Chapter 2
Fuzzy Logic

2.1 Introduction

The real world is complex; this complexity generally arises from uncertainty. Humans have unconsciously been able to address complex, ambiguous, and uncertain problems thanks to the gift of thinking. This thought process is possible because humans do not need the complete description of the problem since they have the capacity to reason approximately. With the advent of computers and their increase in computation power, engineers and scientists are more and more interested in the creation of methods and techniques that will allow computers to reason with uncertainty.

Classical set theory is based on the fundamental concept of a *set*, in which individuals are either a member or not a member. A sharp, crisp, and ambiguous distinction exists between a member and a non-member for any well-defined set of entities in this theory, and there is a very precise and clear boundary to indicate if an entity belongs to a set. Thus, in classical set theory an element is not allowed to be in a set (1) or not in a set (0) at the same time. This means that many real-world problems cannot be handled by classical set theory. On the contrary, the fuzzy set theory accepts partial membership values $\mu_f \in [0, +1]$, and therefore, in a sense generalizes the classical set theory to some extent.

As Prof. Lotfi A. Zadeh suggests by his *principle of incompatibility*: "The closer one looks at a real-world problem, the fuzzier becomes the solution," and thus, imprecision and complexity are correlated [1]. Complexity is inversely related to the understanding we can have of a problem or system. When little complexity is presented, closed-loop forms are enough to describe the systems. More complex systems need methods such as neural networks that can reduce some uncertainty. When systems are complex enough that only few numerical data exist and the majority of this information is vague, fuzzy reasoning can be used for manipulating this information.

2.2 Industrial Applications

The imprecision in fuzzy models is generally quite high. However, when precision is apparent, fuzzy systems are less efficient than more precise algorithms in providing

us with the best understanding of the system. In the following examples, we explain how many industries have taken advantage of the fuzzy theory [2].

Example 2.1. Mitsubishi manufactures a fuzzy air conditioner. While conventional air conditioners use on/off controllers that work and stop working based on a range of temperatures, the Mitsubishi machine takes advantage of fuzzy rules; the machine operates smoother as a result. The machine becomes mistreated by sudden changes of state, more consistent room temperatures are achieved, and less energy is consumed. These were first released in 1989. □

Example 2.2. Fisher, Sanyo, Panasonic, and Canon make fuzzy video cameras. These have a digital image stabilizer to remove hand jitter, and the video camera can determine the best focus and lightning. Fuzzy decision making is used to control these actions. The present image is compared with the previous frame in memory, stationary objects are detected, and its shift coordinates are computed. This shift is subtracted from the image to compensate for the hand jitter. □

Example 2.3. Fujitec and Toshiba have a fuzzy scheme that evaluates the passenger traffic and the elevator variables to determine car announcement and stopping time. This helps reduce the waiting time and improves the efficiency and reliability of the systems. The patent for this type of system was issued in 1998. □

Example 2.4. The automotive industry has also taken advantage of the theory. Nissan has had an anti-lock braking system since 1997 that senses wheel speed, road conditions, and driving pattern, and the fuzzy ABS determines the braking action, with skid control [3]. □

Example 2.5. Since 1988 Hitachi has turned over the control of the Sendai subway system to a fuzzy system. It has reduced the judgment on errors in acceleration and braking by 70%. The Ministry of International Trade and Industry estimates that in 1992 Japan produced about $2 billion worth of fuzzy products. US and European companies still lag far behind. The market of products is enormous, ranging from fuzzy toasters to fuzzy golf diagnostic systems. □

2.3 Background

Prof. Lotfi A. Zadeh introduced the seminal paper on fuzzy sets in 1965 [4]. Since then, many developments have taken place in different parts of the world. Since the 1970s Japanese researchers have been the primary force in the implementation of fuzzy theory and now have thousands of patents in the area.

The world response to fuzzy logic has been varied. On the one hand, western cultures are mired with the *yes* or *no*, *guilty* or *not guilty*, of the binary Aristotelian logic world and their interpretation of the fuzziness causes a conflict because they are given a negative connotation. On the other hand, Eastern cultures easily accommodate the concept of fuzziness because it does not imply disorganization and imprecision in their languages as it does in English.

2.3.1 Uncertainty in Information

The uncertainties in a problem should be carefully studied by engineers prior to selecting an appropriate method to represent the uncertainty and to solve the problem. Fuzzy sets provide a way that is very similar to the human reasoning system. In universities most of the material taught in engineering classes is based on the presumption that knowledge is deterministic. Then when students graduate and enter "the real world," they fear that they will forget the correct formula.

However, one must realize that all information contains a certain degree of uncertainty. Uncertainty can arise from many factors, such as complexity, randomness, ignorance, or imprecision. We all use vague information and imprecision to solve problems. Hence, our computational methods should be able to represent and manipulate fuzzy and statistical uncertainties.

2.3.2 Concept of Fuzziness

In our everyday language we use a great deal of vagueness and imprecision, that can also be called fuzziness. We are concerned with how we can represent and manipulate inferences with this kind of information. Some examples are: a person's size is *tall*, and their age is classified as *young*.

Terms such as *tall* and *young* are fuzzy because they cannot be crisply defined, although as humans we use this information to make decisions. When we want to classify a person as tall or young it is impossible to decide if the person is in a set or not. By giving a *degree* of pertinence to the subset, no information is lost when the classification is made.

2.4 Foundations of Fuzzy Set Theory

Mathematical foundations of fuzzy logic rest in fuzzy set theory, which can be seen as a generalization of classical set theory. Fuzziness is a language concept; its main strength is its vagueness using symbols and defining them.

Consider a set of tables in a lobby. In classical set theory we would ask: Is it a table? And we would have only two answers, *yes* or *no*. If we code *yes* with a 1 and *no* with a 0 then we would have the pair of answers as {0,1}. At the end we would collect all the elements with 1 and have the set of tables in the lobby.

We may then ask what objects in the lobby can *function* as a table? We could answer that tables, boxes, desks, among others can function as a table. The set is not uniquely defined, and it all depends on what we mean by the word *function*. Words like this have many shades of meaning and depend on the circumstances of the situation. Thus, we may say that the set of objects in the lobby that can

function as a table is a *fuzzy set*, because we have not crisply defined the criteria to define the *membership* of an element to the set. Objects such as tables, desks, boxes may function as a table with a certain degree, although the fuzziness is a feature of their representation in symbols and is normally a property of models, or languages.

2.4.1 Fuzzy Sets

In 1965 Prof. Lotfi A. Zadeh introduced *fuzzy sets*, where many degrees of membership are allowed, and indicated with a number between 0 and 1. The point of departure for fuzzy sets is simply the generalization of the valuation set from the pair of numbers {0,1} to all the numbers in [0,1]. This is called a *membership function* and is denoted as $\mu_A(x)$, and in this way we have fuzzy sets.

Membership functions are mathematical tools for indicating flexible membership to a set, modeling and quantifying the meaning of symbols. They can represent a subjective notion of a vague class, such as chairs in a room, size of people, and performance among others. Commonly there are two ways to denote a fuzzy set. If X is the universe of discourse, and x is a particular element of X, then a fuzzy set A defined on X may be written as a collection of ordered pairs:

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

where each pair $(x, \mu_A(x))$ is a *singleton*. In a crisp set singletons are only x, but in fuzzy sets it is two things: x and $\mu_A(x)$. For example, the set A may be the collection of the following integers, as in (2.2):

$$A = \{(1, 1.0), (3, 0.7), (5, 0.3)\}\,. \tag{2.2}$$

Thus, the second element of A expresses that 3 belongs to A to a degree of 0.7. The *support set* of a fuzzy set A is the set of elements that have a membership function different from zero. Alternative notations for the fuzzy sets are *summations* or *integrals* to indicate the *union* of the fuzzy set, depending if the universe of discourse is *discrete* or *continuous*. The notation of a fuzzy set with a discrete universe of discourse is $A = \sum_{x_i \in X} \mu_A(x_i)/x_i$ which is the union of all the singletons. For a continuous universe of discourse we write the set as $A = \int_X \mu_A(x)/x$, where the integral sign indicates the union of all $\mu_A(x)/x$ singletons.

Now we will show how to create a triangular membership function using the Intelligent Control Toolkit for LabVIEW (ICTL). This triangular function must be between 0 and 3 with the maximum point at 1.5; we can do this using the **triangfunction.vi**. To evaluate and graph the function we must use a 1D array that can be easily created using the **rampVector.vi**. We can find this VI (as shown in Fig. 2.1)

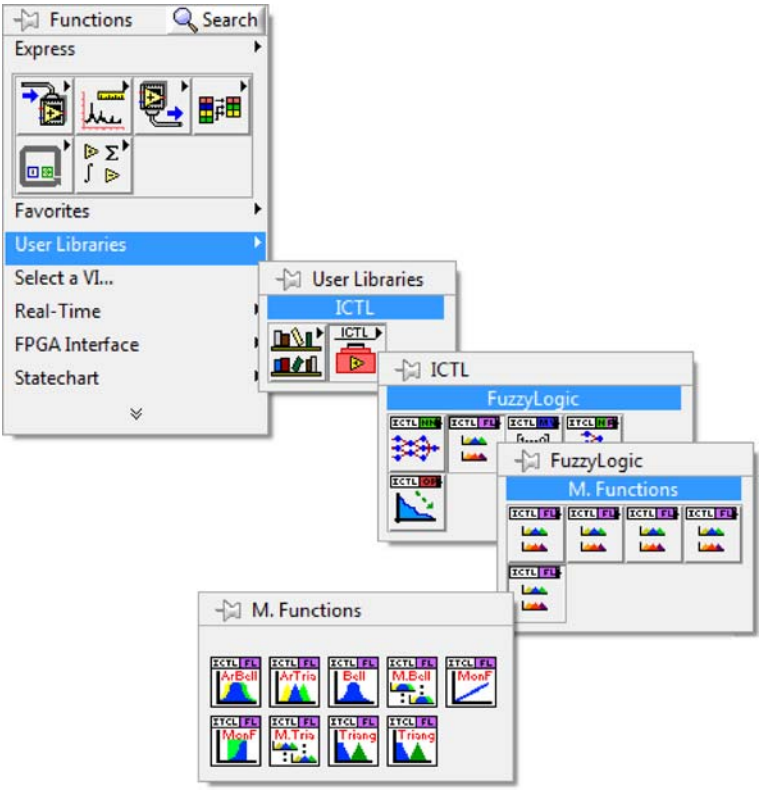

Fig. 2.1 Fuzzy function location on the ICTL

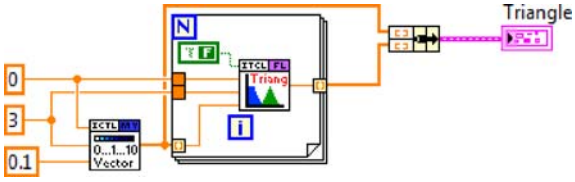

Fig. 2.2 Construction and evaluation of a triangular membership

in the fuzzy logic palette of the toolkit. The block diagram of the program that will create and evaluate the triangular function is shown in Fig. 2.2. The triangular function will be as the one shown in Fig. 2.3.

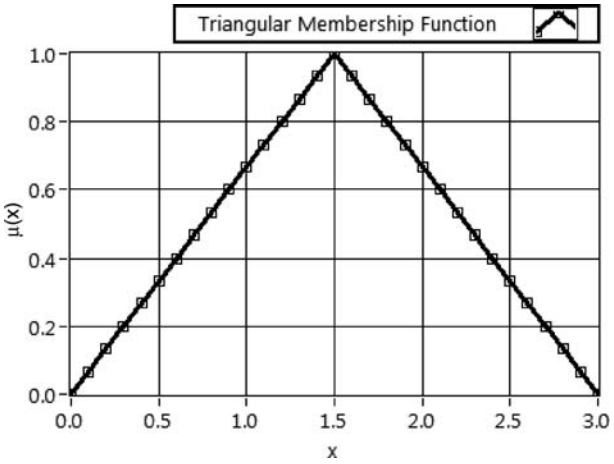

Fig. 2.3 Triangular membership function created with the ICTL

2.4.2 Boolean Operations and Terms

The two-valued logic is called *Boolean algebra*, named after George Boole, a nineteenth century mathematician and logician. In this algebra there are only three basic logic operations: *NOT* $\neg$, *AND* $\wedge$ and *OR* $\vee$. It is also common to use the symbols: $-$, $\cdot$, and $+$. Boolean algebraic formulas can be described by a truth table, where all the variables in the formula are the inputs and the value of the formula is the output. Conversely, a formula can be written from a truth table. For example the truth table for *AND* is shown in Table 2.1.

Complex Boolean formulas can be reduced to simpler equivalent ones using some properties. It is important to note that some rules of the Boolean algebra are the same as those of the ordinary algebra (e. g., $a \cdot 0 = 0$, $a \cdot 1 = a$), but others are quite different ($a + 1 = 1$). Table 2.2 shows the most important properties of Boolean algebra.

Table 2.1 Truth table of the *AND* Boolean operation

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

Table 2.2 The most important properties of Boolean algebra

Laws	Formulas
Characteristics	$a \cdot 0 = 0, a \cdot 1 = a, a + 0 = a$ and $a + 1 = 1$
Commutative law	$a + b = b + a$ and $a \cdot b = b \cdot a$
Associative law	$a + b + c = a + (b + c) = (a + b) + c$
	$a \cdot b \cdot c = a \cdot (b \cdot c) = (a \cdot b) \cdot c$
Distributive law	$a \cdot (b + c) = a \cdot b + a \cdot c$
Idempotence	$a \cdot a = a$ and $a + a = a$
Negation	$\overline{\overline{a}} = a$
Inclusion	$a \cdot \overline{a} = 0$ and $a + \overline{a} = 1$
Absorptive law	$a + a \cdot b = a$ and $a \cdot (a + b) = a$
Reflective law	$a + \overline{a} \cdot b = a + b, a \cdot (\overline{a} + b) = a \cdot b$, and $a \cdot b + \overline{a} \cdot b \cdot c = a \cdot b + b \cdot c$
Consistency	$a \cdot b + a \cdot \overline{b} = a$ and $(a + b) \cdot \left(a + \overline{b}\right) = a$
De Morgan's law	$\overline{a \cdot b} = \overline{a} + \overline{b}$ and $\overline{a + b} = \overline{a} \cdot \overline{b}$

2.4.3 Fuzzy Operations and Terms

Operations such as *intersection* and *union* are defined through the *min* ($\wedge$) and *max* ($\vee$) operators, which are analogous to *product* and *sum* in algebra. Formally the min and max of an element, where $\equiv$ stands for "by definition," are denoted by (2.3) and (2.4):

$$\mu_a \wedge \mu_b = \min(\mu_a, \mu_b) \equiv \begin{cases} \mu_a & \text{if and only if } \mu_a \leq \mu_b \\ \mu_b & \text{if and only if } \mu_a > \mu_b \end{cases} \tag{2.3}$$

$$\mu_a \vee \mu_b = \max(\mu_a, \mu_b) \equiv \begin{cases} \mu_a & \text{if and only if } \mu_a \geq \mu_b \\ \mu_b & \text{if and only if } \mu_a < \mu_b \end{cases}. \tag{2.4}$$

The most important fuzzy operations are shown in Table 2.3. The following functions in (2.5) are two fuzzy sets, a triangular and a bell-shaped membership function:

$$\mu_{\text{triangle}}(x) = \begin{cases} \frac{2(x-1)}{7}; & 1 \leq x \leq \frac{9}{2} \\ -\frac{2(x-8)}{7}; & \frac{9}{2} \leq x \leq 8 \end{cases} \qquad \mu_{\text{bell}}(x) = \frac{1}{1 + \left|\frac{x-0.1}{3}\right|^6}. \tag{2.5}$$

The diagrams for the membership functions can be found in Fig. 2.4, a union between the triangular and bell functions is shown in Fig. 2.5, and an intersection is shown in Fig. 2.6. The bell function and the complement are shown in Fig. 2.7.

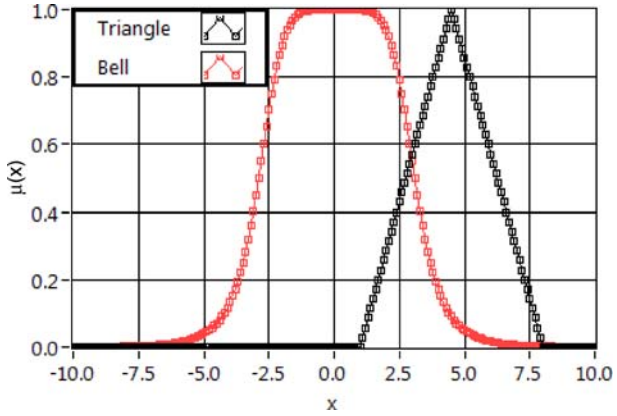

Fig. 2.4 Diagram of triangular and bell membership functions

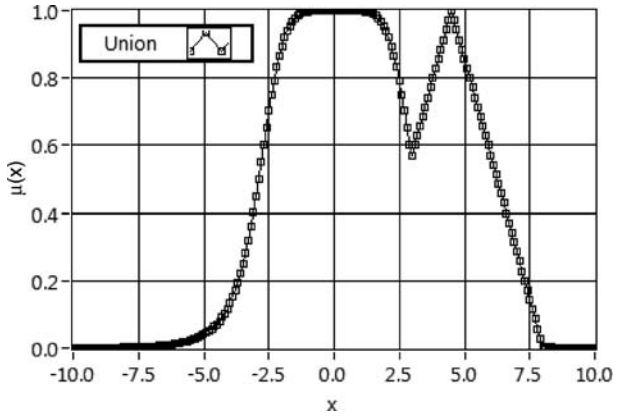

Fig. 2.5 Union of functions

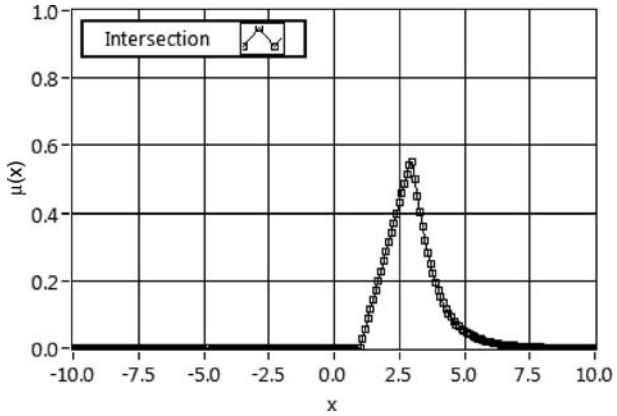

Fig. 2.6 Intersection of sets

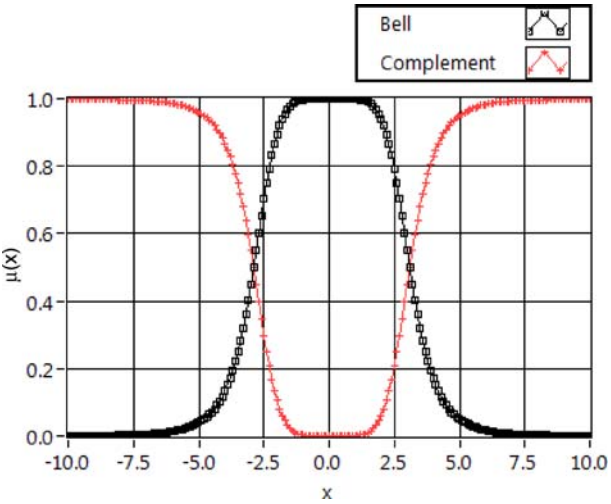

Fig. 2.7 Bell and complement of the bell function

Table 2.3 The most important fuzzy operations

Empty fuzzy set	It is empty if its membership function is zero everywhere in the universe of discourse.	$A \equiv \emptyset$ if $\mu_A(x) = 0, \forall x \in X$
Normal fuzzy set	It is normal if there is at least one element in the universe of discourse where its membership function equals one.	$\mu_A(x_a) = 1$
Union of two fuzzy sets	The union of two fuzzy sets A and B over the same universe of discourse X is a fuzzy set $A \cup B$ in X with a membership function which is the maximum of the grades of membership of every x and A and B. This operation is related to the *OR* operation in fuzzy logic.	$\mu_{A \cup B}(x) \equiv \mu_A(x) \vee \mu_B(x)$
Intersection of fuzzy sets	It is the minimum of the grades of every x in X to the sets A and B. The *intersection* of two fuzzy sets is related to the *AND*.	$\mu_{A \cap B}(x) \equiv \mu_A(x) \wedge \mu_B(x)$
Complement of a fuzzy set	The complement of a fuzzy set A is denoted as $\bar{A}$.	$\mu_{\bar{A}}(x) \equiv 1 - \mu_A(x)$
Product of two fuzzy sets	$A \cdot B$ denotes the product of two fuzzy sets with a membership function that equals the algebraic product of the membership function A and B.	$\mu_{A \cdot B}(x) \equiv \mu_A(x) \cdot \mu_B(x)$

Table 2.3 (continued)

Power of a fuzzy set	The β power of A (A^{β}) has the equivalence to linguistically modify the set with *VERY*.	$\mu_{A^{\beta}}(x) \equiv [\mu_A(x)]^{\beta}$
Concentration	Squaring the set is called *concentration CON*.	$\mu_{CON(\beta)}(x) \equiv (\mu_A(x))^2$
Dilation	Taking the square root is called *dilation* or *DIL*.	$\mu_{DIL(A)}(x) \equiv \sqrt{\mu_A(x)}$

2.4.4 Properties of Fuzzy Sets

Fuzzy sets are useful in performing operations using membership functions. Properties listed in Table 2.4 are valid for crisp and fuzzy sets, although some are specific for fuzzy sets only. Sets A, B, and C must be considered as defined over a common universe of discourse X.

All of these properties can be expressed using the membership function of the sets involved and the definitions of union, intersection and complement. De Morgan's law says that the intersection of the complement of two fuzzy sets equal the complement of their union. There are also some properties not valid for fuzzy sets such as the law of contradiction and the law of the excluded middle.

Table 2.4 The most important fuzzy properties

Double negation law	$\overline{(\bar{A})} = A$
Idempotency	$A \cup A = A \quad A \cap A = A$
Commutativity	$A \cap B = B \cap A \quad A \cup B = B \cup A$
Associative property	$(A \cup B) \cup C = A \cup (B \cup C)$ $(A \cap B) \cap C = A \cap (B \cap C)$
Distributive property	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)\, A \cap$ $(B \cup C) = (A \cap B) \cup (A \cap C)$
Absorption	$A \cap (A \cup B) = A$ $A \cup (A \cap B) = A$
De Morgan's laws	$\overline{A \cup B} = \overline{A} \cap \overline{B}$ $\overline{A \cap B} = \overline{A} \cup \overline{B}$

2.4.5 Fuzzification

This process is mainly used to transform a crisp set to a fuzzy set, although it can also be used to increase the fuzziness of a fuzzy set. A *fuzzifier function F* is used to control the fuzziness of the set. As an example the fuzzy set A can be defined with

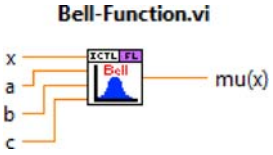

Fig. 2.8 Diagram of the **Bell-Function.vi**

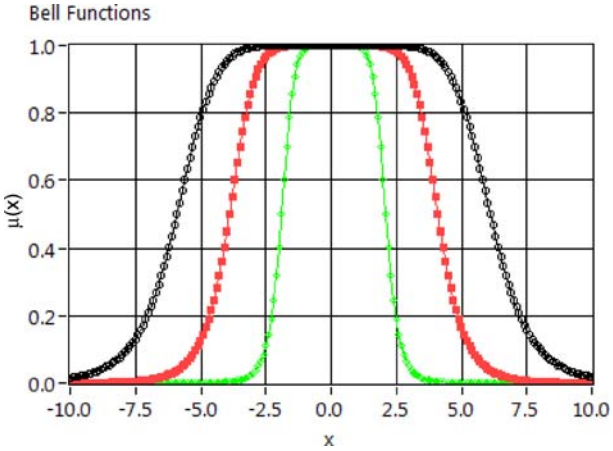

Fig. 2.9 Different forms of bell membership functions

the function in (2.6):

$$\mu_A(x) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}}, \tag{2.6}$$

where x is any positive real number, and the parameters a, b, c shape the form of the bell membership function. The fuzzy set A can be written as:

$$A \equiv \int_x \left[\frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}}\right] / x\,. \tag{2.7}$$

This is an example of the bell function with different parameters using the ICTL. We can use the **Bell-Function.vi** (shown in Fig. 2.8) to create the membership functions. The a, b and c parameters can be changed and the form of the function will be different. The membership functions are shown in Fig. 2.9. The code that generates the membership functions is shown in Fig. 2.10. Basically, a 1D array is used to evaluate each one of the bell functions and generate their different forms.

Example 2.6. The productivity of people can be modeled using a bell function. It will increase depending on their age, then it will remain on the top for several years and it will decrease when the person reaches a certain age. This model is shown in the membership function (Triangular function with saturation) given in Fig. 2.11. □

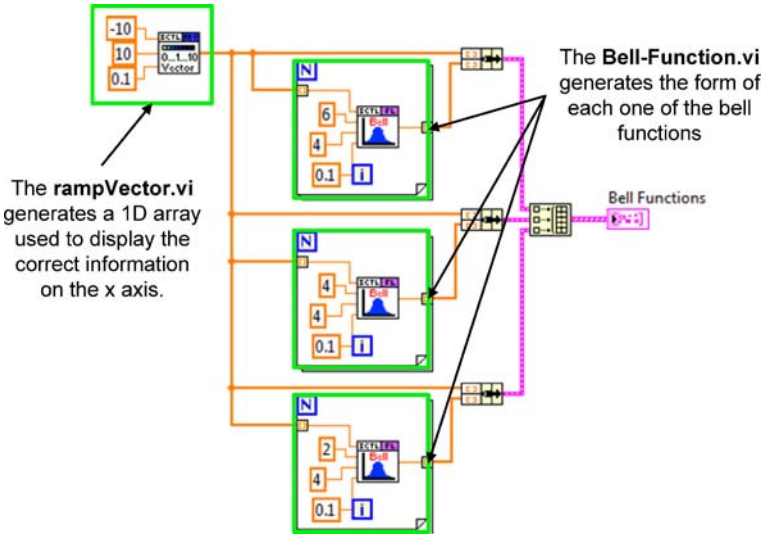

Fig. 2.10 Block diagram for the generation of bell membership functions

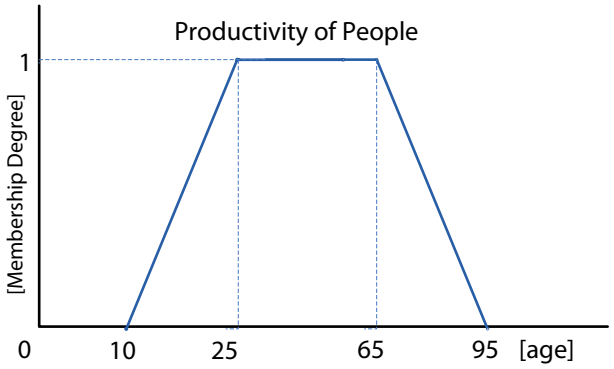

Fig. 2.11 Productivity of people fuzzy model

Why do not we select a conventional triangular membership function? The answer is because triangular functions reach their maximum at only one number and we are trying to model a range in which the productivity reaches its maximum. Thus, if we use triangular functions we would be representing the maximum of the productivity for a certain age of people (Fig. 2.12).

We can use a shoulder function to model a process, where after a certain level, the degree of membership remains the same (Fig. 2.13). We may want to model the level

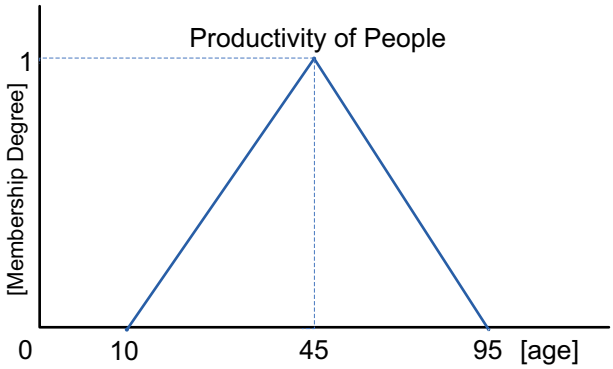

Fig. 2.12 Productivity of people modeled with a conventional triangular membership function

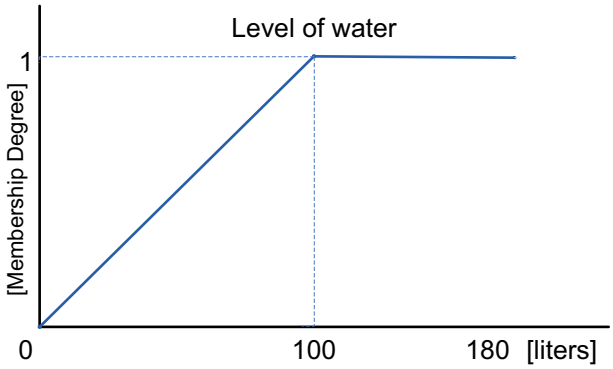

Fig. 2.13 Productivity modeled with a shoulder function

of water in a tank, which gets full after a certain number of liters are poured into the tank. Once we pass beyond that level, the degree of the level of water remains the same; the same happens if the tank is completely drained.

2.4.6 Extension Principle

This is a mathematical tool used to extend crisp mathematical notions and operations to the fuzzy realm, by fuzzifying the parameters of a function, resulting in computable fuzzy sets. Suppose that we have a function f that maps elements x_1, x_2, …, x_n of a universe of discourse X to another universe of discourse Y, and

a fuzzy set A defined on the inputs of f in (2.8):

$$\begin{array}{l} y_1 = f\,(x_1) \\ y_2 = f\,(x_2) \\ \cdots \\ y_n = f\,(x_n) \end{array} \qquad A = \mu_A\,(x_1)/x_1 + \mu_A\,(x_2)/x_2 + \cdots + \mu_A\,(x_n)\,/x_n\,. \tag{2.8}$$

What could happen if the input of function f becomes fuzzy? Would the output be fuzzy? The extension principle then tells us that there is a fuzzy output given by (2.9):

$$B = f\,(B) = \mu_A\,(x_1)/f\,(x_1) + \mu_A\,(x_2)/f\,(x_2) + \cdots + \mu_A\,(x_n)\,/\,f\,(x_n)\,, \tag{2.9}$$

where every single image of x_i under f becomes fuzzy to a degree $\mu_A\,(x_i)$. Most of the functions out there are *many-to-one*, meaning several x map the same y. We then have to decide which of the two or more membership values we should take as the membership value of the output. The extension principle says that the *maximum* of the membership values of these elements of the fuzzy set A should be chosen as the membership of the desired output. In the other case if no element x in X is mapped to the output, then the membership value of the set B at the output is zero.

Example 2.7. Suppose that $f\,(x) = ax + b$ and $a \in A = \{1, 2, 3\}$ and $b \in B = \{2, 3, 5\}$ with $x = 6$. Then $f\,(x) = 6A + B = \{8, 15, 23\}$. □

Example 2.8. Consider the following function $y = F\,(s) = -2s^2 + 1$ with domain $S = R$ and range $Y = (-\infty, 1]$. Suppose that $S_f = [0, 2]$ is a fuzzy subset with the

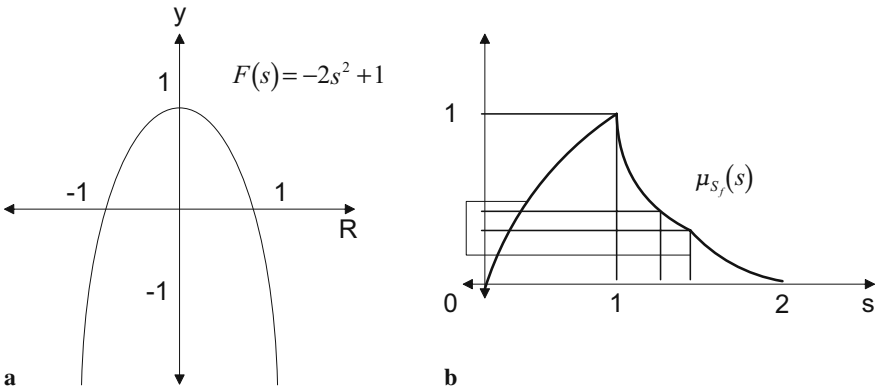

Fig. 2.14a,b Extension principle example. **a** Function: $F\,(s) = -2s^2 + 1$. **b** Fuzzy membership function of function $F\,(s) = -2s^2 + 1$

membership function shown in Fig. 2.14. The fuzzy subset $Y_f = F\left(S_f\right)$ is given by (2.10):

$$\begin{aligned} Y_f =& F\left(S_f\right) = F\left([0,2]\right) = -2\,[0,2]\cdot[0,2] + 1 \\ =& -2\,[0,4] + 1 = [-8,0] \\ =& [-7,1]\ . \end{aligned} \tag{2.10}$$

The membership function $\mu_{Y_f}(s)$ associated with Y_f is determined as follows. Let y run through from -7 to 1. For each y, find the corresponding $s \in S_f$ satisfying $y = F(s)$, then $\mu_{Y_f}(s) = \sup_{s:F(s)=y} \mu_{S_f}(s)$. It is clear that for any $y \in [-7,1]$, there is always one $s \in [0,2]$ satisfying $y = F(s) = -2s^2 + 1$. Therefore, it can be easily verified that the membership function is the one shown in Fig. 2.15. □

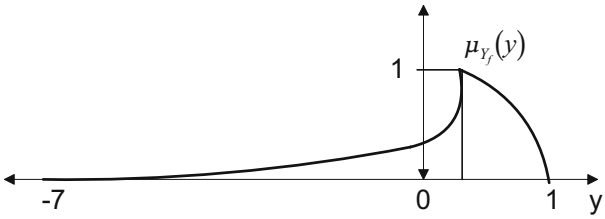

Fig. 2.15 Resulting function when the extension principle is applied

2.4.7 Alpha Cuts

An alpha cut (α-*cut*) is a crisp set of elements of A belonging to the fuzzy set to a degree α. The α-*cut* of a fuzzy set A is the crisp set comprised of all elements x of universe X for which the membership function of A is greater or equal to α (2.11):

$$A_\alpha = \{x \in X \,|\, \mu_A(x) \geq \alpha\}\ , \tag{2.11}$$

where α is in the range of $0 < \alpha \leq 1$ and "|" stands for "such that."

Example 2.9. A triangular membership function with an α-*cut* at 0.4 is shown in Fig. 2.16. Figure 2.17 shows the block diagram. □

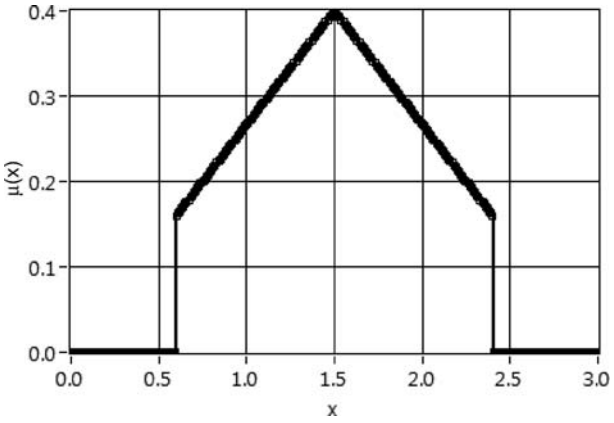

Fig. 2.16 Triangular membership function with alpha cut of 0.4

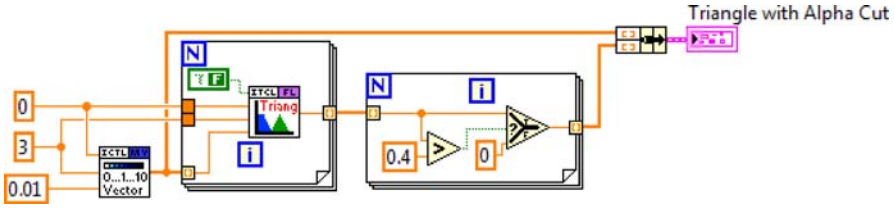

Fig. 2.17 Block diagram of the triangular membership function with alpha cut of 0.4

2.4.8 The Resolution Principle

This principle offers a way of representing membership to fuzzy sets by means of α-*cuts* as (2.12) denotes:

$$\mu_A(x) = \bigvee_{0<\alpha\leq 1} [\alpha \cdot \mu_{A_\alpha}(x)] \ . \tag{2.12}$$

The maximum is taken over all α-cut, and the equation indicates that the membership function of A is the union of all α-cut, after each one of them has been multiplied by α. Figure 2.18 shows these functions.

2.4.9 Fuzziness of Uncertainty

Many kinds of uncertainties arise in the real world and there are many techniques to model them. Randomness is one kind, which is typically modeled using probability theory. Outcomes are assumed to be observations of random variables and these variables have distribution laws. Fuzziness manipulates uncertainty by dealing with

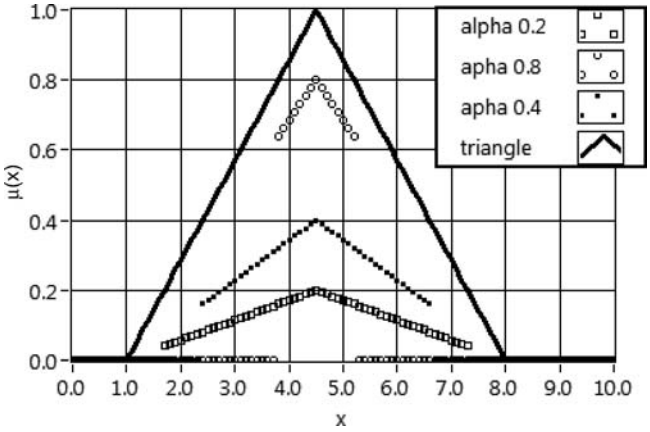

Fig. 2.18 A triangular function composed of multiple alpha cuts

the boundaries of a set that are not clearly defined. The membership in such classes is a matter of degree rather than certainty specified by fuzzy sets.

2.4.10 Possibility and Probability Theories

Possibility theory emphasizes the quantification of the semantic or meaning rather than the measure of information. The theory of possibility is analogous and yet conceptually different from the theory of probability. Probability is a measure of frequency of occurrence of an event, which has a physical event basis. Thus, probabilities have a physical event basis and are related to statistical experiments; they are primarily used for quantifying how frequently a sample occurs in a population.

Possibility theory attempts to quantify how accurately a sample resembles a *stereotype* element of a population. This stereotype is a prototypical class of the population and is known as a fuzzy set. This theory focuses more on the *imprecision* intrinsic in the language, while probability theory focuses more on the *uncertainty* of events, in the sense of its randomness in nature.

Probabilistic methods have been the instrument for quantifying equipment and human reliability as well, in which two concepts are very important: the failure rate and the error rate. Knowing these concepts and being able to control them leads to the correct understanding and function of machines, which allows industries to save money. But the correct estimation of these parameters requires a large amount of data, thus in practice they are estimated by experts based on their engineering judgment. Here is where fuzzy probabilities and possibilities can be used to model these judgments.

Over the years a new concept has emerged: the possibility theory. It is known as a fuzzy measure which is a function assigning a value between 0 and 1 to each

crisp set of the universe of discourse that signifies the degree to which a particular element belongs to a set. Sugeno introduced this concept in 1974 as part of his Ph.D. dissertation.

Possibility measures are softer than probability measures, and their interpretation is quite different. On the one hand, probability is used to quantify the frequency of occurrence of an event, while on the other hand, possibility is used to quantify the meaning of an event. Possibility is an upper bound of probability, i.e., a higher degree of possibility does not imply a higher degree of probability. But if an event is not possible, then it is not probable.

If we attempt to use both probability and possibility theories to describe a similar thing, we can use the *possibility/probability consistency principle* as a guide. It will help us draw the difference between the *objectivistic* use of probability measures and *subjectivist* use of possibility or fuzzy measures.

2.5 Fuzzy Logic Theory

2.5.1 From Classical to Fuzzy Logic

Logic refers to the study of methods and principles of human reasoning. Classical logic deals with propositions that are either *true* or *false*, where each proposition has an opposite. Thus, classical logic deals with combinations of variables that represent propositions. As each variable stands for a hypothetical proposition any combination eventually assumes a truth value (true or false), but never in between the two.

The main content of classical logic is the study of *rules* that allow new logical variables to be produced as *functions* of certain existing variables. An example of a rule is shown in (2.13):

$$IF\ x_1 \text{ is } true\ AND\ x_2 \text{ is } false\ AND\ \ldots\ AND\ x_n \text{ is } false\ THEN\ y \text{ is } false\,. \quad (2.13)$$

The fundamental assumption upon which the classical logic is based is that every proposition is either true or false. Now it is well understood that many propositions are both partially true and false. Multi-valued logic was a first attempt to extend and generalize classical logic. In the 1930s an n-valued logic was invented by Lukasiewicz [5], which even allowed for $n = \infty$. More recently, it has been understood that there exists an isomorphism between classical logic and crisp set theory and similarly between Lukasiewicz and the fuzzy set theory.

2.5.2 Fuzzy Logic and Approximate Reasoning

The ultimate goal of fuzzy logic is to provide foundations for approximate reasoning using imprecise propositions based on fuzzy set theory, similar to classical reasoning

using precise propositions based on classical set theory. We will first recall how classical reasoning works, where the following syllogism is an example of such reasoning in linguistic terms:

1. Everyone who is 70 years old is old.
2. Hiram is 70 years old and Miriam is 39 years old.
3. Hiram is old but Miriam is not.

This is an example of a precise deductive inference that is correct in the sense of classical two-valued logic. When the output variable represented by a logical formula is always true regardless of the truth values of the inputs, it is called a *tautology*. If it is the contrary then it is called a *contradiction*. Various tautologies can be used for making deductive inferences, and are referred to as *inference rules*. They can also be expressed with truth tables. The four most frequent are:

- *Modus ponens:* $(a \wedge (a \Rightarrow b)) \Rightarrow b$
- *Modus tollens:* $\left(\overline{b} \wedge (a \Rightarrow b)\right) \Rightarrow \overline{a}$
- *Syllogism:* $(a \Rightarrow b) \wedge (b \Rightarrow c) \Rightarrow (a \Rightarrow c)$
- *Contraposition:* $(a \Rightarrow b) \Rightarrow \left(\overline{b} \Rightarrow \overline{a}\right)$.

We will now consider an example of approximate reasoning in linguistic terms that cannot be handled by the classical reasoning logic:

1. Everyone who is 60 to 70 years old is old, but very old if 71 years old or above; everyone who is 20 to 39 is young but very young if 19 years old or below.
2. Hiram is 70 years old and Miriam is 39 years old.
3. Hiram is old but not very old; Miriam is young but not very young.

This is an example of *approximate reasoning*; in order to deal with an imprecise inference, fuzzy logic can be employed. It allows imprecise linguistic terms such as:

- Fuzzy predicates: rare, expensive, fast, high
- Fuzzy quantifiers: few, usually, much, little
- Fuzzy truth values: true, unlikely true, false, and mostly false.

To describe fuzzy logic mathematically, the following concepts and notations are introduced. Let **S** be a universe set and A a fuzzy set associated with a membership function $\mu_A(x)$, $x \in \mathrm{S}$. If $y = \mu_A(x_0)$ is a point in $[0, 1]$, representing the truth value of the proposition "x_0 is a," then the value for "x_0 is not a" is:

$$\overline{y} = \mu_A(x_0 \text{ is not } a) = 1 - \mu_A(x_0 \text{ is } a) = 1 - \mu_A(x_0) = 1 - y\,. \tag{2.14}$$

Consequently for n members $x_1, \ldots, x_n$ in **S** with n corresponding truth values $y_i = \mu_A(x_i)$ in $[0, 1]$, $i = 1, \ldots, n$ by applying the extension principle, the truth values of "not a" get defined as $\overline{y}_i = 1 - y_i$, $i = 1, \ldots, n$. We note that $n = \infty$ is allowed. The same can be applied to other logical operators.

For instance in the modus ponens $(a \wedge (a \Rightarrow b)) \Rightarrow b$, the inference rule is: *IF* $\mu_A(a) > 0$ *AND* $\mu_A(a \Rightarrow b) = \min\{1, 1 + \mu(b) - \mu(a)\} > 0$ *THEN*

$\mu_B(a) > 0$, where $\mu > 0$ is equivalent to $\mu \in (0, 1]$. This fuzzy logic inference can be interpreted as follows: *IF* a is true with a certain degree of confidence *THEN* b is true with a certain degree of confidence. All of these degrees of confidence can be quantitatively evaluated by using the corresponding membership functions. This is a *generalized modus ponens*, called *fuzzy modus ponens*.

2.5.3 Fuzzy Relations

In fuzzy relations we consider n-tuples of elements that are related to a *degree*. Just as the question of whether some element belongs to a set may be considered a matter of degree, whether some elements are associated may also be a matter of degree. Fuzzy relations are fuzzy sets defined on Cartesian products. While fuzzy sets are defined on a single universe of discourse, fuzzy relations are defined on higher-dimensional universes of discourse.

If $\mathbf{S}$ is the universe set and A and B are subsets, $A \times B$ will denote a product set in the universe $\mathbf{S} \times \mathbf{S}$. A fuzzy relation is a relation between elements of A and elements of B, described by a membership function $\mu_{AxB}(a, b)$, $a \in A$ and $b \in B$.

A discrete example of a fuzzy relation can be defined as: $\mathbf{S} = R$, $A = \{a_1, a_2, a_3, a_4\} = \{1, 2, 3, 4\}$ and $B = \{b_1, b_2, b_3\} = \{0, 0.1, 2\}$. Table 2.5 defines a fuzzy relation: a is considerably larger than b.

Table 2.5 Definition of relation: a is considerably larger than b

	b_1	b_2	b_3
a_1	0.6	0.6	0.0
a_2	0.8	0.7	0.0
a_3	0.9	0.8	0.4
a_4	1.0	0.9	0.5

2.5.4 Properties of Relations

Fuzzy relations can be represented in many ways: linguistically, listed, in a directed graph, tabular, matrix, among others. Crisp and fuzzy relations are classified on the basis of the mathematical properties they possess. In fuzzy relations, different properties call for different requirements for the membership function of a relation. The following are some of the properties that a relation can have:

- *Reflexive*. We say that a relation R is *reflexive* if any arbitrary element x in S for which xRx is *valid*.

- *Anti-reflexive.* A relation R is *anti-reflexive* if there is no x in S for which xRx is valid.
- *Symmetric*. A relation R is *symmetric* if for all x and y in S, the following is true: if xRy then yRx is valid also.
- *Anti-symmetric*. A relation R is *anti-symmetric* if for all x and y in S, when xRy is valid and yRx is also valid, then $x = y$.
- *Transitive.* A relation R is called *transitive* if the following for all x, y, z in S: if xRy is valid and yRx is also valid, then xRz is valid as well.
- *Connected.* A relation R is *connected* when for all x, y in S, the following is true: if $x \neq y$, then either xRy is valid or yRx is valid.
- *Left unique.* A relation R is *left unique* when for all x, y, z in S the following is true: if xRy is valid and yRx is also valid, then we can infer that $x = y$.
- *Right unique.* A relation R is *right unique* when for all x, y, z in S the following is true: if xRy is valid and xRz is also valid, then we can infer that $y = z$.
- *Biunique.* A relation R that is both *left unique* and *right unique* is called *biunique.*

2.5.5 Max–Min Composition

Let R, R^1, R^2, R^3 be fuzzy relations defined on the same product set $A \times A$ and let $\circ$ be the max–min composition operation for these fuzzy relations. Then:

1. The max–min composition is *associative* (2.15):

$$\left(R^1 \circ R^2\right) \circ R^3 = R^1 \circ \left(R^2 \circ R^3\right) . \tag{2.15}$$

2. If R^1 is reflexive and is arbitrary R^2 is arbitrary, then $\mu_{R^2}(a,b) \leq \mu_{R^1 \circ R^2}(a,b)$ for all a,$b \in A$ and $\mu_{R^2}(a,b) \leq \mu_{R^2 \circ R^1}(a,b)$ for all $a, b \in A$.
3. If R^1 and R^2 are reflexive, then so are $R^1 \circ R^2$ and $R^2 \circ R^1$.
4. If R^1 and R^2 are symmetric and $R^1 \circ R^2 = R^2 \circ R^1$, then $R^1 \circ R^2$ is symmetric. In particular if R is symmetric then so is $R \circ R$.
5. If R is symmetric and transitive, then $\mu_R(a,b) \leq \mu_R(a,a)$ for all $a, b \in A$.
6. If R is reflexive and transitive, then: $R \circ R = R$.
7. If R^1 and R^2 are transitive and $R^1 \circ R^2 = R^2 \circ R^1$, then $R^1 \circ R^2$ is transitive.

An example will show how the max–min composition works for the approximate reasoning.

Example 2.10. Supposing that we have two relations R^1 and R^2, we want to compute the max–min composition of the two, $R = R^1 \circ R^2$. The relationships to be composed are described in Tables 2.6 and 2.7. □

To find the new relation we use the definition of the max–min composition: $\mu_{R_1 \circ R_2}(x,z) = \bigvee_y [\mu_{R_1}(x,y) \wedge \mu_{R_2}(y,z)]$. To make the composition we proceed in the following manner. First, we fix x and z and vary y. Next we evaluate the following

Table 2.6 Definition of relation: R^1

R^1	y_1	y_2	y_3	y_4	y_5
x_1	0.1	0.2	0.0	1.0	0.7
x_2	0.3	0.5	0.0	0.2	1.0
x_3	0.8	0.0	1.0	0.4	0.3

Table 2.7 Definition of relation: R^2

R^2	y_1	y_2	y_3	y_4
x_1	0.9	0.0	0.3	0.4
x_2	0.2	1.0	0.8	0.0
x_3	0.8	0.0	0.7	1.0
x_4	0.4	0.2	0.3	0.0
x_5	0.0	1.0	0.0	0.8

pairs of minima, as shown in (2.16):

$$\begin{aligned} \mu_{R_1}(x_1, y_1) \wedge \mu_{R_2}(y_1, z_1) &= 0.1 \wedge 0.9 = 0.1 \\ &\vdots \\ \mu_{R_1}(x_1, y_5) \wedge \mu_{R_2}(y_5, z_1) &= 0.7 \wedge 0.0 = 0.0 \end{aligned} \quad . \tag{2.16}$$

We take the maximum of these terms and obtain the value of the (x_1, z_1) element of the relation as in: $\mu_{R_1 \circ R_2}(x_1, z_1) = 0.1 \vee 0.2 \vee 0.0 \vee 0.4 = 0.4$. We then determine the grades of membership for all other pairs and we finally obtain R as shown in Table 2.8.

Table 2.8 Definition of min–max composition: R

$R = R^1 \circ R^2$	z_1	z_2	z_3	z_4
x_1	0.4	0.7	0.3	0.7
x_2	0.3	1.0	0.5	0.8
x_3	0.8	0.3	0.7	1.0

2.5.6 Max–Star Composition

Different operations can be used in place of min in the max–min composition while still performing maximization. This type of composition is known as max–star or max-$*$-composition. It is defined in (2.17). The integral sign in this equation is replaced by summation when the product is discrete.

$$R^1 * R^2 \equiv \int_{X \times Z} \bigvee_y [\mu_{R^1}(x, y) * \mu_{R^2}(y, z)]/(x, z) \, . \tag{2.17}$$

2.5.7 Max–Average Composition

In a max–average composition the arithmetic sum divided by 2 is used. Thus the max–average composition of R^1 with R^2 is a new relation $R^1 \langle + \rangle R^2$ (2.18):

$$R^1 \langle + \rangle R^2 \equiv \int_{X \times Z} \bigvee_y \left[\frac{1}{2} \left(\mu_{R^1}(x, y) + \mu_{R^2}(y, z) \right) \right] / (x, z) \,. \tag{2.18}$$

2.6 Fuzzy Linguistic Descriptions

Fuzzy linguistic descriptions are often called *fuzzy systems* or *linguistic descriptions*. They are formal representations of systems made through fuzzy *IF–THEN* rules. A linguistic variable is a variable whose arguments are words modeled by *fuzzy sets*, which are called *fuzzy values*. They are an alternative to analytical modeling systems. Informal linguistic descriptions used by humans in daily life as well as in the performance of skilled tasks are usually the starting point for the development of fuzzy linguistic descriptions. Although fuzzy linguistic descriptions are formulated in a human-like language, they have rigorous mathematical foundations involving fuzzy sets and relations. The knowledge is encoded in a statement of the form shown in (2.19):

IF (a set of conditions is satisfied) *THEN* (a set of consequences can be inferred) . (2.19)

A *general fuzzy IF–THEN rule* has the form:

$$IF\ a_1 \text{ is } A_1\ AND \ldots AND\ a_n \text{ is } A_n\ THEN\ b \text{ is } B \,. \tag{2.20}$$

Using the fuzzy logic *AND* operation, this rule is implemented by:

$$\mu_A(a_1) \wedge \ldots \mu_{A_n}(a_n) \Rightarrow \mu_B(b) \,. \tag{2.21}$$

Reasoning with Fuzzy Rules

Fuzzy reasoning includes two distinct parts: evaluating the rule antecedent (*IF* part of the rule) and implication or applying the result to the consequent, the *THEN* part of the rule. While in classical rule-based systems if the antecedent of the rule is true, then the consequent is also true, but in fuzzy systems the evaluation is different. In fuzzy systems the antecedent is a fuzzy statement, this means all the rules fire at some extent. If the antecedent is true in some degree of membership, then the consequent is also true in some degree.

Example 2.11. Consider two fuzzy sets, *tall men* and *heavy men* represented in Fig. 2.19. These fuzzy sets provide the basis for a weight estimation model. The model is based on a relationship between a man's height and his weight, which can be expressed with the following rule: *IF* height is *tall*, *THEN* weight is *heavy*. The value of the output or the membership grade of the rule consequent can be estimated directly from a corresponding membership grade in the antecedent. Fuzzy rules can have multiple antecedents, as the consequent of the rule, which can also include multiple parts. □

In general, fuzzy expert systems incorporate not one but several rules that describe expert knowledge. The output of each rule is a fuzzy set, but usually we need to obtain a single number representing the expert system output, the crisp solution. To obtain a single crisp output a fuzzy expert system first aggregates all output fuzzy sets into a single output fuzzy set, and then defuzzifies the resulting set into a single number.

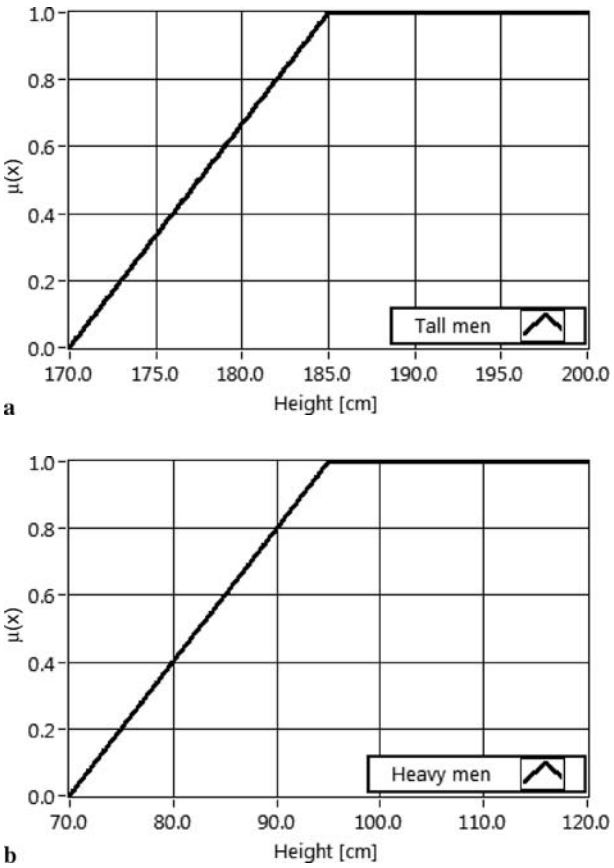

Fig. 2.19a,b Fuzzy sets. **a** Tall men. **b** Heavy men

2.7 The Fuzzy Logic Controller

With traditional sets an element either belongs to the set or does not belong to the set {0,1}, while in fuzzy sets the degree to which the element belongs to the set is analyzed and it is called the membership degree, giving values in the range [0,1], where 1 indicates that the element belongs completely to the set. The *fuzzy logic controllers* (FLC) make a non-linear mapping between the input and the output using *membership functions* and *linguistic rules* (normally in the form if__then__).

In order to use a FLC, knowledge is needed and this can be represented as two different types:

1. *Objective* information is what can be somehow quantifiable by mathematical models and equations.
2. *Subjective* information is represented with linguistic rules and design requirements.

2.7.1 Linguistic Variables

Just like in human thinking, in fuzzy logic systems (FLS) linguistic variables are utilized to give a "value" to the element, some examples are *much*, *tall*, *cold*, etc. FLS require the linguistic variables in relation to their numeric values, their quantification and the connections between variables and the possible implications.

2.7.2 Membership Functions

In FLS the membership functions are utilized to find the degree of membership of the element in a given set.

2.7.3 Rules Evaluation

The rules used in the FLS are of the *IF–THEN* type, for example, *IF* x_1 is big *THEN* y_1 is small. To define the rules you need an expert, or you must be able to extract the information from a mathematic formula. The main elements of a FLC are fuzzification, rules evaluation, and defuzzification, as shown in Fig. 2.20.

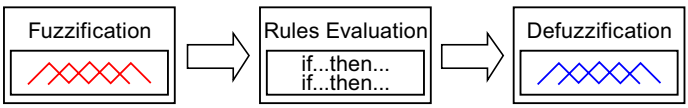

Fig. 2.20 Elements of the FLC

2.7.4 Mamdani Fuzzy Controller

The Mamdani is one kind of fuzzy controller. This section gives an introduction to the Mamdani fuzzy controller.

2.7.5 Structure

This controller consists of three main parts: fuzzification, rules evaluation and defuzzification. The inputs have to be crisp values in order to allow the fuzzification using membership functions, and the outputs of this controller are also crisp values. This controller is shown in Fig. 2.21.

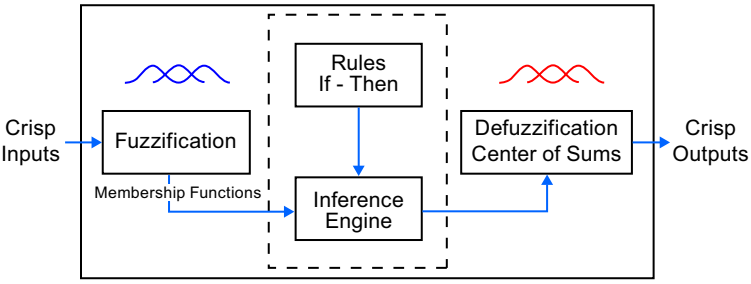

Fig. 2.21 Mamdani block diagram

2.7.6 Fuzzification

For mapping the crisp values to fuzzy ones, you have to evaluate their membership degree using membership functions. With this you get one fuzzy value for each crisp input.

An example is presented in Fig. 2.22 where $\mu(a)$ is the membership value, and the crisp values are a_0.

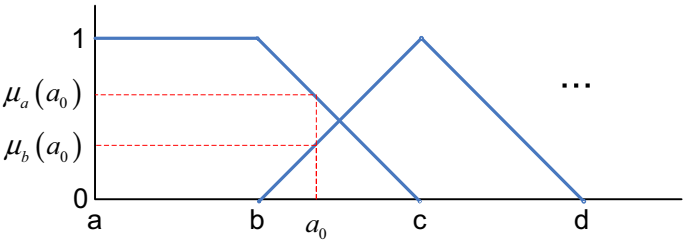

Fig. 2.22 Membership functions (input)

2.7.7 Rules Evaluation

After getting the membership values of the inputs, they are evaluated using *IF–THEN* rules: *IF* a is x *AND* b is y *AND* c is z *THEN* w, where a, b and c are the crisp inputs, x, y and z are the fuzzy clusters to which the inputs may correspond, and w is the output fuzzy cluster used to defuzzify. To be able to obtain the fuzzy values of the outputs, the system has to use an inference engine. The min–max composition is used which takes the minimum of the premises and the maximum of the consequences.

2.7.8 Defuzzification

To defuzzify the outputs we use the center of sums method. In this method we take the output from each contributing rule, and then we add them. The center of sums is one of the most popular methods for defuzzificating because it is very easy to implement and gives good results. With (2.22) we get the crisp value of the outputs, and Fig. 2.23 shows the graphical discrete representation.

$$u^* = \frac{\sum_{i=1}^{N} u_i \cdot \sum_{i=1}^{N} \mu_{A_k}(u_i)}{\sum_{i=1}^{N} \sum_{k=1}^{n} \mu_{A_k}(u_i)} . \tag{2.22}$$

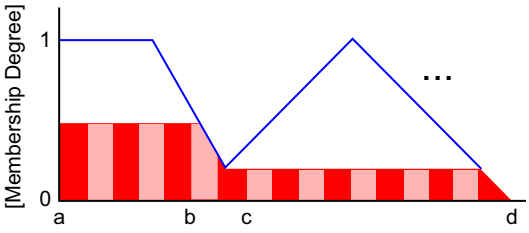

Fig. 2.23 Membership functions (output)

2.7.9 Tsukamoto Fuzzy Controller

Tsukamoto controllers are like the Mamdani controllers but with monotonic input and output membership functions. A monotonic function is a function that preserves the given order. In other words, it increases or decreases, and if its first derivative (needs not be continuous) does not change in sign.

2.7.10 Takagi–Sugeno Fuzzy Controller

The Takagi–Sugeno is another kind of fuzzy controller; the following gives an introduction to this kind of controller. The main difference in Sugeno type is in the defuzzificatio stage. In this case we do not use membership functions anymore.

2.7.11 Structure

This controller consists of three main parts: fuzzification, rules evaluation and defuzzification (see Fig. 2.24). The inputs have to be crisp values in order to allow the fuzzification to use membership functions, and the outputs of this controller are also crisp values.

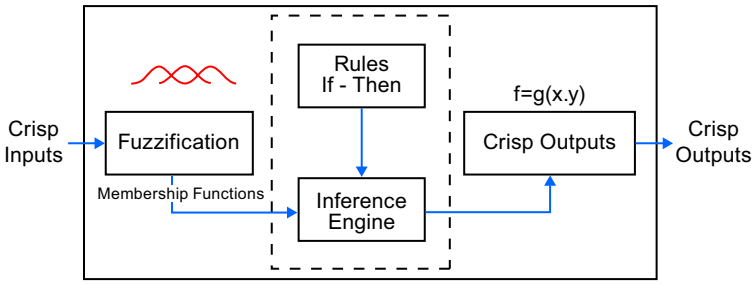

Fig. 2.24 Takagi–Sugeno diagram

2.7.12 Fuzzification

In order to transform the crisp values into fuzzy ones, you have to evaluate those using membership functions. With this you get one fuzzy value for each crisp input. The membership functions could be either conventional ones or non-conventional, the selection of the membership function depends on the specific problem. As result the first step in fuzzy logic is to select the best membership function for describing the problem.

2.7.13 Rules Evaluation

After getting the fuzzy values of the inputs, they are evaluated using *IF–THEN* rules: *IF* a is x *AND* b is y *AND* c is z *THEN* $u = f(a,b,c)$, where a, b and c are the crisp

inputs, x, y and z are the fuzzy clusters to which the inputs could correspond, and $u = f(a,b,c)$ is a polynomial. Instead of evaluating a, b and c, we evaluate in the polynomial the number of the fired rules.

These polynomials are calculated using regressions in order to adjust them to the desired output functions. For the inference system we use the minimum of the premises, but because the consequence of the rule is not fuzzy, the maximum of the consequences could not be used.

2.7.14 Crisp Outputs

To defuzzify take the product of the sum of the minimum of the antecedents of every rule fired times the value of the polynomial evaluated in that fired rule, all this divided by the sum of the minimum of the antecedents of every rule fired:

$$\text{Output} = \frac{\sum_{i-1}^{r} [\min(\mu_{ix,y,z}) u(a,b,c)]}{\sum_{i-1}^{r} \min(\mu_{ix,y,z})} . \tag{2.23}$$

2.8 Implementation of the Fuzzy Logic Controllers Using the Intelligent Control Toolkit for LabVIEW

We will now create a FLC using the Intelligent Control Toolkit for LabVIEW (ICTL). The fuzzy logic VI includes the following parts:

- Mamdani controller.
- Takagi–Sugeno controller.
- Tsukamoto controller.

It is important to state that there are no limitations on the number of linguistic variables and linguistic terms, nor on the type of membership functions supported. For a better understanding of how to use the FLC VIs, the implementation of FLCs in a mobile robot will be explained step-by-step in the following.

A robot named Wheel sends via Bluetooth® the distance measured by right, center, and left ultrasonic distance sensors. It has two servomotors that receive 0 for no movement, 1 to go clockwise, and 2 for counterclockwise, as in the diagram in Fig. 2.25.

The robot is driven by a BASIC Stamp® controller, the Bluetooth antenna is an EmbeddedBlue™ Transceiver from Parallax, and the two servos and three Ping)))™ sensors are also from Parallax. For the computer we use a Belkin Bluetooth antenna. Figure 2.26 plots the information flow.

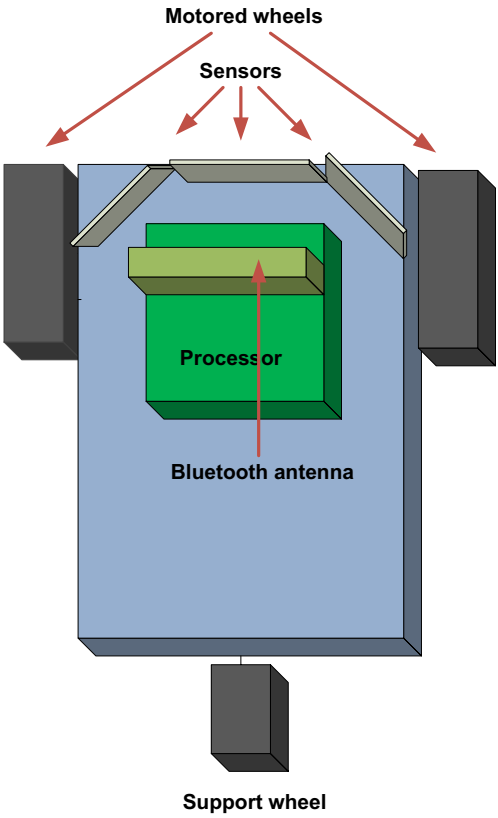

Fig. 2.25 Three-wheeled robot parts

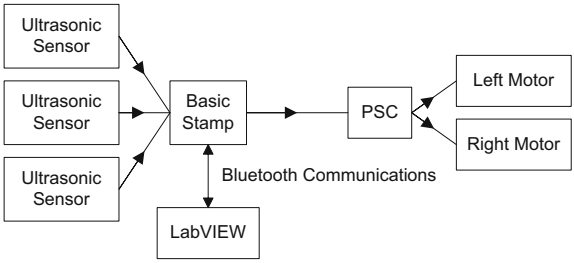

Fig. 2.26 Three-wheeled robot diagram

2.8.1 Fuzzification

We know that the controller for the robot receives three inputs (left, center, and right) and generates two outputs (sLeft and sRight). With this information we can calculate the number of rules, which is $32 = 9$.

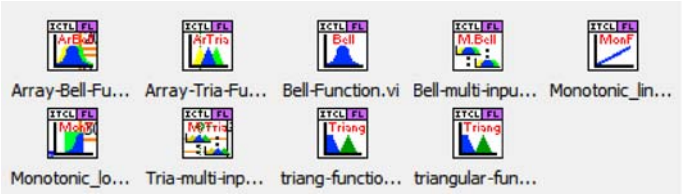

Fig. 2.27 Function VI blocks

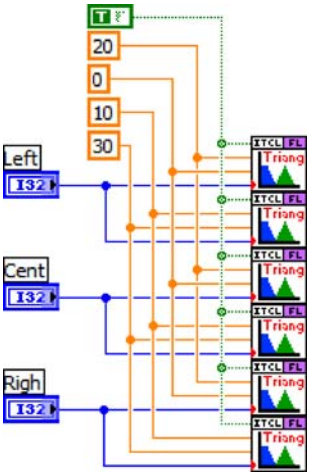

Fig. 2.28 Block diagram of the membership function

Next thing we need to do is generate the fuzzy membership functions for the fuzzification of the inputs; as stated before, we will be using two membership functions with the linguistic values of *close* and *far*. We can choose from different forms for the functions in the fuzzification process, as shown in Fig. 2.27.

For our controller we will choose the triangular function, called **triang-function.vi**, because we will only have two membership functions per input (Fig. 2.28). We will use them as shoulder triangular functions, and because our robot is small and fast we will set the *close* cluster within the limits of 0 to 20 cm, and the *far* cluster limits will be 10 to 30 cm. Because of the programming of the triangular function, the decreasing or increasing of the functions will begin just between the limit where the membership value starts to be zero a and the final limit where it starts to be one b. If $a < b$ the function will be a left shoulder function, otherwise it will be a right shoulder.

Now we need to create a 2D array from the results obtained from the fuzzification process. Figure 2.29 shows the wiring part of the code, and Fig. 2.30 shows the wiring process. Basically for each input an array with their values must be created and then a second array must be created containing all the input information.

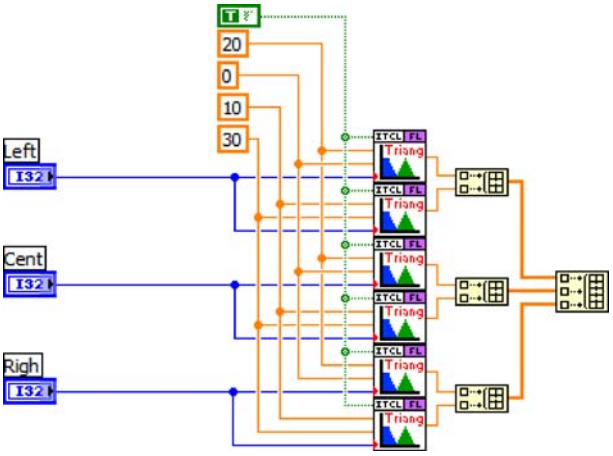

Fig. 2.29 Complete block diagram of the membership function

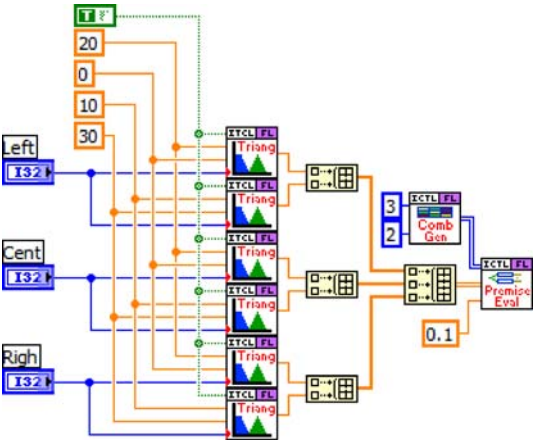

Fig. 2.30 Block diagram until rules evaluation

2.8.2 Rules Evaluation

With this we will have the information of the membership value for *left* in the row of the 2D array, the second row will have the information for *center*, and the third array will have the information for the *right* array. For each row, the first value will be for the *close* set, and the second value will be for the *far* set (Table 2.9). This will be pretty much like creating a table for the evaluation of the inputs. There is a VI that will automatically generate this table for us: it is the input combinatory generator and we must feed it the number of inputs and membership functions.

This way we have generated the premise part of the rule: *IF left* is *close AND center* is *close AND right* is *close, THEN*... Now we need to use the **premise_evaluation.vi** to obtain the min operation for each rule. Sometimes we will want to consider that a rule is activated only if its min value is above a certain level; that is why we have

Table 2.9 Rules base

Input from the sensors		
Left	Center	Right
Close	Close	Close
Close	Close	Far
Close	Far	Close
Close	Far	Far
Far	Close	Close
Far	Close	Far
Far	Far	Close
Far	Far	Far

set the "value" input from the premise VI to 0.1. This part of the code is shown in Fig. 2.30.

Now we need to generate the combinations that will make the robot move around. We have previously established that 2 = *move forward*, 1 = *move backward*, and 0 = *stopped*. With this in mind we can generate a table with the consequences of the rules. In order to obtain the values we must generate 1D arrays for the defuzzification of each output (Table 2.10).

Table 2.10 Rules of outputs for the Takagi–Sugeno and Tsukamoto controllers

	Output for the wheels	
Action	Left wheel	Right wheel
Go backward	2	2
Turn right back	2	0
Go front	1	1
Turn right	2	1
Turn left back	0	2
Go backward	2	2
Turn left	1	2
Go front	1	1

2.8.3 Defuzzification: Crisp Outputs

In order to obtain the crisp outputs for a controller using singleton output functions or a Takagi–Sugeno inference, we must use the **defuzzifier_constants.vi** that will help us obtain the final outputs sLeft and sRight. In case we would like to use equations instead of constants, we would need to evaluate these equations and pass the constant value to the **defuzzifier_constants.vi**; this way we are able to use any kind of equation. Figure 2.31 shows the complete block diagram of Takagi

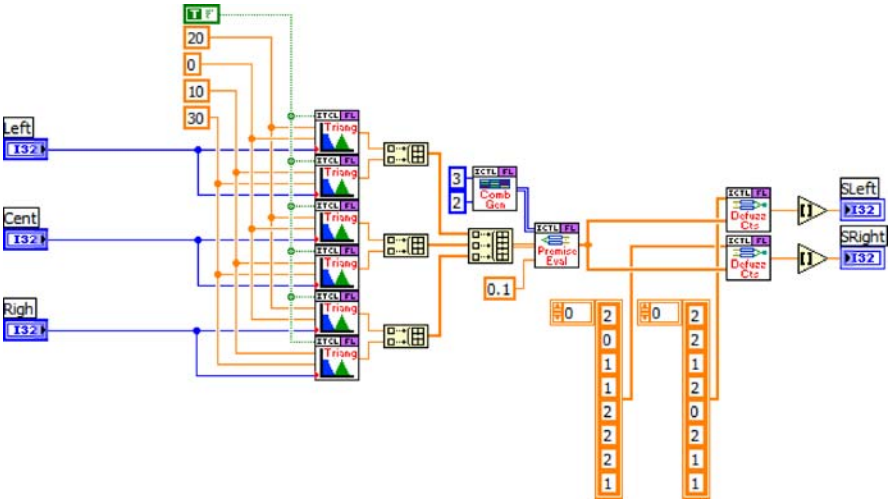

Fig. 2.31 Complete block diagram of the Takagi–Sugeno and Tsukamoto controllers

Table 2.11 Rules set for the Takagi–Sugeno and Tsukamoto controllers

Input from the sensors			Output for the wheels		
Left	Center	Right	Action	Left wheel	Right wheel
Close	Close	Close	Go backward	2	2
Close	Close	Far	Turn right back	2	0
Close	Far	Close	Go front	1	1
Close	Far	Far	Turn right	2	1
Far	Close	Close	Turn left back	0	2
Far	Close	Far	Go backward	2	2
Far	Far	Close	Turn left	1	2
Far	Far	Far	Go front	1	1

and Tsukamoto controllers. This way we have created a controller that will behave according to the set of rules shown in Table 2.11.

If we decide to create a Mamdani controller, the premise part is pretty much the same, only the consequences part will have to change, i.e., instead of using singletons we will use three triangular functions for the defuzzification process called *stopped*, *forward*, and *backward*. We have to use the **general_defuzzifier_mamdani.vi** for each output that we would like to generate. This VI will take the number of membership functions that will be used for defuzzification, the same set created with the desired output for each fired rule and the result of the evaluation of the premises, and it will return the max for each of the desired sets. With this information we will then have to create the defuzzifying functions and pass them their respective max value. Finally we have to combine the results to obtain the desired outputs. Figure 2.32 shows the complete block diagram of the Mamdani controller. This controller will behave according to the set of rules shown in Table 2.12.

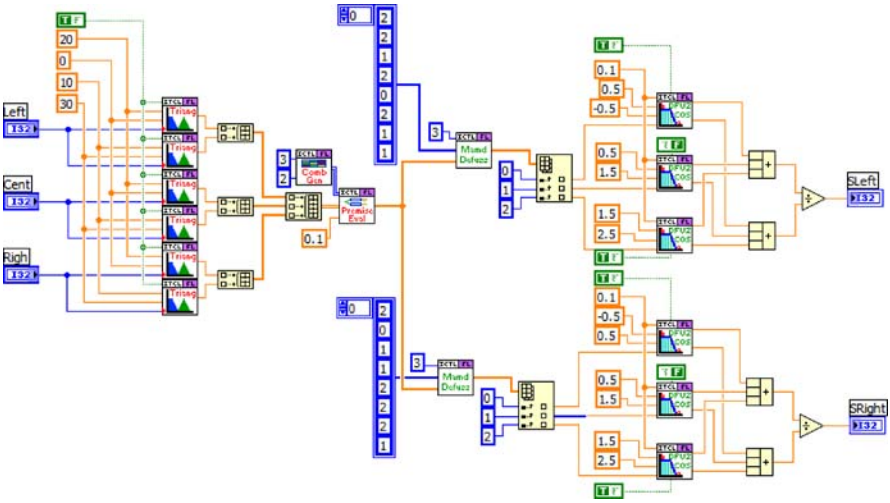

Fig. 2.32 Complete block diagram of a Mamdani controller

Table 2.12 Rules set for the Mamdani controller

Input from the sensors			Output for the wheels		
Left	Center	Right	Action	Left wheel	Right wheel
Close	Close	Close	Go backward	Backward	Backward
Close	Close	Far	Turn right back	Backward	Stopped
Close	Far	Close	Go front	Forward	Forward
Close	Far	Far	Turn right	Backward	Forward
Far	Close	Close	Turn left back	Stopped	Backward
Far	Close	Far	Go backward	Backward	Backward
Far	Far	Close	Turn left	Forward	Backward
Far	Far	Far	Go front	Forward	Forward

2.9 Classical Control Example

Here fuzzy logic is used to control the speed of the discrete model of a direct current motor. A fuzzy proportional derivative controller was used along with the model of a direct current motor whose transfer function in continuous time is shown in (2.24):

$$G(s) = \frac{2.445}{s^2 + 6.846s}. \tag{2.24}$$

The model of the direct current motor was digitalized with a period of 0.1 s and converted to difference equations, then programmed as shown in Fig. 2.33. The diagram for the closed-loop controller is shown in Fig. 2.34.

The error and the difference of the error is calculated and sent to the fuzzy controller that will create the appropriate signal to control the speed reference of the motor. The motor will then react to the control signals of the controller and the speed will be measured and receive feedback.

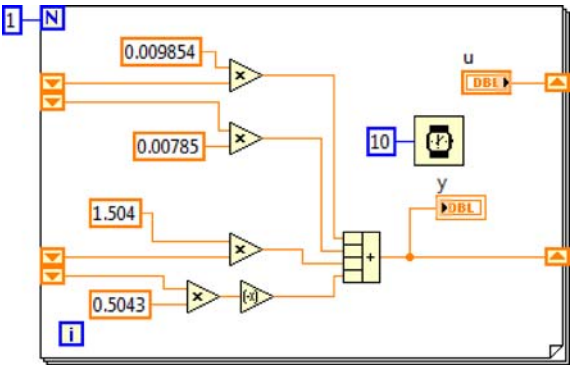

Fig. 2.33 Digital model of the plant in difference equation form

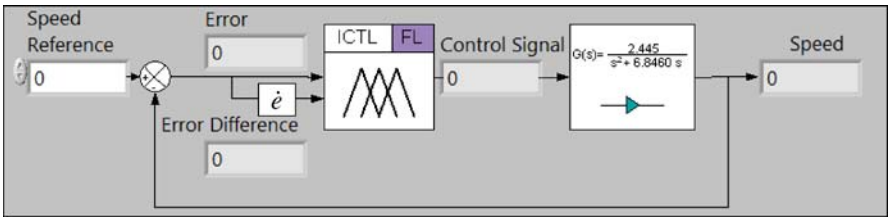

Fig. 2.34 Closed-loop diagram of the controller

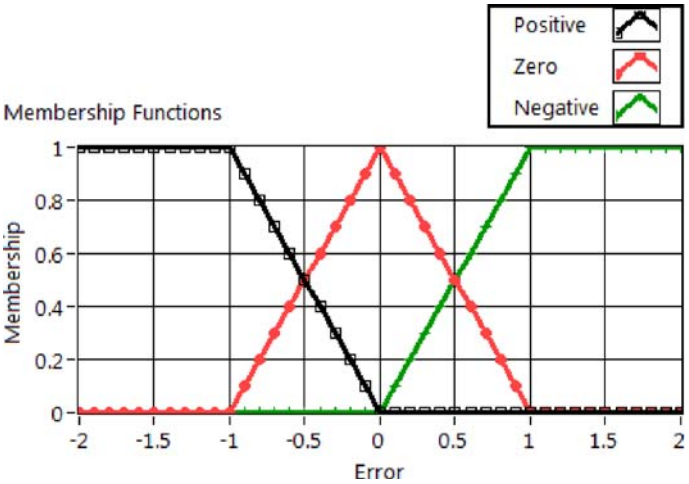

Fig. 2.35 Membership functions for fuzzification and defuzzification

The proportional derivative fuzzy controller is a Mamdani controller with two inputs and one output. Three membership functions for fuzzification and defuzzification are used of triangular form, as shown in Fig. 2.35. The diagram of the fuzzy controller is shown and explained in Fig. 2.36.

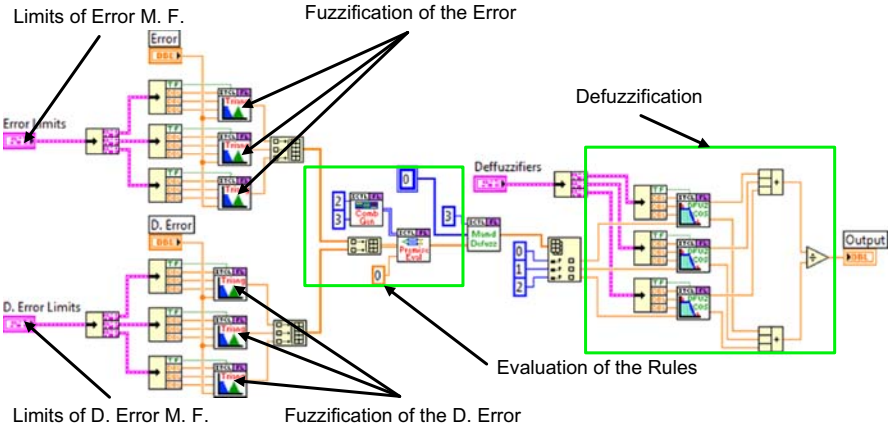

Fig. 2.36 Block diagram of fuzzy controller

The fuzzy controller works like this:

1. The limits of the membership functions are set.
2. Then the error and the difference error are evaluated in three triangular membership functions; their outputs are then gathered in an array to be used.
3. After that the process of evaluation of the rules is complete.
4. Finally three triangular membership functions are used to defuzzify the output of the controller.

The code for the complete program is shown in Fig. 2.37.

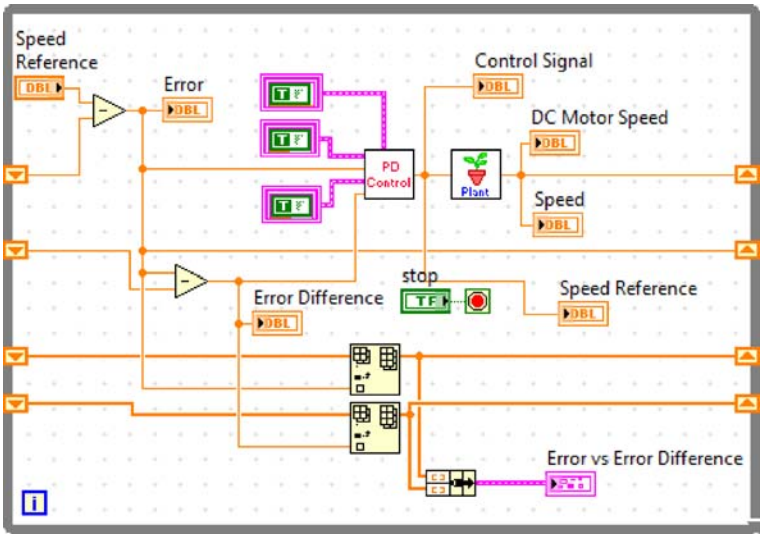

Fig. 2.37 Block diagram of the closed-loop

References

1. Zadeh LA (1973) Outline of a new approach to the analysis of complex systems and decision processes, IEEE Trans Syst Man Cyber SMC 3:28–44
2. Karray FO, de Silva C (2004) Soft computing and intelligent system design. Addison Wesley Longman, Boston
3. von Altrock C (1994) Fuzzy logic technologies in automotive engineering. IEEE Proceedings of WESCON, Anaheim, CA, 27–29 Sept 1994
4. Zadeh LA (1965) Fuzzy sets. Info Control 8(3):338–353
5. Tsoukalas LH, Uhrig RE (1996) Fuzzy and neural approaches in engineering. Wiley, New York

Futher Reading

Guanrong C, Trung P (2000) Introduction to fuzzy sets, fuzzy logic, and fuzzy control systems. CRC, Boca Raton, FL

Hung NT, Elbert W (1999) A first course in fuzzy logic, 2nd edn. CRC, Boca Raton, FL

Timothy RJ (1995) Fuzzy logic with engineering applications. McGraw-Hill, Boston

http://www.springer.com/978-1-84882-683-0

Intelligent Control Systems with LabVIEW™

Ponce-Cruz, P.; Ramírez-Figueroa, F.D.

2010, XIII, 216 p., Hardcover
ISBN: 978-1-84882-683-0