nu + D^3(-1+\nu)\big(-1+Q(4-3\nu)+\nu+Q^2(-4+3\nu)\big) -$$
$$D^2(-1+\nu)\big(-3+Q(10-8\nu)+2\nu+2Q^2(-5+4\nu)\big) + D(-1+\nu)\big(-2+Q(6-5\nu)+\nu+Q^2(-6+5\nu)\big)$$

3.6.7 Frank

Based on Frank norms the next state of the Choi type D F^3 is

$$Q(t+1) = (D)\ i_F\ (D\ u_F\ Q)\ i_F\ ((1-Q)\ u_F\ D) = \tag{3.45}$$

$$= \log_s\left(1+\left(-1+s^D\right)\cdot M\cdot N/\left(-1+s\right)^2\right)$$

$$M = \left(-1+s/\left(1+\left(-1+s^{1-D}\right)\left(-1+s^{1-Q}\right)/\left(-1+s\right)\right)\right)$$

$$N = \left(-1+s/\left(1+\left(-1+s^{1-D}\right)\left(-1+s^{Q}\right)/\left(-1+s\right)\right)\right)$$

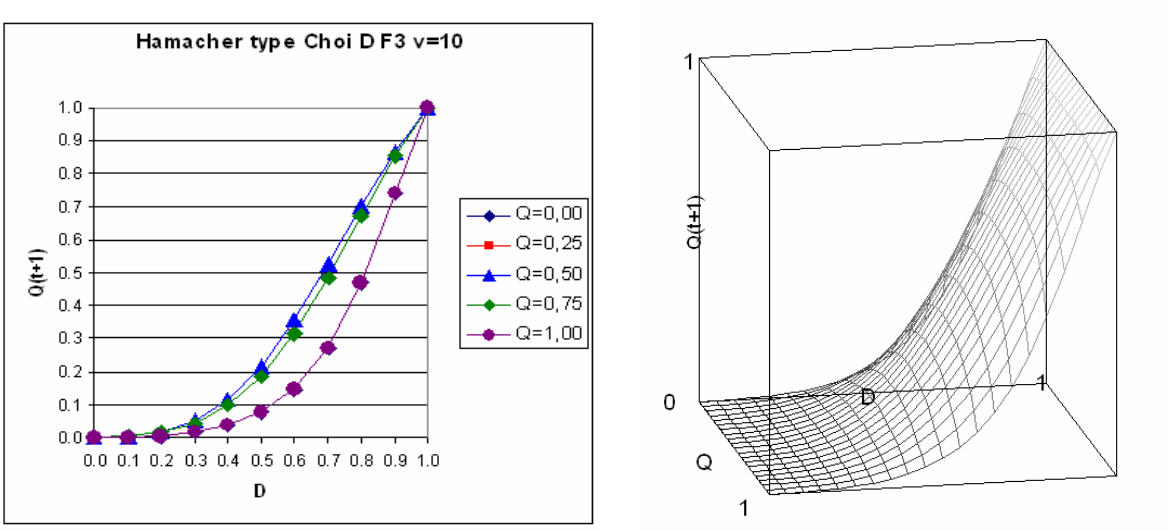


Figure 3.32
Hamacher Choi D F^3s

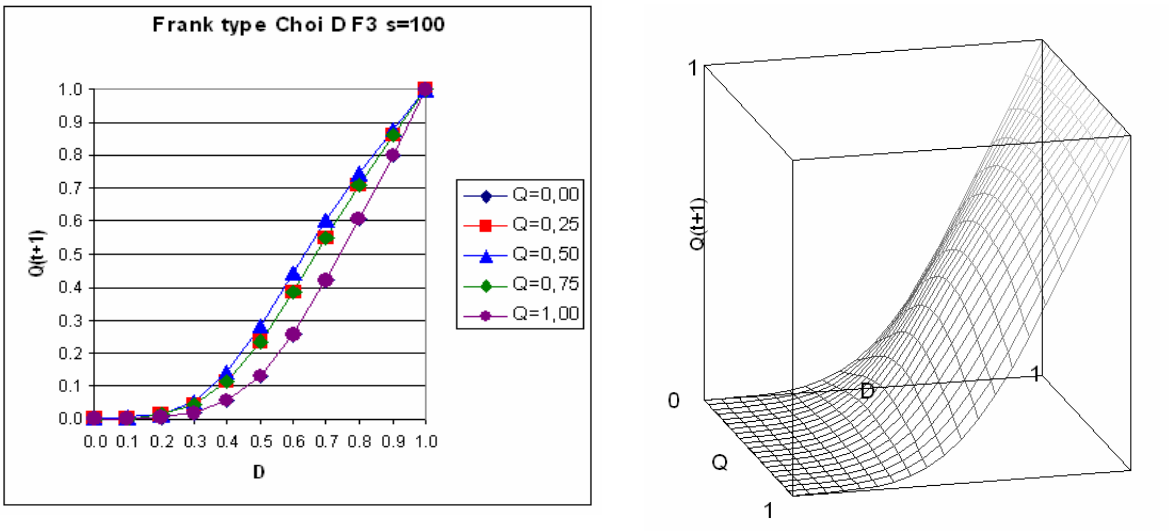


Figure 3.33
Frank Choi D F^3s

Figures 3.32 and 3.33 show that transfer characteristics of Choi D F^3s based on Hamacher and Frank norms are more or less s-shaped.

3.6.8 Dubois-Prade

Using the parameterized families of Dubois-Prade norm the Choi fuzzy D flip-flop is defined as

$$Q(t+1) = (D)\ i_{DP}\ (D\ u_{DP}\ Q)\ i_{DP}\ ((1-Q)\ u_{DP}\ D) = \tag{3.46}$$
$$= D\left(1+D-D(1-Q)-Q-\min(1-,D,1-Q)\right)\left(D+Q-DQ-\min(1-d,D,Q)\right)/M$$
$$M = \max(d,1-D,1-Q)\max(d,1-D,Q)\max\left(d,D,\left(D+Q-DQ-\min(1-d,D,Q)\right)/N\right)\cdot O$$
$$N = \max(d,1-D,1-Q)$$
$$O = \max\left(d,\left(1+D-D\left(1-Q-Q-\min(1-d,D,1-Q)\right)\right)/\max(d,1-D,Q),P\right)$$
$$P = D\left(D+Q-DQ-\min(1-d,D,Q)\right)/N\max\left(d,D,\left(D+Q-DQ-\min(1-d,D,Q)\right)/N\right)$$

3.6.9 Schweizer-Sklar

The next equation is defined by proposing the Schweizer-Sklar norms in the maxterm form of Choi D F^3 formulae

$$Q(t+1) = (D)\ i_{SS}\ (D\ u_{SS}\ Q)\ i_{SS}\ ((1-Q)\ u_{SS}\ D) = \tag{3.47}$$
$$= \max\left(0,-1+\left(1-\max\left(0,-1+(1-D)^p+Q^p\right)^{1/p}\right)^p+M\right)^{1/p}$$
$$M = \left(\max\left(0,-1+D^p+\left(1-\max\left(0,-1+(1-D)^p+(1-Q)^p\right)^{1/p}\right)^p\right)^{1/p}\right)^p$$

3.6.10 Fodor

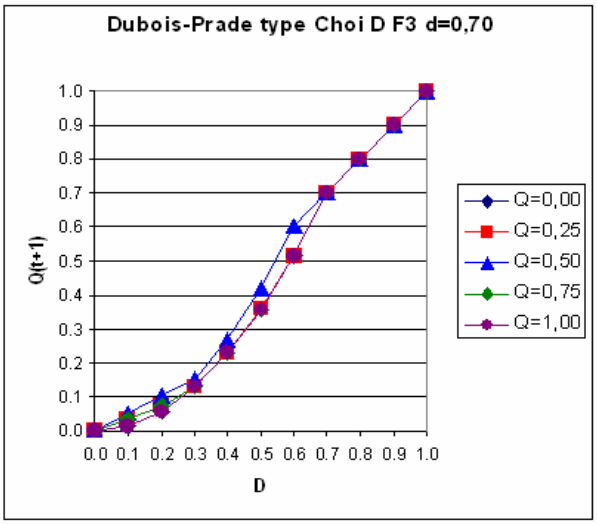
Based on Fodor norms the corresponding unified $Q(t+1)$ definition is

$$Q(t+1) = (D)\ i_F\ (D\ u_F\ Q)\ i_F\ ((1-Q)\ u_F\ D) = \tag{3.48}$$
$$= \frac{1}{2}\left(\max\left(0,\frac{1}{2}\left(-2+\max(0,D-Q)+M+N\right)\right)+O\right)$$
$$M = \max\left(0,-1+D+\frac{1}{2}\left(\max(0,-1+D+Q)+\min(D,Q)\right)\right)$$
$$N = \min(D,1-Q)+\min\left(D,\frac{1}{2}\left(\max(0,-1+D+Q)+\min(D,Q)\right)\right)$$
$$O = \min\left(\frac{1}{2}\left(\max(0,D-Q)+\min(D,1-Q)\right),\frac{1}{2}P\right)$$

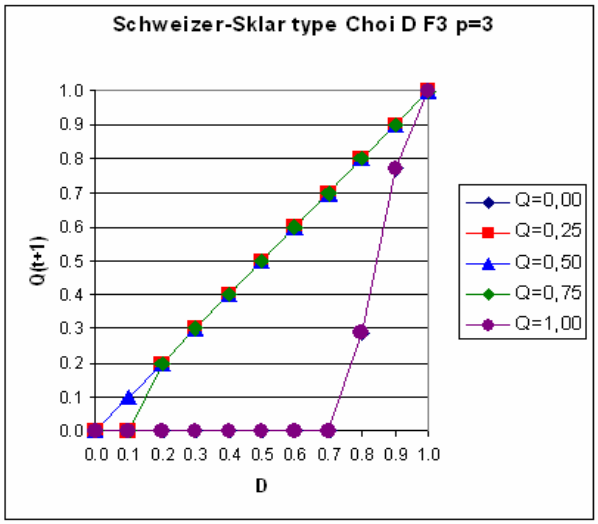
$$P = \left(\max\left(0, -1 + D + \frac{1}{2}\left(\max\left(0, -1 + D + Q\right) + \min\left(D, Q\right)\right)\right) + T \right)$$

$$T = \min\left(D, \frac{1}{2}\left(\max\left(0, -1 + D + Q\right) + \min\left(D, Q\right)\right)\right)$$

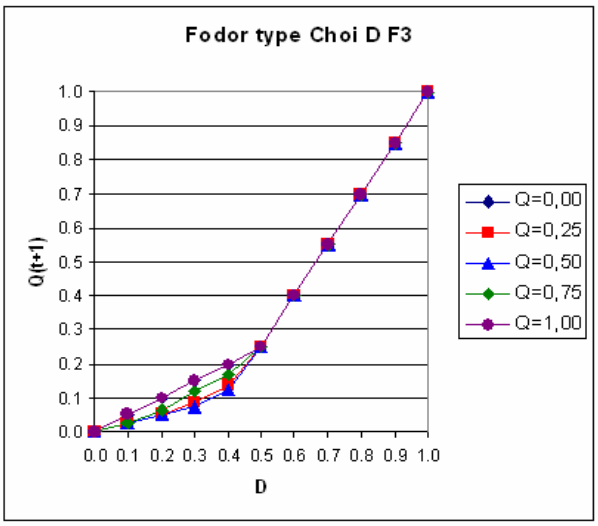
Figure 3.34 depicts the character of the next states of Choi D F^3s based on Dubois-Prade, Schweizer-Sklar and Fodor norms.



Dubois-Prade Choi D F^3s



Schweizer-Sklar Choi D F^3s



Fodor Choi D F^3

Figure 3.34

Comparing the characteristic equation of the new fuzzy D flip-flop (equation 3.33), with the traditional expression proposed by Choi et al. (equation 2.50), there is an essential difference between the two fuzzy flip-flops. Substitute $D = J = 1\text{-}K$, the two formulas differ in

the first member. and $D = D \vee D$ hold only in the exceptional case when the t-conorm is idempotent [14]:

$T(x, x) = x$ and $S(x, x) = x$ for all $x \in [0,1]$;

It can be proved [25] that t-norm T is idempotent iff T = min, and t-conorm S is idempotent if and only if S = max. For example, using the algebraic norm

$$u_A(a,a) = a + a - a \cdot a = 2 \cdot a - a^2 = a \qquad (3.49)$$

is true only in the borderline cases, i.e. when $a = 0$, or $a = 1$. It is surprising how much the satisfaction of idempotence influences the behavior of the fuzzy D flip-flops. Although, the $D \to Q(t+1)$ Choi type fuzzy D flip-flop characteristics for Łukasiewicz, Yager, Hamacher, Frank and Dubois-Prade norms also present more or less sigmoid behavior. Comparing the figures belonging to the two types of fuzzy D flip-flop with the same norms it can be seen that for the same value of Q, the curvature differs, which fact leads to a rather different behavior in the applications.

3.7 Summary

This chapter introduces new types of fuzzy flip-flop. First defines a collection of new Interval Valued - , then the reset and set type - , and modified non-associative type of fuzzy J-K flip-flop. The introduction of the unified equation of reset and set type has four advantages: 1) From a practical aspect it is confusing that reset and set type F^3s sometimes do have very different behavior. (The unique exception is the Fodor F^3.) 2) This equation simultaneously involved both set and reset characteristics. 3) In order to extend the binary J-K flip-flop to a fuzzy flip-flop smoothly. 4) The aim of the unification was the use of fuzzy flip-flops as neuron. In order to test the behavior of the curves with regard to the shape and curvature, several characteristics were performed.

The unified approach of fuzzy J-K flip-flops is defined in section 3.4 where a set of norms, combined with the standard negation, was analyzed in order to investigate, whether and to what degree they present more or less sigmoid (s-shaped) $J \to Q(t+1)$ characteristics in particular cases, when K = 1 - Q (unified fuzzy J-K flip-flop with feedback) with fixed value of Q. In section 3.5 the new fuzzy D flip-flop derived from the unified fuzzy J-K one,

$K = 1 - J$ was introduced. The transfer characteristics $D \to Q(t+1)$ behavior is studied based on the above mentioned eleven norms. Finally, in section 3.6, the characteristic equations of Choi type fuzzy D flip-flop are determined, and investigated their properties.

Table 3.2 summarizes the transfer characteristics shape for every combination of unified J-K, D and Choi D type F^3s with all above mentioned fuzzy operation pairs [71]. The parameters of the norms were chosen for suitable values. The S notation refers to sigmoid character, the NS to non-sigmoid shape. In conclusion, the fuzzy flip-flops based on Łukasiewicz, Yager, Hamacher, Frank and Dubois-Prade norms give quasi sigmoid transfer characteristics for selected *Q* values, independently from the types of F^3 additionally, the Dombi one in case of fuzzy J-K flip-flop with feedback, and all other fuzzy flip-flops investigated have non-sigmoid behavior. The next chapter deals with investigation of the above mentioned types of fuzzy flip-flop based networks as a novel implementation possibility of multilayer perceptron neural networks.

TABLE 3.2 VARIOUS FUZZY FLIP-FLOPS CHARACTERISTICS

Fuzzy operation	J-K (unified; *K*=1-*Q*)	D (*K*=1-*J*)	Choi D
Standard (min-max)	NS	NS	NS
Algebraic	NS	NS	NS
Drastic	NS	NS	NS
Łukasiewicz	**S**	**S**	**S**
Yager	**S**	**S**	**S**
Dombi	**S**	**NS**	**NS**
Hamacher	**S**	**S**	**S**
Frank	**S**	**S**	**S**
Dubois-Prade	**S**	**S**	**S**
Schweizer-Sklar	NS	NS	NS
Fodor	NS	NS	NS

The proposed new types of fuzzy flip-flop could play important role in applications like: distributed modeling, fuzzy algorithmic state machines, dynamic controllers, classifiers etc. A fuzzy logical circuit can be characterized as the combination of fuzzy flip-flops as fundamental circuits (e.g. fuzzy inference circuit or fuzzy inference chip). The fuzzy extension of a sequential circuit combined with fuzzy memory modules should be the fundamental idea of the realization of fuzzy computer hardware.

The main results of chapter 3 can be summarized as follows:

Statement 1.

I defined a series of new fuzzy flip-flops and I investigated their properties.

1.1 *I have defined the following new concepts:*

- *interval valued fuzzy J-K flip-flops and I investigated their properties based on standard and algebraic operations,*
- *reset and set type fuzzy J-K flip-flops based on Yager, and Dombi operations,*
- *unified fuzzy J-K flip-flops based on Yager, Dombi, Hamacher, Frank, Dubois-Prade Schweizer-Sklar and Fodor fuzzy operations,*
- *fuzzy D flip-flops and I investigated their properties based on standard, algebraic, drastic, Łukasiewicz, Yager, Dombi, Hamacher, Frank, Dubois-Prade, Schweizer-Sklar and Fodor fuzzy operations,*
- *fuzzy Choi type D flip-flops based on algebraic, drastic, Łukasiewicz, Yager, Dombi, Hamacher, Frank, Dubois-Prade, Schweizer-Sklar and Fodor fuzzy operations.*

I determined the characteristic equations and I investigated the properties of all the above mentioned 32 new types of fuzzy flip-flop.

1.2 *I proved analytically that the "modified Fodor fuzzy J-K flip-flop" satisfies the very special property of the reset and set formula being equivalent. This is the only F^3 type as far when the two expressions (minterm and maxterm) lead to the same flip-flop definition.*

1.3 *I have conducted extensive investigations and I found that the $J \to Q(t+1)$ transfer characteristics of fuzzy J-K flip-flops with feedback based on Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade norms, further the $D \to Q(t+1)$ characteristics of (new) fuzzy D flip-flop and fuzzy Choi D flip-flop of Łukasiewicz, Yager, Hamacher, Frank and Dubois-Prade operations show quasi sigmoid curvature for some selected Q values, while all other F^3s investigated (including those to be found in the proceeding literature and novel ones defined by myself) have non-sigmoid behavior.*

Chapter 4

Multilayer Perceptrons Based on Fuzzy Flip-Flops

In the last decades many different approaches regarding to the hybridization of neural networks and fuzzy systems have been introduced and studied [51], [64], [99], [100] as new neuro-fuzzy structures. Based on this idea, in this chapter the concept of the fuzzy flip-flop neuron has been introduced. The Fuzzy Flip-Flop based Neural Networks (FNNs) as a novel implementation possibility of multilayer perceptron neural networks is proposed. The FNN neuron element may be any F^3 derived from the original F^3 with more or less sigmoid transfer characteristics. After simplifications the fuzzy J-K, D and Choi D flip-flop neurons block diagram for a fix Q value is given in section 4.1. The proposed fuzzy flip-flop based multilayer perceptron architecture used for function approximation is presented. The Levenberg-Marquardt algorithm is applied to demonstrate that the proposed FNNs built up from fuzzy J-K, D and Choi D flip-flops based on algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade operations can be used for learning and approximating one and two dimensional trigonometric functions and a benchmark *p*H problem model.

4.1 Fuzzy Neural Networks

4.1.1 Fuzzy Flip-Flop Neurons

I proposed the construction of a neuron unit, a combinational sigmoid generator derived from arbitrary fuzzy J-K flip-flop where $\overline{Q}$ is fed back to K and (old) Q is fixed (Figure 4.1).

In this approach, the output of fuzzy J-K flip-flop neuron depends from the value of Q_{fix} and input values of J. Substitute $\overline{K} = Q$ (1 - K = Q) in the unified formula of the fuzzy J-K flip-flop (eq. 2.45), for a fix Q value, the characteristic equation of fuzzy J-K flip-flop neuron is

$$Q_{out} = (J\ u\ Q_{\text{fix}})\ i\ (J\ u\ Q_{\text{fix}})\ i\ \left(Q_{\text{fix}}\ u\ \left(1-Q_{\text{fix}}\right)\right) \tag{4.1}$$

where *i* and *u* denote t-norm and t-conorm.

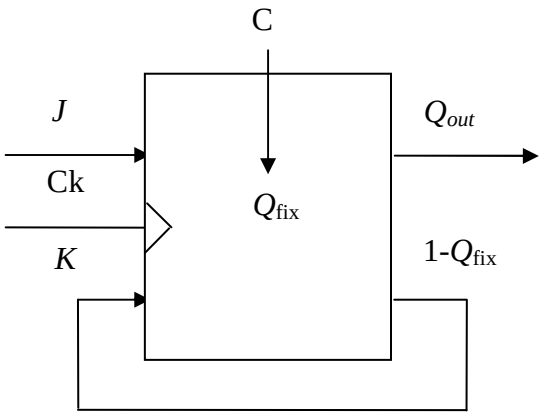


Figure 4.1 Fuzzy J-K flip-flop neuron

The clocked fuzzy J-K flip-flop neuron circuit can be implemented using hardware blocks (denoted by *i*, *u* and *n* symbols) to realize various t-norms, t-conorms and fuzzy negations [123]. Since t-norms and t-conorms are functions from the unit square into the unit interval, the fuzzy J-K flip-flop neuron block diagram differs from the binary J-K flip-flop structure. The input *J* is driven by a synchronized clock pulse in the sample-and-hold (S/H) circuit (Figure 4.2) which could be a simple D flip-flop used as register.

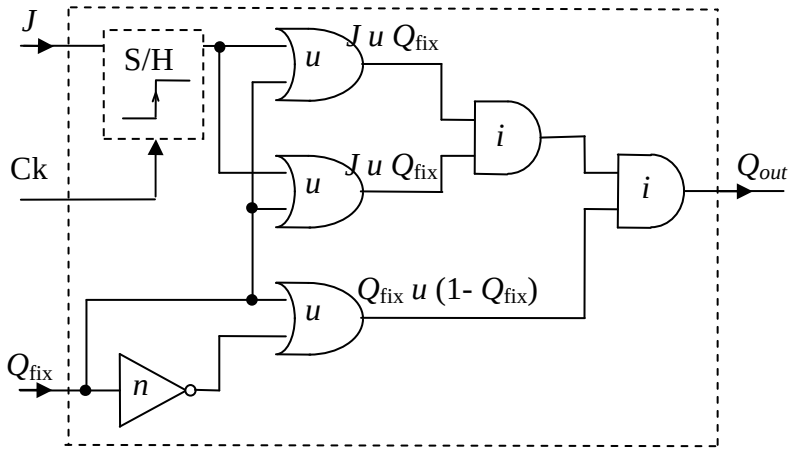


Figure 4.2 Fuzzy J-K flip-flop neuron block diagram (after simplification)

I proposed the construction of the fuzzy D flip-flop neuron (Figure 4.3) which is a combinational sigmoid generator. This unit is derived from arbitrary fuzzy J-K flip-flop by connecting the inputs of the fuzzy J-K flip-flop in a particular way, namely, by applying an inverter in the connection of the input *J* to *K*. Starting from the fundamental equation of the

fuzzy J-K flip-flop and substitute $\overline{K} = J$ in equation (2.45) and let $D = J$ for a fix Q value, the characteristic equation of fuzzy D flip-flop neuron is

$$Q_{out} = (D \; u \; D) \; i \; (D \; u \; Q_{\text{fix}}) \; i \; \left(D \; u \; \left(1 - Q_{\text{fix}}\right)\right) \tag{4.2}$$

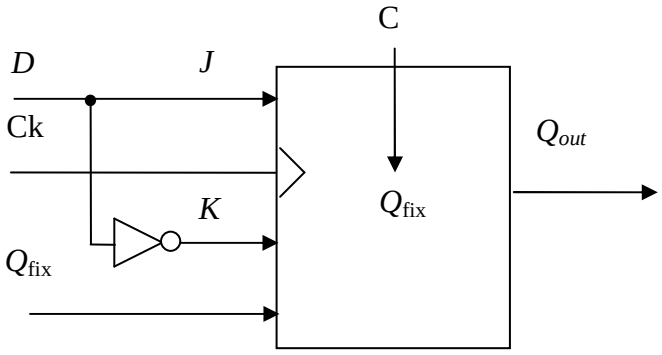


Figure 4.3 Fuzzy D flip-flop neuron

Interconnecting the blocks of fuzzy operations in a different way the fuzzy D flip-flop neuron block diagram is obtained (Figure 4.4).

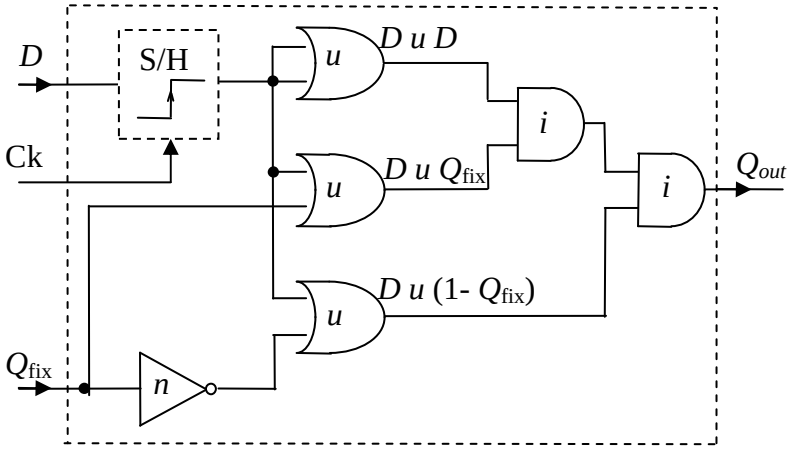


Figure 4.4 Fuzzy D flip-flop neuron block diagram (after simplification)

Finally, the fuzzy Choi D flip-flop neuron is defined. This combinational sigmoid generator is derived from arbitrary fuzzy Choi D flip-flop. The characteristic equation of fuzzy Choi D flip-flop neuron derived from equation (2.50) is

$$Q_{out} = (D) \; i \; (D \; u \; Q_{\text{fix}}) \; i \; \left(D \; u \; \left(1 - Q_{\text{fix}}\right)\right) \tag{4.3}$$

The fuzzy Choi D flip-flop neuron and the corresponding neuron block diagram are presented in Figures 4.5 and 4.6.

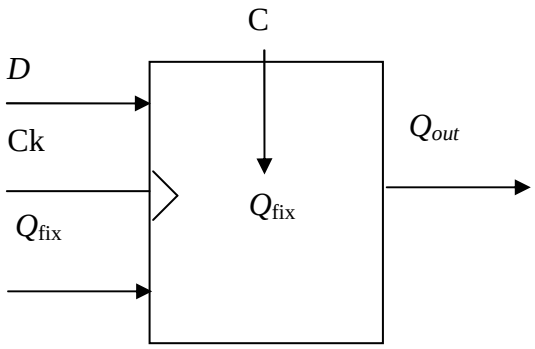


Figure 4.5 Fuzzy Choi D flip-flop neuron

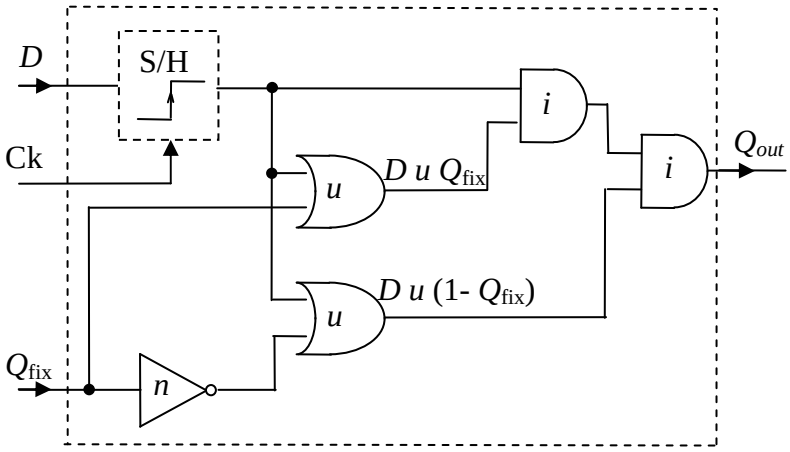


Figure 4.6 Fuzzy Choi D flip-flop neuron block diagram (after simplification)

The fuzzy J-K, D and Choi D flip-flop neurons are obviously combinational circuits. From the neural networks perspective (regarding the ability to use the learning and adaptation mechanisms used with classic neuron models), suitable norms may be deployable for defining fuzzy neurons. It is known that simple parametrical t-norms, furthermore simple fuzzy flip-flop characteristic equation are uncomplicated for tuning and hardware realization. Rudas et al. [97] proposed the hardware implementation of a generation of parametric families of fuzzy connectives together with min-max, Łukasiewicz and drastic t-norms and t-conorms. In [123] Zavala et al. used FPGA technology to implement the above mentioned norms into an 8 bit single circuit that allows operation selection. The same technology was applied for hardware implementation of fuzzy D flip-flop neurons based on Łukasiewicz norms published in [76]. Simulation results, the hardware resources, number of logical gates, logic levels required, I/O used and timing delay consumed by Łukasiewicz operations and overall fuzzy D flip-flop neurons were presented. The proposed circuit was built up using core blocks which realize Łukasiewicz t-norm and t-conorm. The implementation was an 8 bit data width architecture

and additionally two bit as selection for operations. These blocks can also be chained to form more complex structures such as FNN.

In the next the investigation of such possible F^3 networks as new alternative types of neural network are presented. The combinational part of the circuit is embodied in a single layer neural network in which weights and biases are adjusted on the basis of training situations. Numerical considerations highlight the performance of the design process.

4.1.2 Architecture of the Multilayer Perceptron Based on Fuzzy Flip-Flops

A very commonly used architecture of neural networks is the multilayer feedforward network, which allows signals to flow from the input units to the output units, in a forward direction. Two trainable layer networks with sigmoid transfer functions in the hidden layer and linear transfer functions in the output layer are continuous functions approximators [27], analogously the neural system model proposed in this research is based on two hidden layers constituted from fuzzy flip-flop neurons. These networks are sensitive to the number of neurons in their hidden layers. Too few neurons can lead to underfitting too many neurons can cause similarly undesired overfitting. Using two hidden layers the function approximation process becomes more manageable. In particular the local features are extracted in the first, while the global features in the second hidden layer [27], [39]. The first hidden layer neurons are used to partition the input space into regions and learn the local features. A neuron in the second layer learns the global features for a particular region of the input space and outputs zero elsewhere.

The functions to be approximated are represented by a set of input/output pairs. The network architecture built up from fuzzy J-K neurons is shown in Figure 4.7. The output of the *j*th summation unit is obtained by first forming a weighted linear combination of the *n* input values and adding a bias, to give $y_j^{(1)} = \sum_{i=1}^{n} w_{ji}^{(1)} x_i + b_j^{(1)}$. Here $w_{ji}^{(1)}$ denotes a weight in the first layer (from input *i* to hidden unit *j*) and $b_j^{(1)}$ denotes the bias from unit *j*. The first layer input vector denoted by $\underline{y}^{(1)}$ is computed as:

$\underline{y}^{(1)} = \mathbf{W}^{(1)} \underline{x} + \underline{b}^{(1)}$ where $\mathbf{W}^{(1)}$ represents the first hidden layer weights vector matrix.

The output vector $a^{(1)}$, of Layer 1, is computed by applying a transfer function to the net input according to $\underline{a}^{(1)} = f^{(1)}\left(\underline{y}^{(1)}\right)$. The output of the network is $\underline{a}^{(3)} = purelin\left(\mathbf{W}^{(3)} \cdot \underline{a}^{(2)} + \underline{b}^{(3)}\right)$

where *purelin* is a linear transfer function.

In this approach the weighted input values are connected to input *J* of the new fuzzy flip-flop neuron based on a pair of t-norm and t-conorm, having quasi sigmoid transfer characteristics [73], [75]. The output signal is then computed as the weighted sum of the input signals transformed by the transfer function.

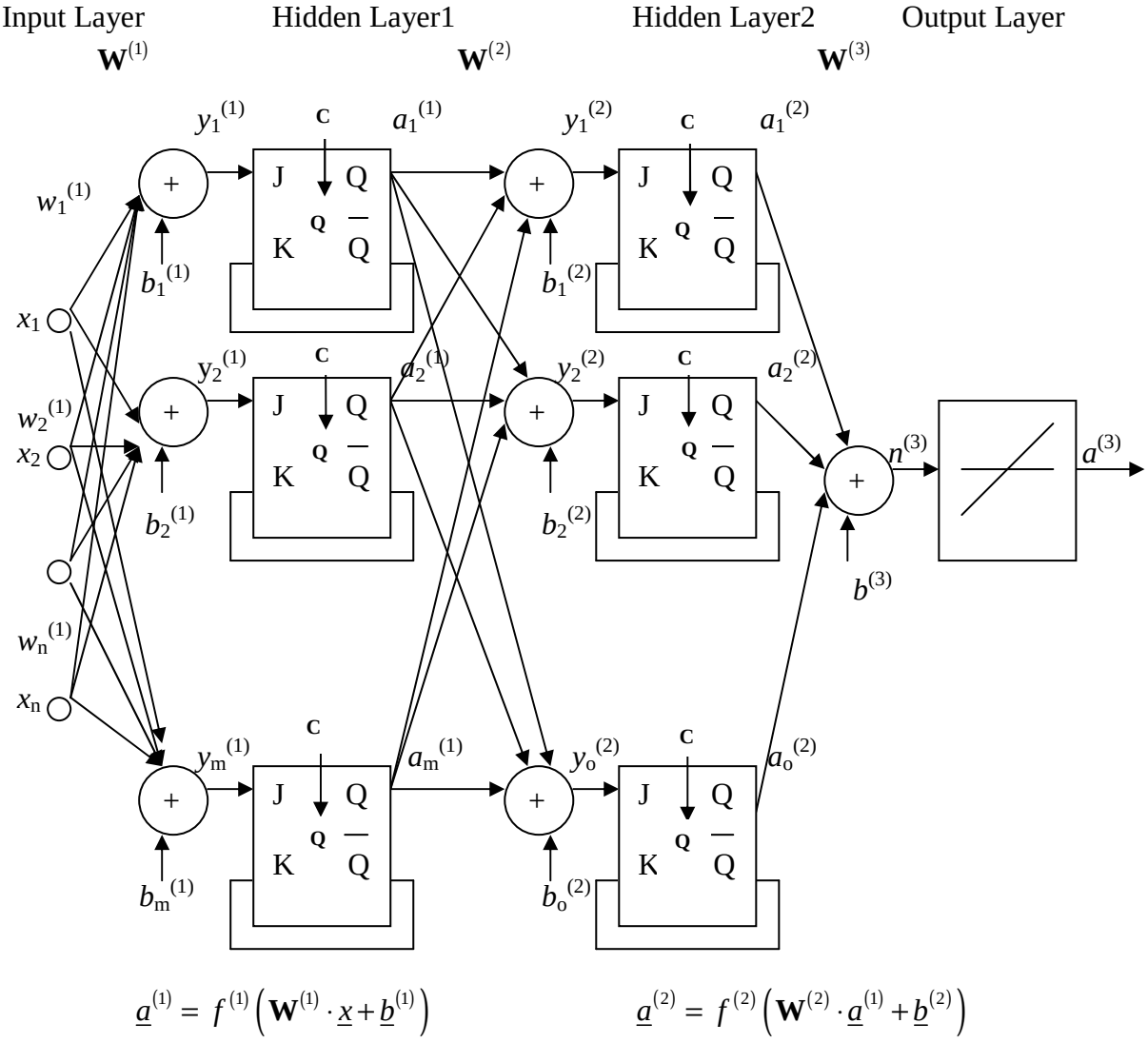


Figure 4.7 Fuzzy flip-flop based neural network

4.2 Training the Fuzzy Flip-Flop Networks with the Levenberg – Marquardt Algorithm

The proposed fuzzy neural network is applied in order to approximate test functions. The nonlinear characteristics exhibited by fuzzy neurons are represented by quasi sigmoid transfer functions given by fuzzy J-K, D and Choi D flip-flop neurons based on algebraic,

Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade operations. The network activation function is the same at each hidden layer from unit to unit. In the simulations the present state value Q belonging to each fuzzy flip-flop has been fixed, $Q = 0.32$.

The number of neurons was chosen after experimenting with different size hidden layers. Smaller neuron numbers in the hidden layer result in worse approximation properties while increasing the neuron number results in better performance but longer simulation time.

The proposed Levenberg-Marquardt algorithm is best suited for function approximation problems where the network has fewer than one hundred weights and the approximation must be very accurate. In many cases it finds a solution even if it starts very far from the final minimum. The LM algorithm is one of the most popular training methods for feedfordward neural networks despite of its high memory requirements and high complexity [34].

To train this kind of network with the usual LM technique the derivatives of the transfer functions have to be calculated. During network training, the weights and thresholds are first initialized to small random values and the network was trained in order to minimize the network performance function in accordance with Levenberg-Marquardt algorithm with 100 maximum numbers of epochs as more or less sufficient. This can be used for training any network as long as its inputs, weights and transfer functions can be differentiated. During the simulations the input data patterns were used like this: the first 60 percent for training and the remaining 20-20 percents for test and validation results [39]. In order to check the goodness of the network training the Mean Squared Error (MSE) as a measure of the error made by the FNN was used. The errors for all input patterns were propagated backwards from the output layer towards the input layer. The corrections to the weights were selected to minimize the residual error between actual and desired outputs. This algorithm can be viewed as a generalized least squares technique applied to the multilayer perceptron. The chosen target activation function is the *tansig* (hyperbolic tangent sigmoid transfer function) a Matlab built in sigmoid transfer function. In order to train the networks with the LM algorithm Matlab2007, one working thread, under Windows 7 with AMD Athlon II X3 435 processor was used.

4.2.1 Performance Tests of Various FNNs

The function approximation performance of various fuzzy neural networks has been studied and compared. The FNNs are built up from fuzzy J-K, D and Choi D flip-flops based on algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade operations.

One and two dimensional test functions have been proposed furthermore, the *p*H benchmark problem consists of 101 datasets.

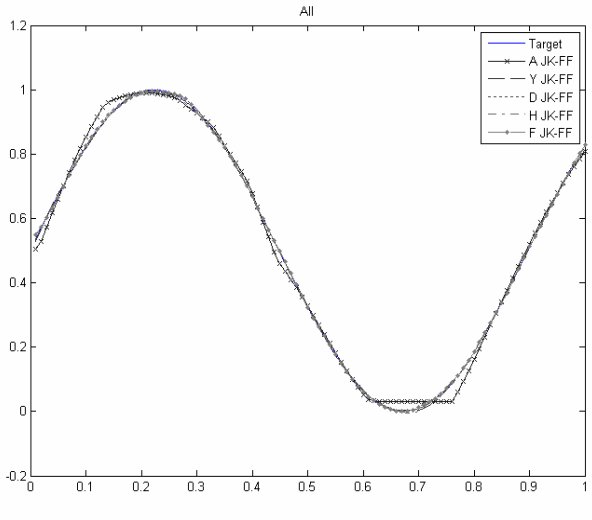
4.2.1.1 Single Sine Wave

A 1-4-4-1 fuzzy J-K flip-flop neurons based neural network is used to approximate a single period of a sine wave in order to demonstrate how the fuzzy operation type affected the function approximation performance. The expression of the function to be approximated was:

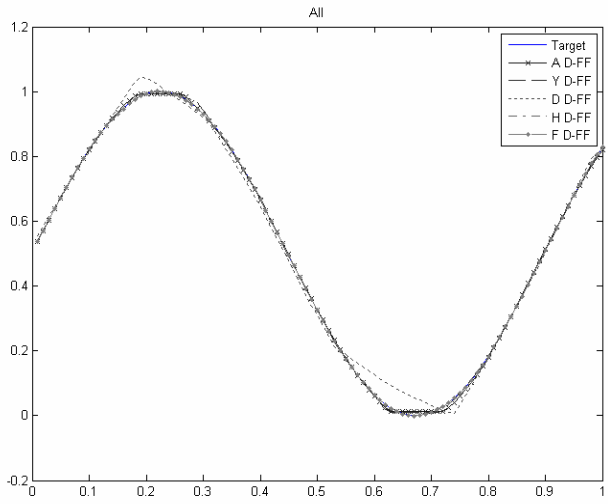
$$y = \sin(c_1 * x)/2+0.5, \tag{4.4}$$

where the input vector x generated a sinusoidal output y. The value of constant c_1 was chosen 0.07, to keep the wavelet in the unit interval. The test function is represented by 100 input/output data sets. The proposed network activation function is the same at each hidden layer from unit to unit. In these experiments parameters of Dombi, Yager, Hamacher, Frank and Dubois-Prade families were quasi optimized by comparing results by a large number of various parameter values, (α = 2, w = 2, v = 10, s = 100 and d = 0.7 resp.), for an universal Q = 0.32 value. Figure 4.8 a) presents the graphs of the simulations in case of fuzzy J-K flip-flop neuron based neural network. Figures 4.8 b) and c) compare the behavior of fuzzy D flip-flop and Choi type fuzzy D flip-flop based NNs. Table 4.1 summarizes the 300 runs average approximation goodness by indicating the minimal and median of the train MSE values for each of the ideal *tansig*, algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade types of FNNs. Comparing the median (median value of the array) MSE values, the Hamacher, Łukasiewicz and Frank types FNNs performed best using fuzzy J-K flip-flop neurons thus they can be considered as rather good function approximators. Dombi and Yager type FNNs also have good approximation properties in this approach. It can be observed that the algebraic and Dubois-Prade F^3 provides a fuzzy neuron with rather bad learning ability. According to the numerical illustrations the average of 300 runs mean squared errors in case of fuzzy D flip-flop type NNs, the best results after the idealistic *tansig* function are given by the Hamacher, Dubois-Prade, Frank and Łukasiewicz F^3s, which are followed by the Yager, algebraic and finally the Dombi one. In case of Choi type fuzzy D flip-flop based NNs the best function approximators are the Dubois-Prade and Łukasiewicz F^3s. It is surprising how much the satisfaction of idempotence influences the behavior of the fuzzy D flip-flop based NN. Comparing the simulation results belonging to the NNs based on the two types of fuzzy

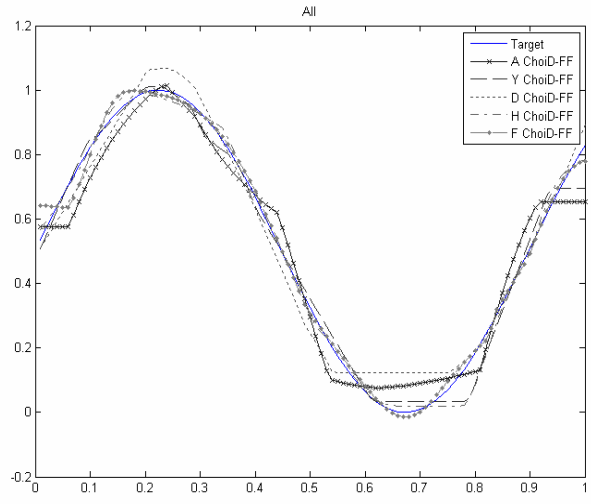
D flip-flop with the same norms, it can be seen that, for the same value of *Q*, the value of the MSE differs, which fact leads to a rather different behavior in the applications.



a) J-K FNN



b) D FNN



c) Choi D FNN

Figure 4.8 Single sine wave

TABLE 4.1 MSE VALUES: SINE WAVE

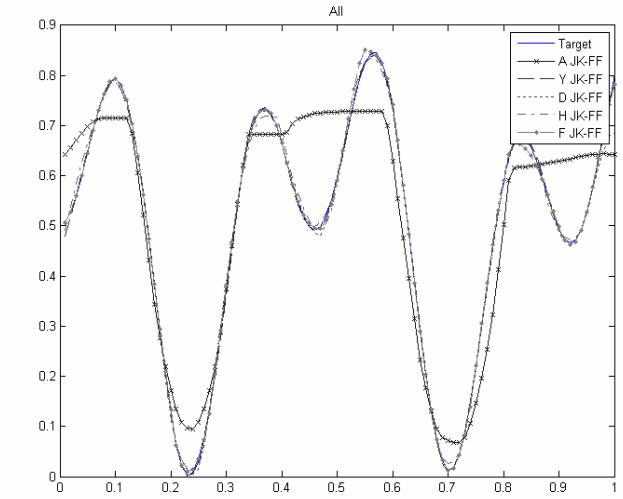
	F^3 Neuron Type					
Fuzzy Operation	J-K		D		Choi D	
	Minimum	Median	Minimum	Median	Minimum	Median
tansig	$2.72\text{x}10^{-11}$	$2.54\text{x}10^{-8}$	$2.72\text{x}10^{-11}$	$2.54\text{x}10^{-8}$	$2.72\text{x}10^{-11}$	$2.54\text{x}10^{-8}$
Algebraic	$5.19\text{x}10^{-3}$	$6.12\text{x}10^{-2}$	$5.43\text{x}10^{-5}$	$3.72\text{x}10^{-2}$	$1.87\text{x}10^{-3}$	$5.64\text{x}10^{-2}$
Łukasiewicz	$7.17\text{x}10^{-6}$	$1.24\text{x}10^{-2}$	$1.92\text{x}10^{-5}$	$2.03\text{x}10^{-2}$	$5.21\text{x}10^{-5}$	$2.19\text{x}10^{-2}$
Yager	$1.11\text{x}10^{-6}$	$3.79\text{x}10^{-2}$	$2.12\text{x}10^{-6}$	$2.94\text{x}10^{-2}$	$8.05\text{x}10^{-3}$	$7.49\text{x}10^{-2}$
Dombi	$1.44\text{x}10^{-8}$	$3.78\text{x}10^{-2}$	$1.13\text{x}10^{-2}$	$8.26\text{x}10^{-2}$	$1.85\text{x}10^{-2}$	$8.41\text{x}10^{-1}$
Hamacher	$3.64\text{x}10^{-6}$	$1.09\text{x}10^{-2}$	$2.46\text{x}10^{-8}$	$9.76\text{x}10^{-3}$	$3.16\text{x}10^{-4}$	$2.75\text{x}10^{-2}$
Frank	$3.09\text{x}10^{-6}$	$1.27\text{x}10^{-2}$	$1.70\text{x}10^{-7}$	$1.55\text{x}10^{-2}$	$2.75\text{x}10^{-4}$	$2.27\text{x}10^{-2}$
Dubois-Prade	$1.59\text{x}10^{-5}$	$6.41\text{x}10^{-2}$	$1.34\text{x}10^{-4}$	$1.13\text{x}10^{-2}$	$4.97\text{x}10^{-5}$	$1.44\text{x}10^{-2}$

4.2.1.2 Two Sine Waves

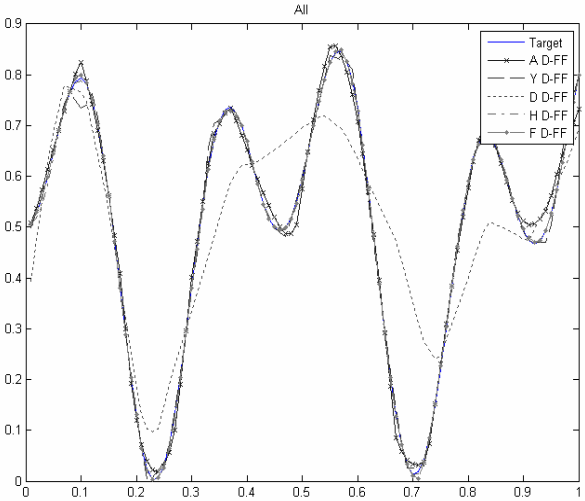
When instead of a single sine wave a more complex wave form was used in order to obtain the same results the neuron numbers in the hidden layers were increased to 8 neurons in each. A 1-8-8-1 F^3 based neural network is proposed to approximate a combination of two sine wave forms with different period lengths described with the equation

$$y = \sin(c_1*x)*\sin(c_2*x)/2+0.5. \quad (4.5)$$

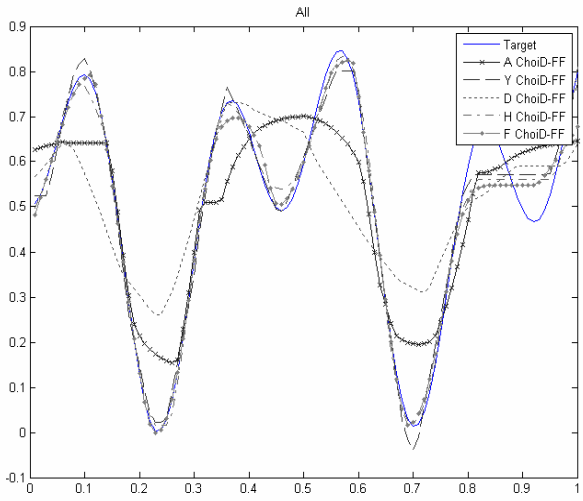
The values of constants c_1 and c_2 were selected to produce a frequency proportion of the two components 1:0.35. The test function is represented by 100 input/output data sets. The network function approximation capability has been compared in case of fuzzy J-K, D and Choi D flip-flop type NNs based on algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade norms.



a) J-K FNN



b) D FNN



c) Choi D FNN

Figure 4.9 Two sine waves

TABLE 4.2 MSE VALUES: TWO SINE WAVES

	F^3 Neuron Type					
Fuzzy Operation	J-K		D		Choi D	
	Minimum	Median	Minimum	Median	Minimum	Median
tansig	$2.96\text{x}10^{-8}$	$1.61\text{x}10^{-6}$	$2.96\text{x}10^{-8}$	$1.61\text{x}10^{-6}$	$2.96\text{x}10^{-8}$	$1.61\text{x}10^{-6}$
Algebraic	$1.13\text{x}10^{-2}$	$4.44\text{x}10^{-2}$	$1.18\text{x}10^{-3}$	$2.66\text{x}10^{-2}$	$8.69\text{x}10^{-3}$	$4.52\text{x}10^{-2}$
Łukasiewicz	$3.14\text{x}10^{-5}$	$6.72\text{x}10^{-3}$	$3.80\text{x}10^{-4}$	$1.72\text{x}10^{-2}$	$1.10\text{x}10^{-3}$	$1.93\text{x}10^{-2}$
Yager	$1.11\text{x}10^{-3}$	$3.35\text{x}10^{-2}$	$2.38\text{x}10^{-5}$	$2.21\text{x}10^{-2}$	$6.32\text{x}10^{-3}$	$4.74\text{x}10^{-2}$
Dombi	$5.75\text{x}10^{-4}$	$2.72\text{x}10^{-2}$	$3.37\text{x}10^{-2}$	$5.97\text{x}10^{-2}$	$3.35\text{x}10^{-2}$	$6.65\text{x}10^{-2}$
Hamacher	$9.84\text{x}10^{-5}$	$6.66\text{x}10^{-3}$	$1.95\text{x}10^{-6}$	$4.43\text{x}10^{-3}$	$1.05\text{x}10^{-3}$	$2.62\text{x}10^{-2}$
Frank	$1.35\text{x}10^{-5}$	$6.78\text{x}10^{-3}$	$3.01\text{x}10^{-6}$	$6.18\text{x}10^{-3}$	$9.24\text{x}10^{-4}$	$2.75\text{x}10^{-2}$
Dubois-Prade	$8.26\text{x}10^{-4}$	$4.29\text{x}10^{-2}$	$5.67\text{x}10^{-4}$	$5.35\text{x}10^{-3}$	$1.24\text{x}10^{-3}$	$1.82\text{x}10^{-2}$

It is interesting that according to the numerical illustrations the average of 300 runs median of the train MSE values (Tables 4.2) in this case; the sequence is almost the same as it was in the case of the single sine wave. By extensive simulation experiments it is proved that the function approximation goodness of FNNs based on fuzzy D flip-flops is superior to FNNs based on fuzzy Choi D flip-flops.

4.2.1.3 Two – Input Trigonometric Function

Now lets turn to more complicated two-input function composed of trigonometrically components to compare two parametrical FNNs, the Yager type with the Dombi one. A 1-20-20-1 feedforward neural network structure based on fuzzy J-K F^3s is proposed to approximate the following two dimensional function

$$y = \sin(c_1 * x_1)^5 * \cos(c_2 * x_2)^3 / 2 + 0.5. \qquad (4.6)$$

represented by 1600 input/output data sets. The 3D scenes using Yager and Dombi operators are depicted in Figure 4.10. The parameter of Dombi operators was fixed $\alpha = 2$, the Yager F^3 parameter was set to $w = 2$, and $Q = 0.32$. Comparing the minimum values of MSEs, the Dombi and Yager type neural networks can be considered as rather good function approximators. The MSEs appearing at the top of the graphs (Yager FF-MSE = 4.8228e-005, and Dombi FF-MSE = 7.7765e-006) are instantaneous values illustrating very well the 100 runs average approximation goodness.

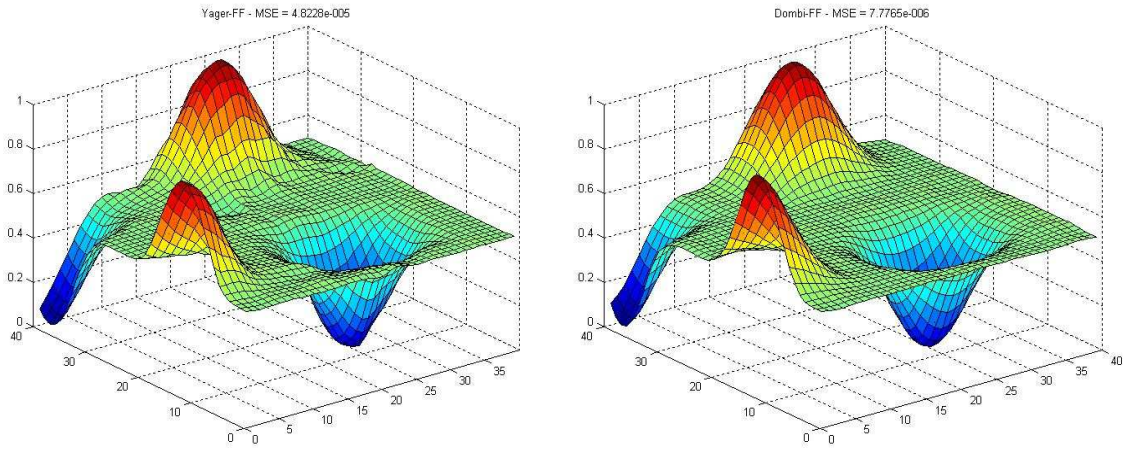


Figure 4.10

Yager FNN (trig. func.) Dombi FNN (trig. func.)

4.2.1.4 Two Dimensional Polynomial Input Function

A two dimensional polynomial input function was used for evaluating and comparing the approximation properties of the Yager and Dombi F^3 based neural networks. For this purpose a 1-20-20-1 feedforward neural network structure based on fuzzy J-K F^3s was proposed. The test function is

$$y = (x_1 - x_2)/(x_1 + x_2)/2 + 0.5. \quad (4.7)$$

represented by 1600 input/output data sets. The combination of the multi-dimensional linear function and the one-dimensional quasi-sigmoid function gave the characteristic sigmoid cliff response. The network with two hidden layers combined a number of response surfaces together through repeated linear combination and nonlinear activation functions. Figure 4.11 illustrates typical response surfaces of two input and a single output units. From these scenes, comparing the minimum MSEs, it is not difficult to ascertain that the best average performance is given again by the Dombi F^3 (Dombi FF-MSE = 3.3442e-006) based neural network which is followed by the Yager one (Yager FF-MSE = 1.643e-005) [72].

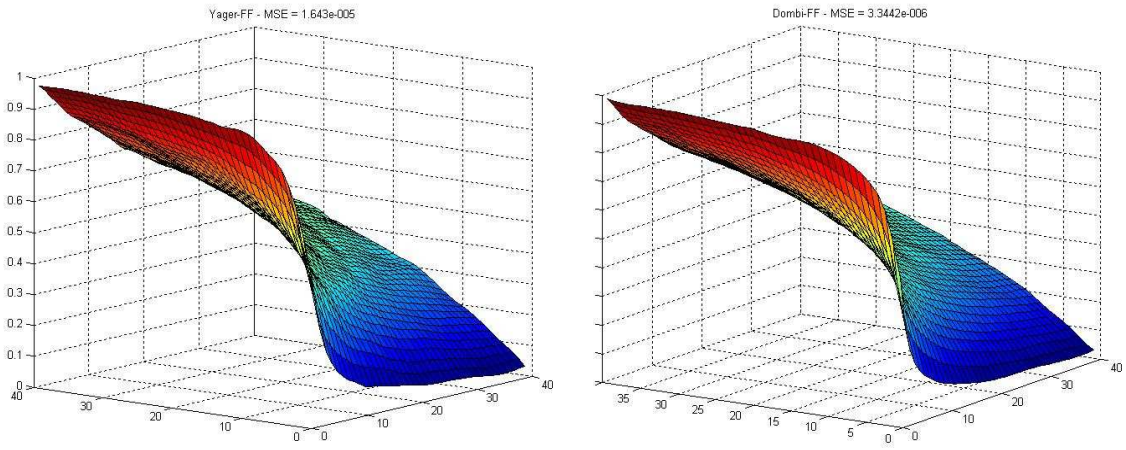


Figure 4.11
Yager FNN (polynomial func.) Dombi FNN (polynomial func.)

4.2.1.5 The *p*H Benchmark Problem

Finally, the performance of Dombi type fuzzy J-K flip-flop neuron based neural network (size 1-3-3-1) was tested on one-input benchmark model the so called *p*H problem [96].

The pattern set consists of a 101 input/output data pairs with very uneven distribution:

Domain: [0.034914, 0.743401]

Range: [0.0001, 1.0000]

No data in (0.19, 0.38); (0.39, 0.59); etc.

In this case the performance of the *tansig* neuron type has been compared with the Dombi one. The parameter of Dombi operators was fixed $\alpha = 2$ and $Q = 0.32$. Figure 4.12 shows that in the domain with only a few data points in the middle area there are outlayer points thus, the curve belonging to *tansig* is deviating from the target and produce overfitting. At the same time, the Dombi one follows very nice by test points interpolating everywhere uniformly well. It is somewhat surprising, that comparing the 300 runs average approximation goodness MSE values (see Table 4.3) belonging to these two cases the results are quite different; the *tansig* based approximation still outperforms the Dombi F^3 network.

TABLE 4.3 THE *P*H PROBLEM

		tansig	Dombi
MSE Train	Minimum	9.67×10^{-10}	5.76×10^{-8}
	Median	1.28×10^{-7}	2.78×10^{-6}
	Mean	1.54×10^{-6}	3.37×10^{-4}
MSE Test	Minimum	3.83×10^{-8}	4.51×10^{-7}
	Median	6.51×10^{-6}	2.17×10^{-5}
	Mean	5.30×10^{-6}	5.79×10^{-4}

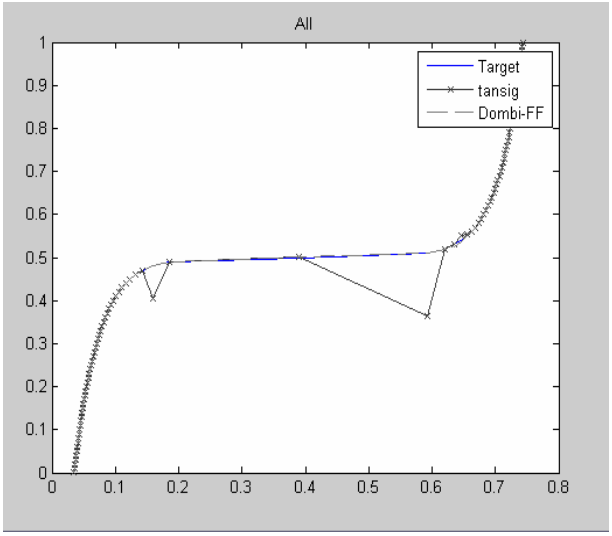


Figure 4.12
Dombi FNN (*p*H problem)

4.3 Summary

This chapter introduces fuzzy flip-flop neurons and novel fuzzy neural networks. In the proposed MLP NN the traditional neurons are implemented with various types of fuzzy flip-flop with quasi sigmoid transfer functions interconnected by weights. An interesting aspect of these F^3s is that they have a certain convergent behavior when their input J is excited repeatedly. This convergent behavior guarantees the learning property of the networks built up in this way. The MLP architecture based on F^3 neurons is given.

The use of fuzzy flip-flop based neural network for performing function approximation based on a combination of test functions has been proposed. The performance of 21 types of FNNs, which depends on the choice of different types of fuzzy flip-flop have been compared. A change of t-norms, Q and fuzzy operations parameter value pairs in the fuzzy flip-flops characteristic equations lead to the modification of the slope of the transfer function, which will affect the learning rate in the implementation of neural networks. Optimizing the parameter values, the FNN structure furthermore, finding an optimal combination of Q and fuzzy operation parameter p values will lead to find the best suitable fuzzy flip-flop type as neuron in the construction of real hardware fuzzy neural network.

The main results of chapter 4 can be summarized as follows:

Statement 2.

I introduced the concept of Fuzzy Flip-Flop based Neural Network (FNN) built up from various types of fuzzy flip-flop neurons with sigmoid transfer functions and I showed that this network family is suitable for learning and function approximation.

2.1 *I proposed the construction of a neuron unit, a combinational sigmoid generator derived from arbitrary*

- *fuzzy J-K flip-flop where* $\overline{Q}$ *is fed back to K, case of K = 1 – Q, and (the old) Q is fixed,*
- *fuzzy J-K flip-flop by applying an inverter in the connection of the input J to K, case of K = 1 – J, the new fuzzy D flip-flop, and (the old) Q is fixed,*
- *fuzzy Choi D flip-flop, and (the old) Q is fixed.*

2.2 *I introduced the concept of the Fuzzy Flip-Flop based Neural Network. The neuron element of FNN may be any fuzzy flip-flop neuron with more or less sigmoid transfer characteristics. Based on previous hardware implementation results FNNs can be stated as easily implementable real hardware neural networks (with fixed structure).*

2.3 *I applied the Levenberg-Marquardt method for investigating the function approximation properties of 21 FNNs built up from fuzzy J-K, D and Choi D flip-flop neurons based on algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade operations by generating a series of transcendental test functions and then by applying them on the pH benchmark problem.*

2.4 *I found that for relatively simple function forms the FNNs generated almost no overfitting problem when applying 2x3 hidden neurons while the standard software implementation (such as tansig based neuron, applying e.g. MATLAB) results in a rather bad overfitting phenomenon.*

I found that for the hardware implementation of a neural network with general purpose (unknown application) FNNs are more suitable than customary (e.g. tansig based) neural networks to avoid overfitting.

2.5 *By extensive simulation experiments I proved that the function approximation goodness of FNNs based on fuzzy D flip-flops is superior to FNNs based on fuzzy Choi D flip-flops.*

Chapter 5

Parameter Optimization in the Fuzzy Neural Networks

A combination of the three main Computational Intelligence (CI) paradigms, like fuzzy systems, neural networks and evolutionary computing has been proposed in this chapter. The goal is to approximate test functions by developing hybrids of the above mentioned three main branches of CI, since no one paradigm is superior to any other. From this point of view the weaknesses of individual components are eliminated while their advantages are increasing. The application of several model identification algorithms for optimizing the fixed value of *Q* and the fuzzy operation parameters in order to achieve as good as possible approximation features of the FNN have been proposed. The Levenberg-Marquardt algorithm is applied for finding the quasi optimal values of *Q* in section 5.1. The effect of the fuzzy neurons number has been studied. It has been shown that the near optimal values of *Q* are quasi independent from the input function complexity and FNN neuron number for a fix fuzzy operation parameter value. The function approximation properties of FNNs built up from fuzzy J-K, D and Choi D flip-flops based on algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade operations are investigated by generating a series of test functions. In section 5.2 the deployment of Bacterial Memetic Algorithm with Modified Operator Execution Order Algorithm (BMAM), developed by László Gál is proposed and applied for FNN variables optimization and training. The most suitable types of F^3 neurons for constructing FNNs have been found. Experimental results and comparisons between different types of FNNs trained with LM and BMAM methods are discussed in section 5.3. This chapter demonstrates that the proposed FNN can be used for learning and approximating simple trigonometric functions; one dimensional real-life application, two dimensional trigonometric functions and a benchmark problem which dates were selected from the input/output test points of a six dimensional non-polynomial function. Experimental results and comparisons between different types of FNN trained with LM and BMAM methods are discussed. In order to train the networks with the mentioned algorithms

Matlab2007, one working thread, under Windows 7 with AMD Athlon II X3 435 processor was used.

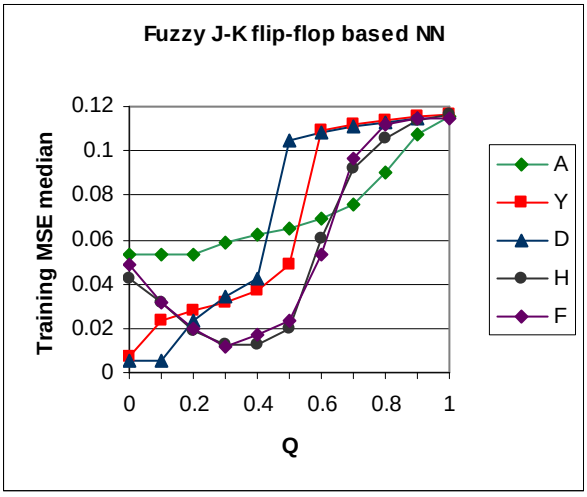
5.1 Levenberg-Marquardt Algorithm Applied for Fuzzy Neural Networks Optimization

5.1.1 Optimization of *Q*

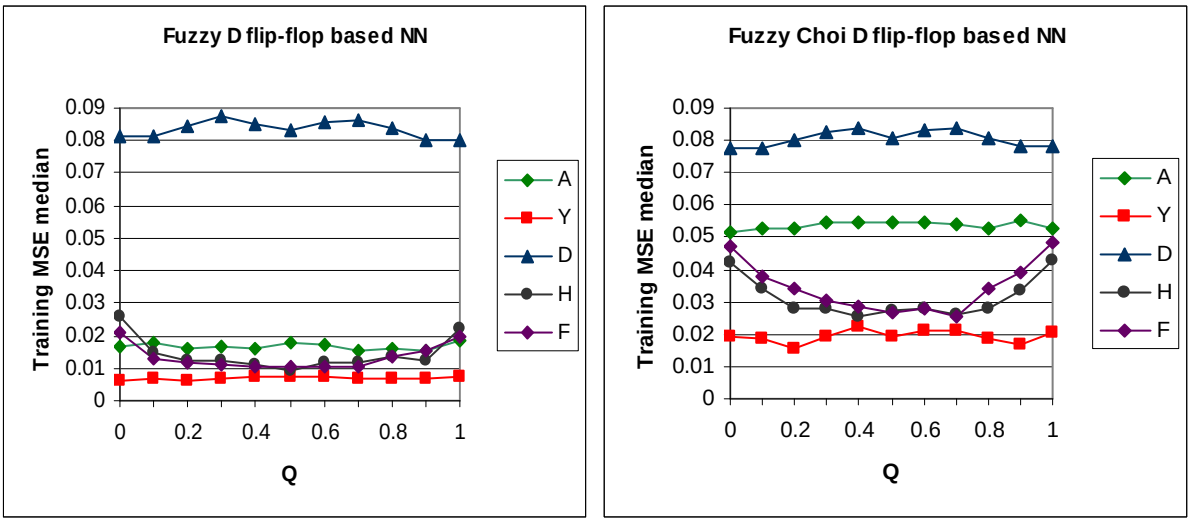
The Levenberg-Marquardt method is applied to find the optimal *Q* values for every combination of fuzzy J-K, D and Choi D flip-flop neuron with algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade fuzzy operations (denoted by the fuzzy operations initials). The norm's parameters have been chosen for suitable values (the parameter of Yager, Dombi, Hamacher, Frank and Dubois-Prade operators were fixed in this case at $w = 2$, $\alpha = 2$, $v = 10$, $s = 100$ and $d = 0.7$ resp.). In these experiments parameters of above mentioned fuzzy operation families were quasi optimized by comparing results by a large number of various parameter values. A 1-8-8-1 size FNN was used to approximate two sine waveforms.

TABLE 5.1 MSE VALUES FOR YAGER FNN

	J-K				D				Choi D			
Q	Train min **x10^{-6}**	Test min **x10^{-6}**	Train median **x 10^{-2}**	Test median **x 10^{-2}**	Train min **x10^{-6}**	Test min **x10^{-6}**	Train median **x 10^{-2}**	Test median **x 10^{-2}**	Train min **x10^{-6}**	Test min **x10^{-6}**	Train median **x 10^{-2}**	Test median **x 10^{-2}**
0.05	0.3449	1.5817	2.2838	2.8808	1.7194	38.604	0.3735	0.5663	17.273	27.334	1.1469	1.2903
0.06	**0.1993**	**0.4866**	**0.6555**	**0.7103**	2.1154	31.142	0.3911	0.4861	34.938	51.431	1.1355	1.2514
0.10	0.5785	1.3701	2.4953	1.4469	2.0411	10.903	0.5131	0.7844	24.298	50.613	0.9507	1.4106
0.17	0.7454	0.5066	1.4509	1.5753	3.4979	19.257	0.4591	0.6759	**11.786**	**16.452**	**0.7770**	**1.1276**
0.18	0.2557	0.6702	0.7525	0.9828	1.9076	12.343	0.3763	0.5936	91.386	39.113	1.1034	1.2310
0.20	0.3529	0.4933	3.1369	3.1731	**1.6111**	**9.4504**	**0.3709**	**0.4824**	99.085	65.979	0.9660	1.1383



a) J-K FNN



b) D FNN c) Choi D FNN

Figure 5.1 Median of the train MSE values

The lowest MSE value indicated the optimal *Q* values which might lead to good learning and approximation properties. The result obtained for the training and test sets give the 300 runs average approximation goodness value. During simulations the median and minimal MSE values have been compared, considering them as the most important indicators of trainability. Table 5.1 summarizes only the significant data intervals belonging to the Yager operation. From the simulation results can be concluded that the quasi optimal *Q* value for J-K FNN is 0.06; for D FNN is 0.20, and finally for the Choi D FNN is 0.17. To specify these values during simulations the parameter optimization steps have been increased.

Figure 5.1 shows the fluctuation of the *Q* values for the proposed types of fuzzy flip-flop neuron. Evaluating the results, low MSE values appears in several domains which fact leads to assess quasi optimal *Q intervals* [74]. The expected ranges of *Q* values are depending on the fuzzy flip-flop neuron and fuzzy operation types. The LM training algorithm is very sensitive to the parameter's initial position in the search space. An inconvenient generated random parameter set leads in a hardly trainable neural network with bad performance, because the LM method is a local searcher and is unable to avoid the local minima.

TABLE 5.2 *Q* OPTIMAL INTERVALS

	F^3 Neuron Type		
Fuzzy Operation	J-K	D	Choi D
Algebraic	0 – 0.2	~0.1; ~0.5; ~0.9	<0.1; ~0.5; >0.8
Łukasiewicz	0.2 – 0.4	0.1 – 0.2 or 0.8 – 0.9	0.4 – 0.5
Yager	0 – 0.3	~0.2; ~0.8	~0.2; ~0.8
Dombi	0.1 – 0.2	<0.1; ~0.5; >0.9	<0.1; ~0.5; >0.9
Hamacher	0.3 – 0.4	0.4 – 0.6	~0.4; ~0.7
Frank	0.2 – 0.4	0.4 – 0.6	0.5 – 0.7
Dubois-Prade	0 – 0.1	0.4 – 0.6	0.4 – 0.6

The optimal *Q* intervals are quasi independent from the input function complexity and FNN neuron number for a fix fuzzy operation parameter value. In particular, the application of LM algorithm is proposed for training various FNNs with different structures in order to approximate a real-life application, two dimensional trigonometric functions and a benchmark problem which dates were selected from the input/output test points of a six dimensional non-polynomial function. The test functions are:

A Simple Real – Life Application: Approximation of a Nickel-Metal Hydride Battery Cell Charging Characteristics

In this particular case, the FNN approximates a Nickel-Metal Hydride (NiMH) Battery Cell charging characteristics [23], a one-input real-life application. The nickel-metal hydride batteries can be repeatedly charged and discharged for about 500 cycles. The charging process duration can be different, from 15 minutes to 20 hours. In this experiment was more than 1 hour. The charge characteristics are affected by current, time and temperature. The test function is a characteristic between the battery capacity input and the cell voltage. The battery type was GP 3.6V, 300mAH, 3x1.2V NiMH, charged for 1.5 hours with 300mA and 25˚C.

B Two – Input Trigonometric Functions

The next two dimensional polynomial input functions were proposed as test functions

$$y_1 = \left(\sin\left(c_1 \cdot x_1\right)^5 \cdot \cos\left(c_2 \cdot x_2\right)^3\right)/2 + 0.5 \tag{5.1}$$

$$y_2 = e^{-\frac{r^2}{100}} \cdot \cos\left(\frac{r}{2}\right); \quad \text{where } r = \sqrt{x_1^{\,2} + x_2^{\,2}}, \tag{5.2}$$

$$x_1,\ x_2 \in [-20, 20]$$

C Six Dimensional Non-Polynomial Function

This widely used six dimensional non-polynomial function as target function originates from paper [5]. Its expression is:

$$y = x_1 + x_2^{0.5} + x_3 x_4 + 2e^{2(x_5 - x_6)} \tag{5.3}$$

where $x_1,\ x_2 \in [1,5],\ x_3 \in [0,4],\ x_4 \in [0,0.6],\ \ x_5 \in [0,1],\ x_6 \in [0,1.2].$

The real-life application (denoted by 1D) is approximated with a 1-2-2-1 FNN size, described by a set of 543 input/output dates selected equidistantly from a set of 2715 test points. A 1-20-20-1 feedforward neural network structure based on F^3 s was proposed to approximate the two input trigonometric functions, (equations 5.1; 5.2; labeled as 2D-1 and 2D-2), represented by 1600 input/output data sets. To approximate the six dimensional benchmark problem I studied a 1-10-5-1 FNN (6D-1) furthermore a 1-10-10-1 FNN (6D-2) sizes given by 200 dates. The number of neurons was chosen after experimenting with different size hidden layers. Figure 5.2 shows the *Q* value intervals for fuzzy J-K and D flip-flop neurons based on Yager norms. The Yager norms parameter values were fixed to *w* = 2. The 2D abbreviation refers to the function described by equation 5.2.

Evaluating the simulation results, low MSE values appears in the same domains which fact leads to assess optimal *Q intervals*, depending less from the type of the input function and FNN neuron number for a fix fuzzy operation parameter value. In case of J-K FNN based on Yager norms the *Q* optimum interval is $[0;\ 0.3]$, in case of D type FNN based on the same norms $Q \in [0.1;\ 0.3]$. An inconvenient generated random parameter set leads in a hardly trainable neural network with bad performance, because the LM method is a local searcher.

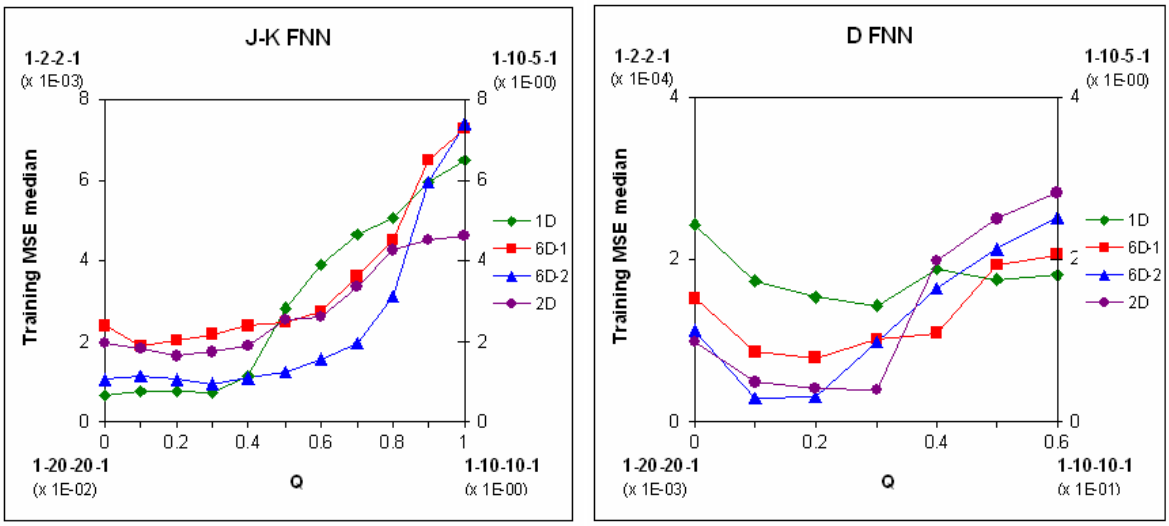


Figure 5.2 J-K FNNs and D FNNs based on Yager norms

5.1.2 Structure Optimization

The function approximation goodness is strongly dependent on the number of fuzzy neurons in the hidden layers. To prove that, a 1-*n*-*n*-1 size FNN based on fuzzy J-K flip-flops containing Dombi norms has been trained, increasing the number of the neurons (*n*) of the hidden layer from 2 to 44. The test function here was a single sine wave represented by a set of 2100 input/output data sets.

During simulations the value of the parameter Q is equal to 0.32 and the parameter of Dombi operators were fixed at $\alpha = 2$, because by repeated simulations it turned out that these values ensured rather good learning abilities. The minimum and median values of the train and test MSE and in addition a useful statistical criterion, the Bayesian Information Criterion (BIC), are shown in Table 5.3 for the proposed network. The BIC balance the accuracy obtained against the model complexity according to the following definition:

$$BIC = m*\ln(MSE_{median}) + k*\ln(m) \tag{5.4}$$

where m is the number of data points and k the number of free parameters to be estimated (i.e. the number of weights and biases of the neurons, the number of the neurons in the hidden layers). During simulations the median MSE values have been considered, considering them as the most important indicators of trainability. The median is a robust estimate of the center of a data sample since outliers have little effect on it.

TABLE 5.3 HIDDEN LAYER NEURON NUMBER EFFECT

n	Train MSE		Test MSE		BIC
	min	median	min	median	
2	$6.39\text{x}10^{-4}$	$1.14\text{x}10^{-1}$	$1.27\text{x}10^{-4}$	$1.14\text{x}10^{-1}$	-4251.9
4	$5.60\text{x}10^{-8}$	$6.72\text{x}10^{-3}$	$6.84\text{x}10^{-8}$	$7.16\text{x}10^{-3}$	-9635.3
6	$8.67\text{x}10^{-8}$	$6.48\text{x}10^{-7}$	$8.31\text{x}10^{-8}$	$7.26\text{x}10^{-7}$	-27843.1
7	$3.39\text{x}10^{-8}$	$2.54\text{x}10^{-7}$	$4.41\text{x}10^{-8}$	$2.54\text{x}10^{-7}$	-29786.6
8	$1.38\text{x}10^{-8}$	$1.62\text{x}10^{-7}$	$1.31\text{x}10^{-8}$	$1.94\text{x}10^{-7}$	-30541.7
9	$4.36\text{x}10^{-9}$	$9.40\text{x}10^{-8}$	$5.71\text{x}10^{-9}$	$1.09\text{x}10^{-7}$	-31174.5
11	$4.08\text{x}10^{-9}$	$4.53\text{x}10^{-8}$	$6.08\text{x}10^{-9}$	$5.88\text{x}10^{-8}$	-32144.1
12	$3.60\text{x}10^{-9}$	$3.81\text{x}10^{-8}$	$4.63\text{x}10^{-9}$	$5.15\text{x}10^{-8}$	-32104.0
15	$1.29\text{x}10^{-9}$	$2.75\text{x}10^{-8}$	$1.63\text{x}10^{-9}$	$3.91\text{x}10^{-8}$	-31948.1
20	$3.86\text{x}10^{-9}$	$2.19\text{x}10^{-8}$	$5.76\text{x}10^{-9}$	$3.32\text{x}10^{-8}$	-30793.2
40	$9.4\text{x}10^{-10}$	$8.82\text{x}10^{-9}$	$3.51\text{x}10^{-9}$	$1.76\text{x}10^{-8}$	-22333.1
44	$8.4\text{x}10^{-10}$	$8.84\text{x}10^{-9}$	$2.65\text{x}10^{-9}$	$1.82\text{x}10^{-8}$	-19590.6

The suggested network is clearly sensitive to the number of neurons in the hidden layers. Too few neurons lead to underfitting while too many neurons cause similarly

undesired overfitting. Smaller neuron numbers in the hidden layer result in worse approximation properties while the increase of the neuron number results in better performance but longer simulation time and a more complex neural network architecture. The overfitting problem emerges when the neuron number in the hidden layers is more then 40 because while the error of the training set is monotonously decreasing the error of the test set is increasing again. In other words the network fits the training points “too well”.

The BIC values calculated with the test and train MSE values are approximately equal. The network with the lower value of BIC is the one to be preferred. As shown in Figure 5.3, varying the number of the neurons in the hidden layer between 7 and 15 the network architecture equilibrates the performance. In this performance test has been show the way how the lowest BIC values indicate the optimal structure complexities.

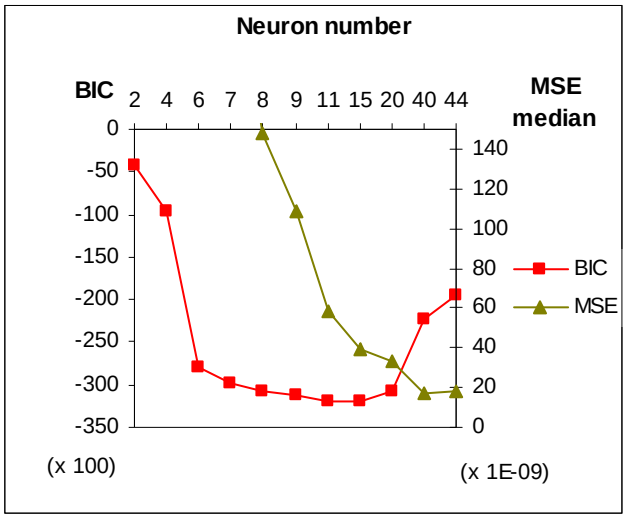


Figure 5.3 Neuron number influences on MSE and BIC values

5.2 Bacterial Memetic Algorithm with Modified Operator Execution Order Algorithm Applied for Fuzzy Neural Networks Parameter Optimization

5.2.1 Optimization of *Q*

In the Bacterial Memetic Algorithm with Modified Operator Execution Order Algorithm the learning of a neural network is formulated as a weight optimization problem, usually using the mean square error as fitness evaluation scheme. It has been shown that evolutionary algorithms work efficient for solving nonlinear and constrained optimization

problems. These methods do not use derivatives of the functions such as the gradient-based training algorithms.

In particular, the application of this evolutionary approach is proposed for training a 1-8-8-1 FNN for finding the optimal *Q* values in the fuzzy J-K, D and Choi D flip-flops based FNNs of algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade operations cases. The combination of two sine waves was used as test function.

The algorithm starts with several alternative solutions to the optimization problem, which are the population individuals. The solutions are coded as binary strings. The basic steps followed by the algorithm embrace the bacterial mutation operation and the LM method.

The FNN weights, biases and *Q* values have been encoded in a bacterium (chromosome) and participated in the bacterial mutation cycle. In this application according to the network size a population with 97+2 parameters was initialized. Therefore a procedure is working on changing the variables testing the model obtained in this way and selecting the best models:

- During simulations 30 generations of 5 individuals with 5 clones were chosen to obtain the best fitting variable values, with the lowest performance. Then the same part or parts of the chromosome is chosen and mutate randomly, except one single clone that remains unchanged during this mutation cycle. The LM method nested into evolutionary algorithm is applied for 5 times for each clone. Several tests have shown that it is enough to run 3 to 5 of LM iterations per mutation to improve the performance of the whole algorithm. The selection of the best clone is made and transfers its parts to the other clones. The part choosing-mutation-LM method-selection-transfer cycle is repeated until all the parts are mutated, improved and tested. The best individual is remaining in the population and all other clones are deleted. This process is repeated until all the individuals have gone through the modified bacterial mutation.
- The *Levenberg-Marquardt method* is applied 7 times for each individual executing several LM cycles during the bacterial mutation after each mutation step. 7-10 LM iterations are required which are done with all the individuals of the population towards reaching the local optimum.
- *Gene transfer operation* is done for 3 times for a partial population.

In this way the local search is done for every global search cycle. The quasi optimal values can be identified at the end of the BMAM training algorithm.

Using the BMAM algorithm, executing several LM cycles after each bacterial mutation step, there is no need to run 300 training cycles to find the lowest MSE value. After only one run the optimal parameter values could be identified. The quasi optimal Q values for various FNNs identified by the BMAM algorithm are listed in Table 5.4. The fuzzy operations parameter values were $w = 2$, $\alpha = 2$, $v = 10$, $s = 100$ and $d = 0.7$ resp.

TABLE 5.4 Q OPTIMAL VALUES

	F^3 Neuron Type		
Fuzzy Operation	J-K	D	Choi D
Algebraic	0.20	0.97	0.13
Łukasiewicz	0.30	0.18	0.48
Yager	0.01	0.22	0.19
Dombi	0.15	0.92	0.06
Hamacher	0.36	0.55	0.36
Frank	0.25	0.51	0.52
Dubois-Prade	0.03	0.49	0.51

The quasi optimal Q intervals obtained with the LM algorithm have been reduced in this case with a single quasi optimal value identified at the end of the training process. In this approach with only one training cycle an acceptable model whose error does not exceed an acceptable level has been obtained.

5.2.2 Simultaneous Optimization of Q and Fuzzy Operation Parameter Values

The BMAM algorithm is proposed for training a 1-8-8-1 size FNN for finding the optimal combination of Q and fuzzy operation parameter p values in the fuzzy J-K, D and Choi D flip-flop neurons of algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade operations cases. The test function and the training attributes are the same as in subsections 5.1.1 and 5.2.1. Table 5.5 shows the optimal combination of Q and fuzzy operations parameter p values found by the BMAM algorithm.

TABLE 5.5 Q AND PARAMETER OPTIMAL VALUES

	F^3 Neuron Type					
Fuzzy Operation	J-K		D		Choi D	
	Q	p	Q	p	Q	p
algebraic	0.20	-	0.97	-	0.13	-
Łukasiewicz	0.30	-	0.18	-	0.48	-
Yager	0.05	1.98	0.26	1.91	0.13	2.02
Dombi	0.24	5.00	0.22	5.73	0.28	5.85
Hamacher	0.29	5.71	0.45	5.45	0.36	2.84
Frank	0.33	6.88	0.52	8.33	0.16	5.46
Dubois-Prade	0.05	0.72	0.53	0.69	0.48	0.71

A change of the t-norms, *Q* and *p* parameter values in the characteristic equations of the fuzzy J-K, D and Choi D flip-flop neurons leads to the modification of the slope of the transfer function, which will affect the learning rate in the implementation of neural networks.

The real-life application, two dimensional trigonometric functions and a benchmark problem as test function have been used to prove that the values of the parameter *Q* are less dependent (with values in a narrow interval) from the input functions and network size for a fixed fuzzy operation parameter value. The quasi optimal *Q* values for different FNNs identified by the BMAM algorithm are listed in Table 5.6. The Yager and Dombi norms parameter values were fixed according to Table 5.5.

TABLE 5.6 QUASI OPTIMAL *Q* VALUES

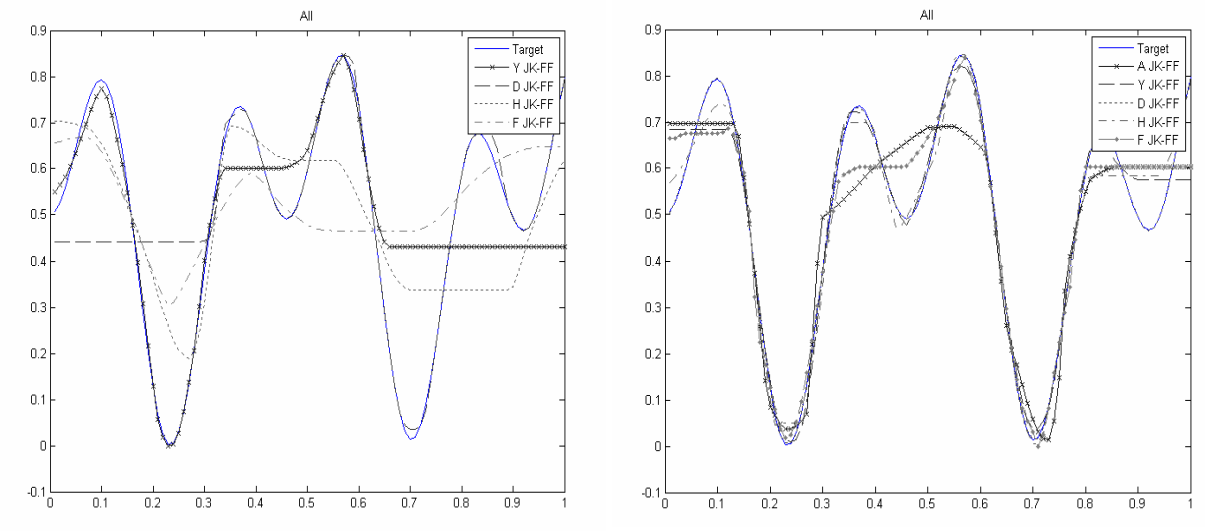
Test Function/ Network size	J-K FNN		D FNN		Choi D FNN	
	Yager type	Dombi type	Yager type	Dombi type	Yager type	Dombi type
1D/1-2-2-1	0.06	0.27	0.26	0.21	0.10	0.25
2D-1/1-20-20-1	0.10	0.24	0.24	0.22	0.18	0.28
2D-2/1-20-20-1	0.13	0.21	0.24	0.22	0.17	0.31
6D-2/1-10-10-1	0.25	0.25	0.28	0.19	0.13	0.27

5.3 Training the Fuzzy Flip-Flop Networks

The function approximation performance of various fuzzy neural networks trained in different ways has been studied. The simulation results are summarized in the next tests. The optimal *Q* and fuzzy operation parameter value pairs have been selected from Table 5.4. First, the network training updates weights, biases and parameters according to Levenberg-Marquardt method. In the second approach the FNN training was made by BMAM algorithm, according to section 5.2 attributes. During the simulations all combinations of fuzzy J-K, D and Choi D type FNNs had been covered with all seven fuzzy operation pairs to approximate the two sine waves as test function (eq. 4.4). The network size is 1-8-8-1.

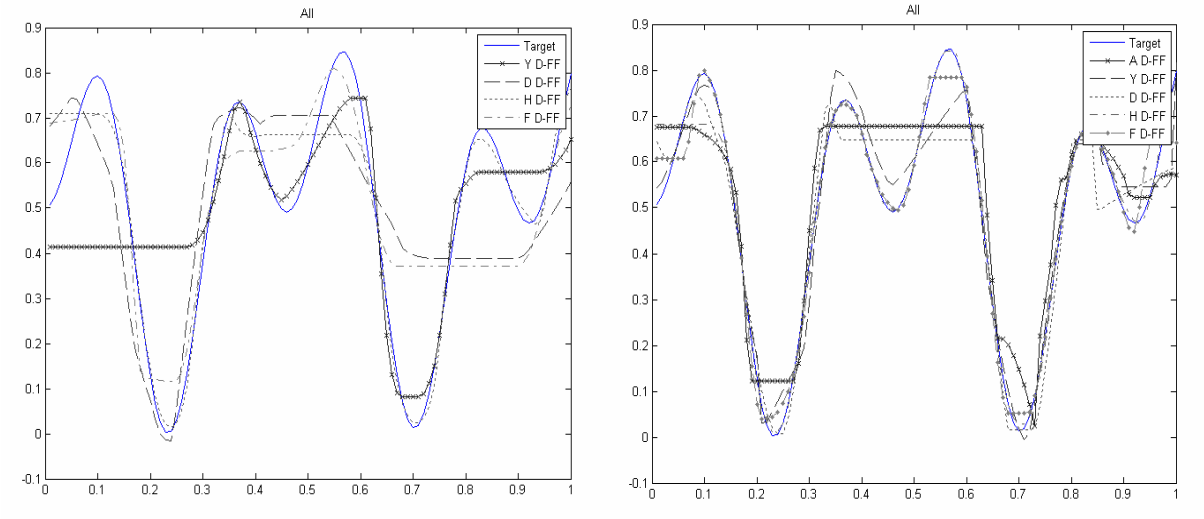
Figure 5.4 presents the graphs of the simulations in case of fuzzy J-K flip-flop neurons, Figures 5.5 and 5.6 compare the behavior of fuzzy D and Choi D flip-flop based NNs trained with LM and BMAM algorithms, respectively. Table 5.7 summarizes the average approximation goodness of 300 and respective 3 runs made by LM and BMAM algorithms, indicating the median MSEs for each of the target *tansig*, furthermore the algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade types of FNNs.

For this network size, with fewer than one hundred weights, the LM method is faster than the BMAM; nevertheless the evolutionary algorithm produces better function approximation capability with lower error. The reason of the training time difference is that the BMAM algorithm is a complex procedure working on changing the parameters, testing the obtained model and finally selecting the best one. Optimizing the population size, clones and generations number, furthermore the LM cycles number in the bacterial mutation and the gene transfer times the training time can be reduced.



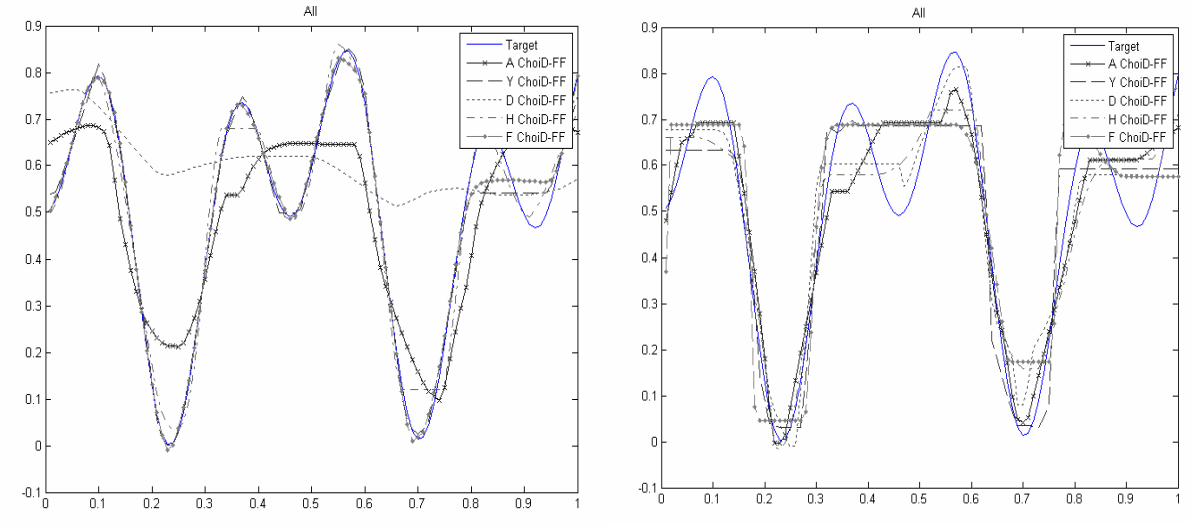
a) LM training b) BMAM training

Figure 5.4 J-K FNNs



a) LM training b) BMAM training

Figure 5.5 D FNNs



a) LM training b) BMAM training

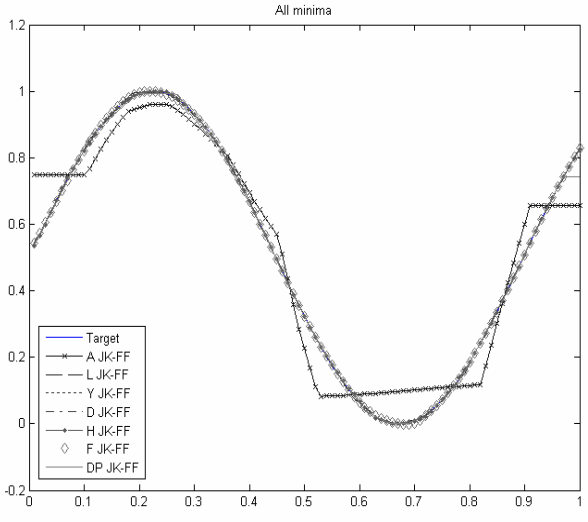
Figure 5.6 Choi D FNNs

TABLE 5.7 MSE MEDIAN VALUES: TWO SINE WAVES

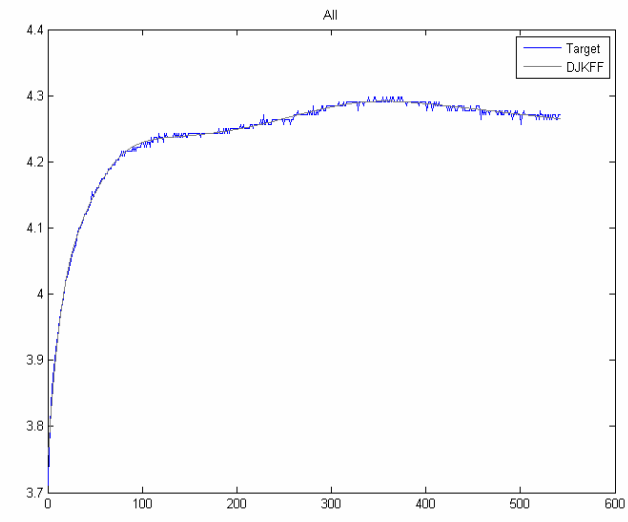
Fuzzy Operation	F^3 Neuron Type					
	J-K		D		Choi D	
	Median MSE					
	LM	BMAM	LM	BMAM	LM	BMAM
tansig	$1.61x10^{-6}$	$1.19x10^{-8}$	$1.61x10^{-6}$	$1.19x10^{-8}$	$1.61x10^{-6}$	$1.19x10^{-8}$
algebraic	$4.34x10^{-2}$	$8.59x10^{-3}$	$2.09x10^{-2}$	$3.59x10^{-3}$	$4.95x10^{-2}$	$1.29x10^{-2}$
Łukasiewicz	$4.93x10^{-3}$	$9.71x10^{-4}$	$2.55x10^{-3}$	$2.55x10^{-4}$	$3.85x10^{-2}$	$4.42x10^{-3}$
Yager	$1.54x10^{-2}$	$1.72x10^{-3}$	$3.18x10^{-3}$	$2.67x10^{-4}$	$4.39x10^{-2}$	$1.22x10^{-2}$
Dombi	$1.23x10^{-3}$	$4.51x10^{-6}$	$1.18x10^{-1}$	$2.98x10^{-2}$	$4.99x10^{-2}$	$2.29x10^{-2}$
Hamacher	$2.58x10^{-2}$	$2.24x10^{-3}$	$4.65x10^{-3}$	$3.95x10^{-4}$	$1.35x10^{-2}$	$3.19x10^{-3}$
Frank	$3.52x10^{-2}$	$3.89x10^{-3}$	$5.09x10^{-3}$	$5.81x10^{-4}$	$4.14x10^{-2}$	$6.12x10^{-3}$
Dubois-Prade	$3.52x10^{-3}$	$2.09x10^{-5}$	$7.62x10^{-3}$	$1.38x10^{-3}$	$3.49x10^{-2}$	$3.85x10^{-3}$

In the next, the application of a recently improved BMAM algorithm is applied for training various FNNs with different structures. This new, complex software is able to train all the FNN parameters which have been encoded in a bacterium (chromosome) with the BMAM, eliminating completely the imprecision accused by training them with the LM algorithm. The simulation results obtained under the same conditions could turn out to be different because the LM method is very sensitive to the initial values of the search space. In particular the algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade operations and three different types of fuzzy flip-flop neuron will be compared from the point of view of the respective fuzzy-neural networks approximation capability. Figures 5.7-5.9 compare the function approximation performance of J-K and two different D type FNNs trained with the BMAM in case of various test functions. The FNNs approximate simple trigonometric functions; one dimensional real-life application, two dimensional trigonometric

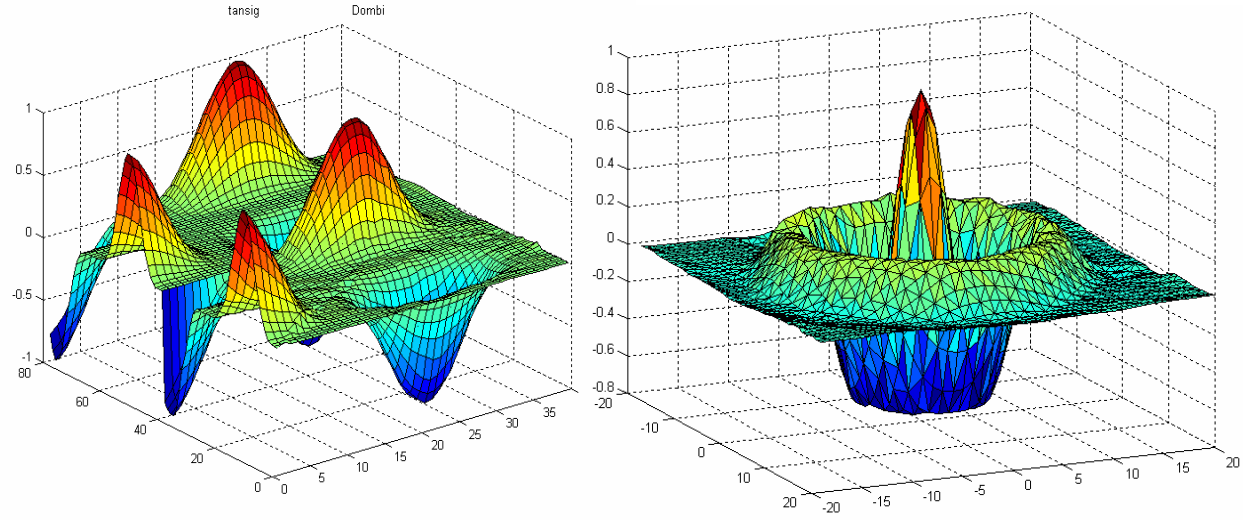
functions and a benchmark problem which dates were selected from the input/output test points of a six dimensional non-polynomial function.



a) all FNNs (sine wave)



b) Dombi FNN (battery cell)



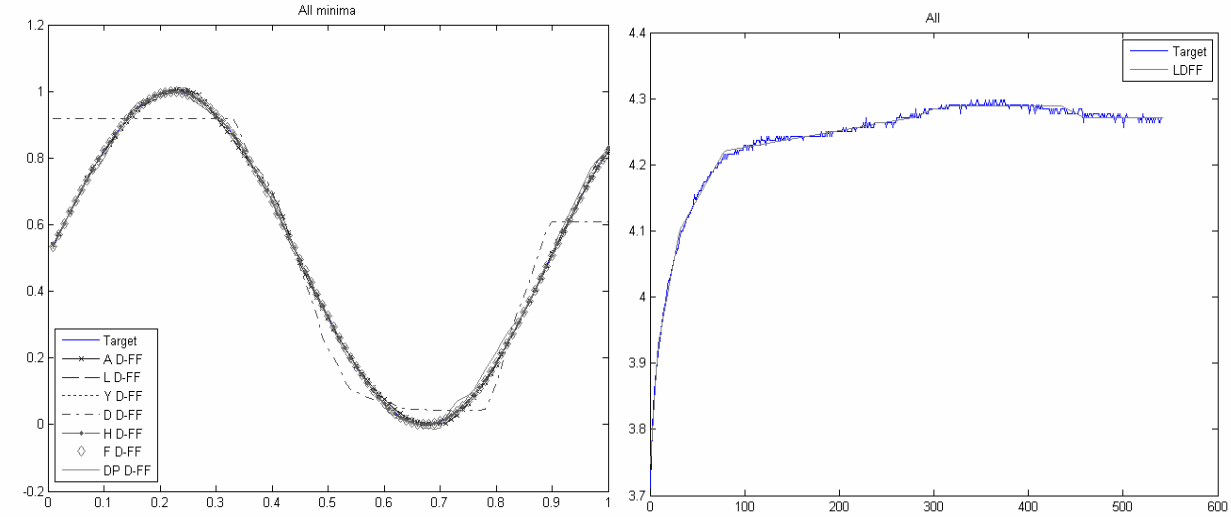
c) tansig and Dombi FNNs (2D trigonometric functions)

Figure 5.7 J-K type FNNs

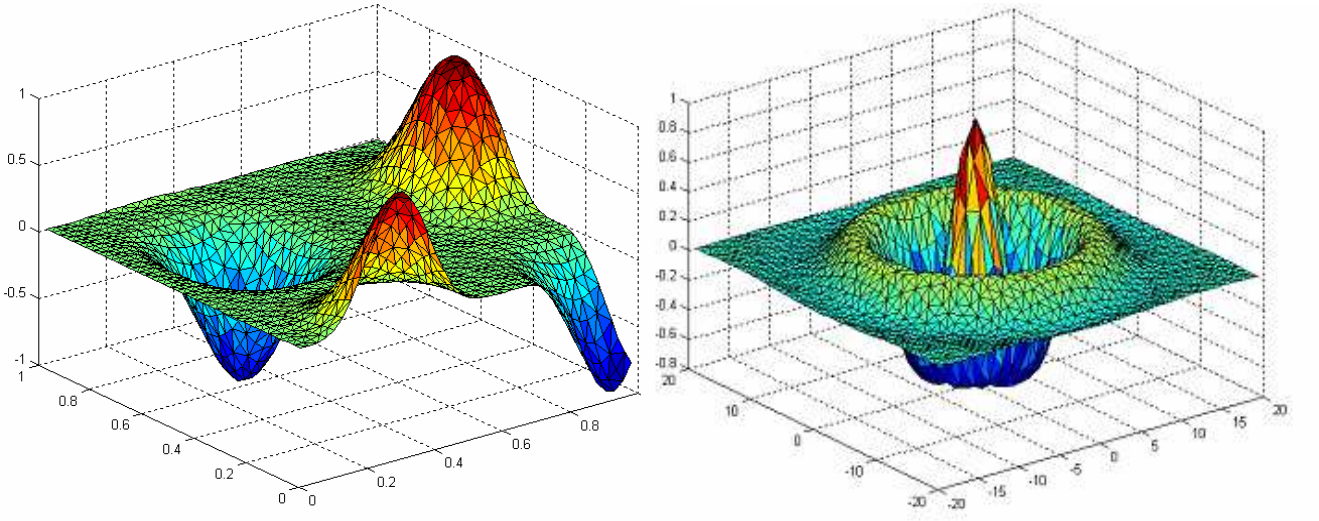
TABLE 5.8 MEDIAN OF THE TRAIN MSE VALUES FOR J-K TYPE FNNS

Fuzzy operation	Sine Wave	1D	2D-1	2D-2	6D-2
tansig	1.88×10^{-9}	1.32×10^{-5}	9.07×10^{-7}	4.26×10^{-7}	1.12×10^{-4}
Algebraic (AJKFF)	9.64×10^{-3}	3.32×10^{-4}	4.32×10^{-2}	5.17×10^{-2}	9.69×10^{-1}
Łukasiewicz (LJKFF)	1.97×10^{-5}	**7.11×10^{-5}**	3.71×10^{-4}	9.46×10^{-4}	5.78×10^{-1}
Yager (YJKFF)	1.14×10^{-4}	1.47×10^{-4}	1.53×10^{-2}	2.49×10^{-2}	5.92×10^{-1}
Dombi (DJKFF)	**1.51×10^{-7}**	**3.52×10^{-5}**	**8.75×10^{-6}**	**1.76×10^{-4}**	**2.98×10^{-1}**
Hamacher (HJKFF)	6.34×10^{-4}	2.59×10^{-4}	2.18×10^{-2}	2.86×10^{-2}	7.43×10^{-1}
Frank (FJKFF)	6.41×10^{-3}	3.89×10^{-4}	2.41×10^{-2}	3.97×10^{-2}	8.83×10^{-1}
Dubois-Prade (DPJKFF)	**7.67×10^{-6}**	**5.91×10^{-5}**	**5.27×10^{-5}**	**4.11×10^{-4}**	5.07×10^{-1}

Tables 5.8-5.10 present the 10 runs average approximation goodness by indicating the median of the train MSE values of the various FNNs. According to the numerical illustrations the function approximation sequence is almost the same in every test function cases. By extensive simulation experiments it is proved that the function approximation goodness of FNNs based on fuzzy J-K flip-flop with Dombi norms are the best ones. The Dubois-Prade**,** Łukasiewicz and Yager type J-K FNNs can be considered as rather good function approximators followed by Hamacher, Frank and algebraic ones. The error of approximation of the battery cell characteristics (1D) obtained by traditional *tansig* function is irrelevant less than that obtained in our simulations. The simulations show that the real-life data-set approximation gives a reasonable solution despite of the large training data sets by a very simple network structure.



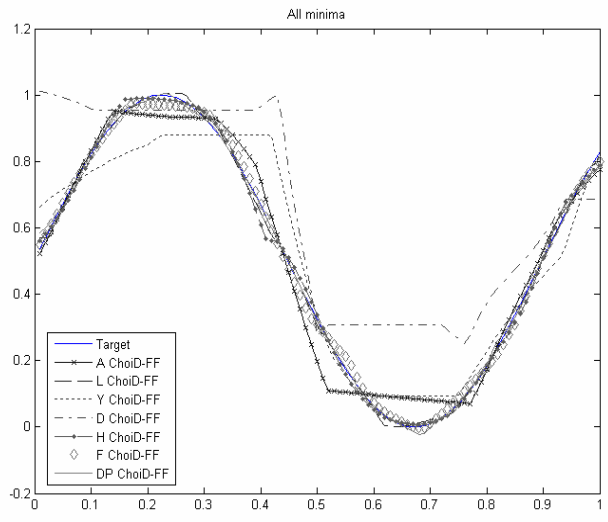
a) all FNNs (sine wave) b) Łukasiewicz FNN (battery cell)



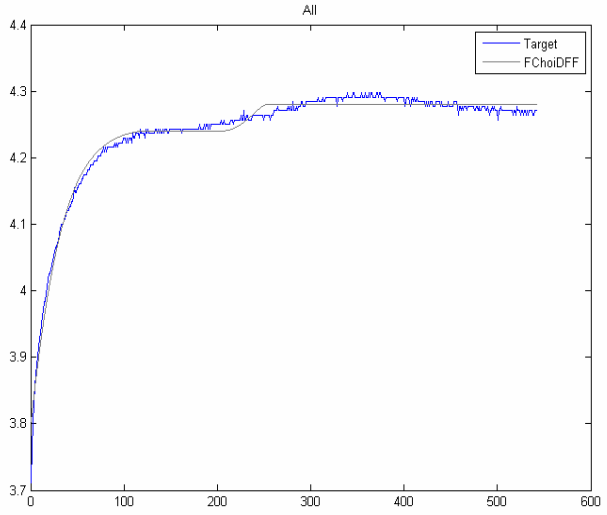
c) Łukasiewicz and Yager FNNs (2D trigonometric functions)
Figure 5.8 D type FNNs

TABLE 5.9 MEDIAN OF THE TRAIN MSE VALUES FOR D TYPE FNNS

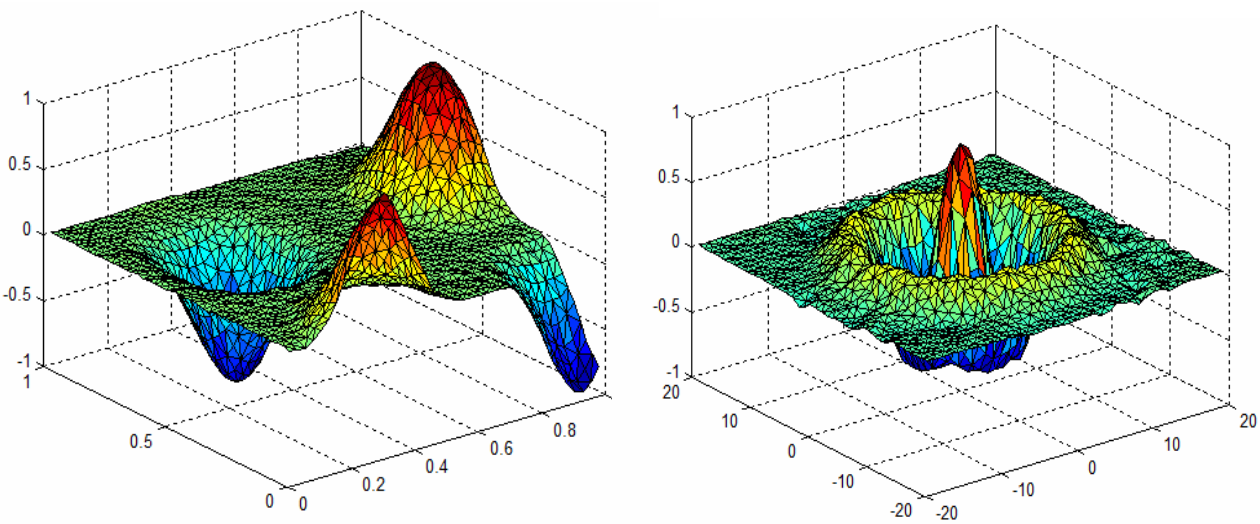
Fuzzy operation	Sine Wave	1D	2D-1	2D-2	6D-2
tansig	1.88×10^{-9}	1.32×10^{-5}	9.07×10^{-7}	4.26×10^{-7}	1.12×10^{-4}
Algebraic (ADFF)	2.11×10^{-3}	5.38×10^{-4}	2.71×10^{-2}	2.94×10^{-2}	6.56×10^{-1}
Łukasiewicz (LDFF)	**4.22×10^{-6}**	**4.95×10^{-5}**	**7.48×10^{-4}**	**1.52×10^{-3}**	**1.05×10^{-1}**
Yager (YDFF)	**5.46×10^{-6}**	1.09×10^{-4}	8.21×10^{-4}	1.48×10^{-2}	1.11×10^{-1}
Dombi (DDFF)	7.62×10^{-3}	6.41×10^{-4}	2.93×10^{-2}	3.23×10^{-2}	1.58×10^{-0}
Hamacher (HDFF)	**7.34×10^{-6}**	1.12×10^{-4}	5.25×10^{-3}	1.86×10^{-2}	2.52×10^{-1}
Frank (FJKFF)	2.27×10^{-5}	2.43×10^{-4}	6.25×10^{-3}	2.25×10^{-2}	3.61×10^{-1}
Dubois-Prade (DPJKFF)	1.29×10^{-3}	4.17×10^{-4}	6.38×10^{-3}	2.53×10^{-2}	3.95×10^{-1}



a) all FNNs (sine wave)



b) Frank FNN (battery cell)



c) Hamacher and Dubois-Prade FNNs (2D trigonometric functions)

Figure 5.9 Choi D type FNNs

TABLE 5.10 MEDIAN OF THE TRAIN MSE VALUES FOR CHOI D TYPE FNNS

Fuzzy operation	Sine Wave	1D	2D-1	2D-2	6D-1
tansig	$1.88x10^{-9}$	$1.32x10^{-5}$	$9.07x10^{-7}$	$4.26x10^{-7}$	$1.12x10^{-4}$
Algebraic (ADFF)	$7.51x10^{-3}$	$8.48x10^{-4}$	$3.87x10^{-2}$	$4.11x10^{-2}$	$1.54x10^{0}$
Łukasiewicz (LDFF)	$5.21x10^{-3}$	$6.89x10^{-4}$	$2.08x10^{-2}$	$2.27x10^{-2}$	$9.48x10^{-1}$
Yager (YDFF)	$7.43x10^{-3}$	$9.37x10^{-4}$	$3.25x10^{-2}$	$3.87x10^{-2}$	$1.11x10^{0}$
Dombi (DDFF)	$1.02x10^{-2}$	$1.73x10^{-3}$	$4.22x10^{-2}$	$4.98x10^{-2}$	$1.62x10^{0}$
Hamacher (HDFF)	$4.11x10^{-3}$	$5.05x10^{-4}$	$1.69x10^{-2}$	$1.93x10^{-2}$	$7.02x10^{-1}$
Frank (FJKFF)	$7.16x10^{-3}$	$7.57x10^{-4}$	$2.48x10^{-2}$	$2.67x10^{-2}$	$1.04x10^{0}$
Dubois-Prade (DPJKFF)	$4.51x10^{-3}$	$6.22x10^{-4}$	$1.84x10^{-2}$	$2.02x10^{-2}$	$8.29x10^{-1}$

By extensive simulation experiments it is proved that the function approximation goodness of FNNs based on fuzzy D flip-flop with Łukasiewicz norms are the best ones. In all experimental results the function approximation by D FNN may be considered sufficiently good in case of the Yager, Hamacher and Frank type fuzzy operations. Here, neither Dombi's nor algebraic operators perform to well. Figures 4.13 and 4.14 compare the behavior of Choi type fuzzy D flip-flop based NNs

My hypothesis is that among these seven Choi type fuzzy D flip-flop based NNs Hamacher neuron is the best and the Dubois-Prade neuron is not much worse, followed by Łukasiewicz and Frank norms. According to the numerical illustrations, the average of 3 runs median of the train MSE values (Tables 5.9, 5.10) sequence is almost the same in every test function cases.

By extensive simulation experiments I proved that the function approximation goodness of FNNs based on fuzzy D flip-flops is superior to FNNs based on fuzzy Choi D flip-flops.

5.4 Summary

This chapter proposes fuzzy flip-flop neurons and novel fuzzy neural networks. In the proposed MLP NNs the traditional neurons are implemented with various types of fuzzy flip-flop with quasi sigmoid transfer functions interconnected by weights. This convergent behavior guarantees the learning property of the networks constructed this way. The MLP architecture constructed of F^3 neurons is given.

In this chapter the LM and BMAM methods were proposed for FNN parameter optimization and network training in order to achieve approximation features of the FNN. Performance comparison of the various FNNs have been given showing how the networks function approximation goodness depends on the *Q* values, on the types of fuzzy flip-flop

neuron, on the hidden layers neuron numbers, on the fuzzy operations and additionally on the training algorithm. Comparison between various types of FNN with respect to the training algorithm cannot be simply used to justify the performance of a training method; it can be used also as a guide to choose suitable methods for particular applications. Numerical results show the superiority of the proposed BMAM algorithm in comparison with the popular LM method.

Optimizing the parameter values, the FNN structure furthermore, finding an optimal combination of *Q* and fuzzy operation parameter *p* values will lead to find the best suitable type of fuzzy flip-flop as neuron in the construction of real hardware fuzzy neural network.

The main results of chapter 5 can be summarized as follows:

Statement 3.

I proposed the application of several model identification algorithms for optimizing the fixed value of Q and the fuzzy operation parameters in order to achieve as good as possible approximation features of the fuzzy neural network.

3.1 *I applied the Levenberg-Marquardt method for finding the quasi optimal parameters Q for every combination of fuzzy J-K, D and Choi D flip-flop based neural network with algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade fuzzy operations.*

I found the following optimal Q intervals:

	F^3 Neuron Type		
Fuzzy Operation	J-K	D	Choi D
Algebraic	0 – 0.2	~0.1; ~0.5; ~0.9	<0.1; ~0.5; >0.8
Łukasiewicz	0.2 – 0.4	0.1 – 0.2 or 0.8 – 0.9	0.4 – 0.5
Yager	0 – 0.3	~0.2; ~0.8	~0.2; ~0.8
Dombi	0.1 – 0.2	<0.1; ~0.5; >0.9	<0.1; ~0.5; >0.9
Hamacher	0.3 – 0.4	0.4 – 0.6	~0.4; ~0.7
Frank	0.2 – 0.4	0.4 – 0.6	0.5 – 0.7
Dubois-Prade	0 – 0.1	0.4 – 0.6	0.4 – 0.6

Experimental results have been provide to demonstrate that the values of Q are less dependent from the input function and network complexity for a fixed fuzzy operation parameter value.

3.2 *I applied the BMAM algorithm for finding the optimal Q values in the fuzzy J-K, D and Choi D flip-flop based neural network with algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade fuzzy operations cases.*

I found the following optimal Q values:

	F^3 Neuron Type		
Fuzzy Operation	J-K	D	Choi D
Algebraic	0.20	0.97	0.13
Łukasiewicz	0.30	0.18	0.48
Yager	0.01	0.22	0.19
Dombi	0.15	0.92	0.06
Hamacher	0.36	0.55	0.36
Frank	0.25	0.51	0.52
Dubois-Prade	0.03	0.49	0.51

3.3 *I applied the BMAM algorithm for simultaneous optimization of Q and fuzzy operation parameter p values in the fuzzy J-K, D and Choi D flip-flop based neural network with algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade fuzzy operations cases.*

I found the following optimal Q and p value pairs:

	F^3 Neuron Type					
Fuzzy Operation	J-K		D		Choi D	
	Q	*p*	*Q*	*p*	*Q*	*p*
algebraic	0.20	-	0.97	-	0.13	-
Łukasiewicz	0.30	-	0.18	-	0.48	-
Yager	0.05	1.98	0.26	1.91	0.13	2.02
Dombi	0.24	5.00	0.22	5.73	0.28	5.85
Hamacher	0.29	5.71	0.45	5.45	0.36	2.84
Frank	0.33	6.88	0.52	8.33	0.16	5.46
Dubois-Prade	0.05	0.72	0.53	0.69	0.48	0.71

3.4 *The very best function approximation results were achieved when applying fuzzy J-K flip-flop neurons based on Dombi norms and fuzzy D flip-flop neurons based on Łukasiewicz norms.*

Thus, I proposed the construction of real hardware fuzzy neural networks build up from Dombi type fuzzy J-K and Łukasiewicz type fuzzy D flip-flop neurons.

The hardware implementation of fuzzy D flip-flop neurons based on Łukasiewicz norms was published jointly with Zavala in [76].

Chapter 6

Conclusions and Future Work

This chapter gives a summary of the results introduced in the dissertation.

In Chapter 3 the concept of several new types of fuzzy flip-flop was defined. Triplets of standard negation, t-norm and t-conorm were used in the different fuzzy flip-flops expressions determining their next state equations, illustrating their behavior by the graphs for typical parameter values. Additionally, this approach was defined for CNF and DNF sets, having advantageous properties, such as interval valued fuzzy flip-flops. The research generalizes the reset and set types later the unified characteristic equation of F^3s.

The corrected fundamental equation for the set type formula in case of Fodor fuzzy flip-flops (F^4) based on a pair of non – associative operations has been given and it has been proved that there exists only one F^4 (set and reset type at the same type). The flip-flops based on various norms have different transfer characteristics slopes. Fixing the value of the present state Q in the characteristic equation often "good" enough sigmoid transfer function characteristics have been obtained. The F^3s were categorized in two groups: with sigmoid and non-sigmoid $J \to Q(t+1)$ characteristics.

The main idea of the study in Chapter 4 is that the fuzzy flip-flops with sigmoid transfer characteristics can be used as neurons in MLPs. The function approximation property of FNN trained with Levenberg-Marquardt method is addressed. In the proposed network the neurons have been substituted with fuzzy J-K and two types of fuzzy D flip-flop based on algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade norms. The effects of the fuzzy operation parameter and fuzzy neuron number have been evaluated.

In Chapter 5 two methods for FNN parameter optimization were given. First, the LM algorithm give expected ranges of optimal values indicating a whole interval; because the FNN function approximation capability is very sensitive to the parameter values. Furthermore, the optimization with BMAM gives accurate optimal values by improving the function approximation capability. In this way a quasi-optimal function approximation result was given as result of a single or only a few training sequences whose error does not exceed

an “acceptable” level. The FNN trained with BMAM algorithm produces a high quality model but requires more computational effort than the network trained with the conventional LM method.

This section presents future work in order to propose new types of fuzzy flip-flop and to improve the function approximation capability of FNNs. As a possible starting point could be the complex fuzzy sequential circuits based on IVFS, MIVFS and midpoint F^3 characteristics investigation. It would be very interesting to examine the behavior of such fuzzy networks to determine the possible divergence behavior and to find the proper, ideal functional interval or point. In the future the research also intends to find the optimal interval or point valued F^3s for practical applications, such as adaptive behavior and learning.

In addition, more developments are desirable for the complex fuzzy sequential circuits based on F^4 behavior investigations and matching logical connectives. Further effort to investigate the behavior of other F^3s based on various ˝famous˝ non – associative operations well known from the literature seems to be necessary. It would be very interesting to find other F^3s with similar advantageous properties.

The proposed FNN function approximation performance could probably be further developed in various ways, particularly in terms of their parameter tuning completed after structure optimization [9], [90], structure and parameter optimization realized at the same time [91], [105] or the application of another neural network types such as the trained Elman network. Furthermore the applications of new hybrid evolutionary methods that combine genetic type algorithms with “classic” local search to perform efficient global search to achieve better results in the function approximation process. It could be a novel version of the BMAM a particular merger of evolutionary and gradient based algorithms combining both global and local search consists of bacterial mutation and, as a second step, the Levenberg-Marquardt (LM) method applied for each clone. This LM step could save in this way some potential solutions that could be lost otherwise after each mutation step. As a third step the LM algorithm could be recalled for a few iterations for each individual of the population towards reaching the local optimum. In this novel algorithm various kinds of fast algorithm with less complexity, like Quasi-Newton algorithm, Conjugate Gradient algorithm, and two Backpropagation training algorithms: Gradient Descent and Gradient Descent with Adaptive Learning Rate and Momentum could be nested in the bacterial mutation.

It could be mentioned as future work to run the tests in another environment based on traditional programming languages such as C, C++ and comparing the results with those obtained in Matlab.

The hardware implementation of min-max, algebraic, drastic and Łukasiewicz norms, furthermore of the fuzzy J-K flip-flops based on standard, algebraic and Łukasiewicz operations was proposed in [11], [16], [89], [93], [97], [123]. These implementations have a very narrow field of use due to their simple expressions and discontinuities. Future research may show that the hardware implementations of more complex network structures which allow parametric or trigonometric operations are advantageous; the early implementations shall be kept relatively simply. Different approaches are needed for the investigation of the floating *K* and *Q* parameter values. The simulations with a wide range of multidimensional input functions, benchmark problems and patterns should reflect the function approximation capability of the proposed FNN. The justified presence of the gene transfer in the BMAM algorithm applied for different FNN training could be also an interesting research topic. Future studies on F^3 based neural networks might include the areas of pattern recognition and computer vision, natural language and text understanding, speech processing, data mining and also general neural computing, machine learning, further fuzzy hardware architectures, software tools and others for possible applications and further investigations.

Summary of statements

Statement 1.

I defined a series of new fuzzy flip-flops and I investigated their properties.

1.1 *I have defined the following new concepts:*

- *interval valued fuzzy J-K flip-flops and I investigated their properties based on standard and algebraic operations,*
- *reset and set type fuzzy J-K flip-flops based on Yager, and Dombi operations,*
- *unified fuzzy J-K flip-flops based on Yager, Dombi, Hamacher, Frank, Dubois-Prade Schweizer-Sklar and Fodor fuzzy operations,*
- *fuzzy D flip-flops and I investigated their properties based on standard, algebraic, drastic, Łukasiewicz, Yager, Dombi, Hamacher, Frank, Dubois-Prade, Schweizer-Sklar and Fodor fuzzy operations,*
- *fuzzy Choi type D flip-flops based on algebraic, drastic, Łukasiewicz, Yager, Dombi, Hamacher, Frank, Dubois-Prade, Schweizer-Sklar and Fodor fuzzy operations.*

I determined the characteristic equations and I investigated the properties of all the above mentioned 32 new types of fuzzy flip-flop.

1.2 *I proved analytically that the "modified Fodor fuzzy J-K flip-flop" satisfies the very special property of the reset and set formula being equivalent. This is the only F^3 type as far when the two expressions (minterm and maxterm) lead to the same flip-flop definition.*

1.3 *I have conducted extensive investigations and I found that the $J \rightarrow Q(t+1)$ transfer characteristics of fuzzy J-K flip-flops with feedback based on Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade norms, further the $D \rightarrow Q(t+1)$ characteristics of (new) fuzzy D flip-flop and fuzzy Choi D flip-flop of Łukasiewicz, Yager, Hamacher, Frank and Dubois-Prade operations show quasi sigmoid curvature for some selected Q values, while all other F^3s investigated (including those to be found in the proceeding literature and novel ones defined by myself) have non-sigmoid behavior.*

Statement 2.

I introduced the concept of Fuzzy Flip-Flop based Neural Network (FNN) built up from various types of fuzzy flip-flop neurons with sigmoid transfer functions and I showed that this network family is suitable for learning and function approximation.

2.1 *I proposed the construction of a neuron unit, a combinational sigmoid generator derived from arbitrary*

- *fuzzy J-K flip-flop where $\overline{Q}$ is fed back to K, case of K = 1 – Q, and (the old) Q is fixed,*
- *fuzzy J-K flip-flop by applying an inverter in the connection of the input J to K, case of K = 1 – J, the new fuzzy D flip-flop, and (the old) Q is fixed,*
- *fuzzy Choi D flip-flop, and (the old) Q is fixed.*

2.2 *I introduced the concept of the Fuzzy Flip-Flop based Neural Network. The neuron element of FNN may be any fuzzy flip-flop neuron with more or less sigmoid transfer characteristics. Based on previous hardware implementation results FNNs can be stated as easily implementable real hardware neural networks (with fixed structure).*

2.3 *I applied the Levenberg-Marquardt method for investigating the function approximation properties of 21 FNNs built up from fuzzy J-K, D and Choi D flip-flop neurons based on algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade operations by generating a series of transcendental test functions and then by applying them on the pH benchmark problem.*

2.4 *I found that for relatively simple function forms the FNNs generated almost no overfitting problem when applying 2x3 hidden neurons while the standard software implementation (such as tansig based neuron, applying e.g. MATLAB) results in a rather bad overfitting phenomenon.*

I found that for the hardware implementation of a neural network with general purpose (unknown application) FNNs are more suitable than customary (e.g. tansig based) neural networks to avoid overfitting.

2.5 *By extensive simulation experiments I proved that the function approximation goodness of FNNs based on fuzzy D flip-flops is superior to FNNs based on fuzzy Choi D flip-flops.*

Statement 3.

I proposed the application of several model identification algorithms for optimizing the fixed value of Q and the fuzzy operation parameters in order to achieve as good as possible approximation features of the fuzzy neural network.

3.1 *I applied the Levenberg-Marquardt method for finding the quasi optimal parameters Q for every combination of fuzzy J-K, D and Choi D flip-flop based neural network with algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade fuzzy operations.*

I found the following optimal Q intervals:

	F^3 Neuron Type		
Fuzzy Operation	J-K	D	Choi D
Algebraic	0 – 0.2	~0.1; ~0.5; ~0.9	<0.1; ~0.5; >0.8
Łukasiewicz	0.2 – 0.4	0.1 – 0.2 or 0.8 – 0.9	0.4 – 0.5
Yager	0 – 0.3	~0.2; ~0.8	~0.2; ~0.8
Dombi	0.1 – 0.2	<0.1; ~0.5; >0.9	<0.1; ~0.5; >0.9
Hamacher	0.3 – 0.4	0.4 – 0.6	~0.4; ~0.7
Frank	0.2 – 0.4	0.4 – 0.6	0.5 – 0.7
Dubois-Prade	0 – 0.1	0.4 – 0.6	0.4 – 0.6

Experimental results have been provide to demonstrate that the values of Q are less dependent from the input function and network complexity for a fixed fuzzy operation parameter value.

3.2 *I applied the BMAM algorithm for finding the optimal Q values in the fuzzy J-K, D and Choi D flip-flop based neural network with algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade fuzzy operations cases.*

I found the following optimal Q values:

	F^3 Neuron Type		
Fuzzy Operation	J-K	D	Choi D
Algebraic	0.20	0.97	0.13
Łukasiewicz	0.30	0.18	0.48
Yager	0.01	0.22	0.19
Dombi	0.15	0.92	0.06
Hamacher	0.36	0.55	0.36
Frank	0.25	0.51	0.52
Dubois-Prade	0.03	0.49	0.51

3.3 *I applied the BMAM algorithm for simultaneous optimization of Q and fuzzy operation parameter p values in the fuzzy J-K, D and Choi D flip-flop based neural network with algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade fuzzy operations cases.*

I found the following optimal Q and p value pairs:

	F^3 Neuron Type					
Fuzzy Operation	J-K		D		Choi D	
	Q	*p*	*Q*	*p*	*Q*	*p*
algebraic	0.20	-	0.97	-	0.13	-
Łukasiewicz	0.30	-	0.18	-	0.48	-
Yager	0.05	1.98	0.26	1.91	0.13	2.02
Dombi	0.24	5.00	0.22	5.73	0.28	5.85
Hamacher	0.29	5.71	0.45	5.45	0.36	2.84
Frank	0.33	6.88	0.52	8.33	0.16	5.46
Dubois-Prade	0.05	0.72	0.53	0.69	0.48	0.71

3.4 *The very best function approximation results were achieved when applying fuzzy J-K flip-flop neurons based on Dombi norms and fuzzy D flip-flop neurons based on Łukasiewicz norms.*

Thus, I proposed the construction of real hardware fuzzy neural networks build up from Dombi type fuzzy J-K and Łukasiewicz type fuzzy D flip-flop neurons.

The hardware implementation of fuzzy D flip-flop neurons based on Łukasiewicz norms was published jointly with Zavala in [76].

List of Figures

List of Tables

References

[1] J. A. Bernard: *Use of rule-based system for process control*, IEEE Control Systems Magazine, 8(5), 1988, pp. 3-13.

[2] C. M. Bishop: *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, UK, 1995.

[3] E. K. Blum and L. K. Li.: *Approximation theory and feedfordward neural networks* Neural Networks, 4(4), 1991, pp. 511-515.

[4] L. B. Booker, D. E. Goldberg and J. H. Holland: *Classifier systems and genetic algorithms*, Artificial Intelligence, 40, 1989, pp.:235-282.

[5] J. Botzheim, B. Hámori, L. T. Kóczy and A. E. Ruano: *Bacterial algorithm applied for fuzzy rule extraction*, in Proc. Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Annecy, France, 2002, pp.1021-1026.

[6] J. Botzheim, M. Drobics and L. T. Kóczy: *Feature selection using bacterial optimization* in Proc. 10th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Perugia, 2004, pp.797-804.

[7] J. Botzheim, C. Cabrita, L. T. Kóczy and A. E. Ruano: *Estimating Fuzzy Membership Functions Parameters by the Levenberg-Marquardt Algorithm*, FUZZ-IEEE 2004, Budapest, Hungary, 2004, pp. 1667-1672.

[8] J. Botzheim, C. Cabrita, L. T. Kóczy and A. E. Ruano: *Fuzzy rule extraction by bacterial memetic algorithms* in Proc. 11th World Congress of International Fuzzy Systems Association, IFSA 2005, Beijing, China, 2005, pp. 1563-1568.

[9] J. Botzheim, C. Cabrita, L. T. Kóczy and A. E. Ruano: *Genetic and bacterial programming for B-spline neural networks design*, Journal of Advanced Computational Intelligence and Intelligent Informatics, 11(2), 2007, pp. 220-231.

[10] Y. J. Cao, N. Ireson, L. Bull and R. Miles: *Design of a Traffic Junction Controller Using Classifier Systems and Fuzzy Logic*, Springer, 1625, 1999, pp. 342-353.

[11] B. Choi and K. Tipnis: *New Components for Building Fuzzy Logic Circuits*, Proc. of the 4th Int. Conf. on Fuzzy Systems and Knowledge Discovery, 2, 2007, pp. 586-590.

[12] I. Ciuca and J. A. Ware: *Layered Neural Networks as Universal Approximators*, Fuzzy Days Dortmund, Germany, 1997, pp. 411-415.

[13] G. Cybenko: *Approximation by superposition of sigmoidal functions*, Math. Contr. Signals. Syst. 2, 1989, pp. 303-314.

[14] E. Czogała and J. Leski: *Fuzzy and Neuro-fuzzy Intelligent Systems*, Physica-Verlag, Springer Verlag, 2000.

[15] L. Davis: *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[16] J. Diamond, W. Pedrycz and D. McLeod: *Fuzzy JK Flip-Flops as Computational Structures: Design and Implementation*, IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, vol.41, no.3, 1994.

[17] J. Dombi: *A general class of fuzzy operators, the De Morgan class of fuzzy operators and fuzziness measures induced by fuzzy operators*, Fuzzy Sets and Systems, 8, 1982, pp. 149-163.

[18] D. Dubois and H. Prade: *Fuzzy Sets and Systems; Theory and Applications*, Academic Press, New York, 1980.

[19] D. Dubois, M. Grabisch and H. Prade: *Gradual rules and the approximation of functions*, in Proc. of the 2nd International Fuzzy Systems Association Congress Iizuka, Japan, 1992, pp. 629-632.

[20] D. Dubois and H. Prade: *Fundamentals of fuzzy sets*, The handbooks of fuzzy sets series, Kluwer Academic Publishers, 2000.

[21] A. E. Eiben: *Towards Automated Parameter Calibration of Evolutionary Algorithm*, Tutorial session in Proc. of IEEE Congress on Evolutionary Computation, Hong Kong 2008.

[22] A. P. Engelbrecht: *Computational Intelligence: An Introduction*, England, Wiley, 2007.

[23] R. Fan: *NiMH Battery Charger Reference Design*, Designer Refernce Manual, 2003.

[24] J. C. Fodor and L. T. Kóczy: *Some remarks on fuzzy flip-flops*, In: L. T. Kóczy, K. Hirota, eds., Proc. of the Joint Hungarian-Japanese Symposium on Fuzzy Systems and Applications (Technical University, Budapest, 1991), pp. 60-63.

[25] J. C. Fodor and M. Roubens: *Fuzzy Preference Modeling and Multicriteria Decision Support*, Kluwer Academic Publishers, Dordrecht, 1994.

[26] M. J. Frank: *On the simultaneous associativity of f(x,y) and x+y+f(x,y)*, Aequationes Mathematicae, 19 (2-3), 1979, pp.194-226.

[27] K. I. Funahashi: *On the approximate realization of continuous mapping by neural networks*, Neural Networks, 2, 1989, pp. 183-192.

[28] L. Gál, J. Botzheim and L. T. Kóczy: *Improvements to the Bacterial Memetic Algorithm used for Fuzzy Rule Base Extraction*, in Proc. of Computational Intelligence for Measurement Systems and Applications, CIMSA, Istanbul, Turkey, 2008, pp. 38-43.

[29] L. Gál and L. T. Kóczy: *Advanced bacterial memetic algorithms*, Acta Technica Jaurinensis Series Intelligentia Computatorica, 1(3), 2008, pp. 481-498.

[30] F. Girosi and T. Poggio: *Representation properties of network: Kolmogorov's theorem is irrelevant*, Neural Computation 1, 1989, pp. 456-469.

[31] J. A. Goguen: *L-fuzzy sets*, Journal of Mathematical Analysis and Applications, 18, 1967, pp. 145-174.

[32] D. E. Goldberg: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison – Wesley Publishing Company, Inc. Reading, Massachusetts, 1989.

[33] I. R. Goodman: *A decision aid for nodes in Command & Control Systems based on cognitive probability logic*, in Proc. of Command& Control Research and Technology Symposium, US Naval War College, Newport, RI, 1999, pp. 898-941.

[34] M. T. Hagan and M. Menhaj: *Training feed-forward networks with the Marquardt algorithm*, IEEE Transactions on Neural Networks, Vol. 5, No. 6, 1994, pp. 989-993.

[35] M .T. Hagan, H. B. Demuth and M. H. Beale: *Neural Network Design*, Boston, MA: PWS Publishing, 1996.

[36] M. T. Hagan and H. B. Demuth: *Neural Networks for Control*, Invited Tutorial, American Control Conference, San Diego, 1999, pp. 1642-1656.

[37] H. Hamacher: *Über logische Vernupfungen unscharfer Aussagen und deren Zugehörige Bewertungsfunctionen*, In R. Trappl, G. J. Klir, and L. Ricciardi editors, Progress in Cybernetics and Systems Research, 3. Hempisphere, Washington D. C. 1978, pp. 276-288.

[38] S. D. Handoko, C. K. Kwoh, Y. S. Ong and M. H. Lim: *A Study on Constrained MA Using GA and SQP: Analytical vs. Finite-Difference Gradients*, in Proc. of IEEE Congress on Evolutionary Computation, Hong Kong, 2008, pp.4032-4039.

[39] S. Haykin: *Neural networks and learning machines*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2009.

[40] R. Hecht-Nielsen: *Theory of the backpropagation neural network*, in Proc. of the Neural Networks, IJCNN, International Joint Conference on Neural Networks, 1989, pp. 593-605.

[41] K. Hirota and K. Ozawa: *Concept of fuzzy flip-flop*, Preprints of 2nd IFSA Congress, Tokyo, 1987, pp. 556-559.

[42] K. Hirota and K. Ozawa: *Fuzzy flip-flop as a basis of fuzzy memory modules*, In: M. M. Gupta et al., eds., Fuzzy Computing. Theory, Hardware and Applications, North Holland, Amsterdam, 1988, pp. 173-183.

[43] K. Hirota and W. Pedrycz: *Neurocomputations with fuzzy flip-flops*, Proc. of International Joint Conference on Neural Networks, 1993, pp.1867-1870

[44] J. H. Holland: *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.

[45] J. H. Holland: *Adaptation in Nature and Artificial Systems: An Introductory Analysis with Application to Biology, Control, and Artificial Intelligence*, The MIT Press, Cambridge, MA 1992.

[46] K. Hornik, M. Stinchcombe and H. White: *Multilayer Feedforward Networks are Universal Approximators*, Neural Networks, 2, 1989, pp. 359-366.

[47] K. Hornik: *Approximation capabilities of multilayer feedfordward networks*, Neural Networks, 4, 1991, pp. 251-257.

[48] I. Hossain, I. Liu and R. Lee: *A study of multilingual speech feature: perspective scalogram based on wavelet analysis*, in Proc. of IEEE Int. Conf. Sys., Man and Cybernetics Vol. II, 1999, Tokyo, Japan, pp.178-183.

[49] C. Isik: *Identification and fuzzy rule based control of a mobile robot motion*, in Proc. of the IEEE Int. Symp. Intelligent Control Philadelphia, 1987.

[50] Y. Ito: *Approximation Capability of Layered Neural Networks with Sigmoid Units on Two Layers*, Neural Computation, 6(6), 1994, pp. 1233-1243.

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

[52] N. K. Kasabov: *Foundations of Neural Networks*, Fuzzy Systems, and Knowledge Engineering, The MIT Press. Cambridge, MA, 1996.

[53] N. K. Kasabov and R. Kozma: *Neuro-Fuzzy Techniques for Intelligent Information Systems*, Physica Verlag, Wurzberg/Springer, Berlin, 1999.

[54] E. P. Klement, L. T. Kóczy and B. Moser: *Are fuzzy systems universal approximators?* International Journal of General Systems 28 (2–3), 1999, pp. 259 – 282.

[55] E. P. Klement, R. Mesiar and E. Pap: *Triangular Norms*, Series: Trends in Logic, 8, 2000.

[56] G. J. Klir and T. A. Folger: *Fuzzy sets, uncertainty, and information*, Prentice Hall, Upper Saddle River, NJ, 1988, pp. 37-637.

[57] G. J. Klir and B. Yuan: *Fuzzy sets and fuzzy logic: Theory and applications*, Prentice Hall, Upper Saddle River, NJ, 1995, pp. 50-88.

[58] L. T. Kóczy and R. Lovassy: *Fuzzy Flip-Flops and Neural Nets?*, in Proc. of IEEE International Conference on Fuzzy Systems - Imperial College, FUZZ 2007, London, England, 2007, pp. 605-610.

[59] L. T. Kóczy and R. Lovassy: *Fuzzy Flip-Flops Revisited*, in Proc. of International Fuzzy Systems Association European Society for Fuzzy Logic and Technology World Congress, IFSA2007, Cancun, Mexico, 2007, pp. 643-652.

[60] Z. Kohavi: *Switching and Finite Automata Theory*, McGraw-Hill Book Company, 1978.

[61] B. Kosko: *Fuzzy Systems are Universal Approximators*, in Proc. of the IEEE International Conference on Fuzzy Systems, San Diego, CA, 1992, pp.1153-1162.

[62] G. K. Kostopoulos: *Digital Engineering*, Wiley-Interscience Publication, 1975.

[63] V. Kurkova: *Kolmogorov's theorem and multilayer neural networks* Neural Networks, 5, 1992, pp. 501-506.

[64] R. S. T. Lee: *Fuzzy-Neuro Approach to Agent Applications*, Springer, 2006.

[65] T. Leephakpreeda: *Car-parking guidance with fuzzy knowledge-based decision making*, Building and Environment, 42(2), 2007, pp.803-809.

[66] K. Levenberg: *A method for the solution of certain non-linear problems in least squares*, Quart Appl. Math. 2(2), 1944, pp. 164-168.

[67] H. Li and M. Gupta: *Fuzzy logic and intelligent systems*, Series: International Series in Intelligent Technologies, 3, 1995.

[68] C. H. Ling: *Representation of associative functions*, Publ. Math. Debrecen 12, 1965, pp. 189-212.

[69] R. Lovassy and L. T. Kóczy: *Fuzzy J-K Flip-Flops Based on Various "Classic" and Non-associative Norms*, Buletinul Ştiinţific al Universitaţii "Politehnica" 51(65), No. 1. Timişoara, Romania, 2006, pp. 11-18.

[70] R. Lovassy and L. T. Kóczy: *Non-associative Fuzzy Flip-Flop with Dual Set-Reset Feature*, in Proc. of 4th Serbian-Hungarian Joint Symposium on Intelligent Systems, , SISY 2006, Subotica, Serbia, 2006, pp. 289-299.

[71] R. Lovassy and L. T. Kóczy: *S-Shaped Fuzzy Flip-Flops*, in Proc. of 8th International Symposium of Hungarian Researchers on Computational Intelligence, CINTI 2007, Budapest, Hungary, 2007, pp. 383-391.

[72] R. Lovassy, L. T. Kóczy and L. Gál: *Analyzing Fuzzy Flip-Flops Based on Various Fuzzy Operations*, Acta Technica Jaurinensis Series Intelligentia Computatorica, 1(3), Győr, Hungary, 2008, pp. 447-465.

[73] R. Lovassy, L. T. Kóczy and L. Gál: *Multilayer Perceptron Implemented by Fuzzy Flip-Flops*, in Proc. of IEEE World Congress on Computational Intelligence, WCCI 2008, Hong Kong, 2008, pp. 1683-1688.

[74] R. Lovassy, L. T. Kóczy and L. Gál: *Parameter Optimization in Fuzzy Flip-Flop Based Neural Networks*, in Proc. of International Symposium on Innovations in Intelligent Systems and Applications, INISTA 2009, Trabzon, Turkey, 2009, pp. 205-209.

[75] R. Lovassy, L. T. Kóczy and L. Gál: *Function Approximation Capability of a Novel Fuzzy Flip-Flop Based Neural Network*, in Proc. of International Joint Conference on Neural Networks, IJCNN 2009, Atlanta, USA, 2009, pp. 1900-1907.

[76] R. Lovassy, A. H. Zavala, L. Gál, O. C. Nieto, L. T. Kóczy and I. Batyrshin: *Hardware Implementation of Fuzzy D Flip-Flop Neurons Based on Łukasiewicz Norms*, in Proc. of the 9th WSEAS Int. Conference on Applied Computer and Applied Computational Science, Penang, Malaysia, 2010, pp.196-201.

[77] G. F. Luger and W. A. Stubbletfield: *Artificial intelligence: structures and strategies for complex problem solving*, Addison-Wesley, 1993.

[78] D. Marquardt: *An Algorithm for Least-Squares Estimation of Nonlinear Parameters*, SIAM J. Appl. Math. 11, 1963, pp. 431-441.

[79] K. Menger: *Statistical Metrics*, in Proc. of the National Academy of Sciences of the USA 28, 1942, pp. 535-537.

[80] W. S. McCulloch and W. Pitts: *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics 5, 1943, pp.115-133.

[81] B. B. Meunier, V. Kreinovich and H. T. Nguyen: *Non-associative Operations*, in Proc. of the Second International Conference on Intelligent Technologies InTech'2001, Bankok, Thailand, 2001, pp.39-46.

[82] P. Moscato: *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms*, Technical Report Caltech Concurrent Computational Program, Report 826, California Institute of Technology, Pasadena, USA, 1989.

[83] T. Munakata: *Fundamentals of the new artificial intelligence: beyond traditional paradigms* Springer 1998.

[84] N. E. Nawa and T. Furuhashi: *Fuzzy System Parameters Discovery by Bacterial Evolutionary Algorithm*, IEEE Trans. on Fuzzy Systems, 7(5), 1999, pp.608-616.

[85] N. E. Nawa, T. Furuhashi, T. Hashiyama and Y. Uchikawa: *A Study on the Discovery of Relevant Fuzzy Rules Using Pseudo-Bacterial Genetic Algorithm*, IEEE Transactions on Industrial Electronics, 46(6), 1999, pp. 1080-1089.

[86] F. Neri, J. Toivanen, G. L. Cascella and Y. S. Ong: *An adaptive multimeme algorithm for designing HIV multidrug therapies*, IEEE/ACM Trans. Comput. Biol. Bioinformatics, 4, 2007, pp. 264-278.

[87] Y. S. Ong, P. B. Nair and K. Y. Lum: *Min-max surrogate-assisted evolutionary algorithm for robust aerodynamic design*, IEEE Trans., Evol., Comput., 10, 2006, pp. 392-404.

[88] K. Ozawa, K. Hirota, L. T. Kóczy and K. Ohmori: *Algebraic fuzzy flip-flop circuits*, Fuzzy Sets and Systems 39(2), 1991, pp. 215-226.

[89] K. Ozawa, K. Hirota and L. T. Kóczy: *Fuzzy flip-flop*, In: M. J. Patyra, D. M. Mlynek, eds., Fuzzy Logic. Implementation and Applications, Wiley, Chichester, 1996, pp. 197-236.

[90] S. E. Papadakis and J. B. Theocharis: *A GA-base fuzzy modeling approach for generating TSK models*, Fuzzy Sets and Systems, 2,2002, pp. 121-152.

[91] K.-J. Park, S.-K. Oh and H.-K. Kim: *Optimization of fuzzy set-fuzzy systems based on IG by means of GAs with successive tuning method*, Journal of Electr. Eng. Teechnol. 3(1), 2008, pp. 101-107.

[92] D. W. Patterson: *Artificial neural networks: theory and applications*, Prentice Hall, 1996.

[93] W. Pedrycz: *Fuzzy sets engineering*, CRC Press, 1995.

[94] F. Rosenblatt: *Principles of Neurodynamics*, Washington D.C., Spartan Press, 1961.

[95] T. J. Ross: *Fuzzy Logic with Engineering Applications*, Second Edition John Wiley & Sons, Ltd, 2004.

[96] A E Ruano, C Cabrita, J V Oliveira, D Tikk and L T Kóczy: *Supervised training algorithms for B-spline neural networks and fuzzy systems*, 9th IFSA World Congress and 20th NAFIPS International Conference (IFSA NAFIPS) Vancouver, 2001, pp. 2830-2835.

[97] I. J. Rudas, I. Z. Batyrshin, A. H. Zavala, O. C. Nieto, L. Horváth and L. V. Vargas: *Generators of Fuzzy Operations for Hardware Implementation of Fuzzy Systems*, Proc. of MICAI 2008, Advances in Artificial Intelligence, 7th Mexican International Conference on Artificial Intelligence, Mexico, October 27-31, 2008, pp.710-719.

[98] D. E. Rumelhart, G. E. Hinton and R. J. Williams: *Parallel Distributed Processing*, eds. J. L. McClelland, D. E. Rumelhart and the PDP Research Group, 1(2), MIT, Cambridge, MA. 1986, pp. 318-362.

[99] L. Rutkowski: *Flexible neuro-fuzzy systems*, Kluwer 2004.

[100] L. Rutkowski: *Computational Intelligence*, Springer 2008.

[101] R. Sambuc: *Fonctions Φ-floues. Application l'aide au diagnostic en pathologie thyroidienne*, Ph. D. Thesis Univ. Marseille, France, 1975.

[102] B. Schweizer and A. Sklar: *Associative functions and statistical triangle inequalities*, Publ. Math. Debrecen 8, 1961, pp. 169-186.

[103] B. Schweizer and A. Sklar: *Associative functions and abstract semigroups*, Publ. Math. Debrecen 10, 1963, pp. 69-81.

[104] A. J. Shepherd: *Second-Order Methods for* Neural Networks*: Fast and Reliable Training Methods for Multi-Layer Perceptrons*, Springer, 1997.

[105] H. Surmann and M. Maniadakis: *Learning feed-forward and recurrent fuzzy systems*: A genetic approach, Journal of Syst. Archit. 47(7), 2001, pp. 649-662.

[106] J. Tang, M. H. Lim and Y. S. *Ong: Diversity-Adaptive Parallel Memetic Algorithm for Solving Large Scale Combinational Optimization Problems*, Soft Computing: A Fusion of Foundations, Methodologies, and Applications, 11(9), 2007, pp. 873-888.

[107] A. Torn and A. Zilinskas: *Global Optimization*, Springer-Verlag, 1989.

[108] I. B. Türkşen and D. D. W. Yao: *Representations of connectives in fuzzy reasoning: the view through normal form*, IEEE Trans. On Systems, Man, and Cybernetics, SMC-14, 1984, pp.146-151.

[109] I. B. Türkşen: *Interval valued fuzzy sets based on normal forms*, Fuzzy Sets and Systems 20, 1986, pp.191-210.

[110] I. B. Türkşen, A. Esper, K. Patel, S.A. Starks and V. Kreinovich: *Selecting a Fuzzy Logic Operation from the DNF-CNF Interval: How Practical Are the Resulting Operations?*, in Proc. of the 21st International Conference of the North American Fuzzy Information Processing Society NAFIPS´2002, New Orleans, 2002, pp. 28-33.

[111] L. X. Wang: *Fuzzy systems are universal approximators*, in Proc. of the IEEE International Conference on Fuzzy Systems, San Diego, CA, 1992, pp. 1163-1169.

[112] L. X. Wang and J. M. Mendel: *Fuzzy basis functions, universal approximations and orthogonal least-squares learning*, IEEE Transactions on Neural Nets, 3, 1992, pp. 807-814.

[113] B. Widrow and F. W. Smith: *Pattern-recognition control systems*, Proc. Computer and Information Science Symposium, Spartan Books, Washington DC, USA, 1964, pp. 163-172.

[114] B. Widrow and M. A. Lehr: *Perceptrons, Adalines, and Backpropagation*, The Handbook of Brain Theory and Neural Network, A Bradford Book, 1995, pp. 719-724.

[115] L. Wiskott and C. Malsburg: *A neural system for the recognition of partially occluded objects in cluttered scenes*, Int. J. Pattern Recognition and Artificial Intelligence, 7(4), 1993, pp. 935-948.

[116] L. Wiskott and C. Malsburg: *Recognizing faces by dynamic link matching*, in Proc. Int. Conf. Artificial Neural Networks ICANN, 1995, pp. 347-352.

[117] R. R. Yager: *On the measure of fuzziness and negation*, part i; Membership in the unit interval, Intern. J. of General Systems 5(4), 1979, pp.221-229.

[118] X. Yao: *Evolving Artificial Neural Networks*, in Proc. of the IEEE, 87(9), 1999, pp. 1423-1447.

[119] F. Yu, J. Z. Feng and J. Li: *A fuzzy logic controller design for vehicle ABS with a on-line optimized target wheel slip ratio*, International Journal of Automotive Technology, 3(4), 2002, pp. 165-170.

[120] L. A. Zadeh: *Fuzzy Sets*, Information and Control 8, 1965, pp. 338-353.

[121] L. A. Zadeh: *Outline of a new approach to the analysis of complex systems and decision processes*, IEEE Trans. on SMC 3, 1973, pp. 28-44.

[122] L. A. Zadeh: *Fuzzy Logic = Computing with Words*, IEEE Transactions on Fuzzy Systems, 2, 1996, pp. 103-111.

[123] A. H. Zavala, O. C. Nieto, I. Batyrshin and L. V. Vargas: *VLSI Implementation of a Module for Realization of Basic t-norms on Fuzzy Hardware*, Proc. of FUZZ-IEEE 2009, IEEE Conference on Fuzzy Systems, Jeju Island, Korea, August 20-24, 2009, pp. 655-659

[124] R. Zayani, R. Bouallegue and D. Roviras: *Levenberg-Marquardt Learning Neural Network for Adaptive Predistortion for Time-Varying HPA with Memory in OFDM Systems*, Proc. of 16th European Signal Processing Conference, Lausanne, Switzerland, 2008.

[125] Z. Zhu, Y. S. Ong and M. Dash: *Wrapper-filter feature selection algorithm using a memetic framework*, IEEE Trans., Syst., Man, Cybern., B., 37, 2007, pp. 70-76.

[126] H. H. Zimmerman and P. Zysno: *Latent connectives in human decision making*, Fuzzy Sets and Systems 4, 1980, pp. 37-51.