2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

Designing Practical Interval Type-2 Fuzzy Logic Systems Made Simple

Dongrui Wu
Machine Learning Laboratory
GE Global Research, Niskayuna, NY
E-mail: wud@ge.com

Jerry M. Mendel
Ming Hsieh Department of Electrical Engineering
University of Southern California, Los Angeles, CA
E-mail: mendel@sipi.usc.edu

***Abstract*—Interval type-2 fuzzy logic systems (IT2 FLSs) have become increasingly popular in the last decade, and have demonstrated superior performance in a number of applications. However, the computations in an IT2 FLS are more complex than those in a type-1 FLS, and there are many choices to be made in designing an IT2 FLS, including the shape of membership functions (Gaussian or trapezoidal), number of membership functions, type of fuzzifier (singleton or non-singleton), kind of rules (Mamdani or Takagi-Sugeno-Kang), type of t-norm (minimum or product), method to compute the output (type-reduction or not), and methods for tuning the parameters (gradient-based methods or evolutionary computation algorithms; one-step or two-step). While these choices give an experienced IT2 FLS researcher extensive freedom to design the optimal IT2 FLS, they may look overwhelming and confusing to IT2 beginners. Such a beginner may make an inappropriate choice, obtain unexpected results, and lose interest, which will hinder the wider applications of IT2 FLSs. In this paper we try to help IT2 beginners navigate through the maze by recommending some representative choices for an IT2 FLS design. We also clarify two myths about IT2 FLSs. This paper will make IT2 FLSs more accessible to IT2 beginners.**

Keywords*—Interval type-2 fuzzy set, interval type-2 fuzzy logic system

I. Introduction

Type-2 fuzzy sets (FSs) were introduced by Zadeh in 1975 [37] but have only become popular during the last decade. Fig. 1 shows the number of publications per year, when searched in Google Scholar using the exact phrase "type 2 fuzzy" excluding citations and patents[1]. Observe that the trend is almost exponential. Another perspective to evaluate the popularity of type-2 fuzzy logic is to look at the awarded Outstanding Papers of the *IEEE Transactions on Fuzzy Systems*[2], the flagship journal on fuzzy logic. It has awarded 14 outstanding papers since 2001, and five of them were on type-2 fuzzy logic. Remarkably, all last three outstanding papers were on type-2 fuzzy logic.

Interval type-2 fuzzy logic systems (IT2 FLSs) have dominated the research and applications of type-2 FLSs so far due to their simpler structure and reduced computational cost. However, although IT2 FLSs are simpler than (general) type-2 FLSs, they are still more complex than type-1 (T1) FLSs.

There are many choices to be made in designing an IT2 FLS, including the number of membership functions (MFs), shape of MFs (Gaussian or trapezoidal), type of fuzzifier (singleton or non-singleton), kind of rules (Mamdani or Takagi-Sugeno-Kang, TSK), type of t-norm (minimum or product), method to compute the output (type-reduction or not), and method for optimizations (gradient-based methods or evolutionary computation algorithms; one-step or two-step). Of these the most difficult one is to select the method to compute the output. In [28] we have presented six methods to compute exact type-reduction outputs as well as 11 alternatives. While these many choices give an experienced IT2 FLS researcher extensive freedom to design an optimal IT2 FLS, they may look overwhelming and confusing to IT2 beginners. Such a beginner may make an inappropriate choice, obtain unexpected results, and lose interest, which will hinder the wider applications of IT2 FLSs. In this paper we try to reduce the learning barrier for an IT2 beginner by recommending some representative choices for an IT2 FLS design.

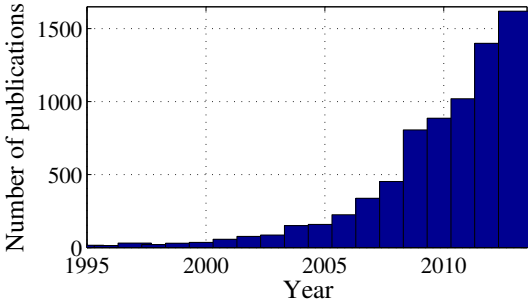

Fig. 1. Number of Google Scholar items on type-2 fuzzy logic.

The remainder of this paper is organized as follows: Because the design of IT2 FLS builds upon the experience of designing a T1 FLS, Section 1 introduces considerations on practical T1 FLS design, and illustrates our recommended choices with an example; Section 2 does the same for IT2 FLSs; Section 3 clarifies two myths about IT2 FLSs; and, Section 4 draws conclusions. We assume the readers have some familiarity with both T1 and IT2 FSs and FLSs, so we will not explain basic concepts like MFs, upper MFs (UMFs), lower MFs (LMFs), and footprint of uncertainty (FOU). These definitions can be found in [16] and [17].

II. Considerations for Practical T1 FLS Designs

A diagram of a T1 FLS is shown in Fig. 2. It consists of four components: fuzzifier, rulebase, inference engine, and

[1]We did not count the number of publications about interval-valued fuzzy sets and systems here. The numbers would be larger if we did that.

[2]http://cis.ieee.org/award-recipients.html#TFSOutstandingPaperAward

defuzzifier. There are many choices to be made in a practical T1 FLS design. The most important ones are described next.

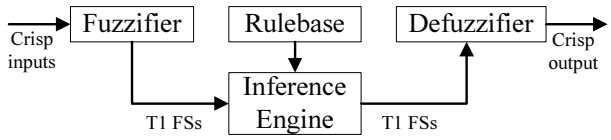

Fig. 2. A T1 FLS.

A. Input MF Shapes: Gaussian or Trapezoidal

The two most commonly used MF shapes for T1 FLSs are Gaussian and trapezoidal. A Gaussian T1 FS is shown in Fig. 3(a), and is described by

$$\mu(x) = e^{-\frac{(x-m)^2}{2\sigma^2}} \tag{1}$$

where m determines the center and σ determines the spread.

A trapezoidal T1 FS is shown in Fig. 3(b). It is determined by four parameters $(a,\ b,\ c,\ d)$. Note that triangular T1 FSs are special cases of trapezoidal T1 FSs when $b = c$.

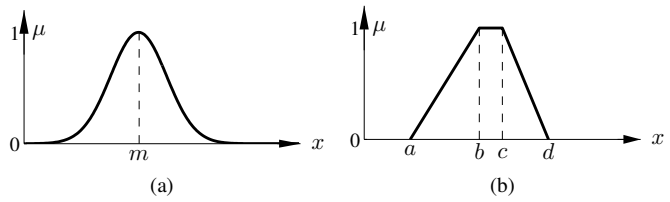

Fig. 3. Examples of T1 FSs. (a) Gaussian; (b) trapezoidal.

Performance is the most important consideration in choosing between Gaussian and trapezoidal MFs in a T1 FLS, and different applications of T1 FLSs have different definitions of performance. The most popular application is fuzzy logic control. There are many studies on comparing the control performance of Gaussian and trapezoidal MFs in T1 fuzzy logic controllers [9], [20]; however, it seems that the conclusion is highly problem dependent, and it is difficult to conclude which MF shape is always better for control performance. We expect the conclusion will be the same for other applications of T1 FLSs, including classification, regression, etc. So, we do not have a preference on the shape of MFs in a T1 FLS in terms of performance. However, we need to point out that the input-output mapping of a T1 FLS may have discontinuities if the input MFs do not cover each input domain completely [31]. This is an advantage of Gaussian MFs over trapezoidal MFs because the former always spread out over the entire input domains.

B. Number of MFs in Each Input Domain

Theoretically, there is no constraint on the number of MFs one should use in each input domain; however, in practice some considerations may prevent one from using too many MFs. First, because the number of rules is an exponential function of the number of MFs in each input domain (e.g., for a 2-input FLS, if each input domain consists of 3 MFs, then the total number of rules is $3^2 = 9$; however, if each input domain consists of 9 MFs, then the total number of rules becomes $9^2 = 81$), the computational cost increases rapidly with the number of MFs. Second, people may prefer FLSs to other black-box controllers, e.g., neural networks, because FLSs can be understood by looking at the rules; however, this advantage diminishes as the number of rules increases.

It is well-known in psychology that the number of objects an average human can hold in working memory is 7 ± 2 [19]. We also suggest ≤ 7 MFs in each input domain to reduce computational cost and to facilitate understanding.

C. Fuzzifier: Singleton or Non-Singleton

The fuzzifier maps an input vector $\mathbf{x} = (x'_1, ..., x'_p)^T$ into p fuzzy sets X_i, $i = 1, 2, ..., p$. There are two categories of fuzzifiers [16]: singleton and non-singleton. For a singleton fuzzifier, $\mu_{X_i}(x_i) = 1$ at $x_i = x'_i$ and $\mu_{X_i}(x_i) = 0$ everywhere else, as shown in Fig. 4(a). For a non-singleton fuzzifier, $\mu_{X_i}(x_i) = 1$ at $x_i = x'_i$ and $\mu_{X_i}(x_i)$ decreases from unity as x_i moves away from x'_i, as shown in Fig. 4(b). Conceptually, the non-singleton fuzzifier implies that the given input value x'_i is the most likely value to be the correct one from all the values in its immediate neighborhood; however, because the input is corrupted by noise, neighboring points are also likely to be the correct value, but to a lesser degree [16]. Usually non-singleton $\mu_{X_i}(x_i)$ is symmetric about x'_i because the effect of noise is most likely to be equivalent on all points.

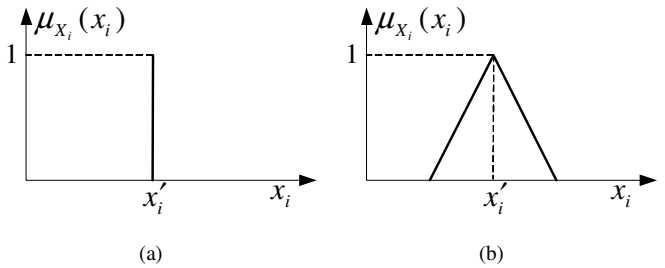

Fig. 4. Examples of (a) a singleton FS, and (b) a non-singleton FS.

Although non-singleton fuzzifiers have demonstrated better performance than singleton fuzzifiers in several applications [3], [16], singleton fuzzifiers are much more popular in practice due to their simplicity. We recommend singleton T1 FLSs, and will only consider singleton T1 FLSs in the sequel.

D. Rules: Mamdani or TSK

There are two kinds of rules: Mamdani [13], where the rule consequents are FSs, and TSK [23], where the rule consequents are crisp functions of the inputs. For example, for a 2-input FLS, a Mamdani rule is of the form

IF x_1 *is* X_1 *and* x_2 *is*X_2, *THEN* y *is* Y

where Y is a T1 FS, whereas a TSK rule is of the form

IF x_1 *is* X_1 *and* x_2 *is* X_2, *THEN* $y = a \cdot x_1 + b \cdot x_2 + c$

where a, b, and c are crisp coefficients.

Mamdani rules were the earliest rules proposed, but TSK rules are much more popular in practice due to their simplicity and flexibility. In many applications people set $a = b = 0$ and each TSK rule consequent is simply represented by a number. As will be shown later in this section, under certain popular defuzzification methods Mamdani rules are equivalent to TSK rules; so, we suggest starting from TSK rules directly.

E. t-Norm: Minimum or Product

t-norms are used by the inference engine to combine the firing levels from multiple antecedents. The two most popular t-norms are the minimum and the product. Assume a rule has two antecedents, and their firing levels are $\mu(x_1')$ and $\mu(x_2')$, respectively. Then, the firing level of the rule is $\min[\mu(x_1'), \mu(x_2')]$ for the minimum t-norm, and $\mu(x_1') \cdot \mu(x_2')$ for the product t-norm.

Both t-norms have been extensively used, and there is no evidence that one t-norm is better than the other. So, one can choose either without worrying about performance.

F. Computing the Output

How to compute the output of a T1 FLS depends heavily on the kind of rules used. For TSK rules, the computation is straightforward: the output is a weighted average of the crisp rule consequents, where the weights are the firing levels of the rules.

There are several different methods for computing the output of a Mamdani FLS. In the height defuzzifier or the center-of-sets defuzzifier, each rule consequent is first replaced by a crisp number, and then a weighted average is used to combine these numbers. In these cases a Mamdani FLS can be viewed as a TSK FLS in which the rule consequents are constants ($a = b = 0$). Another defuzzifier used in the early days of Mamdani FLSs is the centroid defuzzifier, which first combines the output T1 FSs using union and then finds the centroid of this set. Its complexity has significantly limited its adoption.

For simplicity and flexibility, we suggest using TSK rules and weighted average to compute the output of a T1 FLS.

G. Summary

In summary, we recommend singleton TSK T1 FLSs, with ≤ 7 Gaussian MFs in each input domain, either product or minimum t-norm, and weighted average defuzzification.

H. Example

The following T1 proportional-integral (PI) fuzzy logic controller (FLC), adapted from [29], is used to illustrate the computations, following our above recommendations.

The MFs of the T1 PI FLC are shown in Fig. 5 as the bold dashed lines, where the standard deviation of all Gaussian MFs is 0.6. Its four rules are:

$$\begin{aligned}
R^1: &\quad \text{IF } \dot{e} \text{ is } X_1^{\dot{e}} \text{ and } e \text{ is } X_1^e, \text{ THEN } \dot{u} \text{ is } y^1.\\
R^2: &\quad \text{IF } \dot{e} \text{ is } X_1^{\dot{e}} \text{ and } e \text{ is } X_2^e, \text{ THEN } \dot{u} \text{ is } y^2.\\
R^3: &\quad \text{IF } \dot{e} \text{ is } X_2^{\dot{e}} \text{ and } e \text{ is } X_1^e, \text{ THEN } \dot{u} \text{ is } y^3.\\
R^4: &\quad \text{IF } \dot{e} \text{ is } X_2^{\dot{e}} \text{ and } e \text{ is } X_2^e, \text{ THEN } \dot{u} \text{ is } y^4.
\end{aligned}$$

where $\dot{u}$ is the *change of the control signal*, e is the *feedback error*, and $\dot{e}$ is the *change of error*. $y^1 - y^4$ are given in Table I.

Consider an input vector $\mathbf{x}' = (\dot{e}',\ e') = (-0.3, -0.6)$, as shown in Fig. 5. The firing levels of the four T1 FSs are:

$$\mu_{X_1^{\dot{e}}}(\dot{e}') = 0.5063, \quad \mu_{X_2^{\dot{e}}}(\dot{e}') = 0.0956$$

TABLE I. THE RULE CONSEQUENTS OF THE T1 AND IT2 FLCS

	X_1^e ($\widetilde{X}_1^e$)	X_2^e ($\widetilde{X}_2^e$)
$X_1^{\dot{e}}$ ($\widetilde{X}_1^{\dot{e}}$)	$y^1 = -1$	$y^2 = -0.5$
$X_2^{\dot{e}}$ ($\widetilde{X}_2^{\dot{e}}$)	$y^3 = .5$	$y^4 = 1$

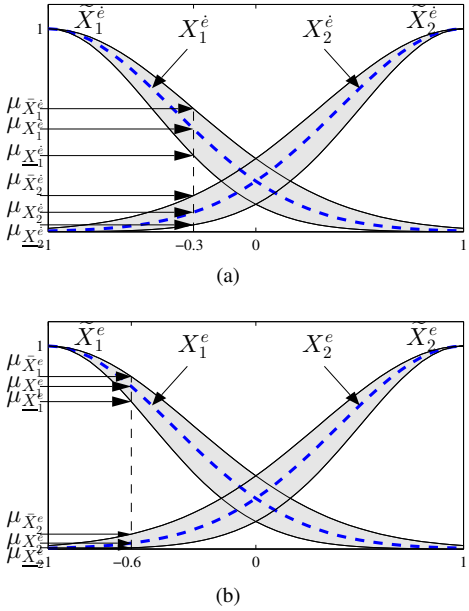

Fig. 5. Firing levels of the T1 FLC, and firing intervals of the IT2 FLC, when $\mathbf{x}' = (\dot{e}', e') = (-0.3, -0.6)$. (a) MFs for $\dot{e}$, and (b) MFs for e.

$$\mu_{X_1^e}(e') = 0.8007, \quad \mu_{X_2^e}(e') = 0.0286$$

The firing levels of its four rules are shown in Table II. The output of the T1 FLS is

$$\dot{u} = \frac{f^1 y^1 + f^2 y^2 + f^3 y^3 + f^4 y^4}{f^1 + f^2 + f^3 + f^4} = -0.3886.$$

III. CONSIDERATIONS FOR PRACTICAL IT2 FLS DESIGNS

The diagram of an IT2 FLS is shown in Fig. 6. Its rules use IT2 FSs instead of T1 FSs; as a result, it needs an extra step called type-reduction before the defuzzifier to reduce IT2 FSs into T1 FSs. Because an IT2 FLS is more complex than a T1 FLS, there are more choices to be made in practical IT2 FLS designs. Next we will describe the most important ones, which will be helpful especially to IT2 beginners.

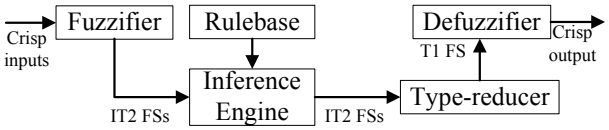

Fig. 6. An IT2 FLS.

A. Input FOUs: Gaussian or Trapezoidal

The FOUs in an IT2 FLS also have two main categories of shapes: Gaussian and trapezoidal. A Gaussian IT2 FS is usually obtained by blurring the mean or standard deviation of a baseline Gaussian T1 FS [34], as shown in Fig. 7. In

TABLE II. FIRING LEVELS OF THE FOUR RULES OF THE T1 FLC

Rule No.:	Firing Level	→	Rule Consequent
R^1 :	$f^1 = \mu_{X_1^{\dot{e}}}(\dot{e}') \cdot \mu_{X_1^{e}}(e') = 0.5063 \times 0.8007 = 0.4054$	→	$y^1 = -1$
R^2 :	$f^2 = \mu_{X_1^{\dot{e}}}(\dot{e}') \cdot \mu_{X_2^{e}}(e') = 0.5063 \times 0.0286 = 0.0484$	→	$y^2 = -0.5$
R^3 :	$f^3 = \mu_{X_2^{\dot{e}}}(\dot{e}') \cdot \mu_{X_1^{e}}(e') = 0.0956 \times 0.8007 = 0.0766$	→	$y^3 = 0.5$
R^4 :	$f^4 = \mu_{X_2^{\dot{e}}}(\dot{e}') \cdot \mu_{X_2^{e}}(e') = 0.0956 \times 0.0286 = 0.0027$	→	$y^4 = 1$

either case, only three parameters $[(m_1, m_2, \sigma)$ or $(m, \sigma_1, \sigma_2)]$ are needed to define a Gaussian IT2 FS. Of course, one can also blur both the mean and the standard deviation to obtain a more general Gaussian FOU, but this approach is rarely used in practice.

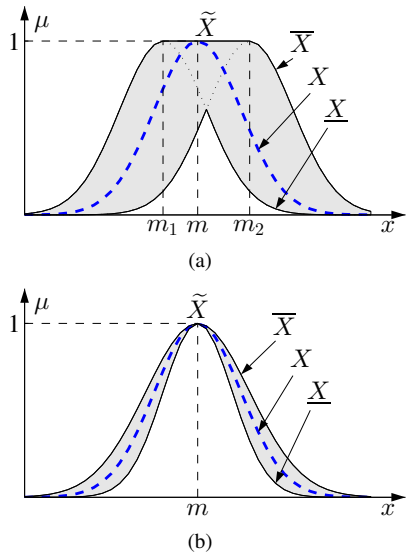

Fig. 7. Gaussian T1 and IT2 FSs. (a) A Gaussian T1 FS (thick dashed curve) and a Gaussian IT2 FS obtained from blurring the mean of the T1 FS; (b) a Gaussian T1 FS (thick dashed curve) and a Gaussian IT2 FS obtained from blurring the standard deviation of the T1 FS.

A trapezoidal IT2 FS can also be obtained by blurring a baseline trapezoidal T1 FS [35], as shown in Fig. 8. Generally, nine parameters are needed to represent a trapezoidal IT2 FS, $(a, b, c, d, e, f, g, i, h)$ shown in Fig. 8, where (a, b, c, d) determines the UMF and (e, f, g, i, h) determines the sub-normal LMF.

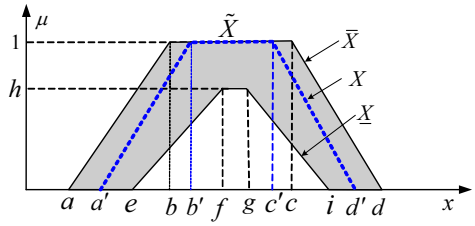

Fig. 8. A trapezoidal T1 FS (thick dashed curve) and a trapezoidal IT2 FS, represented by nine parameters.

From the above description we can conclude that generally it is simpler to represent a Gaussian IT2 FS because it only needs three or four parameters, whereas a trapezoidal IT2 FS needs nine parameters.

In [27] we presented 12 considerations about choosing between Gaussian and trapezoidal FOUs for an IT2 FLS, including representation, construction, optimization, adaptiveness, novelty, analytical structure, continuity, monotonicity, stability, robustness, computational cost, and control performance. Each MF type has its own advantages: Gaussian IT2 FLSs are simpler in design because they are easier to represent and optimize, always continuous, and faster to compute for small rulebases, whereas trapezoidal IT2 FLSs are easier to analyze, although the analysis is still very complex.

Here we focus only on continuity [31] because it was widely ignored before. The following example illustrates the input-output mappings of three 2-input IT2 FLSs using the popular Karnik-Mendel type-reducer[3] [12], [15]. Fig. 9(a) shows the three MFs in each input domain, and Fig. 9(b) shows the corresponding input-output mappings. The FOUs for x_1 are the same in all the cases, where the FOUs for x_2 are not. Observe that:

1) When the input UMFs and LMFs for both x_1 and x_2 fully cover their domains, as shown in the first column of Fig. 9(a), the corresponding input-output mapping is continuous.
2) When the two input domains are fully covered by the UMFs but at least one point in the domain of x_2 is not covered by the LMFs, as shown in the middle column of Fig. 9(a), the corresponding input-output mapping has discontinuities.
3) When the input UMFs and LMFs for x_2 do not fully cover its domain, as shown in the last column of Fig. 9(a), the corresponding input-output mapping has more obvious discontinuities.

In summary, we recommend Gaussian FOUs for their simplicity and automatic guarantee of continuity.

B. Number of MFs in Each Input Domain

Like T1 FLSs, theoretically one can use an arbitrary number of FOUs in each input domain of an IT2 FLS. However, we again suggest ≤ 7 MFs in each input domain to reduce computational cost and to facilitate understanding.

C. Fuzzifier: Singleton or Non-Singleton

The fuzzifier of an IT2 FLS maps an input vector $\mathbf{x} = (x'_1, ..., x'_p)^T$ into p IT2 FSs $\tilde{X}_i$, $i = 1, 2, ..., p$. Like its T1 counterpart, the fuzzifier of an IT2 FLS can also be singleton or non-singleton [16].

[3]The EIASC algorithm introduced later in this section is a more efficient implementation of this type-reducer; however, the outputs are identical from both.

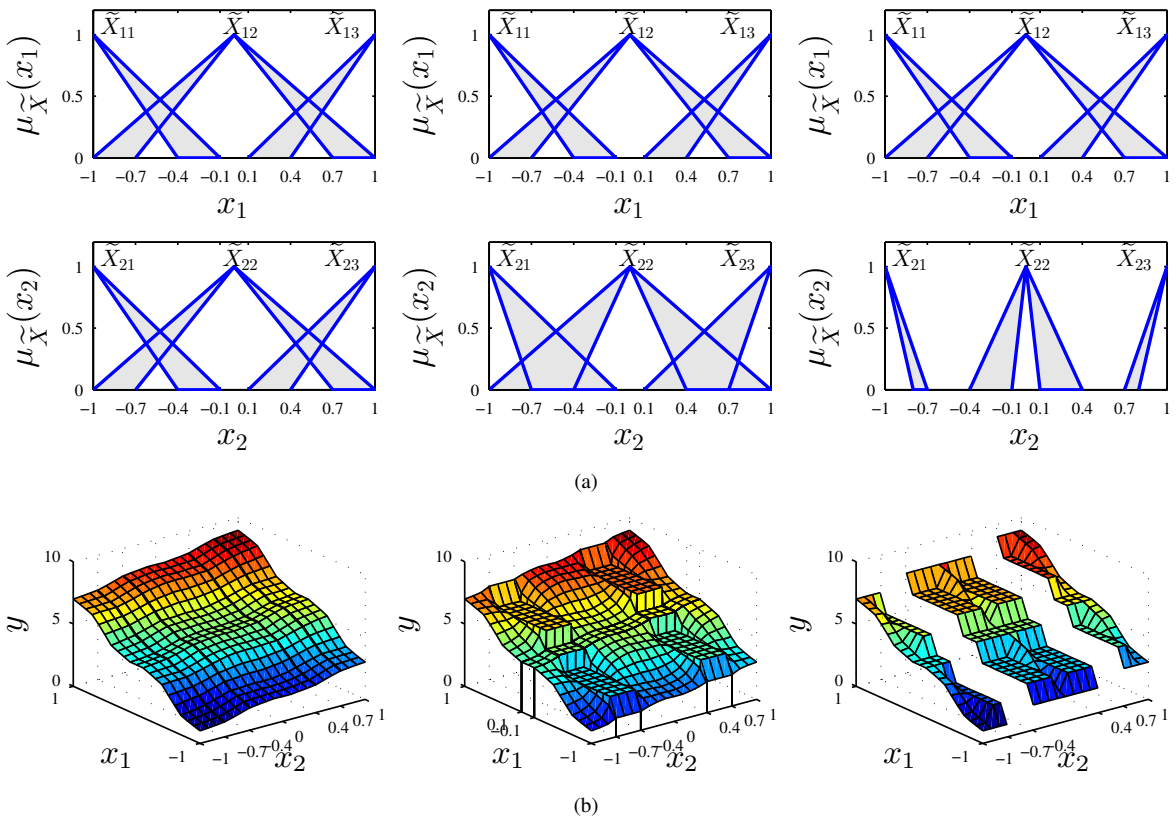

Fig. 9. Example input-output mappings of 2-input IT2 FLSs. (a) The input MFs; (b) The input-output mappings computed by the Karnik-Mendel method.

For a singleton fuzzifier, $\mu_{\tilde{X}_i}(x_i) = 1/1$ at $x_i = x_i'$ and $\mu_{\tilde{X}_i}(x_i) = 1/0$ everywhere else. For a non-singleton fuzzifier, the output can be a T1 FS, as shown in Fig. 4(b), or even an IT2 FS, as shown in Fig. 7.

Although IT2 FLSs using non-singleton fuzzifiers have demonstrated better performance than singleton fuzzifiers in several applications [16], singleton fuzzifiers are much more popular in practice due to their simplicity. We recommend singleton IT2 FLSs, and will only consider singleton IT2 FLSs in the sequel.

D. Rules: Mamdani or TSK

There are two kinds of rules for an IT2 FLS: Mamdani, where the rule consequents are IT2 FSs, and TSK, where the rule consequents are crisp functions of the inputs. For example, for a 2-input IT2 FLS, a Mamdani rule is of the form *"IF* x_1 *is* $\tilde{X}_1$ *and* x_2 *is* $\tilde{X}_2$*, THEN* y *is* $\tilde{Y}$*"*, where $\tilde{Y}$ is an IT2 FS, whereas a TSK rule is of the form *"IF* x_1 *is* $\tilde{X}_1$ *and* x_2 *is* $\tilde{X}_2$*, THEN* $y = [\underline{a} \cdot x_1 + \underline{b} \cdot x_2 + \underline{c},\ \overline{a} \cdot x_1 + \overline{b} \cdot x_2 + \overline{c}]$*"*, where $\underline{a}$, $\overline{a}$, $\underline{b}$, $\overline{b}$, $\underline{c}$, and $\overline{c}$ are crisp numbers. For simplicity, one may set $\underline{a} = \overline{a}$, $\underline{b} = \overline{b}$, and $\underline{c} = \overline{c}$, in which case each rule consequent becomes a single function of the inputs instead of an interval of functions. One can also set $\underline{a} = \overline{a} = 0$ and $\underline{b} = \overline{b} = 0$, in which case each rule consequent becomes a constant interval $[\underline{c},\ \overline{c}]$. In the simplest case, one sets $\underline{a} = \overline{a} = 0$, $\underline{b} = \overline{b} = 0$, and $\underline{c} = \overline{c}$, i.e., each rule consequent becomes a single number. The latter two approaches are much more popular in practice due to their simplicity, and are our recommended forms to use.

E. t-Norm: Minimum or Product

Both minimum and product t-norms have been extensively used in IT2 FLSs, and there is no evidence that one t-norm is better than the other. So, one can choose either without worrying about performance.

F. Computing the Output

How to compute the output of an IT2 FLS depends heavily on the kinds of rules used. There are several different methods for computing the output of Mamdani FLSs [16]. The most popular method uses center-of-sets type-reduction, in which the centroid of each rule consequent IT2 FS is computed to replace the actual FOU. This is equivalent to the simplified TSK model, where each rule consequent is an interval $[\underline{c},\ \overline{c}]$. This is also our recommended approach.

Next we describe two ways to compute the output in this case.

1) Type-Reduction and Defuzzification: The classical IT2 FLS, as shown in Fig. 6, has separate type-reduction and defuzzification steps.

Type-reduction combines F^n, the firing interval of the rules, and Y^n, the corresponding rule consequents. There are many such methods [16]. The most commonly used one is the center-of-sets type-reducer [16]:

$$Y = \frac{\sum_{n=1}^{N} Y^n F^n}{\sum_{n=1}^{N} F^n} = \bigcup_{\substack{y^n \in Y^n \\ f^n \in F^n}} \frac{\sum_{n=1}^{N} y^n f^n}{\sum_{n=1}^{N} f^n} = [y_l,\ y_r] \quad (2)$$

where,

$$y_l = \min_{k\in[1,N-1]} \frac{\sum_{n=1}^{k} \underline{y}^n \overline{f}^n + \sum_{n=k+1}^{N} \underline{y}^n \underline{f}^n}{\sum_{n=1}^{k} \overline{f}^n + \sum_{n=k+1}^{N} \underline{f}^n} \quad (3)$$

$$y_r = \max_{k\in[1,N-1]} \frac{\sum_{n=1}^{k} \overline{y}^n \underline{f}^n + \sum_{n=k+1}^{N} \overline{y}^n \overline{f}^n}{\sum_{n=1}^{k} \underline{f}^n + \sum_{n=k+1}^{N} \overline{f}^n} \quad (4)$$

Several efficient methods have been proposed for computing y_l and y_r [6], [10]–[12], [14], [30], [32], [36]. Comprehensive descriptions and comparisons are given in [15], [28]. The Enhanced Iterative Algorithm with Stop Condition (EIASC), presented in Table III, makes the best comprise between speed and simplicity, and is the suggested algorithm to use. A Matlab implementation of EIASC can be found in [32].

Once y_l and y_r are obtained, the final defuzzified output is:

$$y = \frac{y_l + y_r}{2}. \quad (5)$$

TABLE III. THE EIASC ALGORITHMS. NOTE THAT $\{\underline{y}^n\}_{n=1,...,N}$ AND $\{\overline{y}^n\}_{n=1,...,N}$ MUST BE SORTED IN ASCENDING ORDER, RESPECTIVELY.

Step	For computing y_l	For computing y_r
1	Initialize	Initialize
	$a = \sum_{n=1}^{N} \underline{y}^n \underline{f}^n$	$a = \sum_{n=1}^{N} \overline{y}^n \underline{f}^n$
	$b = \sum_{n=1}^{N} \underline{f}^n$	$b = \sum_{n=1}^{N} \underline{f}^n$
	$L = 0$	$R = N$
2	Compute	Compute
	$L = L + 1$	$a = a + \overline{y}^R(\overline{f}^R - \underline{f}^R)$
	$a = a + \underline{y}^L(\overline{f}^L - \underline{f}^L)$	$b = b + \overline{f}^R - \underline{f}^R$
	$b = b + \overline{f}^L - \underline{f}^L$	$y_r = a/b$
	$y_l = a/b$	$R = R - 1$
3	If $y_l \le y^{L+1}$, stop;	If $y_r \ge y^R$, stop;
	otherwise, go to Step 2.	otherwise, go to Step 2.

2) Defuzzification Directly: There are also many proposals to by-pass type-reduction[4] and compute the defuzzified output directly [1], [4], [5], [7], [8], [15], [21], [24], [33]. A comprehensive description and comparison is also given in [28]. The Nie-Tan method [21], which computes the output as

$$y = \frac{\sum_{n=1}^{N} y^n(\underline{f}^n + \overline{f}^n)}{\sum_{n=1}^{N}(\underline{f}^n + \overline{f}^n)} \quad (6)$$

gives the best comprise between speed and complexity, and is our recommended one to use.

G. Optimization: Gradient-Based Methods or Evolutionary Computation Methods

Because an IT2 FLS usually has many parameters to optimize, e.g., shape of the MFs, rule consequents, etc, there is no way to tune them manually. Automatic optimization is needed. Generally there are two major categories of optimization algorithms for IT2 FLSs: gradient-based algorithms, and heuristic algorithms, particularly evolutionary computation (EC) algorithms. EC algorithms are recommended for the optimization of IT2 FLSs because derivatives are difficult to compute in an IT2 FLS, and such algorithms are globally convergent [18]. There are many such EC algorithms, e.g., genetic algorithms, simulated annealing, particle swarm optimization, etc, and we have no preference for any one of them.

H. Optimization: One-Step Approach or Two-Step Approach

Once an EC algorithm is chosen, there are two tuning strategies: one-step approach, where an IT2 FLS is tuned from scratch, or two-step approach, where an optimal baseline T1 FLS is tuned first and then optimal FOUs are added to it. In the two-step approach one can include the optimal T1 FLS in the population for the IT2 FLS, which guarantees that the performance of the resulting IT2 FLS is at least as good as the T1 FLS. Details on how to do that using quantum particle swarm optimization are given in [18]. Additionally, the two-step approach also reveals how much performance improvement an IT2 FLS gets over the optimal T1 FLS, and hence is very useful to practitioners: if the performance improvement is not significant, a practitioner may choose to use the T1 FLS for simplicity and speed. For these reasons, we recommend the two-step approach.

I. Summary

In summary, we recommend singleton TSK IT2 FLSs, with ≤ 7 Gaussian FOUs in each input domain, and either product or minimum t-norm. The output can be computed by using: 1) EIASC type-reduction and then defuzzification; or, 2) the Nie-Tan method directly. The optimization should be done by EC algorithms using a two-step approach.

More specifically, assume an IT2 FLS has p inputs and N rules of the form:

$$\widetilde{R}^n: \quad \text{IF } x_1 \text{ is } \widetilde{X}_1^n \text{ and ... and } x_p \text{ is } \widetilde{X}_p^n, \text{ THEN } y \text{ is } Y^n \quad n = 1, ..., N$$

For an input vector $\mathbf{x}' = (x_1', x_2', ..., x_p')^{\mathrm{T}}$, our recommended procedure for computing the output of the IT2 FLS is:

1) Compute the membership interval of x_i' for each $\widetilde{X}_i^n$, $[\mu_{\underline{X}_i^n}(x_i'), \mu_{\overline{X}_i^n}(x_i')]$, $i = 1, 2, ..., p$ and $n = 1, 2, ..., N$.
2) Compute the firing interval of the n^{th} rule, F^n:

$$F^n = [\mu_{\underline{X}_1^n}(x_1') \times \cdots \times \mu_{\underline{X}_I^n}(x_I'), \mu_{\overline{X}_1^n}(x_1') \times \cdots \times \mu_{\overline{X}_I^n}(x_I')] \equiv [\underline{f}^n, \overline{f}^n], \quad n = 1, ..., N \quad (7)$$

   Note that the *minimum* t-norm can also be used in (7).
3) Compute the output by combining F^n and the corresponding rule consequents. This can be done by EIASC type-reduction and defuzzification separately, or by defuzzification directly using (6).

J. Example

The following example, adapted from [29], is used to illustrate the computations in an IT2 FLC, following our recommendations.

[4]Although the type-reduced set provides a measure of the uncertainties that have flowed through all of the IT2 FLS computations, it does not have to be used in practical applications.

An IT2 PI FLC may be constructed by blurring the T1 FSs of a T1 FLC to IT2 FSs. In this paper we blur the standard deviation of the T1 Gaussian MFs from 0.6 to an interval $[0.5,\ 0.7]$, as shown in Fig. 5. The rulebase of the IT2 FLC is

$$\begin{aligned}
\widetilde{R}^1: &\quad \text{IF } \dot{e} \text{ is } \widetilde{X}_1^{\dot{e}} \text{ and } e \text{ is } \widetilde{X}_1^{e},\ \text{THEN } \dot{u} \text{ is } y^1.\\
\widetilde{R}^2: &\quad \text{IF } \dot{e} \text{ is } \widetilde{X}_1^{\dot{e}} \text{ and } e \text{ is } \widetilde{X}_2^{e},\ \text{THEN } \dot{u} \text{ is } y^2.\\
\widetilde{R}^3: &\quad \text{IF } \dot{e} \text{ is } \widetilde{X}_2^{\dot{e}} \text{ and } e \text{ is } \widetilde{X}_1^{e},\ \text{THEN } \dot{u} \text{ is } y^3.\\
\widetilde{R}^4: &\quad \text{IF } \dot{e} \text{ is } \widetilde{X}_2^{\dot{e}} \text{ and } e \text{ is } \widetilde{X}_2^{e},\ \text{THEN } \dot{u} \text{ is } y^4.
\end{aligned}$$

$y^1 - y^4$ have been given in Table I.

Consider again the input vector $\mathbf{x}' = (\dot{e}',\ e') = (-0.3, -0.6)$, as shown in Fig. 5. The firing intervals of the four IT2 FSs are:

$$\begin{aligned}
[\mu_{\underline{X}_1^{\dot{e}}}(\dot{e}'),\ \mu_{\overline{X}_1^{\dot{e}}}(\dot{e}')] &= [0.3753,\ 0.6065]\\
[\mu_{\underline{X}_2^{\dot{e}}}(\dot{e}'),\ \mu_{\overline{X}_2^{\dot{e}}}(\dot{e}')] &= [0.0340,\ 0.1783]\\
[\mu_{\underline{X}_1^{e}}(e'),\ \mu_{\overline{X}_1^{e}}(e')] &= [0.7261,\ 0.8494]\\
[\mu_{\underline{X}_2^{e}}(e'),\ \mu_{\overline{X}_2^{e}}(e')] &= [0.0060,\ 0.0734]
\end{aligned}$$

The firing intervals of the four rules are shown in Table IV. When type-reduction and defuzzification are performed separately, the EIASC algorithm gives $y_l = -0.9288$ and $y_r = -0.4209$, and the final defuzzified output is $\dot{u} = \frac{y_l + y_r}{2} = -0.6748$. When (6) is used, the final output is -0.6932.

Note that the outputs computed from the two approaches are different, given the same rules and MFs. However, in practice one first determines which defuzzification method to use, and then tunes the rules and MFs accordingly, so the optimal rules and MFs for the two defuzzification methods will be different.

K. Software

Matlab has a Fuzzy Logic toolbox which considers only T1 FLSs. Several researchers have developed their own Matlab toolboxes/packages for IT2 FLSs, e.g., Mendel's software[5], Castillo et al.'s toolbox [2], Wu's package [25], Ozek and Akpolat's toolbox [22], etc. Additionally, Wagner developed a Java based toolkit, Juzzy, for T1, IT2 and general T2 FLSs[6].

IV. Myths about IT2 FLSs

The successful applications of IT2 FLSs have created some myths about their performance. Here we shall clarify two of them. To do this we will use IT2 FLCs as an example in the illustrations, but the clarifications can also be extended to other applications of IT2 FLSs.

A. Myth 1: Changing T1 FSs to IT2 FSs Automatically Improves Performance

Many applications have shown that IT2 FLCs can achieve better control performance than their T1 counterparts. This has been attributed to the FOUs (which is true), so an IT2 FLC beginner may get the impression that by changing T1 FSs to IT2 FSs the resulting IT2 FLC will automatically have better performance. Unfortunately, this is not the case. Carefully designed FOUs usually improves performance, but arbitrary FOUs almost never do. To achieve better performance, one needs to re-tune the IT2 FLCs, either from scratch, or using the T1 FLC as a baseline [18], [34]. There is no black magic that, by changing T1 FSs to IT2 FSs, an IT2 FLC will automatically outperform a T1 FLC. The fundamental differences between T1 and IT2 FLCs are explained in [26]; these will help the IT2 FLC beginner to understand the effects of the FOUs.

B. Myth 2: Optimizing an IT2 FLC in Known Scenarios Guarantees Its Optimal Performance in All Unknown Scenarios

We have seen many cases where people tune an IT2 FLC for some operating conditions but then apply it to different operating conditions, and claim that the performance of the IT2 FLC is not as good as expected. This is because the design procedure is not correct. If one wants the IT2 FLC to have good performance under a variety of operating conditions, then all these conditions must be considered during the tuning phase. For example, in [34] we wanted the IT2 FLC to be able to respond quickly to setpoint changes, and also to robustly handle modeling uncertainties including time delay and parameter variations of the underlying physics-based model. All these different scenarios were considered during the design phase. As a result, experimental results were consistent with simulation results. If any of those scenarios was not considered in the design phase, e.g., if the IT2 FLC was tuned without considering time delay, but was then applied to a plant with time delay, then very likely the performance would have been much worse.

V. Conclusions

There are many choices to be made in designing a well-performing IT2 FLS, including the shape of membership functions (Gaussian or trapezoidal), number of membership functions, type of fuzzifier (singleton or non-singleton), kind of rules (Mamdani or TSK), type of t-norm (minimum or product), method to compute the output (type-reduction or not), and optimization method (gradient-based methods or evolutionary computation algorithms; one-step or two-step). While these choices give an experienced IT2 FLS researcher extensive freedom to design the optimal IT2 FLS, they may look overwhelming and confusing to IT2 beginners. To help the IT2 beginner overcome the IT2 learning barrier, this paper recommends the following representative choices for IT2 FLSs based on our experience: singleton fuzzifier, TSK rules, $\leq$ 7 Gaussian FOUs in each input domain, either product or minimum t-norm, computing the output by using EIASC type-reduction and then defuzzification, or, the Nie-Tan method directly, and two-step EC algorithms for optimization. We are not claiming our recommendations are always the best. An experienced researcher on IT2 FLSs may be able to design better IT2 FLSs by using other choices; however, our recommendations have a high chance of leading to IT2 FLSs that can outperform T1 FLSs. We have also clarified two myths about IT2 FLSs.

We hope that this paper will be very useful to IT2 beginners, and will also help to promote wider applications of IT2 FLSs.

[5] http://sipi.usc.edu/∼mendel/software/

[6] http://juzzy.wagnerweb.net/

TABLE IV. FIRING INTERVALS OF THE FOUR RULES OF THE IT2 FLC

Rule No.:	Firing Interval	→	Rule Consequent
$\widetilde{R}^1:$	$[\underline{f}^1,\ \overline{f}^1] = [\mu_{\underline{X}_1^{\dot{e}}}(\dot{e}') \cdot \mu_{\underline{X}_1^{e}}(e'), \mu_{\overline{X}_1^{\dot{e}}}(\dot{e}') \cdot \mu_{\overline{X}_1^{e}}(e')] = [0.3753 \times 0.7261, 0.6065 \times 0.8494] = [0.2725,\ 0.5152]$	$\rightarrow$	$y^1 = -1$
$\widetilde{R}^2:$	$[\underline{f}^2,\ \overline{f}^2] = [\mu_{\underline{X}_1^{\dot{e}}}(\dot{e}') \cdot \mu_{\underline{X}_2^{e}}(e'), \mu_{\overline{X}_1^{\dot{e}}}(\dot{e}') \cdot \mu_{\overline{X}_2^{e}}(e')] = [0.3753 \times 0.0060, 0.6065 \times 0.0734] = [0.0022,\ 0.0445]$	$\rightarrow$	$y^2 = -0.5$
$\widetilde{R}^3:$	$[\underline{f}^3,\ \overline{f}^3] = [\mu_{\underline{X}_2^{\dot{e}}}(\dot{e}') \cdot \mu_{\underline{X}_1^{e}}(e'), \mu_{\overline{X}_2^{\dot{e}}}(\dot{e}') \cdot \mu_{\overline{X}_1^{e}}(e')] = [0.0340 \times 0.7261, 0.1783 \times 0.8494] = [0.0247,\ 0.1514]$	$\rightarrow$	$y^3 = 0.5$
$\widetilde{R}^4:$	$[\underline{f}^4,\ \overline{f}^4] = [\mu_{\underline{X}_2^{\dot{e}}}(\dot{e}') \cdot \mu_{\underline{X}_2^{e}}(e'), \mu_{\overline{X}_2^{\dot{e}}}(\dot{e}') \cdot \mu_{\overline{X}_2^{e}}(e')] = [0.0340 \times 0.0060, 0.1783 \times 0.0734] = [0.0002,\ 0.0131]$	$\rightarrow$	$y^4 = 1$

REFERENCES

[1] M. Begian, W. Melek, and J. Mendel, "Stability analysis of type-2 fuzzy systems," in *Proc. IEEE Int'l Conf. on Fuzzy Systems*, Hong Kong, June 2008, pp. 947–953.

[2] O. Castillo, P. Melin, and J. R. Castro, "Computational intelligence software for interval type-2 fuzzy logic," *Computer Applications in Engineering Education*, vol. 21, no. 4, pp. 737–747, 2013.

[3] T. W. Chua and W. W. Tan, "GA optimisation of non-singleton fuzzy logic system for ECG classification," in *Proc. IEEE Congress on Evolutionary Computation*, Singapore, September 2007, pp. 1677–1684.

[4] S. Coupland and R. I. John, "Geometric type-1 and type-2 fuzzy logic systems," *IEEE Trans. on Fuzzy Systems*, vol. 15, no. 1, pp. 3–15, 2007.

[5] X. Du and H. Ying, "Derivation and analysis of the analytical structures of the interval type-2 fuzzy-PI and PD controllers," *IEEE Trans. on Fuzzy Systems*, vol. 18, no. 4, pp. 802–814, 2010.

[6] K. Duran, H. Bernal, and M. Melgarejo, "Improved iterative algorithm for computing the generalized centroid of an interval type-2 fuzzy set," in *Proc. NAFIPS*, New York, May 2008, pp. 1–5.

[7] M. Gorzalczany, "Decision making in signal transmission problems with interval-valued fuzzy sets," *Fuzzy Sets and Systems*, vol. 23, pp. 191–203, 1987.

[8] S. Greenfield, F. Chiclana, S. Coupland, and R. John, "The collapsing method of defuzzification for discretised interval type-2 fuzzy sets," *Information Sciences*, vol. 179, no. 13, pp. 2055–2069, 2008.

[9] N. Gupta and S. K. Jain, "Comparative analysis of fuzzy power system stabilizer using different membership functions," *Int'l Journal of Computer and Electrical Engineering*, vol. 2, no. 2, pp. 262–267, 2010.

[10] H. Z. Hu, G. Zhao, and H. N. Yang, "Fast algorithm to calculate generalized centroid of interval type-2 fuzzy set," *Control Decis.*, vol. 25, no. 4, pp. 637–640, 2010.

[11] H. Hu, Y. Wang, and Y. Cai, "Advantages of the enhanced opposite direction searching algorithm for computing the centroid of an interval type-2 fuzzy set," *Asian Journal of Control*, vol. 14, no. 6, pp. 1–9, 2012.

[12] N. N. Karnik and J. M. Mendel, "Centroid of a type-2 fuzzy set," *Information Sciences*, vol. 132, pp. 195–220, 2001.

[13] E. H. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant," *Proc. IEE*, vol. 121, no. 12, pp. 1585–1588, 1974.

[14] M. Melgarejo, "A fast recursive method to compute the generalized centroid of an interval type-2 fuzzy set," in *Proc. NAFIPS*, San Diego, CA, June 2007, pp. 190–194.

[15] J. M. Mendel, "On KM algorithms for solving type-2 fuzzy set problems," *IEEE Trans. on Fuzzy Systems*, vol. 21, no. 3, pp. 426–446, 2013.

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

[17] J. M. Mendel, "Type-2 fuzzy sets and systems: An overview," *IEEE Computational Intelligence Magazine*, vol. 2, no. 1, pp. 20–29, 2007.

[18] J. Mendel, "General type-2 fuzzy logic systems made simple: a tutorial," *IEEE Trans. on Fuzzy Systems*, 2014, in press.

[19] G. Miller, "The magical number seven plus or minus two: some limits on the capacity for processing information," *Psychological Review*, vol. 63, no. 2, pp. 81–97, 1956.

[20] J. G. Monicka, N. Sekhar, and K. R. Kumar, "Performance evaluation of membership functions on fuzzy logic controlled AC voltage controller for speed control of induction motor drive," *Int'l Journal of Computer Applications*, vol. 13, no. 5, 2011.

[21] M. Nie and W. W. Tan, "Towards an efficient type-reduction method for interval type-2 fuzzy logic systems," in *Proc. IEEE Int'l Conf. on Fuzzy Systems*, Hong Kong, June 2008, pp. 1425–1432.

[22] M. B. Ozek and Z. H. Akpolat, "A software tool: Type-2 fuzzy logic toolbox," *Computer Applications in Engineering Education*, vol. 16, no. 2, pp. 137–146, 2008.

[23] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 15, pp. 116–132, 1985.

[24] C. W. Tao, J. S. Taur, C.-W. Chang, and Y.-H. Chang, "Simplified type-2 fuzzy sliding controller for wing rock system," *Fuzzy sets and systems*, vol. 207, no. 16, pp. 111–129, 2012.

[25] D. Wu, "A brief tutorial on interval type-2 fuzzy sets and systems." [Online]. Available: https://sites.google.com/site/drwu09/publications/completepubs

[26] D. Wu, "On the fundamental differences between interval type-2 and type-1 fuzzy logic controllers," *IEEE Trans. on Fuzzy Systems*, vol. 20, no. 5, pp. 832–848, 2012.

[27] D. Wu, "Twelve considerations in choosing between Gaussian and trapezoidal membership functions in interval type-2 fuzzy logic controllers," in *Proc. IEEE World Congress on Computational Intelligence*, Brisbane, Australia, June 2012.

[28] D. Wu, "Approaches for reducing the computational cost of interval type-2 fuzzy logic systems: Overview and comparisons," *IEEE Trans. on Fuzzy Systems*, vol. 21, no. 1, pp. 80–99, 2013.

[29] D. Wu, "Two differences between interval type-2 and type-1 fuzzy logic controllers: Adaptiveness and novelty," in *Advances in Type-2 Fuzzy Sets: Theory and Applications*, A. Sadeghian, J. M. Mendel, and H. Tahayori, Eds. Springer, 2013.

[30] D. Wu and J. M. Mendel, "Enhanced Karnik-Mendel Algorithms," *IEEE Trans. on Fuzzy Systems*, vol. 17, no. 4, pp. 923–934, 2009.

[31] D. Wu and J. M. Mendel, "On the continuity of type-1 and interval type-2 fuzzy logic systems," *IEEE Trans. on Fuzzy Systems*, vol. 19, no. 1, pp. 179–192, 2011.

[32] D. Wu and M. Nie, "Comparison and practical implementation of type-reduction algorithms for type-2 fuzzy sets and systems," in *Proc. IEEE Int'l Conf. on Fuzzy Systems*, Taipei, Taiwan, June 2011, pp. 2131–2138.

[33] D. Wu and W. W. Tan, "Computationally efficient type-reduction strategies for a type-2 fuzzy logic controller," in *Proc. IEEE Int'l Conf. on Fuzzy Systems*, Reno, NV, May 2005, pp. 353–358.

[34] D. Wu and W. W. Tan, "Genetic learning and performance evaluation of type-2 fuzzy logic controllers," *Engineering Applications of Artificial Intelligence*, vol. 19, no. 8, pp. 829–841, 2006.

[35] D. Wu and W. W. Tan, "A simplified type-2 fuzzy controller for real-time control," *ISA Transactions*, vol. 15, no. 4, pp. 503–516, 2006.

[36] C.-Y. Yeh, W.-H. Jeng, and S.-J. Lee, "An enhanced type-reduction algorithm for type-2 fuzzy sets," *IEEE Trans. on Fuzzy Systems*, vol. 19, no. 2, pp. 227–240, 2011.

[37] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning-1," *Information Sciences*, vol. 8, pp. 199–249, 1975.