Chapter 2
Fuzzy Logic Systems

2.1 Introduction

The human mind has always shown a remarkable capability of coordinating a wide variety of physical and mental tasks without using any explicit measurements and computations. Considerable efforts were made since the early 1950s towards the development of a scientific theory of intelligence and the development of an artificial model of the brain capable to mimic our perception, cognition and behavioral systems [1, 2]. Despite the accomplishments of system's theory and artificial intelligence, that are increasingly present in our daily activities, in practice computational systems still present several limitations that keep them behind human capabilities. The high dimensionality of information structures stored in computational systems resulting from the use of crisp measurements is one of the major burdens that the development of an intelligence framework must overcome. Uncertainties and imprecisions on information can at first instance be seen as a disadvantage for a decision process but they are important information compression mechanisms that let people make choices in a quick way. Without such tools, taking a decision would be a never ending process, requiring every infinitesimal part of information and its respective combinations to be considered. Therefore, the development of intelligent systems has to focus on the human capability of manipulating imprecise, uncertain and sometimes incomplete information.

Zadeh was challenged by this problem in 1965 and, in his seminal paper [3], he lays the foundation-stone of a methodology known as FL, where the objects of computation are words and propositions drawn from natural language. While Boolean logic results are restricted to values "0 and 1", FL defines for the first time a computational framework to efficiently manipulate intermediate results between the values of absolute true and absolute false. The fuzzy information representation is based on Fuzzy Sets, which are no more than a simple way to translate a crisp measurement into a degree of belonging in a linguistic label. This means that fuzzy

R. Antão, *Type-2 Fuzzy Logic*, Nonlinear Physical Science,
DOI 10.1007/978-981-10-4633-9_2

sets can handle some concepts that we commonly deal with in daily life, like "very cold", "cold", "hot", "very hot", without having to know the specific temperature ranges each concept refers to. Therefore, Fuzzy Logic is more like human thinking because of its reliance on degrees of truth and the use of linguistic variables.

Initially, FL theory was not well-received by the peer community in engineering domains due to its unusual vagueness. Nonetheless, since 1970, it has been widely applied in control applications, establishing successive milestones. Its principles were used to control a laboratory-built steam engine by Mamdani at the University of London in 1974 [4] and the first industrial application was a cement kiln controller built in Denmark in 1979 [5]. Despite born in the USA and theoretically validated in Europe, it was in Japan that FL gained broad notoriety when several Japanese companies pioneered successful practical applications with high impact in the society. One of the most renowned projects was presented in 1987, when Hitachi turned over control of a subway in Sendai, Japan, to a fuzzy system (Fig. 2.1). Fuzzy control techniques were used in all the critical operations of the train's control system, such as accelerating, breaking, and stopping operations [6] but also in traffic planning and predicting customer's usage of subway facilities. In 1987, Yamakawa successfully developed a fuzzy controller applied to a inverted pendulum experiment—a classic control problem [7]. A few years later, NASA took fuzzy logic beyond our planet aboard the Endeavour space shuttle, transporting a Commercial Refrigerator Incubator Module as an experimental payload, which successfully allowed the control of a test chamber's air temperature according to a pre-programmed profile [8]. Since then, several companies have been using fuzzy logic to control hundreds of household appliances, implement decision making systems and improve the performance of many other electronic devices present in our daily life such as air conditioners, video cameras, televisions, washing machines, bus time tables, medical diagnoses or anti-lock braking systems.

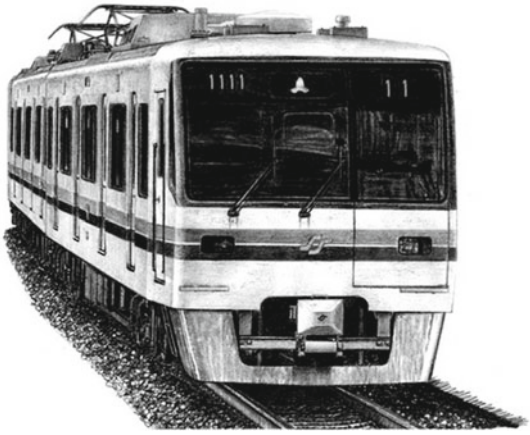


Fig. 2.1 Sendai subway 1000 Series—The first subway coach using a fuzzy control system

Fuzzy Logic Systems are typically developed around two main types of inference frameworks: the Mamdani [9] and the Takagi-Sugeno [10] methods. Despite the differences between the two methods, when non-linear Fuzzy Sets are used to model linguistic labels, FLSs become non-linear structures with universal approximation capabilities [11], a property of major importance when they are used as support for modeling and control techniques. Since both inference mechanisms share several theoretical principles, this chapter will firstly introduce FL's fundamentals based on the Mamdani inference.

2.2 Type-1 Fuzzy Sets

With the development of FL, the Type-1 FSs were defined for the first time. Type-1 FSs are a computational formalism that mimics our tendency to group crisp measurements displayed under a numeric scale using the same linguistic term when a more specific distinction is not required for a good understanding. For example, we describe temperatures using a linguistic terms that go from "Very Cold" to "Very Hot", speed using terms from "Very Slow" to "Very Fast" or the visible colors from "Violet" to "Red". With these descriptions, we are capable of abstain ourselves from the crispness of numeric scales expressed in °C, km/h or nm. Each Type-1 Fs is syntactically represented by a label F_i characterized by a Membership Function (MF), a two-dimensional function that defines the degree of association of a numeric value under the respective linguistic label using a crisp number in the range [0–1]. Different shapes of MFs can be considered, namely triangular, trapezoidal or gaussian shaped functions. Figure 2.2 depicts an example of a generic input domain partitioned using gaussian shaped MFs.

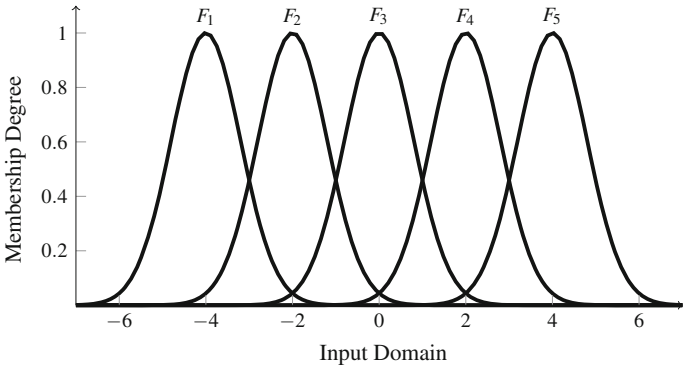


Fig. 2.2 Generic input domain partition using Type-1 Fuzzy Sets

2.3 Type-1 Fuzzy Logic Systems

One of the most appreciated methods of manipulating linguistic information is to use FLSs based on *If-Then* rules, a method that can be easily used to develop models and control algorithms in a way closer to human perception and thinking. There exist also alternatives to the use of rule based systems [12] involving arithmetic approaches using the Extension Principle [13]. To enunciate the required logic statements, this principle redefines common algebraic operations such as addition, multiplication, among many others to the domain of Fuzzy Sets. Sometimes it is hard to define *If-Then* rules using compact algebraic operations. Nevertheless, this approach is particularly useful in situations where the problem has a high dimensionality, i.e. the number of existing rules used to describe the system is so high that it may result in a computationally inefficient process.

A FLS based on Type-1 FSs consists of four main elements, as depicted in Fig. 2.3 and briefly described as follows.

- The Fuzzifier, which is an interface which maps a crisp number into a fuzzy domain defined by a Fuzzy Set. The most widely method is the *singleton fuzzifier*, in which measurements are considered perfect and therefore modeled as crisp values, i.e., as singletons.
- The Rule-Base, which is the heart of a FLS and is composed by information given by experts or extracted from numerical data, is often organized as several *If-Then* statements, where the *If* part of a rule is its antecedent, and the *Then* part of the rule is its consequent.
- The Inference Engine, which is the mechanism that implements the algebras required to manipulate Fuzzy Sets. In the same way humans use many inferential procedures, there exist several methods to do so based on FL. The Mamdani and the Takagi-Sugeno inference mechanisms are the two most popular ones [14].
- The Output Processor, which is the final stage of the FLS and implements the defuzzification procedures to aggregate the output fuzzy set into a single crisp value adequate to the FLS application scenario (usually a process's output prediction in modeling applications or the actuation value of a control system).

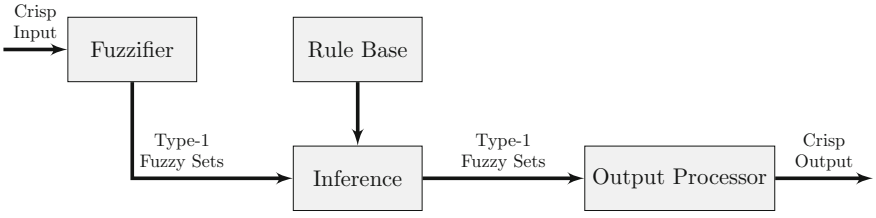


Fig. 2.3 Type-1 fuzzy Logic System structure

2.3.1 Fuzzifier

As pointed out previously, the singleton fuzzifier is the most used method to implement the Fuzzifier stage of a FLS due to its conceptual simplicity and easy data manipulation in the subsequent stages of the FLS. Other functions could be used to perform this operation, as gaussian or triangular functions, but then calculating the firing levels of each antecedent membership function would become a far more complex process [13]. For this reason, singletons are considered in this work, and defined as:

$$\mu_{A_x}(x) = \begin{cases} 1, & x = x' \\ 0, & \text{otherwise} \end{cases}, \tag{2.1}$$

where x' is the input value. This concept is depicted in Fig. 2.4.

2.3.2 Rule-Base

Rules play a central role in the structure of a FLS and are a simple way to gather the knowledge that define the behavior of a fuzzy system in a specific application. These rules are developed using different types of FSs, that are associated with the linguistic terms that appear in the rule's antecedent and consequent parts and are interconnected by operators that establish the relationship dependencies between fuzzy terms.

The most used rule structure is presented in Eq. (2.2)

$$R^i : If\ x_1\ is\ F_1^i\ and\ \cdots\ and\ x_j\ is\ F_j^i,\ Then\ y^i\ is\ G^i\ , \tag{2.2}$$

where R^i represents the i^{th} fuzzy rule, F_j^l and G^i are linguistic terms characterized by Type-1 Fuzzy Sets, $i = \{1, \cdots, M\}$ where M is the number of fuzzy rules,

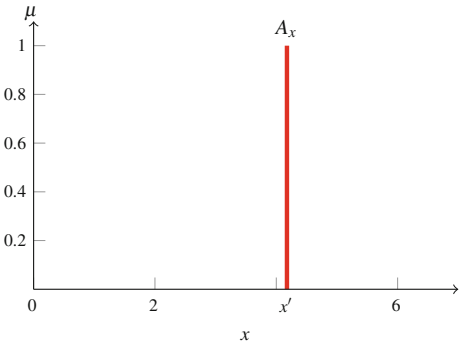


Fig. 2.4 Depiction of a singleton input

$j = \{1, \cdots, N\}$ where N is the number of antecedents, x_j are the fuzzy system inputs and y^i is the rule output.

The linguistic terms F and G can assume several different shapes such as triangular, trapezoidal or gaussian. The latter form was employed in this work, being defined as presented in Eq. (2.3).

$$g(c, \sigma, x) = \exp\left[-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2\right] \tag{2.3}$$

The rule structure (2.2) is just one example of many prospective ways to embed knowledge in a FLS. Similarly to our natural language, one can employ other combinations of FSs using for example *or* relationships, consider the negation of fuzzy sets, or even using *non-obvious* connectives like *unless* or comparative terms [13]. Though, in most scenarios, such logical statements can be represented using the more regular structure of Eq. (2.2).

2.3.3 Inference Engine

The manipulation of fuzzy sets can be performed according to several different algebraic operations depending on the possible combinations of operators used for implementing the rule's connective terms, implication methods and rule aggregation. Let us consider two Type-1 FS F_1 and F_2 characterized by the MFs $\mu_{F_1}(x)$ and $\mu_{F_2}(x)$

$$F_1 = \int_{x \in \mathbb{R}} \mu_{F_1}(x)dx \ , \qquad F_2 = \int_{x \in \mathbb{R}} \mu_{F_2}(x)dx \ . \tag{2.4}$$

The basic logic operations (union, (*s-norm*), intersection (*t-norm*) and complement (*c-norm*)) that provide the support for FS's manipulation can be defined as follows:

$$\begin{aligned} \mu_{F_1 \cup F_2}(x) &= max\left[\mu_{F_1}(x), \mu_{F_2}(x)\right], \ for\ x \in \mathbb{R} \\ \mu_{F_1 \cap F_2}(x) &= min\left[\mu_{F_1}(x), \mu_{F_2}(x)\right] , \\ \mu_{\overline{F}}(x) &= 1 - \mu_F(x) \ . \end{aligned} \tag{2.5}$$

The intersection operator can also be implemented based on the algebraic product and is defined as:

$$\mu_{F_1 \cap F_2}(x) = \mu_{F_1}(x) * \mu_{F_2}(x), \ x \in \mathbb{R} \ . \tag{2.6}$$

Since the usual way of constructing a rule is to use the *and* connectives, the *t-norm* operators are the most used ones. Regardless the implementation followed, since the membership grades $\mu_{F_1}(x)$ and $\mu_{F_2}(x)$ are crisp numbers, any of the operations presented in Eqs. (2.5) and (2.6) yields a crisp number. More particularly, when these

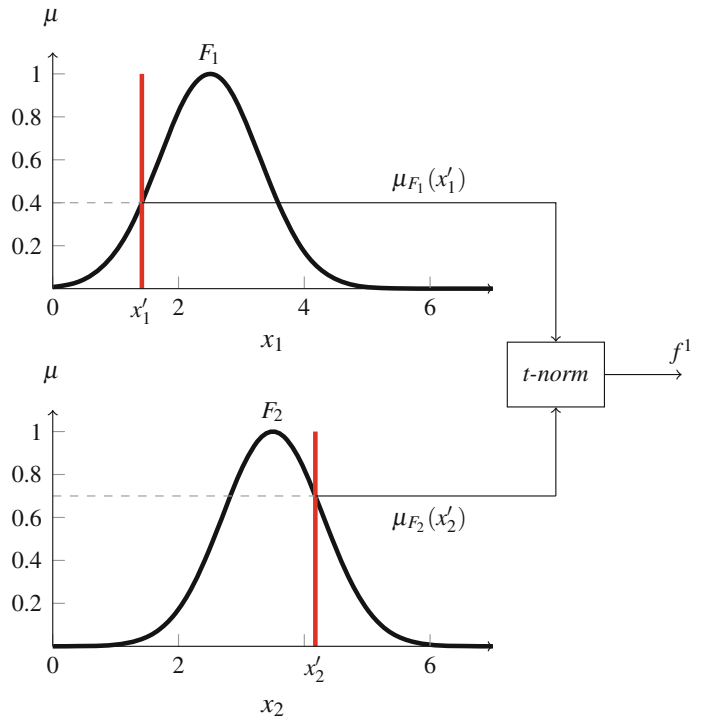


Fig. 2.5 Operation between singleton input and the antecedents of a Type-1 FLS using a *t-norm* operator (minimum or product)

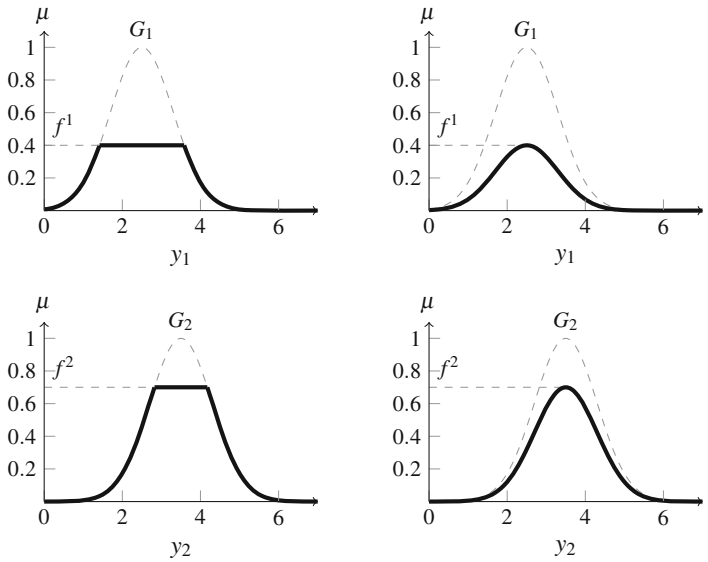
operators are used to aggregate several fired FSs from the antecedent part of the rule, the obtained result is typically referred to as the i^{th} rule firing level f^i. Figure 2.5 depicts the use of the *t-norm* in a two antecedent operation.

When both rule's antecedents and consequent are expressed using Type-1 FS, the implication of the rule's firing level over the consequent FS is typically obtained using one of the Mamdani implication methods, namely the Mamdani minimum or the Mamdani product. These methods are based on the *t-norm* operators previously described and are applied between the rule's firing level f^i and its consequent FS, $G^i(x)$

$$\begin{aligned} f^i \rightarrow G^i(y) &= min\{f^i, \mu_{G^i}(y)\}, \quad for\ y \in \mathbb{R} \\ f^i \rightarrow G^i(y) &= f^i \mu_{G^i}(y)\ . \end{aligned} \tag{2.7}$$

Depending whether the minimum or the product *t-norm* is used, one obtains a clipped or a scaled version of the consequent MF, as depicted in Fig. 2.6.

The inference process ends up by obtaining a fuzzy set determined by the aggregation of the output of all the fired FLSs rules. One of the most used methods to do so is to use a *t-conorm*—the fuzzy union, which is no more than a method for finding the maximum value of the overlapped FSs. This is depicted in Fig. 2.7.



(a) Fired output sets using the minimum *t-norm*.

(b) Fired output sets using the product *t-norm*.

Fig. 2.6 Mamdani inference operations using Type-1 FSs

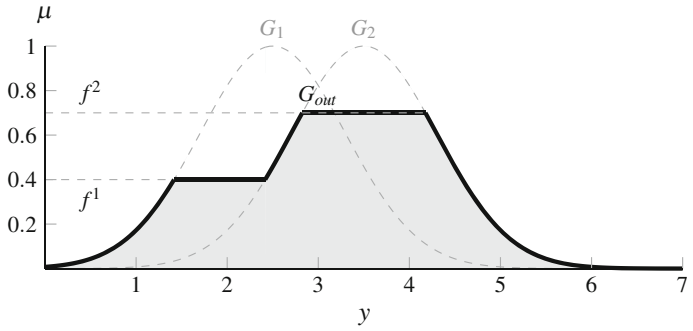


Fig. 2.7 Type-1 Fuzzy Set fired consequents' aggregation procedure, after using the mamdani minimum implication

Still, this final step is not consensual among authors, existing some that give preference to aggregate the output of every rule before defuzzification while others perform the aggregation as part of the defuzzification procedure. Given this fact there exist several different methods to implement FLS defuzzification stage, as will be following discussed.

2.3.4 Output Processor

The Output Processor is employed to obtain a crisp output from the FS resulting from the inference procedure, a process known as defuzzification. The literature is rich in defuzzifiers methods but, considering applications in modeling and control domains, their computational efficiency is often the main exclusion criterion. From the available methods, the Centroid and the Center-of-Sets defuzzifiers are the most frequently used ones [13], and represent good examples of the two different aggregation/defuzzification approaches.

The Centroid defuzzifier obtains a crisp value by finding the centroid of a Type-1 FS resulting from the union of the consequent fuzzy sets. By sampling the resulting G_{out} FS into K points, as depicted in Fig. 2.8, its centroid is given by:

$$y_C = \frac{\sum_{i=1}^{K} y_i \mu_{G_{out}}(y_i)}{\sum_{i=1}^{K} \mu_{G_{out}}(y_i)} . \tag{2.8}$$

Alternatively, the Center-of-Sets defuzzifier does not rely on prior aggregation of every fired consequent FS. Instead, it replaces every rule's consequent Type-1 FS by a singleton placed at its centroid which amplitude is given by the respective rule's firing level. The defuzzifier output value is then obtained by calculating the centroid of the Type-1 FS comprised of these singletons, and is given by:

$$y_{COS} = \frac{\sum_{i=1}^{M} c^i f^i}{\sum_{i=1}^{M} f^i} , \tag{2.9}$$

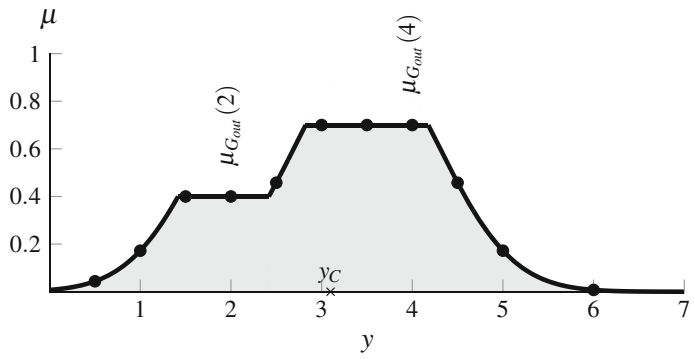


Fig. 2.8 Defuzzification procedure based on the centroid method applied to the G_{out} fuzzy set

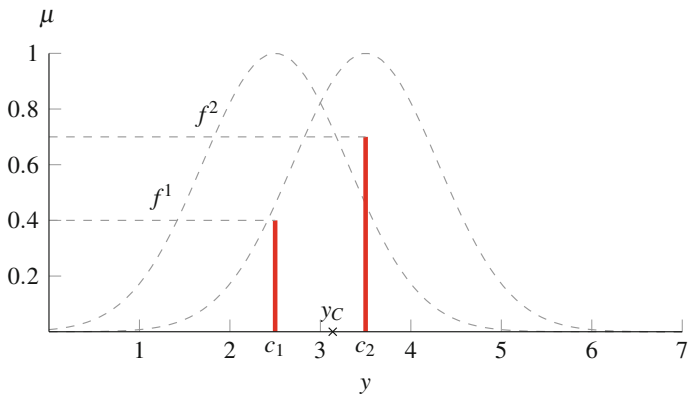


Fig. 2.9 Defuzzification procedure based on the Center-of-Sets method, using the centroid of each fired consequent fuzzy sets separately

where c^i is the centroid of the i^th consequent FS, M is the number of rules of the FLS and f_i is each rule's firing level. This procedure is depicted in Fig. 2.9.

The latter approach is usually preferred when implemented in computationally constrained systems since the centroid of each consequent's FS can be calculated *a priori* and stored in the system, before the fuzzy system is deployed. After performing this step, the FLS's output is obtained as a weighted average of the pre-calculated centroids.

2.3.5 Considerations About Type-1 Fuzzy Logic Systems

The importance of the additive structure that a Fuzzy Logic System presents goes far beyond its rules' intelligibility. By using several rules, one is in fact defining several fuzzy patches and average the ones that overlap, ultimately performing a multi-variable function approximation. Such procedure's accuracy improves as the fuzzy patches grow in number and shrink in size.

The Type-1 FSs have been found to provide good results in uncertainty modeling, but there are several opinions referring that using them as a model for a linguistic label is an incorrect scientific theory [15]. As is pointed out in Jerry Mendel's line of reasoning [15], it is easy to understand the reasons for this refutation:

- A Type-1 FS representation for a word is well-defined by its Membership Function that is totally certain once all of its parameters are specified.
- Words mean different things to different people and so are uncertain.
- Therefore, it is a contradiction to say that something that is certain can model something that is uncertain.

The same way the variance provides a measure of dispersion about the mean, the uncertainty of a linguistic term also needs to be captured, and is not possible to represent such when a single static MF is used. In Fuzzy Set's theory, this second order uncertainty can be modeled by Type-2 Fuzzy Sets.

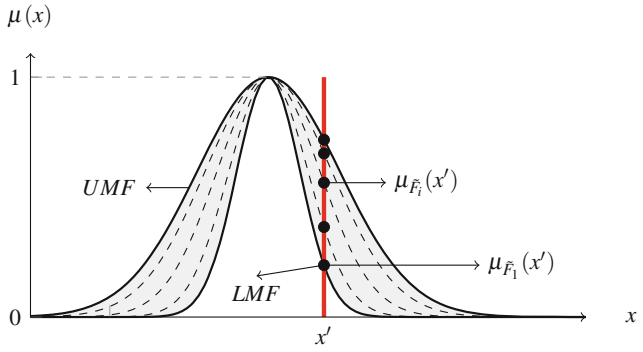
2.4 Type-2 Fuzzy Sets

The concept of Type-2 FSs was developed by Zadeh in 1975 as an extension of Type-1 FSs [16], but it only gained broader audience much more recently with the several developments proposed by Mendel and Karnik [17]. Type-1 FSs introduced an important degree of fuzziness to create a linguistic partition of a crisp domain. Nonetheless, the MFs used to do so are themselves crisp since they are totally defined without considering any uncertainty on their parameters. Type-2 FS overcome this limitation by defining a secondary degree of fuzziness. In this case, the membership value for each input of a FS is itself defined as a FS in the [0, 1] domain [13]. To better understand this new dimensionality, suppose the process of defining a concept as a Type-1 FS by polling a group of experts. After gathering all the responses, certainly it will be noticed that the endpoints of the membership function will vary from person to person. The union of all embedded Type-1 FSs eventually will end up in a blurred area, known as FOU, that is bounded by two MFs, namely the Upper Membership Function (UMF) and the Lower Membership Function (LMF). Furthermore, each membership function given by a person can be assigned with a variable weight according to the amount of confidence associated to its opinion, defining this way the secondary degree of fuzziness. For this reason, a Type-2 FS representation embeds additional degrees of freedom which can better handle uncertainties caused by noisy data and changing environments as is required for example when developing a process's model. Figure 2.10 gives a better overview of the new concepts introduced by Type-2 FSs, which can be generically represented by:

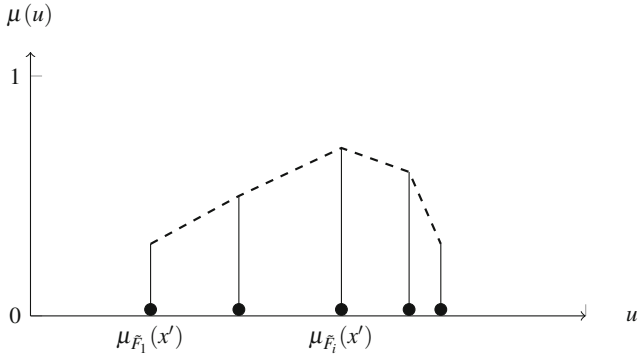
$$\tilde{F} = \int_{u \in X} \mu(u)du = \int_{u \in X} \int_{x \in J_x} g(x)dxdu \qquad J_x \subseteq \Re,\, X \subseteq [0, 1]\,, \tag{2.10}$$

where the $g(x)$ is one of the possible primary MFs, x is the FS input value and u is the primary membership degree.

Until the late nineties, research work on Type-2 FSs was of highly mathematical and theoretical nature, having few publications dedicated to it [18]. The main investigation line was focused on the development of logical operators, with important works from Mizumoto and Tanaka [19], Dubois and Prade [20] and more recently Karnik and Mendel [17]. Another important topic that has received little attention in the literature is the process of acquisition of the Type-2 FSs' membership functions. From the few works published about this topic, Turksen [21] proposes that a



(a) FOU of a Type-2 Fuzzy Set, evincing several embedded Fuzzy Sets.



(b) Vertical slice over the FOU, evincing the variable secondary membership value of each embedded Fuzzy Set.

Fig. 2.10 Representation of a Type-2 Fuzzy Set

Type-2 FS representation can be constructed with the mean and standard deviations of scatter points obtained from surveys. More recently Wagner and Hagras [22] proposed a recursive algorithm to define an optimal approximation of the second degree membership function based on the collected data histogram for the cases of linguistic variables and noisy sensor measurements. The representation and manipulation algebras of Type-2 FSs are also non-closed problems, existing some recent proposals such as [23–25] that introduce simplifications in the Type-2 FSs's representation while maintaining the uncertainty in information representation over the inference stages.

2.5 Type-2 Fuzzy Logic Systems

The success of Type-1 FLSs naturally led to the development of FLSs based on Type-2 FSs. The structure of a Type-2 FLSs shares the same core components of its Type-1 counterpart, namely: a Fuzzifier, a Rule-Base, an Inference Engine and an Output Processor. While in Type-1 FLSs the final stage resumes to a defuzzification procedure, in the Type-2 case the Output Processor embraces an additional stage so that Type-2 FS is firstly converted into an equivalent Type-1 FS. This procedure is implemented by a Type-Reduction (TR) algorithm, which will be presented further in this document. The interdependency of the referred blocks is depicted in Fig. 2.11.

Type-2 FSs can be used either on antecedent, consequent or both levels of the Type-2 FLS, depending on whether is advantageous to account for uncertainties at the referred parts of the rule. As a consequence of its additional degrees of freedom, it has been argued that Type-2 FLSs have a great potential to produce better performing systems. The main reasons for this statement are the following:

- Given the fact that a Type-2 FS embeds itself a large number of Type-1 FS under the same label, it is possible to cover the same range of operation of a Type-1 FS with a smaller number of labels and rules, reducing the complexity of modeling, tuning and understanding a rule-base system comparatively to a similar performing Type-1 FS. This rule reduction capability is particularly advantageous in situations when the number of system inputs tend to increase, as it reduces the number of possible combinations of the linguistic labels that describe each input.
- In a Type-2 FLS, since each input and output is indirectly represented by a large number of Type-1 FSs, more complex input/output relationships that could not be obtained with in a Type-1 FLS can now be modeled without necessarily increase the number of rules [26].

While the additional degrees of freedom of Type-2 FSs revealed a considerable potential of supplanting conventional methodologies [23, 25], especially in complex non-linear modeling tasks, generally their use calls for a greater computational effort. Since Type-2 FS membership degrees are given as a function of a Type-1 FS, the formalisms of elementary fuzzy computations such as the union, intersection and

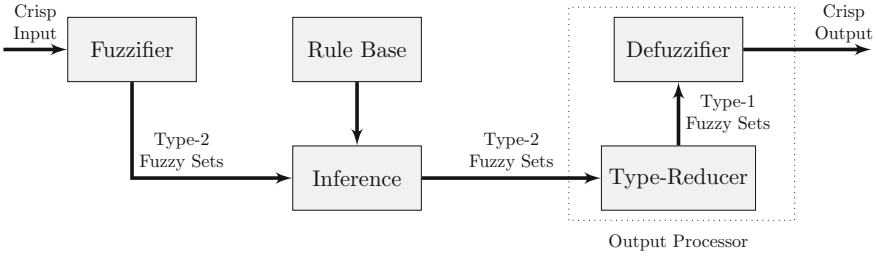


Fig. 2.11 Type-2 Fuzzy Logic System structure

complement are performed in a three-dimensional space, requiring far more complex procedures based on Zadeh's Extension Principle to implement such algebras. Moreover, the accuracy of the TR procedure depends on the number of discretization points of the FS input domain, which naturally is as better as many evaluation points are used - but the computational complexity also increases likewise. For that reason, a great amount of research in Type-2 Fuzzy Logic domains has been put towards developing more efficient representations in order to overcome this bottleneck and further extend its applicability to scenarios where the available computational resources are insufficient to cope with the time constants of the application scenarios.

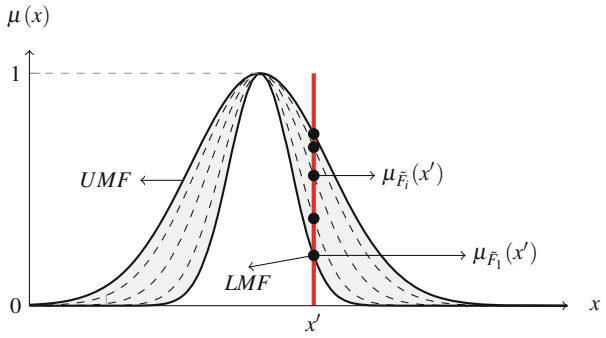
In the past decade, Type-2 FSs' theory publication rate increased significantly, putting stronger efforts towards the reduction of its theoretical complexity and inherent computational effort - the two main problems that kept this uncertainty modeling tool away from real world applications until recent years. Shortly after its first publications, most of the authors focused on a simplified representation known as Interval Type-2 Fuzzy Sets (IT2FSs). In a IT2FS, the MFs uncertainty is restricted to the FOU and considers the third dimension of the Fuzzy Set as uniformly distributed with a membership value "1" [18]. As so, each primary membership is associated with the same third dimension, and each fuzzy set is characterized solely by its LMF and UMF. This concept is depicted in Fig. 2.12.

The Interval Type-2 FS can also be represented based on triangular, gaussian, trapezoidal or sigmoidal MFs. However, while one can define an arbitrary FOU as a piecewise function, the use of the referred classical shapes simplify further model adjustments in training procedures. For this reason, the literature mostly uses gaussian MFs since their FOU can be modeled by varying their mean and standard deviation, as follows:

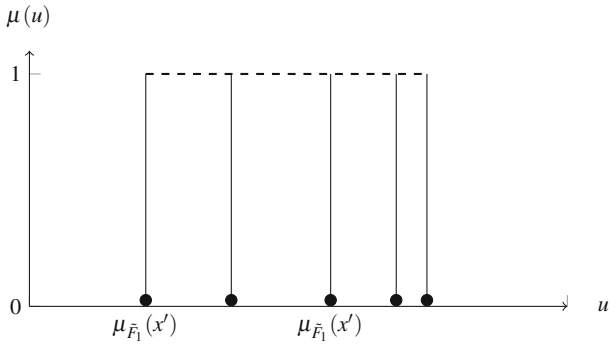
$$\begin{aligned}\tilde{F}_i(c_i, \sigma_i, x) &= \exp\left[-\frac{1}{2}\left(\frac{x - c_i}{\sigma_i}\right)^2\right] \\ &= G(c_i, \sigma_i, x),\ \sigma_j^i \in [\sigma 1_i, \sigma 2_i]\ and\ c_i \in [c1_i, c2_i]\ .\end{aligned} \tag{2.11}$$

Figure 2.13 illustrates the resulting Type-2 FS by varying each one of the referred parameters individually.

The use of an interval based representation significantly reduced the complexity of all the calculations required in the FLSs and, for that reason, turned Interval Type-2 Fuzzy Logic Systems (IT2FLSs) feasible in practical scenarios. Despite the changes in the nature of the MFs, the basic principles of fuzzy logic remain valid and, consequently, IT2FSs' manipulation procedures are very similar to the ones already presented regarding its Type-1 counterpart. Following, a brief analysis of the Type-2 FLS based on the Mamdani inference will be performed assuming that both antecedent and consequent FSs are of Type-2 nature.

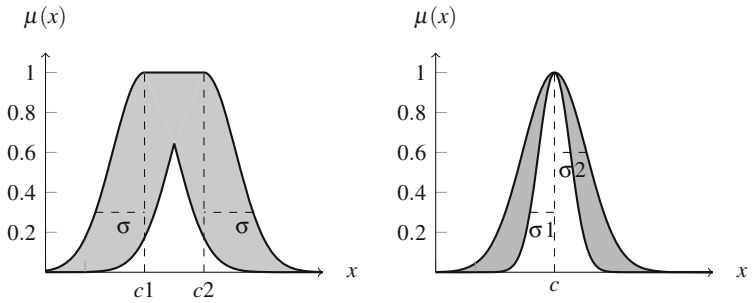


(a) FOU of an Interval Type-2 Fuzzy Set, evincing several embedded Fuzzy Sets.



(b) Vertical slice over the FOU, evincing the constant secondary membership value of each embedded Fuzzy Set.

Fig. 2.12 Representation of an interval Type-2 Fuzzy Set



(a) Gaussian MF with uncertain mean.

(b) Gaussian MF with uncertain standard deviation.

Fig. 2.13 Two possible representations of an interval Type-2 Fuzzy Set based on gaussian membership functions

2.5.1 Fuzzifier

Similarly to the Type-1 FLS, the simplest way of implementing the fuzzifier of a Type-2 FLS is to map a crisp input into a Singleton FS, as defined in Eq. (2.12). While information uncertainty is not explicitly considered in the fuzzification stage, it is indirectly accounted for in the rule's FSs representation

$$\mu_{\tilde{A}_x}(x) = \begin{cases} 1, & x = x' \\ 0, & \text{otherwise} \end{cases} \tag{2.12}$$

where x' is the system's input value.

2.5.2 Rule-Base

As a natural extension of Type-1 FLS, the Type-2 FLSs also synthesize their Rule-Base in a set of *If-Then* rules, establishing the relations between the system's input and output. Regardless the Fuzzy Sets nature, the procedure how rules are created remains the same. Therefore, a Type-2 FLS rule is represented as follows:

$$R^i : If\ x_1\ is\ \tilde{F}_1^i\ and\ \cdots\ and\ x_j\ is\ \tilde{F}_j^i,\ Then\ y^i\ is\ \tilde{G}^i\ , \tag{2.13}$$

where R^i represents the i^{th} fuzzy rule, $\tilde{F}_j^i$ and $\tilde{G}^i$ are linguistic terms characterized by Interval Type-2 FSs, $i = \{1, \cdots, M\}$ where M is the number of rules, $j = \{1, \cdots, N\}$ where N is the number of antecedents, x_j are the FLS inputs and y^i is the rule output.

2.5.3 Inference Engine

The main difference between a Type-1 FLS and a Type-2 FLS lies in their inference engine. In Sect. 2.2, it was concluded that the result of the j^{th} input and corresponding antecedent operations in the i^{th} rule yields a crisp number (μ_j^i) referred as membership degree. In an IT2FS the result of this operation is an interval given by $\tilde{\mu}_j^i$ as follows:

$$\tilde{\mu}_{F_j^i}(x_j) = \left[\underline{\mu}_{\tilde{F}_j^i}(x_j), \overline{\mu}_{\tilde{F}_j^i}(x_j) \right] \tag{2.14}$$

where x_j is the j^{th} FLS system input.

Despite the apparent complexity of this result, an interval based representation allows the direct use of the basic fuzzy logic operations (union (*s-norm*), intersection (*t-norm*) and complement (*c-norm*)) as previously defined in Eqs. (2.5) and (2.6) by

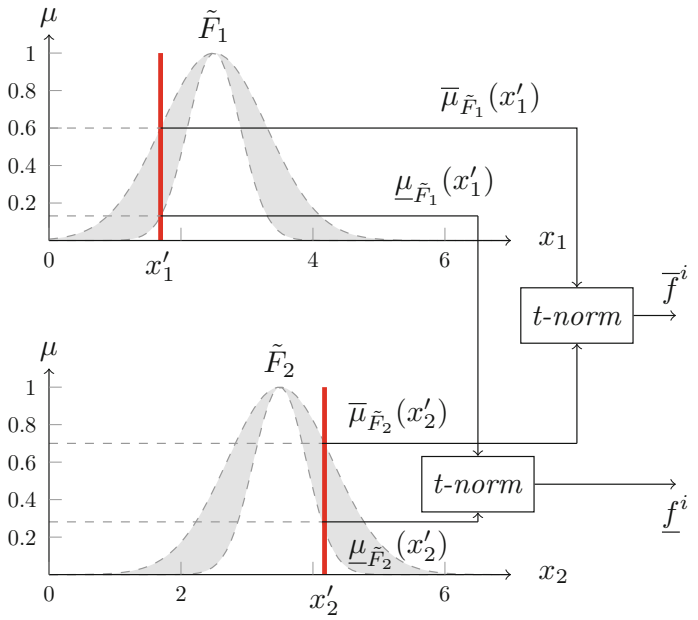


Fig. 2.14 Representation of the operation between singleton input and the antecedents of a Type-2 FLS using a *t-norm* operator (minimum or product)

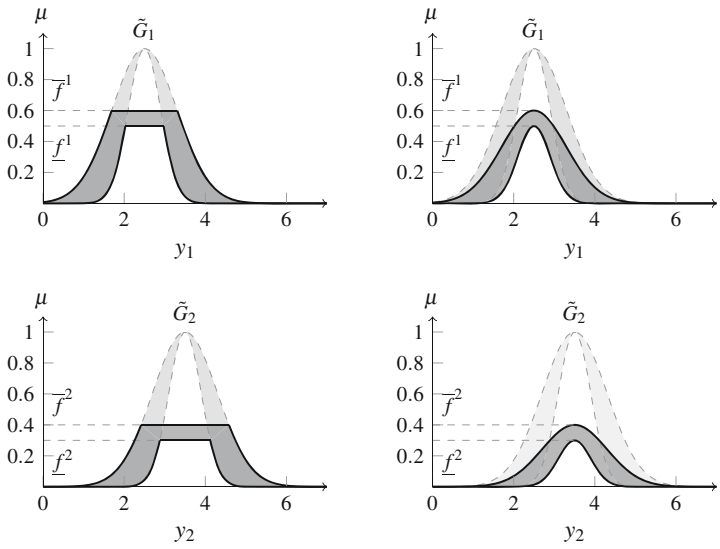
considering the upper and lower bounds of the IT2FS separately. As so, the *t-norm* operator, which is used to perform the intersection of the antecedent FS is defined as:

$$\underline{f}^i = T_{j=1}^{N} \underline{\mu}_{\tilde{F}_j^i}(x_j) \qquad \overline{f}^i = T_{j=1}^{N} \overline{\mu}_{\tilde{F}_j^i}(x_j) \tag{2.15}$$

where T is a *t-norm* (product or minimum). The result of input and antecedent operations (for the minimum and product *t-norm*) is depicted in Fig. 2.14.

Similarly, the Mamdani implication methods (the Mamdani's minimum and product) can be directly used with IT2FS by applying the *t-norm* operator to the rule's firing level $\tilde{f}^i$ and the consequent $\tilde{G}_i$. This procedure is performed by considering the upper and lower bounds of $\tilde{f}^i$ and $\tilde{G}_i$ separately, as presented in Fig. 2.15 for the minimum and product *t-norms*.

The inference process yields a FS determined by the aggregation of the output of all the fired fuzzy sets. Similarly to the Type-1 FLS case, one can merge the contribution of each rule by finding the maximum value of the overlapped FSs, as depicted in Fig. 2.16. To obtain a crisp output after this procedure, one will have to apply a TR algorithm firstly, as will be discussed in the following subsection.



(a) Fired output sets using the minimum *t-norm*.

(b) Fired output sets using the product *t-norm*.

Fig. 2.15 Mamdani inference operations using Type-2 FSs

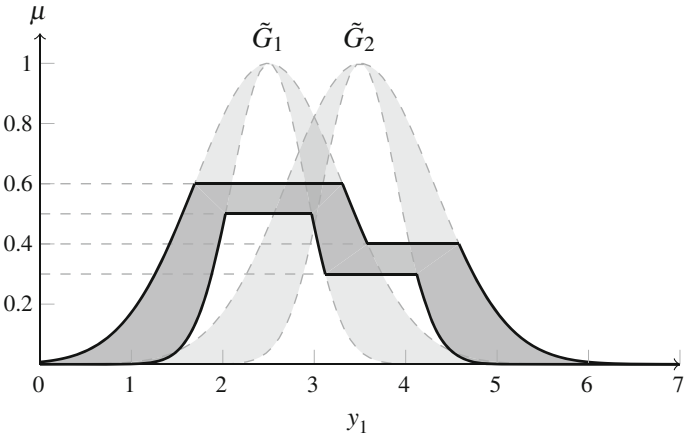


Fig. 2.16 Type-2 Fuzzy Sets fired consequents' aggregation procedure, after using the Mamdani minimum implication

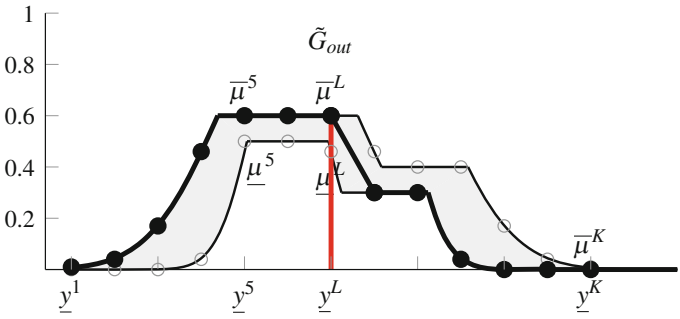
2.5.4 Type-Reduction

In order to develop practical applications based on the Type-2 FL, it becomes necessary to obtain a crisp value from the combination of all fired FS. To accomplish this goal, it is a prerequisit to obtain the centroid of a Type-2 FS, represented as an interval often referred to as type-reduced set. The Karnik-Mendel (KM) algorithm [27], which can be seen as an extension of Type-1 defuzzification procedure, is currently the most accurate TR method found in literature. Though, given its iterative nature, it is the most complex stage of the fuzzy inference process, requiring extensive calculations even when the simpler IT2FSs are used.

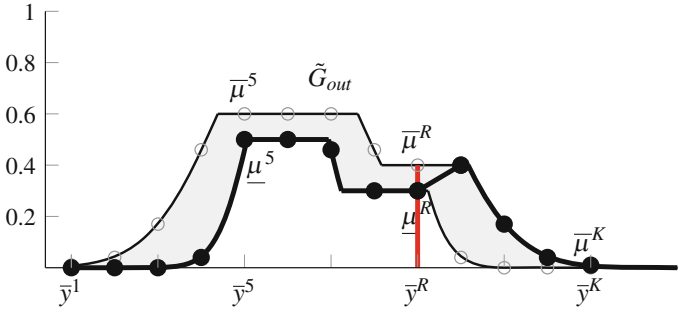
2.5.4.1 Karnik-Mendel Type-Reduction

The KM algorithm is an iterative process which allows one to obtain an interval of uncertainty given by $[y_l, y_r]$ for the centroid of an Interval Type-2 FS. Similarly to the Type-1 FSs defuzzification case, Karnik and Mendel [13] also proposed several methods to perform the Type-2 FSs' TR based on approaches of the Type-1 FLS defuzzification procedures, namely: Height and Modified Height TR, Centroid TR and Center-of-Sets TR. The choice of the defuzzification method has significant implications in the result's quality. The Height and Modified Height are the less complex ones to implement. However, it is known that when a single rule is triggered, these methods may return inconsistent results [13]. The Centroid one requires a large amount of calculations because, for each new system input, it has firstly to merge the consequent part FS of every rule and only then obtain the centroid of the resulting FS. Finally, the Center-of-Sets TR is usually the employed method since it performs a smaller amount of operations when compared with the Centroid one. Its efficiency is due to the *a priori* computation of each consequent FS's centroid. Since the FS's centroid value is independent from the system's input variables, this result can be used as a constant in the Center-of-Sets TR. Therefore, the only procedure that has to be performed after each new input into the system is a weighted average of the stored centroids according to a combination of the upper and lower firing levels of each rule. Since the Center-of-Sets TR inevitably requires one to compute the centroid of each consequent Type-2 FS once, the Centroid TR will be hereby presented.

Similarly to the Centroid defuzzification procedure, the Centroid TR starts by obtaining K samples from a Type-2 FS. Since the FOU of a Type-2 FS embeds several Type-1 FS, to perform the TR one has firstly to obtain two Type-1 FS whose centroid best approximates the upper and lower bounds of the Type-2 FS centroid. Considering for example the $\tilde{G}_{out}$ FS, the procedure starts by using its sampled upper and lower bounds to find the optimal values for the switching points [L, R], as depicted in Fig. 2.17.



(a) Computing y_l: Switching from the upper bounds of the firing intervals to the lower bounds.



(b) Computing y_r: Switching from the lower bounds of the firing intervals to the upper bounds.

Fig. 2.17 Switching points in computing y_l and y_r

The candidate points are obtained as follows in Eqs. (2.16) and (2.17).

$$y_l(k) = \frac{\sum_{i=1}^{k} \underline{y}^i \overline{\mu}^i_{\tilde{G}_{out}} + \sum_{i=k+1}^{K} \underline{y}^i \underline{\mu}^i_{\tilde{G}_{out}}}{\sum_{i=1}^{k} \overline{\mu}^i_{\tilde{G}_{out}} + \sum_{i=k+1}^{K} \underline{\mu}^i_{\tilde{G}_{out}}}, \tag{2.16}$$

$$y_r(k) = \frac{\sum_{i=1}^{k} \overline{y}^i \underline{\mu}^i_{\tilde{G}_{out}} + \sum_{i=k+1}^{K} \overline{y}^i \overline{\mu}^i_{\tilde{G}_{out}}}{\sum_{i=1}^{k} \underline{\mu}^i_{\tilde{G}_{out}} + \sum_{i=k+1}^{K} \overline{\mu}^i_{\tilde{G}_{out}}}, \tag{2.17}$$

where k is an integer in $[1, K-1]$ interval, and K represents the number of discretization points. Then, the optimal interval bounds can be obtained by y_l and y_r, as follows:

$$y_l = \min_{k\in[1,M-1]} y_l(k) \equiv y(L) \equiv \frac{\sum_{i=1}^{L} \underline{y}^i \overline{\mu}^i_{\tilde{G}_{out}} + \sum_{i=L+1}^{K} \underline{y}^i \underline{\mu}^i_{\tilde{G}_{out}}}{\sum_{i=1}^{L} \overline{\mu}^i_{\tilde{G}_{out}} + \sum_{i=L+1}^{K} \underline{\mu}^i_{\tilde{G}_{out}}}, \tag{2.18}$$

$$y_r = \max_{k\in[1,M-1]} y_r(k) \equiv y(R) \equiv \frac{\sum_{i=1}^{R} \overline{y}^i \underline{\mu}^i_{\tilde{G}_{out}} + \sum_{i=R+1}^{K} \overline{y}^i \overline{\mu}^i_{\tilde{G}_{out}}}{\sum_{i=1}^{R} \underline{\mu}^i_{\tilde{G}_{out}} + \sum_{i=R+1}^{K} \overline{\mu}^i_{\tilde{G}_{out}}}, \tag{2.19}$$

where L and R are switch points satisfying

$$y^L \leq y_l < y^{L+1}, \tag{2.20}$$

$$y^R \leq y_r < y^{R+1}. \tag{2.21}$$

The choice of whether we start from the upper or lower firing levels when finding the left and right bounds of each switching point has a very simple explanation. Take y_l as an example: y_l has to be the minimum value of the FLS output. Since $\underline{y}^i$ is ordered ascendantly along the horizontal axis of Fig. 2.12, a large weight (upper bound of the firing interval) should be chosen in the left of the switch point and a small weight (lower bound of the firing interval) for its right side. As finding all the centroid $[y_l, y_r]$ candidates is a computationally inefficient approach, an iterative procedure to find the optimal switching points is presented in Table 2.1.

Despite the improvements brought by Interval Type-2 FS representations, the KM algorithm still requires a large number of iterations to find the optimal type-reduced FS. Therefore, several enhancements and simplifications were proposed in the recent years for the sake of reducing its computational footprint.

2.5.4.2 Optimized Type-Reduction Algorithms

With the development of simpler and alternative algorithms, Type-2 FL definitely gathered the attention of a broader number of researchers, having a direct impact in an increasing number of applications in domains such as modeling, control and classification and pattern recognition observed in recent years.

Table 2.1 Iterative Karnik-Mendel algorithm

Step	For computing y_l	For computing y_r
1.	Initialize $\mu^i_{\tilde{G}_{out}} = \frac{\underline{\mu}^i_{\tilde{G}_{out}} + \overline{\mu}^i_{\tilde{G}_{out}}}{2}$ and compute $y = \frac{\sum_{i=1}^{M} \underline{y}^i \mu^i_{\tilde{G}_{out}}}{\sum_{i=1}^{M} \mu^i_{\tilde{G}_{out}}}$	Initialize $\mu^i_{\tilde{G}_{out}} = \frac{\underline{\mu}^i_{\tilde{G}_{out}} + \overline{\mu}^i_{\tilde{G}_{out}}}{2}$ and compute $y = \frac{\sum_{i=1}^{M} \overline{y}^i \mu^i_{\tilde{G}_{out}}}{\sum_{i=1}^{M} \mu^i_{\tilde{G}_{out}}}$
2.	Find $l \in [1, M-1]$ such that $\underline{y}^l < y < \underline{y}^{l+1}$	Find $r \in [1, M-1]$ such that $\overline{y}^r < y < \overline{y}^{r+1}$
3.	Set $\mu^i_{\tilde{G}_{out}} = \begin{cases} \overline{\mu}^i_{\tilde{G}_{out}}, & n \leq l \\ \underline{\mu}^i_{\tilde{G}_{out}}, & n > l \end{cases}$ and compute $y' = \frac{\sum_{i=1}^{M} \underline{y}^i \mu^i_{\tilde{G}_{out}}}{\sum_{i=1}^{M} \mu^i_{\tilde{G}_{out}}}$	Set $\mu^i_{\tilde{G}_{out}} = \begin{cases} \underline{\mu}^i_{\tilde{G}_{out}}, & n \leq r \\ \overline{\mu}^i_{\tilde{G}_{out}}, & n > r \end{cases}$ and compute $y' = \frac{\sum_{i=1}^{M} \overline{y}^i \mu^i_{\tilde{G}_{out}}}{\sum_{i=1}^{M} \mu^i_{\tilde{G}_{out}}}$
4.	If $y' = y$, stop and set $y_l = y$ and $L = l$; otherwise, set $y = y'$; and go to step 2	If $y' = y$, stop and set $y_r = y$ and $R = r$; otherwise, set $y = y'$; and go to step 2

The Type-Reduction methods found in literature can be grouped into two main categories:

- Enhancements to the KM algorithm, that improve directly the original formulation of the KM by choosing a better initialization and termination conditions, to reduce the number of iterations and to optimize the computing technique speeding up each iteration of the TR process.
- Alternative TR algorithms, which unlike the iterative KM algorithms, are mostly presented in a closed-form representation and provide faster results than the KM method.

In [28] a thorough analysis about the current TR algorithms' state-of-the-art is done and it was observed that enhanced versions of the KM algorithm are, in general, faster than its original formulation. Yet, the gains may vary with the size of the FLS rule base. From the presented approaches, the Enhanced Opposite Direction Searching Algorithm (EODS) [29] shown itself as the fastest one achieving gains up to 70% (relative to the original KM) when the FLS has less than 100 rules—as is used in most part of non-linear processes' modeling applications. It is important

to highlight that, despite their algorithmic differences, enhanced versions of the KM algorithms give exactly the same outputs as its original formulation.

While KM algorithms have been widely adopted, some closed-form methods that bypass this TR procedure have been proposed. However, since their methodologies may be significantly different than the original KM algorithms their outputs may also be quite different. Therefore, a compromise between accuracy and complexity of the method may be necessary. For example, Wu and Tan [30] introduced a method which eliminates TR by defining a collection of Type-1 FS embedded by the footprint of uncertainty. Alternatively, Wu-Mendel Uncertainty Bound method [31] directly uses the uncertainty bounds of the FS and was shown to be the closed-form method giving the closest approximation to the KM and presented an execution turnaround time very close to a similar sized Type-1 FLS. Despite their good performance, a closed-form approach is not adopted in this work since the best performing ones hinder the decomposition of the model output as sum of locally linear models. The capability of decomposing the system in such way is of great advantage to the present work as it allows an efficient implementation of the online training procedures and the synthesis of the control law based on the Generalized Predictive Control theory.

2.5.5 *Defuzzifier*

After applying one of the possible TR methods, the obtained Interval Fuzzy Set still has to be converted into a crisp number so it becomes suited to the most part of the FLS application scenarios. Anyway, this procedure is fairly straightforward, and the defuzzified value obtained by simply computing the average of the interval's left and right endpoints as:

$$y_{out} = \frac{y_r + y_l}{2} \,. \tag{2.22}$$

2.6 Comparative Analysis

The noise reduction properties of Type-2 Fuzzy MFs have been several times pointed in literature as one of its main advantages when compared to its Type-1 counterparts. To study the influence of the antecedent part membership functions' FOU width in the rule activation level when the FLS's inputs are corrupted by a disturbance, a comparative analysis will be performed by probing its input domain with different magnitude noise levels. This comparison will be based on Gaussian-shaped FS with uncertain mean (as presented in Fig. 2.13, where its FOU is bounded by the upper and lower MFs as defined by the equations:

$$\underline{\mu}_{\tilde{F}}(x) = \begin{cases} G(c2, \sigma, x), & x < \frac{c1+c2}{2} \\ \\ G(c1, \sigma, x), & x \geq \frac{c1+c2}{2} \end{cases}, \tag{2.23}$$

$$\overline{\mu}_{\tilde{F}}(x) = \begin{cases} G(c1, \sigma, x), & x < c1 \\ 1, & c1 \leq x \leq c2 \\ G(c2, \sigma, x), & x > c2 \end{cases}. \tag{2.24}$$

In a similar approach as presented in [32], this procedure will be based on a simple FLS comprising a single input and two rules defined by two overlapping MFs ($\tilde{F}_1$ and $\tilde{F}_2$) such that, for a certain input value x, the following conditions are satisfied:

$$\overline{\mu}_2 = 1 - \underline{\mu}_1 , \tag{2.25a}$$

$$\underline{\mu}_2 = 1 - \overline{\mu}_1 , \tag{2.25b}$$

where $\overline{\mu}_i$ and $\underline{\mu}_i$ are the upper and lower firing levels of $\tilde{F}_i(x)$, respectively.

To simplify this evaluation, it is considered that the output is given by Nie Tan closed form Type-Reduction [33],

$$y = \frac{\sum_{i=1}^{M} (\underline{f}^i + \overline{f}^i) y^i}{\sum_{i=1}^{M} (\underline{f}^i + \overline{f}^i)} \tag{2.26}$$

where y_i is the output of each rule and $\underline{f}^i$ and $\overline{f}^i$ are equivalent to $\overline{\mu}_i$ and $\underline{\mu}_i$, respectively, since the system solely has one antecedent. Therefore, based on Eqs. (2.25) and (2.26), the contribution of each rule to the model output is weighted by

$$\begin{aligned} r_i(x) &= \frac{\underline{f}^i + \overline{f}^i}{\sum_{k=1}^{M} (\underline{f}^k + \overline{f}^k)} \\ &= \frac{\underline{\mu}^i(x) + \overline{\mu}^i(x)}{2} . \end{aligned} \tag{2.27}$$

If the system's input is corrupted by a gaussian noise of magnitude n, then the firing strength becomes:

$$r_i(x+n) = \frac{\underline{\mu}_i(x+n) + \overline{\mu}_i(x+n)}{2} . \tag{2.28}$$

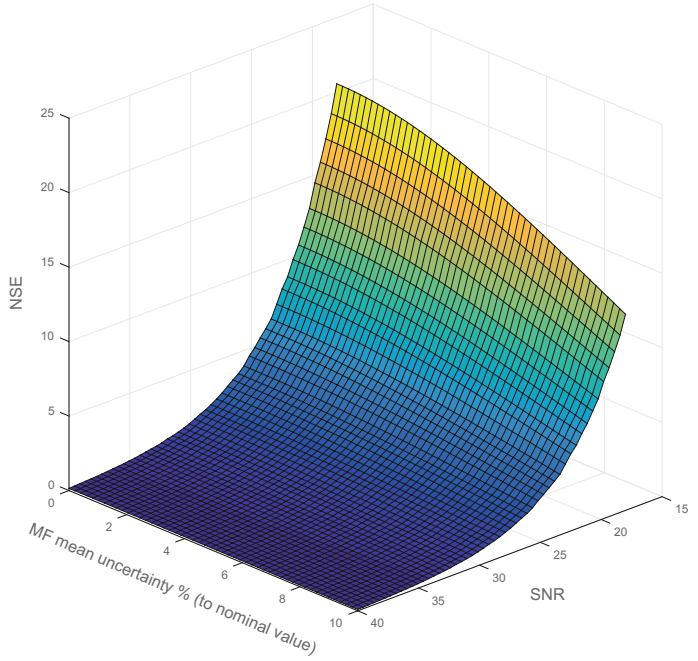


Fig. 2.18 Surface of the NSE of the evaluated system depending on the input noise level and the antecedent parameter's FOU width

Based on Eqs. (2.27) and (2.28), one can evaluate the relationship between the firing level distortion caused by the input noise and the FOU width for a single rule by obtaining the Normalized Squared Error (NSE) as

$$NSE = \int_{n=-n_1}^{n_1} \int_{x=x_1}^{x_2} \left[\frac{r_i(x) - r_i(x+n)}{r_i(x)} \right]^2 dxdn, \qquad for\ n \in \mathbb{R} \qquad x \in \mathbb{R}\ . \quad (2.29)$$

The solution of Eq. (2.29) is obtained numerically by varying in the same proportion the values c_1 and c_2 relatively to a Type-1 FS initially centred in c and then corrupting the input variable with a noise signal with different Signal-to-Noise Ratios (SNRs), defined relatively to the maximum input value of the fuzzy set's domain. The maximum uncertainty percentage relatively to the FS upper and lower MF's center is limited to 10%, as with higher values a great portion of the input space a large portion of the upper and lower membership degrees become closer to one and zero respectively, thus not providing a desirable input space partition for a TS system. Figure 2.18 presents the dependency of the NSE on the noise level and the uncertainty ratio, and the results obtained by averaging 50 trials for every parameters' combination.

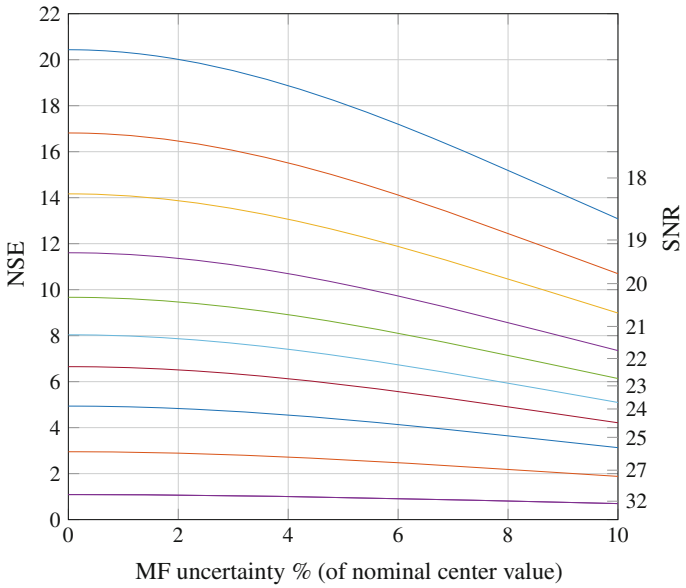


Fig. 2.19 Dependency of the firing degree NSE on the membership function's center uncertainty ratio and the noise level corrupting the input signals

Table 2.2 Distortion level for different SNR considering 10% uncertainty over the membership function's center.

SNR	18	19	20	21	22	23	24	25	27	32
NSE	13.11	10.64	8.98	7.35	6.13	5.09	4.21	3.13	1.88	0.70
σ	0.98	0.81	0.72	0.47	0.46	0.39	0.38	0.24	0.13	0.05

For a better comparison between the different scenarios, a bi-dimensional projection perspective of the previous surface is presented in Fig. 2.19 and the results relative to the 10% uncertainty level summarized in Table 2.2.

From the presented results two main conclusions can be obtained regarding the noise properties of Type-2 FSs:

- For a given SNR, increasing its FOU reduces the distortion observed at each rule firing level when comparing to the Type-1 counterpart which served as starting point (when a 0% uncertainty factor is considered in the MF's center);
- For reduced noise levels, the use of Type-2 FS at the antecedent part of the rule base does not bring significant improvements comparing to its Type-1 counterpart, as is evinced by the flatter region observed in Fig. 2.19 as SNR is increased.

2.7 Conclusions

This chapter presented the fundamental theory of Type-1 Fuzzy Logic Systems and how its extension to the Type-2 Fuzzy Logic formalisms can be performed. In the recent years, Type-2 Fuzzy Logic has been acclaimed as a significant improvement over the fundamental Fuzzy Logic theory. Despite the lack of an irrefutable theoretical proof of it, the fact is that most recent publications present practical applications where the use of Type-2 Fuzzy Logic is shown to be advantageous, mostly in scenarios where uncertain information representations are manipulated. The evaluative scenario concluding this chapter points also in that direction. Therefore, the concepts here introduced will be further developed in the succeeding chapters, by integrating them with more flexible structures in terms of learning capabilities.

References

1. Albus, J.: A therory of cerebellar function. Math. Biosci. **10**, 25–61 (1971)
2. Newell, A., Simon, H.: Human Problem Solving. Prentice-Hall (1972)
3. Zadeh, L.: Fuzzy sets. Inf. Control **8**, 338–353 (1965)
4. Mamdani, E.: Applications of fuzzy algorithms for control of a simple dynamic plant. Proc. IEEE **121**, 1585–1588 (1974)
5. Verbruggen H., Babuska, R.: Fuzzy Logic Control: Advances in Applications. World Scientific (1999)
6. Yasunobu, S., Miyamoto, S., Ihara, S.: Train automatic operation system by fuzzy theory. In: Proceedings of 20th SICE, pp. 467–468 (1981)
7. Yamakawa, T.: Stabilization of an inverted pendulum by a high-speed fuzzy logic controller hardware system. Fuzzy Sets Syst. **32**(2), 161–180. Elsevier (1989)
8. Rumerman, J.: NASA Launch Systems, Space Transportation/Human Spaceflight, and Space Science 1989–1998, NASA Historical Data Book, vol. VII, The NASA History Series, Volume VII, (2009)
9. Mamdani, E.: Applications of fuzzy logic to approximate reasoning using linguistic systems. IEEE Trans. Syst. Man Cybern. **26**(12), 1182–1191 (1977)
10. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybern. **15**(1), 116–132 (1985)
11. Kosko, B.: Fuzzy systems as universal approximators. IEEE Trans. Comput. **43**(11), 1329–1333 (1994)
12. Delgado, M., Duarte, O., Requena, I.: Arithmetic approach for the computing with words paradigm. Int. J. Intell. Syst. **21**, 121–142. Wiley (2006)
13. Mendel, J.: Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions. Prentice-Hall (2001)
14. Passino, K., Yurkovich, S.: Fuzzy Control. Addison-Wesley (1998)
15. Mendel, J., Zadeh, L.: Trillas, E., Yager, R., Lawry, J., Hagras, H., Guadarrama, S.: What computing with words means to me. IEEE Comput. Intell. Mag. (2010)
16. Zadeh, L.: The concept of linguistic variable and its applications to approximate reasoning. Inf. Sci., Part I–III, pp. 199–249, pp. 301–357, pp. 43–80 (1975)
17. Karnik, N., Mendel, J., Liang, Q.: Type-2 fuzzy logic systems. IEEE Trans. Fuzzy Syst. **7**(6), 643–658 (1999)
18. John, R., Coupland, S.: Type 2 fuzzy logic: a historical view. IEEE Comput. Intell. Mag. (2007)

19. Mizumoto, M., Tanaka, K.: Fuzzy sets of type-2 under algebraic product and algebraic sum. Fuzzy Sets Syst. **5**, 277–290 (1981)
20. Dubois, D., Prade, H.: Fuzzy Sets and Systems: Theory and Applications. Academic Press (1982)
21. Turksen, I., Norwich, A.: Measurement of fuzziness, measurement of fuzziness. In: Proceedings of the International Conference on Policy Analysis and Information Systems, pp. 745–754 (1981)
22. Wagner, C., Hagras, H.: Novel methods for the design of general type-2 fuzzy sets based on device characteristics and linguistic labels surveys. In: Proceedings of the 2009 International Fuzzy Systems Association World Congress and the 2009 European Society for Fuzzy Logic and Technology Conference, pp. 537–543 (2009)
23. Wagner, C., Hagras, H.: Towards general type-2 fuzzy logic systems based on zSlices. IEEE Trans. Fuzzy Syst. **18**(4) (2010)
24. Coupland, S., John, R.: Geometric type-1 and type-2 fuzzy logic. IEEE Trans. Fuzzy Syst. **15**, 3–15 (2007)
25. Wagner, C., Miller, S., Garibaldi, J., Anderson, D.: From interval-valued data to general type-2 fuzzy sets. IEEE Trans. Fuzzy Syst. **23**(2), 248–269 (2015)
26. Wu, D.; Tan, W.: Type-2 FLC modeling capability analysis. In: Proceeding of the 2005 IEEE International Conference on Fuzzy Systems, pp. 242–247 (2005)
27. Karnik, N., Mendel, J.: Centroid of a type-2 fuzzy set. Inf. Sci. **132**, 195–220 (2001)
28. Wu, D.: Approaches for reducing the computational cost of interval type-2 fuzzy logic systems: overview and comparisons. IEEE Trans. Fuzzy Syst. **21**(1) (2013)
29. Hu, H., Wang, Y., Cai, Y.: Advantages of enhanced opposite direction searching algorithms for computing the centroid of an interval type-2 fuzzy set. Asian J. Control **14**(6), 1–9 (2012)
30. Wu, D.; Tan, W.: Computationally efficient type-reduction strategies for a type-2 fuzzy logic controller. In: Proceedings of IEEE International Conference in Fuzzy Systems, pp. 353–358 (2005)
31. Wu, D., Mendel, J.: Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems. IEEE Trans. Fuzzy Syst. **10**(5), 622–639 (2002)
32. Kayacan, E.: interval type-2 fuzzy logic systems: theory and design, Ph.D. Thesis, Bogaziçi University, Istanbul, Turkey (2011)
33. Nie, M.; Tan, W.: Towards an efficient type-reduction method for interval type-2 fuzzy logic systems. In: Proceedings of IEEE International Conference on Fuzzy Systems, pp. 1425–1432 (2008)