An Overview of Alternative Type-Reduction Approaches for Reducing the Computational Cost of Interval Type-2 Fuzzy Logic Controllers

Dongrui Wu

Machine Learning Lab, GE Global Research, Niskayuna, NY, USA
E-mail: wud@ge.com

***Abstract*—Interval type-2 fuzzy logic controllers have demonstrated better abilities to handle uncertainties than their type-1 counterparts in many applications; however, the high computational cost of the iterative Karnik-Mendel algorithms in type-reduction may hinder them from certain real-time applications. This paper provides an overview and comparison of 11 alternative type-reducers, which have closed-form representations and are more convenient in analysis. Experiments demonstrate that 10 of them are faster than the Karnik-Mendel algorithms. Among them, the Wu-Tan and Nie-Tan methods are the fastest, and they are only about 1.2-1.7 times slower than a type-1 fuzzy logic controller.**

***Index Terms*—Interval type-2 fuzzy logic controller, type-reduction, computational cost**

I. Introduction

Type-2 fuzzy sets (T2 FSs) were first proposed by Zadeh in 1975 as an extension to type-1 (T1) FSs. The main difference between a T2 FS and a T1 FS is that the memberships in a T2 FS are themselves T1 FSs instead of crisp numbers in a T1 FS. T2 FSs and systems have been gaining considerable attentions recently. Particularly, people are interested in interval type-2 (IT2) FSs [19], whose memberships are intervals (instead of T1 FSs in a general T2 FS), for their simplicity and reduced computational cost. IT2 FSs and systems have been used in a variety of applications [19], [20], [23], [28]. In this paper we focus on IT2 fuzzy logic controllers (FLCs), which have demonstrated better abilities to handle uncertainties than their T1 counterparts in many applications [2], [10], [11], [16], [18], [34], [35].

Fig. 1 shows the schematic diagram of an IT2 FLC. It is similar to its T1 counterpart, the major difference being that at least one of the FSs in the rulebase is an IT2 FS. Hence, the outputs of the inference engine are IT2 FSs, and a type-reducer [13], [19] is needed to convert them into a T1 FS before defuzzification can be carried out. Type-reduction (TR) is usually performed by the iterative Karnik-Mendel (KM) algorithms [13], which are computationally intensive and may hinder IT2 FLCs from certain real-time applications.

There have been many different approaches for reducing the computational cost of IT2 FLCs. They can be grouped into three categories:

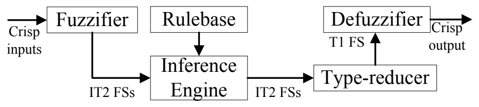


Fig. 1. The schematic diagram of an IT2 FLC.

1) *Enhancements to the KM TR algorithms* [5], [27], [29], [38], which improve directly over the original KM TR algorithms to speed them up. [29] gives an overview and comparison of four such enhancements. It shows that the enhanced iterative algorithm with stop condition (EIASC), also proposed in [29], is the fastest. It also gives the Matlab implementation of the EIASC.
2) *Simplified IT2 FLCs* [30], [35], in which the architecture of an IT2 FLC is simplified by using only a small number of (usually only one) IT2 FSs for the most critical input regions and T1 FSs for the rest. In [30], a simplified IT2 FLC, which used one IT2 FS around $e = 0$ and one around $\dot{e} = 0$, outperformed a T1 FLC with the same number of membership functions (MFs) and showed similar performance as an IT2 FLC whose MFs are all IT2 FSs. In [35], a simplified IT2 FLC, which used only one IT2 FS in the $\dot{e}$ domain, outperformed a T1 FLC with the same number of MFs and showed similar performance as a T1 FLC with more MFs and an IT2 FLC whose all MFs are IT2 FSs.
3) *Alternative TR algorithms* [1], [3], [4], [6], [9], [15], [17], [21], [24], [32], [37]. Unlike the iterative KM algorithms, these alternative TR algorithms have closed-form representations. They are usually fast approximations of the KM algorithms.

This paper focuses on the third category and gives an overview and comparison of 11 alternative TR algorithms. A comprehensive overview and comparison of all three categories of methods is presented in a forthcoming journal paper [25].

We need to point out an important difference between the methods presented in this paper and those enhanced versions to the KM algorithms in [29]: all the enhanced versions of the KM algorithms give exactly the same outputs as the original KM algorithms, which have some fundamentally different characteristics from T1 FLCs [26]. On the other hand, the alternative TR algorithms presented in this paper have very different characteristics from the original KM algorithms, and hence their outputs are different. Both categories of methods have demonstrated good performance in the literature; however, which category is better is an open problem, and is beyond the scope of this paper.

The rest of this paper is organized as follows: Section II presents background materials on IT2 FSs and FLCs. Section III introduces 11 alternative TR algorithms. Section IV compares the computational costs of these 11 algorithms. Section V draws conclusions.

II. IT2 FSs and FLCs

For the completeness of this paper, background materials on IT2 FSs and FLCs are presented in this section.

A. *IT2 FSs*

An IT2 FS [19] $\tilde{X}$ is characterized by its MF $\mu_{\tilde{X}}(x,u)$, i.e.,

$$\widetilde{X} = \int_{x\in D_{\tilde{X}}} \int_{u\in J_x\subseteq[0,1]} \mu_{\tilde{X}}(x,u)/(x,u) \tag{1}$$

where x, called the *primary variable*, has domain $D_{\tilde{X}}$; $u \in [0,1]$, called the *secondary variable*, has domain $J_x \subseteq [0,1]$ at each $x \in D_{\tilde{X}}$; J_x is also called the *support of the secondary MF*; and, the amplitude of $\mu_{\tilde{X}}(x,u)$, called a *secondary grade* of $\tilde{X}$, equals 1 for $\forall x \in D_{\tilde{X}}$ and $\forall u \in J_x \subseteq [0,1]$.

An example of an IT2 FS, $\tilde{X}$, is shown in Fig. 2. Observe that unlike a T1 FS, whose membership grade for each x is a number, the membership of an IT2 FS is an interval. Observe also that an IT2 FS is bounded from above and below by two T1 FSs, $\overline{X}$ and $\underline{X}$, which are called *upper membership function* (UMF) and *lower membership function* (LMF), respectively. The area between $\overline{X}$ and $\underline{X}$ is the *footprint of uncertainty* (FOU). An *embedded T1 FS* is any T1 FS within the FOU. $\underline{X}$ and $\overline{X}$ are two such sets.

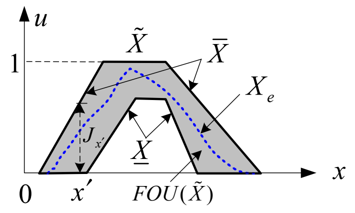


Fig. 2. An IT2 FS. $\underline{X}$ (the LMF), $\overline{X}$ (the UMF), and X_e are three embedded T1 FSs.

B. *IT2 FLCs*

An IT2 FLC is a FLC containing at least one IT2 FSs. Without loss of generality, consider the rulebase of an IT2 FLC consisting of N rules assuming the following form:

$$\widetilde{R}^n: \quad \text{IF } x_1 \text{ is } \widetilde{X}_1^n \text{ and } \cdots \text{ and } x_I \text{ is } \widetilde{X}_I^n, \quad \text{THEN } y \text{ is } Y^n.$$

where $\widetilde{X}_i^n$ $(i = 1,\ldots,I)$ are IT2 FSs, and $Y^n = [\underline{y}^n, \overline{y}^n]$ is an interval, which can be understood as the centroid [12], [19] of a consequent IT2 FS[1], or the simplest TSK model. In many applications [31], [34], [35] we use $\underline{y}^n = \overline{y}^n$, i.e., each rule consequent is represented by a crisp number.

For an input vector $\mathbf{x}' = (x_1', x_2', ..., x_I')$, typical computations in an IT2 FLC involve the following steps:

1) Compute the membership interval of x_i' on each X_i^n, $[\mu_{\underline{X}_i^n}(x_i'), \mu_{\overline{X}_i^n}(x_i')]$, $i = 1,2,...,I$, $n = 1,2,...,N$.
2) Compute the firing interval of the nth rule, F^n:

$$\begin{aligned} F^n = [&\mu_{\underline{X}_1^n}(x_1') \times \cdots \times \mu_{\underline{X}_I^n}(x_I'), \\ &\mu_{\overline{X}_1^n}(x_1') \times \cdots \times \mu_{\overline{X}_I^n}(x_I')] \\ \equiv [&\underline{f}^n, \overline{f}^n], \quad n = 1,...,N \end{aligned} \tag{2}$$

Note that the *minimum* t-norm may also be used in (2). However, this paper focuses only on the *product* t-norm.

3) Perform type-reduction to combine F^n and the corresponding rule consequents. There are many such methods [19]. The most commonly used one is the center-of-sets type-reducer [19]:

$$Y_{\cos} = \frac{\sum_{n=1}^N Y^n F^n}{\sum_{n=1}^N F^n} = \bigcup_{\substack{y^n\in Y^n \\ f^n\in F^n}} \frac{\sum_{n=1}^N y^n f^n}{\sum_{n=1}^N f^n} \tag{3}$$

$$= [y_l, \; y_r] \tag{4}$$

where,

$$y_l = \min_{k\in[1,N-1]} \frac{\sum_{n=1}^k \underline{y}^n \overline{f}^n + \sum_{n=k+1}^N \underline{y}^n \underline{f}^n}{\sum_{n=1}^k \overline{f}^n + \sum_{n=k+1}^N \underline{f}^n} \tag{5}$$

$$y_r = \max_{k\in[1,N-1]} \frac{\sum_{n=1}^k \overline{y}^n \underline{f}^n + \sum_{n=k+1}^N \overline{y}^n \overline{f}^n}{\sum_{n=1}^k \underline{f}^n + \sum_{n=k+1}^N \overline{f}^n} \tag{6}$$

y_l and y_r can be computed by the KM algorithms [12], [19], or their many enhanced versions introduced in [29].

4) Compute the defuzzified output as:

$$y = \frac{y_l + y_r}{2}. \tag{7}$$

III. Alternative TR Algorithms

As the iterative KM algorithms have high computational cost, and also their iterative nature makes them difficult in analysis, people have proposed many alternative TR algorithms, which have closed-form expressions and are usually faster than the KM algorithms. Eleven of them [1], [3], [4], [6], [9], [15], [17], [21], [24], [32], [37] are introduced in this section. They are presented in chronological order.

A. *The Gorzalczany Method*

Gorzalczany [7] proposed two defuzzification methods to obtain a number from the output of the Mamdani inference engine using interval-valued FSs. Since the focus of this paper is the TSK model, we adapt his methods to TSK models. Note that both of his methods only apply to $\underline{y}^n = \overline{y}^n \equiv y^n$.

[1]The rule consequents can be IT2 FSs; however, when the popular center-of-sets TR method [19] is used, these consequent IT2 FSs are replaced by their centroids in the computation; so, it is more convenient to represent the rule consequents as intervals directly.

Given y^n and the firing intervals of the rules, $[\underline{f}^n, \overline{f}^n]$, $n = 1, 2, ..., N$, first we construct a polygon shown in Fig. 3, which can be viewed as a special IT2 FS. For each point in $[y^1, y^N]$, we compute

$$\mu(y) = \frac{(\underline{f} + \overline{f})}{2} \cdot [1 - (\overline{f} - \underline{f})] \tag{8}$$

where $\overline{f} - \underline{f}$ is called the *bandwidth*. Then, the defuzzified output can be computed as

$$y_G = \arg\max_y \mu(y). \tag{9}$$

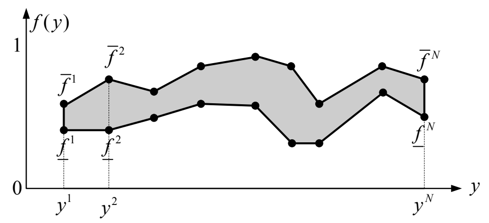


Fig. 3. The polygon used in Gorzalczany's method for computing $\mu(y)$.

Gorzalczany [7] explained that (9) provides an element y_G which most adequately satisfies the compromise between the maximization of the mean value and the minimization of the bandwidth of the inference engine output. He also pointed out that (9) yielded a constant error in his FLC. So, he proposed another method to prevent such a situation, where the defuzzified output is chosen as the point that divides in half the region under the curve $\mu(y)$, i.e., y_G is the solution to the following equation:

$$\int_{y^1}^{y_G} \mu(y)dy = \int_{y_G}^{y^N} \mu(y)dy \tag{10}$$

Gorzalczany did not point out how to efficiently compute y_G in (10). However, if we form a granule from $\mu(y)$, as shown in Fig. 4, then y_G in (10) is its centroid. Coupland and John's method for computing the geometric centroid of an IT2 FS, introduced later in this section, can be used for this purpose, and it is used in our experiment.

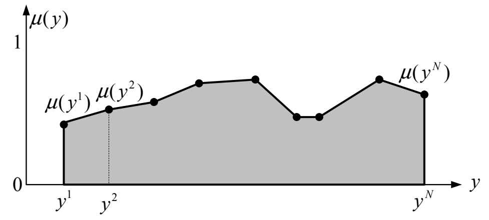


Fig. 4. The polygon used in computing y_G in (10).

B. The Liang-Mendel (LM) Unnormalized Method

Liang and Mendel [17] proposed an *unnormalized* TR method, in which the defuzzified output is still computed by (7), but,

$$y_l = \sum_{n=1}^{N} \underline{f}^n y^n \tag{11}$$

$$y_r = \sum_{n=1}^{N} \overline{f}^n y^n \tag{12}$$

where $\underline{y}^n = \overline{y}^n \equiv y_n$ and $\{y^n\}$ do not need to be sorted. This method is called unnormalized because neither y_l nor y_r is normalized by the sum of the firing levels. In the literature and practice most FLCs use normalized defuzzification.

C. The Wu-Mendel (WM) Uncertainty Bound Method

The uncertainty bound method, proposed by Wu and Mendel [37], computes the output of the IT2 FLC by (7), but,

$$y_l = \frac{\underline{y}_l + \overline{y}_l}{2}, \qquad y_r = \frac{\underline{y}_r + \overline{y}_r}{2}$$

where

$$\overline{y}_l = \min\{\underline{y}^{(0)}, \underline{y}^{(N)}\}, \qquad \underline{y}_r = \max\{\overline{y}^{(0)}, \overline{y}^{(N)}\} \tag{13}$$

$$\underline{y}_l = \overline{y}_l - \frac{\sum_{n=1}^N (\overline{f}^n - \underline{f}^n)}{\sum_{n=1}^N \overline{f}^n \sum_{n=1}^N \underline{f}^n} \times \frac{\sum_{n=1}^N \underline{f}^n (\underline{y}^n - \underline{y}^1) \sum_{n=1}^N \overline{f}^n (\underline{y}^N - \underline{y}^n)}{\sum_{n=1}^N \underline{f}^n (\underline{y}^n - \underline{y}^1) + \sum_{n=1}^N \overline{f}^n (\underline{y}^N - \underline{y}^n)} \tag{14}$$

$$\overline{y}_r = \underline{y}_r + \frac{\sum_{n=1}^N (\overline{f}^n - \underline{f}^n)}{\sum_{n=1}^N \overline{f}^n \sum_{n=1}^N \underline{f}^n} \times \frac{\sum_{n=1}^N \overline{f}^n (\overline{y}^n - \overline{y}^1) \sum_{n=1}^N \underline{f}^n (\overline{y}^N - \overline{y}^n)}{\sum_{n=1}^N \overline{f}^n (\overline{y}^n - \overline{y}^1) + \sum_{n=1}^N \underline{f}^n (\overline{y}^N - \overline{y}^n)} \tag{15}$$

in which

$$\underline{y}^{(0)} = \frac{\sum_{n=1}^N \underline{y}^n \underline{f}^n}{\sum_{n=1}^N \underline{f}^n}, \qquad \underline{y}^{(N)} = \frac{\sum_{n=1}^N \underline{y}^n \overline{f}^n}{\sum_{n=1}^N \overline{f}^n}$$

$$\overline{y}^{(N)} = \frac{\sum_{n=1}^N \overline{y}^n \underline{f}^n}{\sum_{n=1}^N \underline{f}^n}, \qquad \overline{y}^{(0)} = \frac{\sum_{n=1}^N \overline{y}^n \overline{f}^n}{\sum_{n=1}^N \overline{f}^n}$$

Unlike the KM algorithms, the uncertainty bound method does not require $\{\underline{y}^n\}$ and $\{\overline{y}^n\}$ to be sorted, though it still needs to identify the minimum and maximum of $\{\underline{y}^n\}$ and $\{\overline{y}^n\}$.

D. The Wu-Tan (WT) Method

Wu and Tan [32] proposed a closed-form TR and defuzzification method by making use of the equivalent T1 membership grades [33]. The basic idea is to first find an equivalent T1 membership grade $\mu_{X_i^n}(x_i)$ to replace each firing interval $[\mu_{\underline{X}_i^n}(x_i), \mu_{\overline{X}_i^n}(x_i)]$, i.e.,

$$\mu_{X_i^n}(x_i) = \mu_{\overline{X}_i^n}(x_i) - h_i^n(\mathbf{x})[\mu_{\overline{X}_i^n}(x_i) - \mu_{\underline{X}_i^n}(x_i)] \tag{16}$$

where $h_i^n(\mathbf{x})$ is a function of the input $\mathbf{x}$, and is different for different IT2 FSs. This property is motivated by the *adaptiveness* of an IT2 FLC [26], which means that the embedded T1 FSs used to compute the bounds of the type-reduced interval change as input changes.

Since the firing strengths of the rules become point (instead of interval) numbers (f^n) computed from these $\mu_{X_i^n}(x_i)$, the

IT2 FLC becomes an adaptive T1 FLC, and its output is computed as

$$y = \frac{\sum_{n=1}^{N} y^n f^n}{\sum_{n=1}^{N} f^n}. \tag{17}$$

The Wu-Tan method also does not require $\{y^n\}$ to be sorted.

E. The Coupland-John (CJ) Geometric Method

Coupland and John [3] proposed a geometric method for TR and defuzzification of Mamdani IT2 FLCs. In this paper we extend it to TSK IT2 FLCs.

The Coupland-John method constructs a closed polygon from y^n and the corresponding firing intervals, and relabels the boundary points as those shown in Fig. 5. For an IT2 FLS with N rules, there are $2N$ points on the boundary of the closed polygon, (y^n, f^n), $n = 1, ..., 2N$. Then, the centroid of the polygon is viewed as the defuzzification output:

$$y = \frac{\sum_{n=1}^{2N} (y^n + y^{n+1})(y^n f^{n+1} - y^{n+1} f^n)}{3\sum_{i=1}^{2N} (y^n f^{n+1} - y^{n+1} f^n)} \tag{18}$$

where (y^{2N+1}, f^{2N+1}) is the same as (y^1, f^1). Observe that the geometric method requires $\{y^n\}$ to be sorted so that the closed polygon can be constructed.

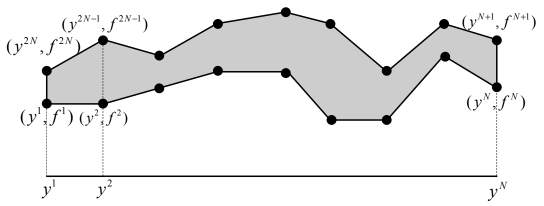


Fig. 5. The closed polygon used in the Coupland-John method.

F. The Nie-Tan (NT) Method

Nie and Tan [21] proposed another closed-form TR and defuzzification method, where the output of an IT2 FLC is computed as:

$$y = \frac{\sum_{n=1}^{N} y^n (\underline{f}^n + \overline{f}^n)}{\sum_{n=1}^{N} (\underline{f}^n + \overline{f}^n)}. \tag{19}$$

Observe that the NT method does not require $\{y^n\}$ to be sorted, and it is a special case of the WT method when $h_i^n(\mathbf{x}) = 0.5$. However, by specifying $h_i^n(\mathbf{x})$ to be a constant for all inputs, the resulting IT2 FLC loses adaptiveness, which is considered as one of the two fundamental differences between IT2 and T1 FLCs [26].

G. The Begian-Melek-Mendel (BMM) Method

Begian, Melek and Mendel [1] proposed another closed-form TR and defuzzification method for IT2 FLCs, i.e.,

$$y = \alpha \frac{\sum_{n=1}^{N} \underline{f}^n y^n}{\sum_{n=1}^{N} \underline{f}^n} + \beta \frac{\sum_{n=1}^{N} \overline{f}^n y^n}{\sum_{n=1}^{N} \overline{f}^n}. \tag{20}$$

where α and β are adjustable coefficients. Observe that it views the output of an IT2 FLC as a combination of the outputs of two T1 FLCs, one constructed only from the LMFs, and the other constructed only from the UMFs.

The BMM method does not require $\{y^n\}$ to be sorted. It is also similar to Niewiadomski et al.'s [22] fourth method for TR of IT2 FSs; however, Niewiadomski et al. have not extended their methods to IT2 FLCs.

The BMM method requires $\underline{y}^n = \overline{y}^n \equiv y^n$. Li et al. [14] extended it to the case that $\underline{y}^n \neq \overline{y}^n$, i.e.,

$$y = \alpha \frac{\sum_{n=1}^{N} \underline{f}^n \underline{y}^n}{\sum_{n=1}^{N} \underline{f}^n} + \beta \frac{\sum_{n=1}^{N} \overline{f}^n \overline{y}^n}{\sum_{n=1}^{N} \overline{f}^n}. \tag{21}$$

Since (21) and (20) have the same computational cost, only the BMM method is considered in this paper.

H. The Greenfield-Chiclana-Coupland-John (GCCJ) Collapsing Method

Greenfield et al. [9] proposed a collapsing method for TR of IT2 FLCs, where each IT2 FS is replaced by a representative embedded T1 FS whose membership grades are computed recursively. To simplify the computation, the representative embedded T1 FS can be approximated by a pseudo representative embedded T1 FS:

$$\mu_X(x) = \frac{\mu_{\underline{X}}(x) + \mu_{\overline{X}}(x)}{2} \tag{22}$$

Once all IT2 FSs in an IT2 FLC are replaced by their pseudo representative embedded T1 FSs, the IT2 FLC is reduced to a T1 FLC, and the defuzzification is straightforward. Again, the collapsing method does not require $\{y^n\}$ to be sorted.

Observe that the collapsing method looks very similar to the NT method. In fact, when there is only one input, these two methods are identical; however, they are different when there are more than one input, because the firing level of $\widetilde{R}^n$ (see Section II-B) in the GCCJ method is

$$f_{GCCJ}^n = \prod_{i=1}^{I} \frac{\mu_{\underline{X}_i^n}(x_i) + \mu_{\overline{X}_i^n}(x_i)}{2}$$

whereas the firing level of $\widetilde{R}^n$ in the NT method is

$$f_{NT}^n = \frac{\prod_{i=1}^{I} \mu_{\underline{X}_i^n}(x_i) + \prod_{i=1}^{I} \mu_{\overline{X}_i^n}(x_i)}{2}$$

Greenfield et al. [8] also proposed a sampling method for TR, where only a relatively small random sample of the totality of embedded T1 FSs is processed. Because its output is not deterministic, it is not considered in this paper.

I. The Li-Yi-Zhao (LYZ) Method

Li et al. [15] proposed a new TR method based on interval analysis without considering the dependency of f^n in the numerator and denominator of (3). They still computed the output of the IT2 FLC by (7), but,

$$y_l = \min \left[\frac{\sum\limits_{n=1}^{N} \min\left(\underline{f}^n \underline{y}^n, \overline{f}^n \underline{y}^n\right)}{\sum_{n=1}^{N} \underline{f}^n}, \frac{\sum\limits_{n=1}^{N} \min\left(\underline{f}^n \underline{y}^n, \overline{f}^n \underline{y}^n\right)}{\sum_{n=1}^{N} \overline{f}^n} \right]$$

$$y_r = \max\left[\frac{\sum_{n=1}^{N}\max\left(\underline{f}^n\overline{y}^n, \overline{f}^n\overline{y}^n\right)}{\sum_{n=1}^{N}\underline{f}^n}, \frac{\sum_{n=1}^{N}\max\left(\underline{f}^n\overline{y}^n, \overline{f}^n\overline{y}^n\right)}{\sum_{n=1}^{N}\overline{f}^n}\right]$$

The LYZ method does not require $\{y^n\}$ to be sorted.

J. The Du-Ying (DY) Method

Du and Ying [4] proposed an average defuzzifier. It first computes 2^N crisp outputs obtained by all possible combinations of the lower and upper firing levels, i.e.,

$$y_m = \frac{\sum_{n=1}^{N} y^n f^{n^*}}{\sum_{n=1}^{N} f^{n^*}}, \quad m = 1, 2, ..., 2^N \tag{23}$$

where $f^{n^*} \in \{\underline{f}^n, \overline{f}^n\}$. The final defuzzified output is then computed as the average of all these 2^N y_m, i.e.,

$$y = \frac{1}{2^N}\sum_{m=1}^{2^N} y_m. \tag{24}$$

The DY method does not require $\{y^n\}$ to be sorted.

Though the DY method makes the analysis of the resulting IT2 FLC easier, it has higher computational cost than the KM type-reducer. Additionally, its computational cost increases exponentially with the number of rules since there are 2^N T1 FLCs to be computed.

K. The Tao-Taur-Chang-Chang (TTCC) Method

Tao et al. [24] proposed a simplified IT2 FLC, whose output is computed as

$$y = \alpha y_{PLM} + (1-\alpha) y_{PLM} \tag{25}$$

where y_{PLM} is the output of a T1 FLC constructed only from the *possible-left-most* embedded T1 FSs, and y_{PRM} is the output of a T1 FLC constructed only from the *possible-right-most* embedded T1 FSs. In [24] an IT2 FS was obtained by blurring a triangular T1 FS left and right, and hence the possible-left-most and possible-right-most embedded T1 FSs can be easily identified, as shown in Fig. 6(a); however, Tao et al. did not discuss how to identify these embedded T1 FSs for IT2 FSs with arbitrary FOUs. In this paper we construct these embedded T1 FSs as shown in Fig. 6(b). It is easy to observe that Tao et al.'s construction method is a special case of ours.

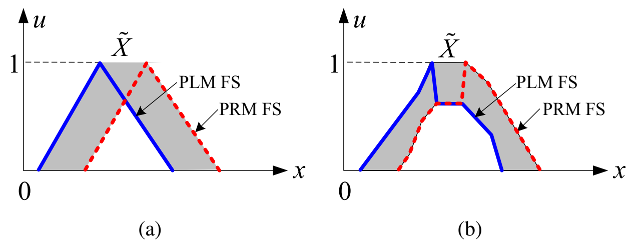


Fig. 6. (a) The possible-left-most and possible-right-most embedded T1 FSs used in [24]; (b) The possible-left-most and possible-right-most embedded T1 FSs for an arbitrary FOU.

The TTCC method does not require $\{y^n\}$ to be sorted. Observe that it is similar to the BMM method in that both methods compute the output of the IT2 FLC as a linear combination of the outputs of two T1 FLCs, which are constructed from the embedded T1 FSs. However, the BMM method uses the upper and lower MFs, whereas the TTCC method uses the possible-left-most and possible-right-most convex and normal embedded T1 FSs.

IV. Computational Cost Comparisons

Four experiments are presented in this section to compare the computational cost of the alternative TR approaches with the KM algorithms and T1 FLCs. The platform was a Lenovo Thinkpad T500 laptop computer with Intel Core2 Duo CPU 8600@2.4G Hz and 3GB memory, running Windows 7 Home Premium 32-bit and Matlab R2009a.

A. Control Surface Computation Using Gaussian MFs

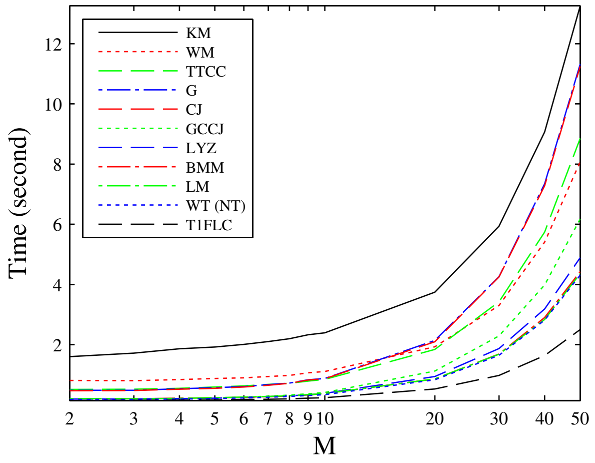
First, two-input single-output IT2 FLCs using Gaussian MFs are considered. Each input domain consisted of M MFs, and $M = \{2, 3, ..., 10, 20, ..., 50\}$ were used. Each input domain was discretized into 10 points, and hence computing a complete control surface requires $10 \times 10 = 100$ type-reductions. To make the results statistically meaningful, we generated 100 random IT2 FLCs for each M and recorded the time that different algorithms were used to perform type-reduction for these $100 \times 100 = 10000$ input pairs. To compare the computational cost of IT2 FLCs with T1 FLCs, we also recorded the computation time for baseline T1 FLCs, whose MFs were the UMFs of the corresponding IT2 FLCs. The results are shown in Fig. 7. We did not include the DY method because its computational cost is much higher than others and increases exponentially with respect to N. Note that for fair comparison with the KM algorithms and T1 FLCs, in Fig. 7 we include the time for computing the firing strengths of the rules, because some algorithms use crisp firing strengths whereas others use interval firing strengths. Observe from Fig. 7 that:

1) All 10 alternative TR approaches in Fig. 7(b) are faster than the KM algorithms, especially when M is small, e.g., $M < 10$.
2) The WT, NT, LM and BMM methods have similar computational cost, and generally they are faster than other alternative TR algorithms.
3) The WT and NT methods are about 1.2-1.7 times slower than the T1 FLC.

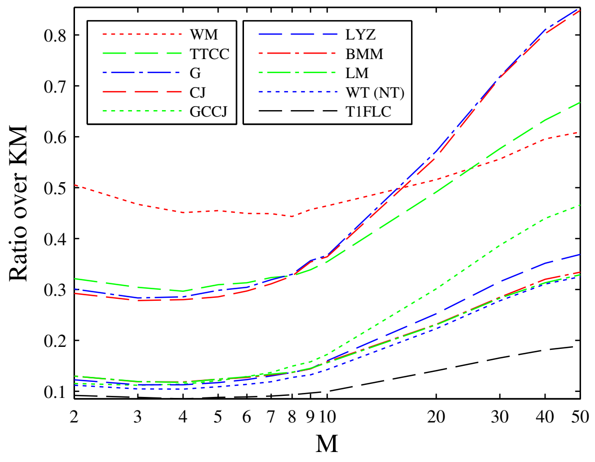
B. Control Surface Computation Using Trapezoidal MFs

Because trapezoidal MFs are also frequently used in practice, in this subsection we compare the computational cost of different approaches in control surface computation using trapezoidal MFs.

There are many different ways to generate trapezoidal IT2 FSs. We used a simple method, as illustrated in Fig. 8 for three IT2 FSs in an input domain. The apexes of the UMFs were generated randomly under the constraint that for any point in the input domain, its firing levels on all UMFs add to 1. After the UMFs were generated, the LMF for each IT2 FS was also generated randomly with some constraints. Take the LMF



(a)



(b)

Fig. 7. Computational cost of the alternative TR algorithms in control surface computation using Gaussian MFs. (a) Total computation time of the 100 IT2 FLCs for different M (the number of IT2 FSs in each input domain). Note that $N = M^2$. (b) The ratio of the computation time of the alternative type-reducers to the KM algorithms.

of the middle IT2 FS as an example. e is a random number between a and b, f and g are two random numbers between b and c, i is a random number between c and d, and h is a random number in $[0,1]$.

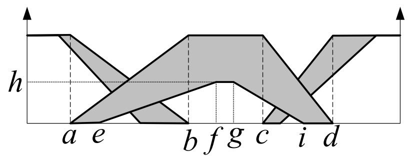
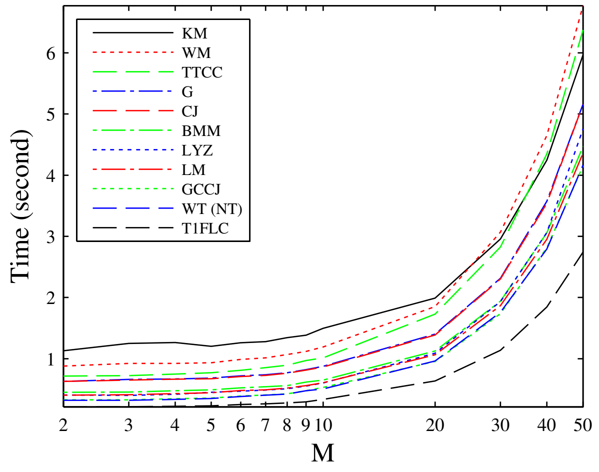


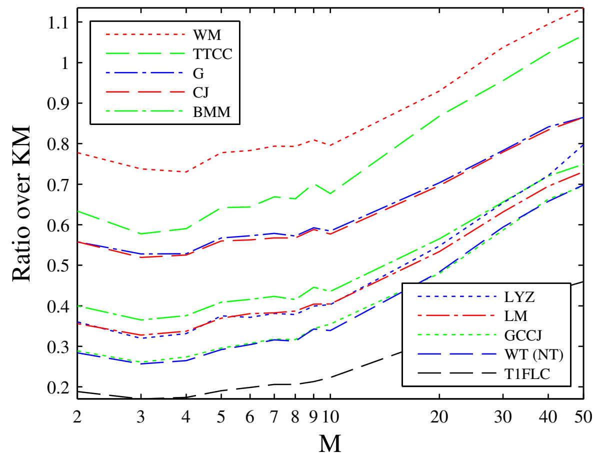
Fig. 8. The trapezoidal IT2 FSs used in the experiments.

We repeated the experiments in the previous subsection for trapezoidal IT2 FSs. The results are shown in Fig. 9. Observe:

1) All 10 alternative TR approaches in Fig. 9(b) are faster than the KM algorithms, especially when M is small, e.g., $M < 10$. However, the computational cost saving of all algorithms, including the T1 FLC, over the KM algorithms is not as large as that in the Gaussian MF case. This is because for trapezoidal MFs at any time at most four rules are fired, so all algorithms converge very quickly.
2) The WT and NT methods are still the fastest; however, the GCCJ method is equally fast. This is because when a very small number of rules are fired, the time used to construct the closed polygon in the GCCJ method is negligible.
3) The WT, NT and GCCJ methods are about 1.5 times slower than the T1 FLC.



(a)



(b)

Fig. 9. Computational cost of the alternative TR algorithms in control surface computation using trapezoidal MFs. (a) Total computation time of the 100 IT2 FLCs for different M. (b) The ratio of the computation time of the alternative type-reducers to the KM algorithms.

C. Evolutionary FLC Design Using Gaussian MFs

We also compare the computational cost of the alternative TR approaches with the KM algorithms and T1 FLCs in FLC design using evolutionary computation, where the performance of a large number of (usually randomly generated) FLCs are evaluated. The following simple first-order plus dead-time plant is employed as the nominal system [36]:

$$G(s) = \frac{K}{\tau s + 1} e^{-Ls} = \frac{1}{10s + 1} e^{-2.5s} \tag{26}$$

The goal is to design an IT2 fuzzy PI controller

$$\dot{u} = k_P \dot{e} + k_I e \tag{27}$$

where $\dot{u}$ is the change in control signal, e is the error, $\dot{e}$ is the change of error, and k_P and k_I are PI gains.

First, assume there are M Gaussian IT2 FSs in each domain (e and $\dot{e}$), and each IT2 FSs is determined by three parameters (one mean, m_m, and two standard deviations, σ_m^1 and σ_m^2, $m = 1, 2, ..., M$). Each of the M^2 rule consequents is represented by a crisp number y^n, $n = 1, ..., M^2$. Then, for each input pair (e, $\dot{e}$), all $N = M^2$ rules are fired, and a TR algorithm is needed to compute the output of the IT2 FLC. The population consisted of 100 randomly generated IT2 FLCs (all m_m, σ_m^1, σ_m^2, and y^n were generated randomly), and the performance of each FLC was evaluated by a step response in the first 100 seconds with sampling frequency 1 Hz. We recorded the time that the alternative TR algorithms were used to perform type-reduction for these $100 \times 100 = 10000$ input pairs. $M = \{2, 3, ..., 10, 20, ..., 50\}$ were used. The results are shown in Fig. 10. Observe that Fig. 10(b) is very similar to Fig. 7(b). The observations made in Section IV-A also hold here.

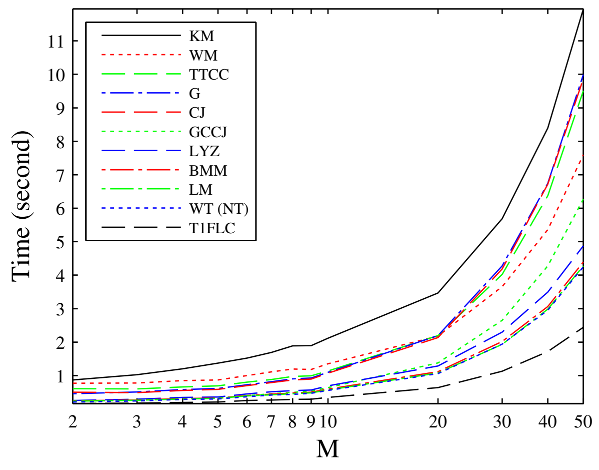
D. Evolutionary FLC Design Using Trapezoidal MFs

We repeated the experiments in the previous subsection for trapezoidal MFs. The results are shown in Fig. 11. Observe that:

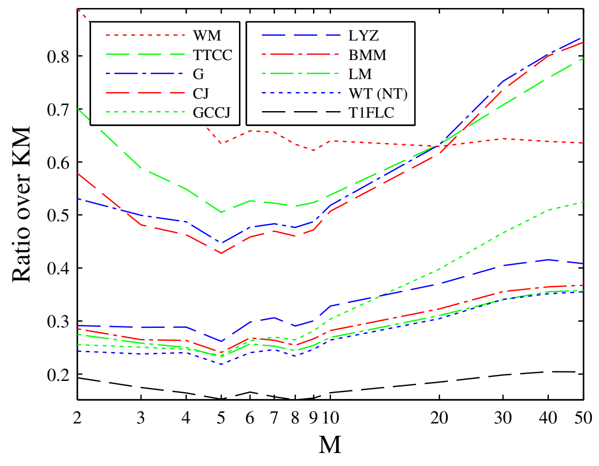
1) Generally all alternative TR methods still outperform the KM algorithms, but the computational cost savings are not as significant as those for the Gaussian MFs in Fig. 10. Again, this is because for trapezoidal MFs at any time at most four rules are fired, so all algorithms converge very quickly.
2) The WT, NT and GCCJ methods are the fastest, but they are about 1.5 times slower than the T1 FLC.

V. Conclusions

IT2 FLCs have demonstrated better abilities to handle uncertainties than their T1 counterparts in many applications; however, the high computational cost of the iterative KM algorithms in type-reduction may hinder them from certain real-time applications. In this paper we have provided an overview and comparison of 11 alternative type-reducers in the literature, which have closed-form representations and are more convenient in analysis. Experiments demonstrated that except for the Du-Ying method, all the other 10 methods are generally faster than the KM algorithms; among them, the Wu-Tan and Nie-Tan methods are the fastest, and they are only about 1.2-1.7 times slower than a T1 FLC.



(a)



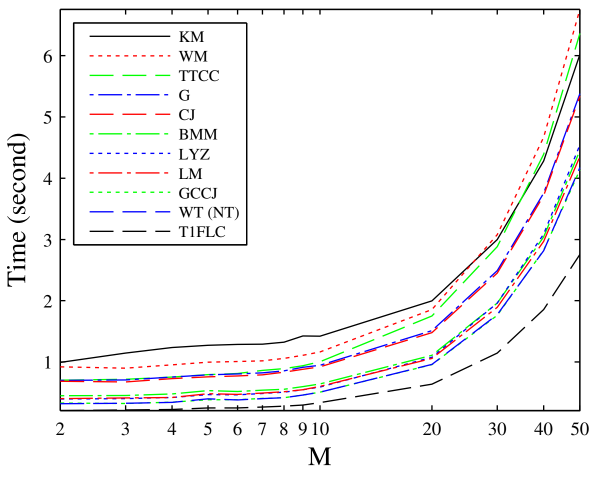
(b)

Fig. 10. Computational cost of the alternative TR algorithms in evolutionary IT2 FLC design using Gaussian MFs. (a) Total computation time of the 100 IT2 FLCs for different M. (b) The ratio of the computation time of the alternative type-reducers to the KM algorithms.

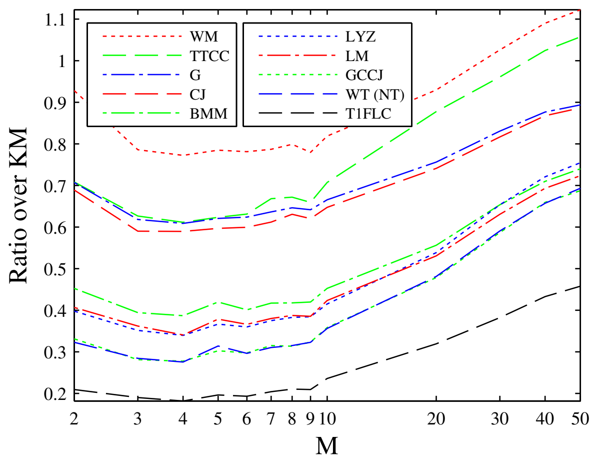
Finally, we need to point out that this paper only compares the computational cost of the alternative type-reducers. Which type-reducer gives the best control performance is an open problem and is beyond the scope of this paper.

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 and P. Melin, *Type-2 Fuzzy Logic Theory and Applications*. Berlin: Springer-Verlag, 2008.
[3] 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.
[4] 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.
[5] 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.



(a)



(b)

Fig. 11. Computational cost of the alternative TR algorithms in evolutionary IT2 FLC design using trapezoidal MFs. (a) Total computation time of the 100 IT2 FLCs for different M. (b) The ratio of the computation time of the alternative type-reducers to the KM algorithms.

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

[7] M. Gorzalczany, "Interval-valued fuzzy controller based on verbal model of object," *Fuzzy Sets and Systems*, vol. 28, pp. 45–53, 1988.

[8] S. Greenfield, R. John, and S. Coupland, "A novel sampling method for type-2 defuzzification," in *Proc. UK Workshop on Computational Intelligence*, London, September 2005, pp. 120–127.

[9] 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.

[10] H. Hagras, "Type-2 FLCs: A new generation of fuzzy controllers," *IEEE Computational Intelligence Magazine*, vol. 2, no. 1, pp. 30–43, 2007.

[11] E. A. Jammeh, M. Fleury, C. Wagner, H. Hagras, and M. Ghanbari, "Interval type-2 fuzzy logic congestion control for video streaming across IP networks," *IEEE Trans. on Fuzzy Systems*, vol. 17, no. 5, pp. 1123–1142, 2009.

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

[13] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Trans. on Fuzzy Systems*, vol. 7, pp. 643–658, 1999.

[14] C. Li, J. Yi, and T. Wang, "Stability analysis of SIRMs based type-2 fuzzy logic control systems," in *Proc. IEEE Int'l. Conf. on Fuzzy Systems*, Taipei, Taiwan, June 2011.

[15] C. Li, J. Yi, and D. Zhao, "A novel type-reduction method for interval type-2 fuzzy logic systems," in *Proc. 5th Int'l. Conf. on Fuzzy Systems and Knowledge Discovery*, vol. 1, Jinan, Shandong, China, March 2008, pp. 157–161.

[16] Q. Liang, N. N. Karnik, and J. M. Mendel, "Connection admission control in ATM networks using survey-based type-2 fuzzy logic systems," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 30, no. 3, pp. 329–339, 2000.

[17] Q. Liang and J. M. Mendel, "Equalization of nonlinear time-varying channels using type-2 fuzzy adaptive filters," *IEEE Trans. on Fuzzy Systems*, vol. 8, no. 5, pp. 551–563, 2000.

[18] Z. Liu, Y. Zhang, and Y. Wang, "A type-2 fuzzy switching control system for biped robots," *IEEE Trans. on Systems, Man, and Cybernetics–C*, vol. 37, no. 6, pp. 1202–1213, 2007.

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

[20] J. M. Mendel and D. Wu, *Perceptual Computing: Aiding People in Making Subjective Judgments*. Hoboken, NJ: Wiley-IEEE Press, 2010.

[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] A. Niewiadomski, J. Ochelska, and P. Szczepaniak, "Interval-valued linguistic summaries of databases," *Control and Cybernetics*, vol. 35, no. 2, pp. 415–443, 2006.

[23] F. C.-H. Rhee, "Uncertainty fuzzy clustering: Insights and recommendations," *IEEE Computational Intelligence Magazine*, vol. 2, no. 1, pp. 44–56, 2007.

[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*, 2012, in press.

[25] D. Wu, "Approaches for reducing the computational cost of interval type-2 fuzzy logic systems: Overview and comparison," *IEEE Trans. on Fuzzy Systems*, 2012, submitted.

[26] D. Wu, "On the fundamental differences between interval type-2 and type-1 fuzzy logic controllers," *IEEE Trans. on Fuzzy Systems*, 2012, in press.

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

[28] D. Wu and J. M. Mendel, "Linguistic summarization using IF-THEN rules and interval type-2 fuzzy sets," *IEEE Trans. on Fuzzy Systems*, vol. 19, no. 1, pp. 136–151, 2011.

[29] 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.

[30] D. Wu and W. W. Tan, "A simplified architecture for type-2 FLSs and its application to nonlinear control," in *Proc. IEEE Conf. on Cybernetics and Intelligent Systems*, Singapore, Dec. 2004, pp. 485–490.

[31] D. Wu and W. W. Tan, "A type-2 fuzzy logic controller for the liquid-level process," in *Proc. IEEE Int'l Conf. on Fuzzy Systems*, vol. 2, Budapest, Hungary, July 2004, pp. 953–958.

[32] 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.

[33] D. Wu and W. W. Tan, "Type-2 FLS modeling capability analysis," in *Proc. IEEE Int'l Conf. on Fuzzy Systems*, Reno, NV, May 2005, pp. 242–247.

[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] D. Wu and W. W. Tan, "Interval type-2 fuzzy PI controllers: Why they are more robust," in *Proc. IEEE Int'l. Conf. on Granular Computing*, San Jose, CA, August 2010, pp. 802–807.

[37] H. Wu and J. M. Mendel, "Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems," *IEEE Trans. on Fuzzy Systems*, vol. 10, no. 5, pp. 622–639, 2002.

[38] 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.