A Fast Algorithm to Compute Precise Type-2 Centroids for Real-Time Control Applications

Sumantra Chakraborty, Amit Konar, Senior Member, IEEE, Anca Ralescu, and Nikhil R. Pal

Abstract—An interval type-2 fuzzy set (IT2 FS) is characterized by its upper and lower membership functions containing all possible embedded fuzzy sets, which together is referred to as the footprint of uncertainty (FOU). The FOU results in a span of uncertainty measured in the defuzzified space and is determined by the positional difference of the centroids of all the embedded fuzzy sets taken together. This paper provides a closed-form formula to evaluate the span of uncertainty of an IT2 FS. The closed-form formula offers a precise measurement of the degree of uncertainty in an IT2 FS with a runtime complexity less than that of the classical iterative Karnik–Mendel algorithm and other formulations employing the iterative Newton–Raphson algorithm. This paper also demonstrates a real-time control application using the proposed closed-form formula of centroids with reduced root mean square error and computational overhead than those of the existing methods. Computer simulations for this real-time control application indicate that parallel realization of the IT2 defuzzification outperforms its competitors with respect to maximum overshoot even at high sampling rates. Furthermore, in the presence of measurement noise in system (plant) states, the proposed IT2 FS based scheme outperforms its type-1 counterpart with respect to peak overshoot and root mean square error in plant response.

Index Terms—Centroid, footprint of uncertainty and fuzzy control, interval type-2 fuzzy sets, type-2 fuzzy sets.

I. INTRODUCTION

T RADITIONAL approaches to system modeling usually employ differential equations to represent input–output relationships of physical systems. Due to inadequate knowledge about system characteristics, system engineers attempt to model many real-world systems using fuzzy logic, the logic based on fuzzy set theory [1]. The input and the output of such systems are represented by fuzzy sets in which the input–output relationship is modeled with fuzzy relations. Performance of a fuzzy model depends on the characterization of the system by appropriate fuzzy rules and reasoning methodology to infer the outputs from the measured inputs.

Type-1 fuzzy set (T1 FS) provides an opportunity to model system’s input–output relation by considering a single knowledge engineer’s perspective. However, individual opinion is not always free from errors. With type-2 fuzzy sets (T2 FS), pioneered by Zadeh [1] and popularized by Mendel et al. [3]–[8], [16]–[20], [22], [32], [34], [39], [40], it is possible to use the knowledge of several experts to construct a space of fuzzy membership functions (MFs) and to determine system outputs by using the measured inputs and the space of these MFs. Two distinct approaches to the ill-defined system modeling using T2 FS are found in the literature. They are referred to as generalized T2 FS (GT2 FS) [15], [33] and interval T2 FS (IT2 FS). Currently, modeling of engineering systems using IT2 FS [10], [14], [21], [23] has attracted an increased interest from the research community. This is due, among other reasons, to the computational simplicity over modeling using GT2 FS. IT2 FS takes care of several type-1 MFs for a fuzzy proposition, obtained from different knowledge sources, and thus can capture both intrasource (within a given MF) and intersource (taking all MFs together) level uncertainty. The representational power of IT2 FS to handle both intrasource and intersource level uncertainty accounts for the popularity of using IT2 FS in several engineering applications, including control [11], diagnosis [12], path planning by a robot [13], and many other engineering problems cited in [2], [24], [31], and [35]–[38].

A GT2 FS is usually represented by a triplet \((\mu_A(x), \mu(x, \mu_A(x)))\), where \(x\) is a linguistic primary variable in a given universe \(X\), \(A\) is a subset of \(X\), \(\mu_A(x)\) is the primary membership of \(x\) to \(A\), and \(\mu(x, \mu_A(x))\) is the secondary membership function of \(x\) and \(\mu_A(x)\). Both the primary and the secondary memberships lie in \([0, 1]\). On the other hand, an IT2 FS is characterized by two MFs \(\tilde{\mu}_A(x)\) and \(\tilde{\mu}_L(x)\), called the upper MF (UMF) and the lower MF (LMF), respectively. The interval \([\mu_L(x), \mu_U(x)]\) containing the union of all possible embedded \(T\) FS [3] in an IT2 FS is referred to as the footprint of uncertainty (FOU) [5].

Usually, the centroid of an IT2 FS is defined by an interval set \([c_l, c_u]\), where \(c_l\) and \(c_u\) are the left and the right end-point centroids, respectively. The span of uncertainty of a given IT2 FS is measured by taking the difference of \(c_l\) and \(c_u\) [19], [20]. It is important to evaluate the exact values of centroids \(c_l\) and \(c_u\) to know the span of uncertainty precisely in an IT2 FS. This paper proposes a novel approach to evaluate the exact value
of the centroids \( c_l \) and \( c_r \) in a given IT2 FS, and to, thereby, obtain the exact span of uncertainty. The main contributions of this paper are briefly outlined in further sections.

Although bounds on \( c_l \) and \( c_r \) are already known [19], closed-form results for \( c_l \) and \( c_r \) are still unknown in the IT2 FS literature. A method to derive a closed-form formula for \( c_l \) and \( c_r \) is presented in this paper. There are two advantages of the closed-form formulas for \( c_l \) and \( c_r \). First, the evaluation of the closed-form formula is computationally time efficient than that of the iterative methods [41] used to evaluate \( c_l \) and \( c_r \). Second, the closed-form formula always yields the exact value of \( c_l \) and \( c_r \), and is thus precise without any predefined error bounds.

The proposed method of computing \( c_l \) and \( c_r \) has successfully been employed in a real-time fuzzy control application. The interval type-2 (IT2) fuzzy controller here embodies five modules (see Fig. 1): one IT2 fuzzifier, one IT2 knowledge base, one IT2 inference engine, one IT2 defuzzifier, and one averaging unit. The IT2 fuzzifier encodes the crisp [45] input variables of the system into IT2 MFs. An IT2 knowledge base is used to describe the mapping between IT2 fuzzy encoded measurements and IT2 fuzzy encoded inferences. An IT2 fuzzy inference engine is used to determine the IT2 MFs of the linguistic variables used in the consequent part of the fired rules. An IT2 defuzzifier determines the left and the right end-point centroids of the IT2 fuzzy inferences. The averaging unit computes the output by taking the average of the left and the right end-point centroids.

The entire computation process, starting at fuzzification and terminating at averaging, should be completed within one sampling interval [44] of a digital controller. Naturally, the whole process shown in Fig. 1 including IT2 defuzzification needs to be repeated in each sampling interval. Traditional methods of iterative centroid computation may not be feasible for the present application for its high computational overhead. The importance of this paper lies here. The proposed method of computation of the left and the right end-point centroids, being very fast, justifies its importance in real-time control applications.

The remainder of this paper is structured as follows. Section II provides preliminaries of IT2 FS, including fundamental definitions. Section III undertakes computation of \( c_l \) and \( c_r \) for FOUs of different geometric types, including triangular, trapezoidal, and left-end and right-end shoulder (RES) MFs, using a closed-form formula. Section IV presents the technique for computation of \( c_l \) and \( c_r \) for FOUs built with piecewise linear (upper and lower) MFs using the closed-form formula. The runtime complexity analysis of an iterative method and the closed-form formula is given in the last part of Section IV. Section V explains a real-time fuzzy control application to justify the importance of the proposed centroid computation. This section also compares the relative performance of the proposed IT2 fuzzy controller with its type-1 counterpart in the presence of noise. Conclusions are listed in Section VI.

II. Preliminaries of T2 FS

The following definitions related to type-2 (T2) and IT2 FS have been used throughout this paper [19], [20].

Definition 1 (T2 FS): A T2 FS \( \tilde{A} \) is characterized by a T2 MF \( \mu_{\tilde{A}}(x, u) \) [19]. Here, \( x \in X \) and \( u \in J_x \) (primary membership value) \( \subseteq [0, 1] \). Here, \( \tilde{A} \) is defined by

\[
\tilde{A} = \{(x, u), \mu_{\tilde{A}}(x, u))|\forall x \in X, \forall u \in J_x \subseteq [0, 1]\}.
\]

Alternatively, \( \tilde{A} \) can be expressed as

\[
\tilde{A} = \int \int_{x \in X, u \in J_x} \mu_{\tilde{A}}(x, u))(x, u), J_x \subseteq [0, 1].
\]

Here, \( \int \int \) denotes the union over all admissible values of \( x \) and \( u \). When the universe of discourse is discrete, \( \int \int \) is replaced by \( \sum \sum \).

Definition 2 (IT2 FS): An IT2 FS is defined by an MF \( \mu_{\tilde{A}}(x, u) = 1 \), where \( x \in X \) and \( u \in J_x \subseteq [0, 1] \). Mathematically, it is expressed as [20]

\[
\tilde{A} = \{(x, u), \mu_{\tilde{A}}(x, u) = 1)|\forall x \in X, \forall u \in J_x \subseteq [0, 1]\}
\]

where \( 0 \leq \mu_{\tilde{A}}(x) \leq 1 \). Alternately, \( \tilde{A} \) can be represented as

\[
\tilde{A} = \int \int_{x \in X, u \in J_x} 1(x, u) = \int_{x \in X} \int_{u \in J_x} 1/u \big|x, J_x \subseteq [0, 1]\}
\]

Again, for a discrete universe of discourse, \( \int \int \) is replaced by \( \sum \sum \) to obtain

\[
\tilde{A} = \sum_{x \in X} \sum_{u \in J_x} 1(x, u) = \sum_{x \in X} \sum_{u \in J_x} 1/u \big|x, J_x \subseteq [0, 1]\}
\]

Definition 3 (FOU): The FOU of \( \tilde{A} \) is defined as the union of all of its primary MFs (embedded T1 FS) [19], [20]. Mathematically, it is expressed as

\[
\text{FOU}(\tilde{A}) = \bigcup_{\forall x \in X} J_x = \{(x, u) : u \in J_x \subseteq [0, 1]\}
\]

Definition 4 (Upper and Lower MF): The upper bound of FOU(\( \tilde{A} \)) is called the UMF(\( \tilde{A} \)) (also UMF) and is denoted by \( \mu_{\tilde{A}}(x) \), \( \forall x \in X \) [19], [20]. Similarly, the lower bound of FOU(\( \tilde{A} \)) is called the LMF(\( \tilde{A} \)) (also LMF) and is denoted by \( \mu_{\tilde{A}}(x) \), \( \forall x \in X \). More precisely

\[
\text{UMF}(\tilde{A}) = \mu_{\tilde{A}}(x), \forall x \in X \Rightarrow \mu_{\tilde{A}}(x), \forall x \in X
\]

\[
\text{LMF}(\tilde{A}) = \mu_{\tilde{A}}(x), \forall x \in X \Rightarrow \min_{\forall x} \mu_{\tilde{A}}(x), \forall x \in X
\]
where \( A_i(x) \) is any embedded type-1 fuzzy set lying in FOU(\( \hat{A} \)). As the primary membership, \( J_x \), is an interval

\[
J_x = \left[ \mu_{A_i}(x), \bar{\mu}_{A_i}(x) \right].
\]

Equations (7) and (8) result in

\[
\text{FOU}(\hat{A}) = \bigcup_{\forall x \in X} \left[ \mu_{A_i}(x), \bar{\mu}_{A_i}(x) \right].
\]

**Definition 5 (Centroid of IT2 FS):** The centroid \( C_{\hat{A}} \) of an IT2 FS \( \hat{A} \) is the union of the centroids of all its embedded T1 FSs (\( A_i \), see Fig. 2) and it is the interval \([c_l, c_r]\) defined as [19], [20]

\[
C_{\hat{A}} = \bigcup_{\forall A_i} c(A_i) = [c_l, c_r]
\]

\[
= \int_{\theta_l \in J_{x_1}} \ldots \int_{\theta_l \in J_{x_n}} 1 \left/ \sum_{\theta_l \in J_{x_i}} \theta_l \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \righthand side is a closed-form formula for \( c_r \) and \( c_l \) evaluation further is questioned in [19] for any general IT2 FS.

This paper attempts to develop a closed-form formula to compute \( c_l \) and \( c_r \) for some of the standard FOU, including triangular, trapezoidal, left-end shoulder (LES), and RES. It makes use of the geometry of the FOU to determine the closed-form formula for \( c_r \) and \( c_l \) for both symmetric and nonsymmetric FOUs. Computation of \( c_r \) and \( c_l \) for triangular FOU is taken up next, whereas the computation of the left and the right end-point centroids of the remaining FOUs stated earlier is presented in the Appendix.

A. Computation of \( c_r \) and \( c_l \) for Triangular FOU

A generalized triangular FOU with parameters \( g, h, a_r, a_l, b_r, b_l \), and \( g \) is shown in Fig. 3. The corresponding UMF and LMF are given as follows:

\[
\mu_{\hat{A}}(x) = gx/b_r + g, \quad \text{if} \quad x \in [-b_r, 0]\]

\[
= -gx/b_r + g, \quad \text{if} \quad x \in [0, b_r]\]

\[
\mu_{\hat{A}}(x) = hx/a_r + h, \quad \text{if} \quad x \in [-a_r, 0]\]

\[
= -hx/a_r + h, \quad \text{if} \quad x \in [0, a_r]\]

where \( g, h, a_r, a_l, b_r, b_l \), and \( g \) are the parameters used to describe the FOU, as indicated in Fig. 3.

The location of \( c_r \) can be inferred in \([-b_r, b_r]\) for the triangular FOU given in Fig. 3. We now partition the possible domain of \( c_r \) into four nonoverlapping segments. They correspond to four alternative cases: Case I, where \( c_r \in [0, a_r] \), Case II, where \( c_r \in [a_r, b_r] \), Case III, where \( c_r \in [-a_l, 0] \), and Case IV, where \( c_r \in [-b_l, -a_l] \). In our subsequent analysis, we will assume all four subintervals as closed intervals as this will not affect the integrals.

**Case I: When \( c_r \in [0, a_r] \)**

Equation (13), when applied to Fig. 3, appears as

\[
c_r = \int_{-a_l}^{0} \mu_{\hat{A}}(x)dx + \int_{0}^{c_r} \mu_{\hat{A}}(x)dx + \int_{c_r}^{b_r} \mu_{\hat{A}}(x)dx
\]

Substituting the values of \( \mu_{\hat{A}}(x) \) from (14) and \( \mu_{\hat{A}}(x) \) from (15) into (16), we obtain the cubic equation

\[
c_r^3 + k_1c_r^2 + k_2c_r + k_3 = 0
\]

where

\[
k_1 = \frac{3a_lb_r(h - g)}{(a_r - b_r)h}, \quad k_2 = \frac{3a_lb_r(a_lh + b_rg)}{(a_r - b_r)h}
\]

\[
and k_3 = \frac{a_lb_r(a_l^2h - b_r^2g)}{(a_r - b_r)h}. \]

\[\text{Fig. 3. Triangular FOU characterized by} \ a_r, b_r, a_l, b_l, g, \text{and} \ h.\]
The solution of (17) is given by (19) [42], [43] subject to the condition that \((a,g - b,h) \neq 0\)

\[
c_r = -2\sqrt{p} \sinh \left( \frac{1}{3} \sinh^{-1} \left( \frac{q}{\sqrt{p}} \right) - \frac{k_1}{3} \right), \text{ if } p > 0
\]

\[
= -2\sqrt{p} \cosh \left( \frac{1}{3} \cosh^{-1} \left( \frac{-q}{\sqrt{p}} \right) \right) - \frac{k_3}{3}, \text{ if } p < 0
\]

\[
= 2\sqrt{p} \cos \left( \frac{1}{3} \cos^{-1} \left( \frac{q}{\sqrt{p}} \right) \right) - n \frac{2\pi}{3} - \frac{k_2}{3}, \text{ for } n = 0, 1, 2, \text{ otherwise}
\]

(19)

In (19), \(p = (3k_2^2 - k_1^2)/9\) and \(q = (2k_1^2 - 9k_1k_2 + 27k_3)/54\). If \((a,g - b,h) = 0\), (17) becomes quadratic that is given by

\[
c_r^2 + k_1c_r + k_2 = 0
\]

where

\[
k_1 = \frac{a_1b_1(a_1h + b_1g)}{a_1b_1(h - g)} \quad \text{and} \quad k_2 = \frac{a_1b_1(a_1^2h - b_1^2g)}{3a_1b_1(h - g)}.
\]

(21)

In this case, the solution for \(c_r\) is obtained as

\[
c_r = -0.5k_1\pm \left( \frac{0.25k_1^2 - k_2}{} \right).
\]

(22)

**Case II: When** \(c_r \in [a_r, b_r]\)

Equation (13), when applied to Fig. 3, appears as

\[
c_r = \frac{\int_{-a_r}^{b_r} \mu_A(x)dx + \int_r^{b_r} \mu_A(x)dx + \int_{-a_r}^{b_r} \mu_A^3(x)dx + \int_r^{b_r} \mu_A^3(x)dx}{\int_{-a_r}^{b_r} \mu_A(x)dx + \int_r^{b_r} \mu_A(x)dx + \int_{-a_r}^{b_r} \mu_A^3(x)dx + \int_r^{b_r} \mu_A^3(x)dx}.
\]

(23)

Substituting the values of \(\hat{\mu}_A(x)\) from (14) and \(\mu_A(x)\) from (15) into the above equation, we obtain

\[
c_r^3 + k_1c_r^2 + k_2c_r + k_3 = 0
\]

(24)

where

\[
k_1 = -3b_r, \quad k_2 = 3b_r(a_rh + a_rh + b_rh)/g
\]

and

\[
k_3 = b_r(a_r^2h - a_r^2h - b_r^2g)/g,
\]

(25)

Here, \(g\) cannot be 0. To solve (24), we use (19).

**Case III: When** \(c_r \in [-a_0, 0]\)

Equation (13), when applied to Fig. 3, appears as

\[
c_r = \frac{\int_{-a_0}^{a_0} \mu_A(x)dx + \int_{a_0}^{b_r} \mu_A(x)dx + \int_{-a_0}^{b_r} \mu_A^3(x)dx + \int_{a_0}^{b_r} \mu_A^3(x)dx}{\int_{-a_0}^{a_0} \mu_A(x)dx + \int_{a_0}^{b_r} \mu_A(x)dx + \int_{-a_0}^{b_r} \mu_A^3(x)dx + \int_{a_0}^{b_r} \mu_A^3(x)dx}.
\]

(26)

Substituting the values of \(\hat{\mu}_A(x)\) from (14) and \(\mu_A(x)\) from (15) into the above equation, we obtain

\[
c_r^3 + k_1c_r^2 + k_2c_r + k_3 = 0
\]

(27)

where

\[
k_1 = \frac{3a_h(b_h - g)}{b_h - a_h}, \quad k_2 = \frac{3a_h(b_h + b_hg)}{b_h - a_h}
\]

and

\[
k_3 = \frac{a_h(a_h^2h - b_h^2g)}{b_h - a_h}.
\]

(28)

The solution of (27) is given by (19) subject to the condition that \((b_h - a_rh) \neq 0\). If \((b_h - a_rh) = 0\), (27) becomes quadratic as follows:

\[
c_r^2 + k_1c_r + k_2 = 0
\]

(29)

where

\[
k_1 = \frac{(a_h + b_r)(a_h + b_r)}{(h - g)} \quad \text{and} \quad k_2 = \frac{(a_h^2h - b_r^2g)}{3(h - g)}.
\]

(30)

To solve (29), we use (22).

**Case IV: When** \(c_r \in [-b_1, -a_1]\)

Equation (13), when applied to Fig. 3, appears as

\[
c_r = \frac{\int_{-b_1}^{a_1} 0dx + \int_{a_1}^{b_1} \hat{\mu}_A(x)dx + \int_{-b_1}^{b_1} \mu_A(x)dx + \int_{a_1}^{b_1} \mu_A(x)dx}{\int_{-b_1}^{a_1} 0dx + \int_{a_1}^{b_1} \mu_A(x)dx + \int_{-b_1}^{b_1} \mu_A(x)dx}.
\]

(31)

Substituting the values of \(\hat{\mu}_A(x)\) from (14) and \(\mu_A(x)\) from (15) in the above equation, we obtain

\[
c_r^3 + k_1c_r^2 + k_2c_r + k_3 = 0
\]

(32)

where \(k_1 = 3b_1, k_2 = 3b_1b_r, \) and \(k_3 = -b_r^2b_r\).

(33)

To solve (32), we use (19).

The closed-form formula for \(c_t\) can be derived in a similar manner.

Theorem 1 in the Appendix shows the uniqueness of \(c_r\) in the entire span of the linguistic variable \(x\). It can be shown analogously that the left end-point centroid, \(c_t\), is also unique for the various IT2 MFs considered here. The above results lead to the development of the following algorithm by taking into account all the possible intervals of \(c_r\) and \(c_t\).

**B. Principles Adopted to Compute \(c_r\) and \(c_t\)**

The following principles are adopted to design the algorithm for computing \(c_r\) and \(c_t\).

1. The domain of \(c_r\) and \(c_t\) is to be partitioned into non-overlapped subintervals or segments. The length of each segment should be as large as possible, satisfying the condition that the UMF and the LMF in each segment has a constant slope.

The \(c_r\) and \(c_t\) space for the triangular FOU of Fig. 3 is partitioned into four intervals of the linguistic variable \(x\) given by \([-b_1, -a_1], [-a_1, 0], [0, a_r], \) and \([a_r, b_r]\).

2. For each possible interval (range) of \(c_r\) (or \(c_t\)), we list the condition for existence of solution in the prescribed interval and the coefficients of the cubic or quadratic equations required to obtain solutions in the selected range.

Table I provides the conditions for existence of solutions along with the coefficients of the appropriate cubic or quadratic equations for each interval of \(c_r\).

3. Given the parameters of the FOU, for each segment of \(c_r\) (or \(c_t\)), we test the condition (in the first column of Table I) for the existence of solution in the selected segment. If the condition is satisfied, we solve the corresponding cubic or quadratic equation with the parameters listed in Table I and then test whether the solution lies in the selected segment of \(c_r\) (or \(c_t\)). If there is no condition for a certain segment of \(c_r\) (or \(c_t\)), we simply solve for \(c_r\) (or \(c_t\)) using the selected parameters of the cubic or quadratic equation and test whether the solution lies in the given range.

4. As \(c_r\) (and \(c_t\)) has a unique value by Theorem 1, only one of the four possible \(c_r\) (and also \(c_t\)) would pass the test of its occurrence in the prescribed interval.
TABLE I

<table>
<thead>
<tr>
<th>Ranges and Condition for existence of solution</th>
<th>Coefficients $k_1$, $k_2$, and $k_3$ for cubic and $k_t$ for quadratic equations used for solving $c_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_r \in [0, a_r)$</td>
<td>$k_1 = 3a_1h(b - g)$</td>
</tr>
<tr>
<td>$k_2 = 3a_2(b + a_r)$</td>
<td>$k_3 = a_2(b - h) a_2(b - h) - a_2(b - h)$</td>
</tr>
<tr>
<td>$c_r \in [0, a_r)$</td>
<td></td>
</tr>
<tr>
<td>$k_2 = (a_1 + b)(b - g)$</td>
<td></td>
</tr>
<tr>
<td>$a_1(b - g)$</td>
<td></td>
</tr>
<tr>
<td>$k_3 = (a_1 - b)(b - g)$</td>
<td></td>
</tr>
<tr>
<td>$c_r \in [a_r, b_r]$</td>
<td>$k_1 = -3b_1$</td>
</tr>
<tr>
<td>$k_2 = 3(a_1 + a_r + b_2)$</td>
<td></td>
</tr>
<tr>
<td>$k_3 = (a_1 - b)(b - g)$</td>
<td></td>
</tr>
<tr>
<td>$c_r \in [-a_r, 0)$</td>
<td>$k_1 = 3a_1h(b - g)$</td>
</tr>
<tr>
<td>$k_2 = 3a_2(b + a_r)$</td>
<td></td>
</tr>
<tr>
<td>$k_3 = a_2(b - h) a_2(b - h) - a_2(b - h)$</td>
<td></td>
</tr>
<tr>
<td>$c_r \in [-a_r, 0)$</td>
<td></td>
</tr>
<tr>
<td>$k_2 = (a_1 + b)(b - g)$</td>
<td></td>
</tr>
<tr>
<td>$a_1(b - g)$</td>
<td></td>
</tr>
<tr>
<td>$k_3 = (a_1 - b)(b - g)$</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4. Here, PR stands for processor. (a) Gantt chart for parallel realization of $c_r$-finding algorithm. (b) Gantt chart for sequential realization (worst case) of the $c_r$-finding algorithm. The former is being mapped to four processors whereas the latter is being mapped to a uniprocessor.

C. Algorithm to Compute $c_r$ and $c_l$ for Triangular FOU

Principles introduced earlier are used to develop an algorithm to compute $c_r$. The algorithm includes five main program segments: $P_0$, $P_1$, $P_2$, $P_3$, and $P_4$. Segment $P_0$ is used to compute $c_r$ when $c_r \in [0, a_r)$, whereas segments $P_2$, $P_3$, and $P_4$ are used to compute $c_r$ when $c_r \in [a_r, b_r)$, $c_r \in [-a_l, 0)$, and $c_r \in [-b_l, -a_l)$, respectively. As $c_r$ is unique, only one particular value of $c_r$ in one of the four segments, $[0, a_r)$, $[a_r, b_r)$, $[-a_l, 0)$ and $[-b_l, -a_l)$, would return the correct answer. The pseudocode of the parallel program is given next.

In the absence of a parallel processing system (with four processing elements), we can run the program for computing $c_r$ (and $c_l$) in a sequential manner. After execution of $P_0$, the other four program segments can be executed in any order. For example, we can execute them in the following order: $P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$, where $P_1 \rightarrow P_2$ indicates that segment $P_2$ has to be executed following the execution of segment $P_1$. Note that whenever a program segment finds the solution, the remaining segments are not required to be executed. The Gantt chart used to schedule the parallel and sequential realization (worst case) of the $c_r$-computing algorithm is given in Fig. 4(a) and (b), respectively.

The trace of the pseudocode for computing $c_r$ for an illustrative triangular FOU is given in Table II.

### Algorithm 1 Parallel Program to Compute $c_r$ for Triangular FOU

**Segment $P_0$:**
- Input required FOU parameters: $a_r$, $b_r$, $a_l$, $g$ and $h$.
- Clobegin
- If $(a_r g - b_r h) = 0$
  - Determine $k_1$ and $k_2$ following 2nd row of Table I;
  - Compute $c_r$ using equation (22);
  - Select $c_r = \text{root}$, if it lies in $[0, a_r)$;
- Else
  - Determine $k_1$, $k_2$, and $k_3$ following 1st row of Table I;
  - Compute $c_r$ using equation (19);
  - Select $c_r = \text{root}$, if it lies in $[0, a_r)$.
- CEnd

**Segment $P_1$:**
- Determine $k_1$, $k_2$, and $k_3$ following 3rd row of Table I;
- Compute $c_r$ using equation (19);
- Select $c_r = \text{root}$, if it lies in $[a_r, b_r)$.

**Segment $P_2$:**
- If $(b_r h - a_r g) = 0$
  - Determine $k_1$ and $k_2$ following 5th row of Table I;
  - Compute $c_r$ using equation (22);
  - Select $c_r = \text{root}$, if it lies in $[-a_l, 0)$;
- Else
  - Determine $k_1$, $k_2$, and $k_3$ following 4th row of Table I;
  - Compute $c_r$ using equation (19);
  - Select $c_r = \text{root}$, if it lies in $[-a_l, 0)$.
- CEnd

**Segment $P_3$:**
- Determine $k_1$, $k_2$, and $k_3$ following 6th row of Table I;
- Compute $c_r$ using equation (19);
- Select $c_r = \text{root}$, if it lies in $[-b_l, -a_l)$.

**Coend.**

The parallel algorithm to determine of $c_l$ can be realized similarly, based on the coefficients in their different ranges.

IV. COMPUTATION OF $c_r$ AND $c_l$ FOR PIECEWISE LINEAR FOU

Now, we extend our investigation to construct a closed-form formula for general piecewise linear FOU. Fig. 5 provides an asymmetric piecewise linear FOU with $n$ segments: $p_1, p_2, \ldots, p_n$. Here, the UMF and the LMF of each segment are linear function of $x$. Let $a_i$ and $a_{i+1}$ denote the lower and upper bound of $x$ in segment $p_i$. Then the expressions of the UMF and the LMF in $p_i$ are given by

$$\mu_i^U(x) = UMF \text{ in } p_i = m_{i1} x + c_{i1}$$

$$\mu_i^L(x) = LMF \text{ in } p_i = m_{i2} x + c_{i2}$$

where $m_{i1}$ and $m_{i2}$ represent slopes and $c_{i1}$ and $c_{i2}$ represent intercepts with the vertical axis.

When $c_r$ exists in segment $p_i$, from (13), we can write the equations shown at the bottom of the next page and

$$C_r = \frac{r_{i1} + \int_{c_{r1}}^{c_{r2}} \mu_i^U(x) dx}{r_{i3} + \int_{c_{r1}}^{c_{r2}} \mu_i^L(x) dx}$$

$$= \frac{r_{i1} + \int_{c_{r1}}^{c_{r2}} \mu_i^U(x) dx + \int_{c_{r1}}^{c_{r2}} \mu_i^L(x) dx + r_{i2}}{r_{i3} + \int_{c_{r1}}^{c_{r2}} \mu_i^L(x) dx + \int_{c_{r1}}^{c_{r2}} \mu_i^U(x) dx + r_{i4}}$$  (36)
Substituting the expression of $i$ Fig. 5. Piecewise linear FOU.

CHAKRABORTY et al. defined as follows:

Similarly, proceeding with (12), we obtain a cubic equation of the form (17). The coefficients $w_i$ here $r_i = \sum_{j=1}^{n_{A_i}} \int_{a_i}^{p_i} j \mu_i(x) dx$, and $r_{i2} = \sum_{j=1}^{n_{A_i}} \int_{a_i}^{p_i} j \tilde{\mu}_i(x) dx$.

Substituting the expression of $i \tilde{\mu}_i(x)$ and $i \mu_i(x)$ from (34) and (35), respectively, into (36) and simplifying the resulting expression, we finally obtain a cubic equation of the form (17). The coefficients $k_1$, $k_2$, and $k_3$ of (17) are given in Table III. Similarly, proceeding with (12), we obtain a cubic equation of the form (17) for $c_r$, whose parameters are also listed in Table III. The parameters, $r_{i2}$, $k=5$ to 8, in Table III are defined as follows:

$$ r_{i5} = \sum_{j=1}^{n_{A_i}} \int_{a_i}^{p_i} j \mu_i(x) dx, \quad r_{i6} = \sum_{j=1}^{n_{A_i}} \int_{a_i}^{p_i} j \tilde{\mu}_i(x) dx, $$

$$ r_{i7} = \sum_{j=1}^{n_{A_i}} \int_{a_i}^{p_i} j \mu_i(x) dx, \quad r_{i8} = \sum_{j=1}^{n_{A_i}} \int_{a_i}^{p_i} j \tilde{\mu}_i(x) dx $$

(37)

Here, we note that when $m_1 = m_2$, the root-finding equations of $c_r$ and $c_i$ become quadratic as in (20), the parameters $k_1$ and $k_2$ of which are given in Table IV.

A. Algorithm for Computation of $c_r$ and $c_i$

It is apparent from Tables III and IV that the equations used to compute $c_r$ (or $c_i$) would have different coefficients for each segment $p_i$, $i$ to $n$. As we do not know the segment $p_i$, where $c_r$ (or $c_i$) would be found, we attempt to solve $c_r$ (or $c_i$) for all the segments in parallel. However, by Theorem 1, only one segment would give the right value of $c_r$ (or $c_i$). Therefore, we need to check whether the obtained solution of $c_r$ for each segment, $p_i$, satisfies the boundary condition $c_r \in [a_i, a_{i+1})$ for $i$ to $n$. The only solution that satisfies the boundary condition of the corresponding segment is the true value of $c_r$. Similar constraints are also applicable to determine $c_i$. The parallel algorithm to compute $c_r$ has the following steps (see Algorithm 2).

B. Time Complexity

In the iterative method [16], the first step is discretization of the primary variable. Obviously, the accuracy of the results depends on the size of the discretization step (width of the interval); a smaller step produces more accurate results for $c_i$ and $c_r$. Suppose that the iterative method is used to evaluate $c_r$. Let the domain of a primary variable of a given FOU be $-b_t$ to $b_t$. CHAKRABORTY et al. defined as follows:

**TABLE II**

<table>
<thead>
<tr>
<th>Segment $P_i$: Get $h=1.0, a_i=0.4, b_i=1.0, a_i=0.7, g=1.0$ and $h=0.7$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment $P_i$: 1. Here, $(a-b-a) \neq 0$.</td>
</tr>
<tr>
<td>2. Following $4^{th}$ row of Table I, $k_1=-4.2666$ and $k_2=0.9866$.</td>
</tr>
<tr>
<td>3. Using equation (22), roots are $0.25587, 1.375+0.65, 1.375-0.65$.</td>
</tr>
<tr>
<td>4. No root lies in $[0.7, 1]$.</td>
</tr>
<tr>
<td>Segment $P_i$: 2. Following last row of Table I, $k_1=3, k_2=3$ and $k_3=1$.</td>
</tr>
<tr>
<td>2. Using equation (19), roots are $0.2599, -1.63+0.09, -1.63-0.09$.</td>
</tr>
</tbody>
</table>
| 3. No root lies in $[-1, -0.4]$.

**TABLE III**

**TABLE IV**

**COEFFICIENTS FOR CUBIC EQUATIONS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$c_r$</th>
<th>$c_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>$(3c_i, c_r) = (m_i, m_i)$</td>
<td>$(3c_i, c_r) = (m_i, m_i)$</td>
</tr>
<tr>
<td>$k_2$</td>
<td>$(3m_1 a_i^2 + 5m_2 a_i^2 - 2c_i c_r) = (c_i, c_i)$</td>
<td>$(3m_1 a_i^2 + 5m_2 a_i^2 - 2c_i c_r) = (c_i, c_i)$</td>
</tr>
<tr>
<td>$k_3$</td>
<td>$(3m_1 a_i^2 + 5m_2 a_i^2 - 2c_i c_r) = (c_i, c_i)$</td>
<td>$(3m_1 a_i^2 + 5m_2 a_i^2 - 2c_i c_r) = (c_i, c_i)$</td>
</tr>
</tbody>
</table>

**COEFFICIENTS FOR QUADRATIC EQUATIONS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$c_r$</th>
<th>$c_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>$(m_1 a_i^2 + 5m_2 a_i^2 - 2c_i c_r) = (c_i, c_i)$</td>
<td>$(m_1 a_i^2 + 5m_2 a_i^2 - 2c_i c_r) = (c_i, c_i)$</td>
</tr>
<tr>
<td>$k_2$</td>
<td>$(m_1 a_i^2 + 5m_2 a_i^2 - 2c_i c_r) = (c_i, c_i)$</td>
<td>$(m_1 a_i^2 + 5m_2 a_i^2 - 2c_i c_r) = (c_i, c_i)$</td>
</tr>
<tr>
<td>$k_3$</td>
<td>$(m_1 a_i^2 + 5m_2 a_i^2 - 2c_i c_r) = (c_i, c_i)$</td>
<td>$(m_1 a_i^2 + 5m_2 a_i^2 - 2c_i c_r) = (c_i, c_i)$</td>
</tr>
</tbody>
</table>

**TABLE III**

**Trace of Algorithm for Triangular FOU**

**TABLE IV**

**Trace of Algorithm for Triangular FOU**

**TABLE III**

CHOICE OF COEFFICIENTS FOR QUADRATIC EQUATIONS

**TABLE IV**

**Trace of Algorithm for Triangular FOU**

**TABLE III**

CHOICE OF COEFFICIENTS FOR CUBIC EQUATIONS

**TABLE IV**

**Trace of Algorithm for Triangular FOU**

**TABLE III**

CHOICE OF COEFFICIENTS FOR QUADRATIC EQUATIONS

**TABLE IV**

**Trace of Algorithm for Triangular FOU**

**TABLE III**

CHOICE OF COEFFICIENTS FOR CUBIC EQUATIONS

**TABLE IV**

**Trace of Algorithm for Triangular FOU**

**TABLE III**

CHOICE OF COEFFICIENTS FOR QUADRATIC EQUATIONS

**TABLE IV**

**Trace of Algorithm for Triangular FOU**

**TABLE III**

CHOICE OF COEFFICIENTS FOR CUBIC EQUATIONS

**TABLE IV**

**Trace of Algorithm for Triangular FOU**

**TABLE III**

CHOICE OF COEFFICIENTS FOR QUADRATIC EQUATIONS

**TABLE IV**

**Trace of Algorithm for Triangular FOU**

**TABLE III**

CHOICE OF COEFFICIENTS FOR CUBIC EQUATIONS

**TABLE IV**

**Trace of Algorithm for Triangular FOU**

**TABLE III**

CHOICE OF COEFFICIENTS FOR QUADRATIC EQUATIONS

**TABLE IV**

**Trace of Algorithm for Triangular FOU**

**TABLE III**

CHOICE OF COEFFICIENTS FOR CUBIC EQUATIONS

**TABLE IV**

**Trace of Algorithm for Triangular FOU**

**TABLE III**

CHOICE OF COEFFICIENTS FOR QUADRATIC EQUATIONS

**TABLE IV**

**Trace of Algorithm for Triangular FOU**

**TABLE III**

CHOICE OF COEFFICIENTS FOR CUBIC EQUATIONS

**TABLE IV**

**Trace of Algorithm for Triangular FOU**
Algorithm 2 Pseudocode of Computing $c_r$ for a Piecewise Linear FOU

**Pseudo code of computing $c_r$ for a piece-wise linear FOU**

**Begin**

**Input:** Piece-wise linear FOU.

**Output:** $c_r$.

1. Partition the FOU into $n$ segments along the $x$-axis, such that each segment can be described by a piece-wise linear UF and LMF. Find the lower and upper bounds $[a_i, b_i]$ on the $x$-axis for each segment.
2. Obtain the equations for UMF and LMF for each segment $p_i$ in $[a_i, b_i]$.
3. For $i = 1$ to $n$

   **Cobegin**
   1. Find coefficients for segment $p_i$ using Table III and IV.
   2. Solve equation (20) when $m_i = m_0$, else solve equation (17) when $m_i \neq m_0$.
   3. Select $c_r = \text{root}$, if it lies in $[a_i, b_i]$.

**Coend**

**End For**

**End.**

<table>
<thead>
<tr>
<th>FOU</th>
<th>CKM when $e = 10^{-6}$</th>
<th>CKM when $e = 10^{-9}$</th>
<th>Proposed Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangular FOU, of Table II</td>
<td>$5 \times 0.0446$</td>
<td>$6 \times 0.0446$</td>
<td>$1 \times 0.0424$</td>
</tr>
<tr>
<td>Trapezoidal FOU, in Appendix.</td>
<td>$6 \times 0.0446$</td>
<td>$8 \times 0.0446$</td>
<td>$1 \times 0.0424$</td>
</tr>
<tr>
<td>Right end Shoulder FOU, in Appendix.</td>
<td>$5 \times 0.0446$</td>
<td>$6 \times 0.0446$</td>
<td>$1 \times 0.0424$</td>
</tr>
<tr>
<td>Left end Shoulder FOU, in Appendix.</td>
<td>$3 \times 0.0446$</td>
<td>$3 \times 0.0446$</td>
<td>$1 \times 0.0424$</td>
</tr>
</tbody>
</table>

$b_r$. Now, the primary variable is discretized into $n$ subintervals of width $q$. Thus, $b_r - (-b_r) = n.q$, and therefore

$$q = \frac{(b_r + b_l)}{n}. \quad (38)$$

Here, $c_r$ can have any value between its upper and lower bounds [41]. Thus, for $c_r$ to have the maximum precision, $q \rightarrow 0$, and by (38), we obtain $n \rightarrow \infty$. Setting $n \rightarrow \infty$ is tedious from computational point of view. Therefore, the complexity of finding $c_r$ with a high precision is quite high. We avoid using the closed-form formula to determine the end-point centroids, and it does not require any discretization. The proposed closed-form formula returns exact values of $c_r$ and $c_l$, which cannot be achieved by the iterative methods at a relatively lower cost.

Liu and Mendel [41] introduced a new approach to evaluate $c_r$ and $c_l$ using Newton–Raphson root-finding method Connect Karnik-Mendel (CKM) algorithm. This is also an iterative method similar to the KM algorithm, and a predefined error bound ($\varepsilon$) is required to terminate the algorithm. This method usually requires a less number of iterations and yields more precise results than that by the KM algorithm. Comparison of the proposed closed-form formula with the CKM algorithm is given next for different FOUs. Table entries here represent time in milliseconds and are expressed as a number of iterations required $\times$ average runtime per iteration. Such a representation makes the comparison easier.

Let us now consider a triangular FOU with the same parameters as given in Table II. For this FOU, we found $c_r = 0.2453596984$ (see Table II) using the closed-form formula. For the same triangular FOU, the CKM method takes five iterations for a predefined error bound of $10^{-5}$ and six iterations for a predefined error bound of $10^{-12}$ (see Table V). However, the CPU run time of the proposed method and the run time for one iteration of the CKM method are approximately the same and these are $0.0424$ ms and $0.0446$ ms, respectively. In Table V, we compare the run time required by the proposed method and that by the CKM method for four common FOUs with two predefined error bounds: $10^{-5}$ and $10^{-12}$. It is apparent from Table V that the proposed method requires the least runtime in comparison to the CKM both for $\varepsilon = 10^{-5}$ and $10^{-12}$.

V. REAL-TIME FUZZY CONTROL APPLICATION

Now we demonstrate the merits of the proposed T2 centroid computation technique over the existing ones with reference to a real-time control application. As a case study, we consider the well-known inverted pendulum system, which describes a standard model for the attitude control problem of a space booster during takeoff.

A. System Description

The inverted pendulum mounted on a motor-driven cart (Fig. 6) has the following system dynamics [45]:

$$(l(4M + m)/3)\ddot{\theta}(t) - (M + m)\dot{\theta}(t)g' = (-180/\pi)F(t) \quad (39)$$

where

1. $m$ = mass of the pole assumed to be concentrated at the center of the pendulum in kilogram
2. $M$ = mass of the cart in kilogram
3. $l$ = length of the pendulum in meter
4. $\theta(t)$ = deviation angle (at time $t$) from the dotted vertical line (Fig. 6) in the clockwise direction in degree
5. $T_p(t)$ = torque applied to the pole in the clockwise direction in Newton meter
6. $F(t)$ = controlled force acting on the cart from left to the right producing the counterclockwise torque $T_p(t)$ in Newton meter
7. $t$ = time in second
8. $g'$ = gravitational acceleration constant

With the parameter setting $l = 3(M + m)g'/4(M + m) + M = (180 - m)g'/\pi$, (39) reduces to

$$\ddot{\theta}(t) = \theta(t) - F(t). \quad (40)$$

Setting state variables $\dot{\theta}_1(t) = \theta(t)$ and $\dot{\theta}_2(t) = \ddot{\theta}(t)$, the dynamics (40) can be represented by the following state equations [44]:

$$\begin{align*}
\dot{\theta}_1(t) &= \theta(t) \\
\dot{\theta}_2(t) &= \ddot{\theta}(t) - F(t)
\end{align*} \quad (41)$$

Approximating $\dot{\theta}_i(t)$ by

$$\dot{\theta}_i(t) \approx \frac{\theta_i((k + 1)T) - \theta_i(kT)}{T}, i = 1, 2, \ldots, \infty \quad (42)$$

where $T$ is the sampling interval, we obtain the following discrete–time state space equation [44]:

$$\begin{align*}
\theta_1(kT + T) &= \theta_1(kT) + T\dot{\theta}_1(kT) \\
\theta_2(kT + T) &= T\dot{\theta}_1(kT) + \theta_2(kT) - TF(kT)
\end{align*} \quad (43)$$

for $k = 0, 1, 2, \ldots, \infty$. 

Fig. 7. (a) Block diagram of the proposed fuzzy control system with $\theta_0 = 0$. (b) Architecture of the fuzzy controller (here $D$ denotes a time-differentiation operation).

B. Fuzzy Control Using the Proposed Defuzzification

Fig. 7 provides a schematic architecture of the proposed fuzzy control for the inverted pendulum system. In Fig. 7(a), we have three modules: the fuzzy controller, a DC gain ($K$), and the plant in the closed-loop [45] inverted pendulum system. The fuzzy controller is driven by an error signal $\theta_e(kT)$, representing the positional difference of the set point, here $\theta_0=0^\circ$, and feedback signal, here $\theta(kT)$. The controller [Fig. 7(b)] inputs $\theta_e(kT)$ and $\theta(kT)$ to evaluate the desired control signal $u(kT)$ through a three-step process: fuzzification, inference generation, and defuzzification, all in IT2 space. The details of the above three steps are given below.

1) Fuzzification: Here, we fuzzify $\theta_1(kT)$, $\theta_2(kT)$, and $u(kT)$ into three grades: negative (N), near zero (NZ), and positive (P) using LES, triangular, and RES-type IT2 FOUs, respectively (see Fig. 8). The assumed domains of $\theta_1(kT)$, $\theta_2(kT)$ and $u(kT)$ are taken as $-5^\circ \leq \theta_1(kT) \leq 5^\circ$, -5 dps $\leq \theta_2(kT) \leq 5$ dps (dps=$\deg$ per second), and -10 Nm $\leq u(kT) \leq 10$ Nm.

2) Fuzzy Inference Generation: We use the following knowledge base for the present application.

- Rule 1: If $\theta_1(kT)$ is N and $\theta_2(kT)$ is P, then $u(kT)$ is NZ.
- Rule 2: If $\theta_1(kT)$ is NZ and $\theta_2(kT)$ is P, then $u(kT)$ is P.
- Rule 3: If $\theta_1(kT)$ is P and $\theta_2(kT)$ is P, then $u(kT)$ is P.
- Rule 4: If $\theta_1(kT)$ is N and $\theta_2(kT)$ is NZ, then $u(kT)$ is N.
- Rule 5: If $\theta_1(kT)$ is NZ and $\theta_2(kT)$ is NZ, then $u(kT)$ is NZ.
- Rule 6: If $\theta_1(kT)$ is P and $\theta_2(kT)$ is NZ, then $u(kT)$ is P.
- Rule 7: If $\theta_1(kT)$ is N and $\theta_2(kT)$ is N, then $u(kT)$ is N.
- Rule 8: If $\theta_1(kT)$ is NZ and $\theta_2(kT)$ is N, then $u(kT)$ is N.
- Rule 9: If $\theta_1(kT)$ is P and $\theta_2(kT)$ is N, then $u(kT)$ is NZ.

The inference engine fires appropriate rules based on the matching of the antecedent of the rules with the measured instantiation of the variables $\theta_1(kT)$ and $\theta_2(kT)$. If a single rule is fired, the inference is obtained directly due to firing of the rule. However, if the measured instantiations match with the premises of multiple rules, more than one rule will fire. The overall inference in that case is the fuzzy aggregation of the T2 inferences produced by each rule. In Example 1, we illustrate the process of generating inference due to firing of both single and multiple rules.

Example 1: Let us define the UMFs (and LMFs) for N/NZ/P MFs by adding the latter as suffix to the former. For example, LMF of set NZ be denoted by LMFNZ. Furthermore, when we refer to the LMFs, for $y = \theta_1$, $\theta_2$, or $u$, we write it as an argument of LMFs, that is, LMF$_y$. For example, when $x = \text{NZ}$ and $y = \theta_2$, we denote it by LMF$_x(NZ)$. Similar nomenclature is adopted for UMFS as well. Now, given $\theta_1(kT) = -1.5$ and $\theta_2(kT) = -3$ for certain non-negative integer $k$, it is apparent from the given rules that Rules 7, 4, 8, and 5 are fired as the antecedent parts of these rules are instantiable with the measured value of $\theta_1(kT)$ and $\theta_2(kT)$. We first explain the procedure for inference generation due to firing of Rule 7 in Fig. 9(a).

Let us define the UMFS (and LMFs) for $\theta_1(kT)$, $\theta_2(kT)$, and $u(kT)$ for N, NZ, and P set.

Fig. 8. IT2 FS of $\theta_1(kT)$, $\theta_2(kT)$, and $u(kT)$ for N, NZ, and P set.
For each segment in Fig. 9(e) for visual clarity, Fig. 9(f) is divided into six segments. The processors select appropriate coefficients \( p_1, p_2, p_3, p_4, p_5, \) and \( p_6 \) to serve the computations for six piecewise linear segments in parallel. The processors select appropriate parameters: \( k_1, k_2, \) and \( k_3 \) with an aim to determine \( c_l \) (or \( c_r \)). As \( c_l \) and \( c_r \) are unique (by Theorem 1), only processor P2 and P4 can determine \( c_l \) and \( c_r \), respectively, given in Table VII.

\[
\text{COEFFICIENTS OF ALL SIX PROCESSORS}
\]

<table>
<thead>
<tr>
<th>Processor</th>
<th>( k_1 )</th>
<th>( k_2 )</th>
<th>( k_3 )</th>
<th>( c_l )</th>
<th>( c_r )</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>64.8</td>
<td>148.88</td>
<td>-144.51</td>
<td>-7.2031</td>
<td>0.0424</td>
</tr>
<tr>
<td>P2</td>
<td>30</td>
<td>1185</td>
<td>1186</td>
<td>44.179</td>
<td>2.3911</td>
</tr>
<tr>
<td>P3</td>
<td>16.6</td>
<td>7353.2</td>
<td>7872.5</td>
<td>111.844</td>
<td>501.6</td>
</tr>
<tr>
<td>P4</td>
<td>40</td>
<td>2458.3</td>
<td>2458.3</td>
<td>2265.6</td>
<td>516.96</td>
</tr>
<tr>
<td>P5</td>
<td>-30</td>
<td>-131.25</td>
<td>-131.25</td>
<td>-4.7971</td>
<td>2.3911</td>
</tr>
<tr>
<td>P6</td>
<td>-30</td>
<td>-4.7971</td>
<td>-4.7971</td>
<td>2.3911</td>
<td>4.7971</td>
</tr>
</tbody>
</table>

For performance analysis, we require a few parameters introduced as follows.

1) Centroid Error \( E_c = |(c_l + c_r)/2 - (c_l + c_r)/2| \).

Here, \( c_l \) and \( c_r \) are the actual left and the right centroids, respectively, produced by the proposed algorithm, while \( c_l' \) and \( c_r' \) are the left and the right centroids, respectively, produced by either KM or CKM algorithm for a given error bound.
TABLE VIII

<table>
<thead>
<tr>
<th>Parameters</th>
<th>KM</th>
<th>CKM</th>
<th>Single Processor</th>
<th>Dual Processor</th>
<th>Fully Parallel Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>(E_0)</td>
<td>0.0816</td>
<td>0</td>
<td>0.2078</td>
<td>0</td>
<td>0.1368</td>
</tr>
<tr>
<td>(T_{CPU}) (ms)</td>
<td>6.77</td>
<td>0.223</td>
<td>0.4294</td>
<td>0.2146</td>
<td>0.0424</td>
</tr>
</tbody>
</table>

TABLE IX

<table>
<thead>
<tr>
<th>Sampling Rate</th>
<th>KM</th>
<th>CKM</th>
<th>Single Processor</th>
<th>Dual Processor</th>
<th>Fully Parallel Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f_s=147) Hz</td>
<td>3.1848</td>
<td>0.1615</td>
<td>0.2078</td>
<td>0.1603</td>
<td>0.1368</td>
</tr>
<tr>
<td>(f_s=2328) Hz</td>
<td>(\infty)</td>
<td>1.215</td>
<td>2.3962</td>
<td>1.1279</td>
<td>0.2361</td>
</tr>
<tr>
<td>(f_s=4484) Hz</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>2.2657</td>
<td>0.5248</td>
</tr>
<tr>
<td>(f_s=4659) Hz</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>2.3911</td>
<td>0.5423</td>
</tr>
<tr>
<td>(f_s=5555) Hz</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>(\infty)</td>
</tr>
</tbody>
</table>

2) CPU Time: \(T_{CPU} = \text{Runtime per iteration in millisecond.}\)

3) RMS: \(\text{error} = \sqrt{\frac{\sum_{t=0}^{T}(\theta(t) - \theta_0)^2}{T_S}}\)
   where \(\theta_0 = 0\) and \(T_S\) is total time span.

4) Maximum Overshoot: The maximum positive deviation of the output, \(\theta(t)\), with respect to its desired value is known as the maximum overshoot and is denoted by \(M_p\). Thus

\[
M_p = \theta(t)_{\text{max}} - \theta_0 = \theta_1(t)_{\text{max}} \quad \text{(Since, } \theta_0 = 0).\]

Now, with the initial condition, \(\theta_1(kT) = -1.5\) and \(\theta_2(kT) = -3\), the centroid error and CPU time for five algorithms including the KM algorithm, the CKM algorithm, single processor (SP) of the proposed algorithm, dual processor (DP) of the proposed algorithm, and fully parallel processing of the proposed algorithm are given in Table VIII. The error bound \(\varepsilon\) was selected as \(10^{-12}\).

Now, we accelerate the inverted pendulum dynamics by increasing the sampling rate \(f_s = 144\) Hz. We determine the maximum overshoot \((M_p)\) and Root Mean Square (RMS) error at this sampling rate. We then continue increasing sampling rate and note special circumstances when \(M_p\) and RMS error approaches to a large value \(>10^3\), hereafter called infinity. The appropriate sampling rates for which we observe \(M_p\) and RMS error to be infinity by traditional algorithms and SP/DP realization of our algorithm are indicated in Tables IX and X, respectively.

It is observed from Table X that for the fully parallel realization of our scheme, the system has an \(M_P\) of \(0.998^\circ\) for a very high sampling rate of \(5555\) Hz.

In Fig. 10, we present the system response for different sampling rates. Here, the \(x\)-axis represents time and the \(y\)-axis represents the system output (\(\theta_1\)). Fig. 10(a)–(l) represents the stable system response that varies around \(0^\circ\) with a maximum overshoot of \(5.3211^\circ\). The infinite values of RMS error or Maximum Overshoot (MP) in Tables IX and X correspond to unstable system. Fig. 10(m) demonstrates the behavior of an unstable system.

D. Performance Merit of IT2 FS Over T1 Fuzzy Controller

Measurement noise often creeps into plant/process parameters or system states because of poor sensor characteristics (due to sensor aging, manufacturing defect, or poor environmental ambience) [44]. It can have different probability distributions, such as Gaussian, Poisson, or random. Here, we inject a random measurement noise in the system state \(\theta_1(kT)\) of the inverted pendulum system (plant). Let \(N(kT)\) be the amplitude of the noise injected into the measurement of \(\theta_1(kT)\) at time \(t = kT\), \(k = 0, 1, 2, \ldots, \infty\). Fig. 11 provides an adaptive tuning scheme of the MFs used by the fuzzy controller. To compare the relative merit of the IT2 FS controller over its T1 counterpart, we repeat the simulation of Fig. 11 for both IT2 and T1 fuzzy systems.

Let \(\theta_{c}(kT) = \theta_0 - \theta_1(kT)\) be the error signal obtained at the output of the error detector (circle in Fig. 11). Here, we attempt to determine the fuzzy controller parameters [the slopes of the straight lines used in the MFs: N, NZ, and P] of the IT2 (or T1) fuzzy systems. For \(\theta_{c}(kT)\), we repeat the simulation of Fig. 11 for both IT2 and T1 fuzzy systems. We initialize trial solutions called parameter vectors, the components of which include slopes of the IT2 (or T1) MFs, N, NZ, and P for both IT2 and T1 fuzzy systems. We use the well-known differential evaluation (DE) algorithm for its computational speed with appreciably good accuracy, small program code, and few control parameters (scale factors [46]).

The evolutionary adaptation scheme for MF refinement for \(\theta_1(kT), \theta_2(kT), \text{ and } u(kT)\) works as follows.

1) Initialize trial solutions called parameter vectors, the components of which include slopes of the IT2 (or T1) MFs, N, NZ, and P for \(\theta_1(kT), \theta_2(kT), \text{ and } u(kT)\), with constant settings of all other parameters in the MFs.

2) Obtain a finite number of error samples \(\theta_{c}(kT), k = 0, 1, 2, \ldots, k_{\text{Max}}, \text{ keeping the MF tuner on top of Fig. 11, switched off (by using switch S) during these sampling intervals.}\)

3) Run one complete iteration of the DE algorithm to optimize the performance index \(J \approx \sum_{k=0}^{k_{\text{Max}}} \theta_{c}^2(kT)\) by adaptation of slope of the MFs: \(\theta_1(kT), \theta_2(kT), \text{ and } u(kT)\).
4) Set $k=0$ and repeat from step 2 until $J$ is smaller than a user-defined threshold.

5) Use the modified slopes of the MFs in the fuzzifier [Fig. 7(b)] of the fuzzy controller (Fig. 11).

The simulation was performed on a uniprocessor (Pentium P5) using MATLAB R2009a under WINDOWS 97. But, in practice, the adaptive tuner (top box in Fig. 11) and the fuzzy controller should be realized on two distinct processors. The objective of such realization is to keep the controller free to execute its task, leaving aside the computational overhead of the MF tuner. The uniprocessor realization undertaken here steals several sampling intervals after every 50 control cycles to execute one iteration of DE. This is repeated until convergence.
of DE is obtained. In our simulation, we allow DE to produce the controller parameters after $k_{\text{max}} = 25$ sampling intervals.

Choice of $k_{\text{max}}$ here is an important issue. A large setting of $k_{\text{max}}$ degrades the performance of the evolutionary MF tuner at the cost of fewer control cycles. A very small setting of $k_{\text{max}}$, on the other hand, degrades the controller performance as several control cycles are lost to improve the performance of the evolutionary MF tuner. A setting of $k_{\text{max}} = 25$ is found to be a good choice that balances the performance of both evolutionary tuner and the controller. Fig. 12 demonstrates adaptation of MFs for IT2 realization of the controller in Fig. 11.

The response of the plant $\theta_1(kT + T)$ is shown in Fig. 13 with measurement noise introduced in $\theta_1(kT)$. It is apparent from Fig. 13 that the IT2 FS realization results in a small peak overshoot in the plant response after injection of noise in $\theta_1(kT)$. The RMS error defined in Section V is found to be smaller in IT2 FS realization in comparison to T1 realization.

VI. CONCLUSION

This paper presents closed-form formulas, $c_r$ and $c_l$, for standard FOUs, such as triangular, trapezoidal, LES, and RES, already introduced by Mendel and Wu [19], [20] along with generic piecewise linear FOU. The merit of the work lies in evaluating the exact, crisp values of $c_r$ and $c_l$, instead of their bounds, $[\bar{c}_r, \underline{c}_r]$ and $[\bar{c}_l, \underline{c}_l]$, as computed in [19]. The evaluated upper and lower values of centroids offer an exact span of uncertainty in the T2 defuzzified space. The formulation for computing $c_r$ and $c_l$ does not require any iterative procedure as in [16] and [41], where the results of computation depends on the interval of the discretized primary variable or predefined error bound. Thus, the current approach makes it possible to measure the exact uncertainty of an IT2 FS using the geometric parameters of FOU of such FS. The proposed technique of fast computation of centroids has been successfully applied in real-time IT2 fuzzy control system. The study undertaken with measurement noise in the plant demonstrates that the IT2 fuzzy control outperforms its T1 counterpart with respect to peak overshoot and RMS error.

APPENDIX

A. Uniqueness of $c_r$

Theorem 1: The right end-point centroid, $c_r$, is unique irrespective of the functional forms of UMF ($\bar{\mu}_A(x)$) and LMF ($\bar{\mu}_C(x)$).

Proof: Suppose, $c_r$ has two possible values $\beta$ and $\gamma$, $\beta > \gamma$. Now, dropping $A$ from $\bar{\mu}_A(x)$ and $\bar{\mu}_C(x)$ in (13) for the sake of simplicity, we rewrite it for $c_r = \beta$ and $c_r = \gamma$ as follows:

$$\beta = \frac{\int_{-\infty}^{\beta} \mu(x)dx + \int_{\beta}^{\infty} \bar{\mu}(x)dx}{\int_{-\infty}^{\beta} \mu(x)dx + \int_{\beta}^{\infty} \bar{\mu}(x)dx}$$ (A1)

and

$$\gamma = \frac{\int_{-\infty}^{\gamma} \mu(x)dx + \int_{\gamma}^{\infty} \bar{\mu}(x)dx}{\int_{-\infty}^{\gamma} \mu(x)dx + \int_{\gamma}^{\infty} \bar{\mu}(x)dx}$$ (A2)

Simplification of (A1) and (A2) yields

$$\int_{-\infty}^{\beta} \mu(x)dx + \int_{\beta}^{\infty} \bar{\mu}(x)dx = \int_{-\infty}^{\beta} \mu(x)dx + \int_{\beta}^{\infty} \bar{\mu}(x)dx$$

and

$$\int_{-\infty}^{\gamma} \mu(x)dx + \int_{\gamma}^{\infty} \bar{\mu}(x)dx = \int_{-\infty}^{\gamma} \mu(x)dx + \int_{\gamma}^{\infty} \bar{\mu}(x)dx.$$ (A3)

By taking difference of (A3) from (A4), we have

$$\int_{-\infty}^{\beta} \mu(x)dx - \int_{-\infty}^{\gamma} \mu(x)dx + \int_{\gamma}^{\infty} \mu(x)dx - \int_{-\infty}^{\gamma} \bar{\mu}(x)dx + \int_{\gamma}^{\infty} \bar{\mu}(x)dx = \int_{-\infty}^{\beta} \mu(x)dx - \int_{-\infty}^{\gamma} \mu(x)dx + \int_{\gamma}^{\infty} \mu(x)dx - \int_{-\infty}^{\gamma} \bar{\mu}(x)dx + \int_{\gamma}^{\infty} \bar{\mu}(x)dx.$$ (A5)

Since $\beta > \gamma$, we set $\beta = \gamma + \delta$ for $\delta > 0$ in (A5) and, thus, obtain

$$\int_{-\infty}^{\beta} (\gamma + \delta)\mu(x)dx - \int_{-\infty}^{\gamma} \gamma \mu(x)dx + \int_{\gamma}^{\infty} (\gamma + \delta)\bar{\mu}(x)dx - \int_{-\infty}^{\gamma} \gamma \bar{\mu}(x)dx + \int_{\gamma}^{\infty} \gamma \mu(x)dx = \int_{-\infty}^{\beta} \mu(x)dx - \int_{-\infty}^{\gamma} \mu(x)dx + \int_{\gamma}^{\infty} \mu(x)dx + \int_{-\infty}^{\gamma} \bar{\mu}(x)dx + \int_{\gamma}^{\infty} \bar{\mu}(x)dx = 0$$ (A6)

$$\Rightarrow \int_{-\infty}^{\beta} \mu(x)dx + \int_{\beta}^{\infty} \mu(x)dx = \int_{-\infty}^{\gamma} \mu(x)dx + \int_{\gamma}^{\infty} \mu(x)dx + \int_{-\infty}^{\gamma} \bar{\mu}(x)dx + \int_{\gamma}^{\infty} \bar{\mu}(x)dx = 0$$

$$\Rightarrow \int_{-\infty}^{\beta} \delta \mu(x)dx + \int_{\beta}^{\infty} \mu(x)dx = 0$$

As $\delta > 0$, $\mu(x) > 0$, $\bar{\mu}(x) > 0$, and $\mu(x) > \mu(x)$, for all $x$ and $\beta > \gamma$, so all the terms of (A6) are non-negative. The sum of these non-negative terms is equal to zero that ensures the following.

1) $\int_{-\infty}^{\beta} \delta \mu(x)dx = 0$ yielding either $\delta = 0$ or $\mu(x) = 0$, for $-\infty \leq x \leq \beta$.
2) $\int_{\beta}^{\infty} \delta \bar{\mu}(x)dx = 0$ yielding either $\delta = 0$ or $\bar{\mu}(x) = 0$, for $x < -\infty$.
3) $\int_{-\infty}^{\gamma} (x - \gamma)\bar{\mu}(x)dx = 0$ yielding either $\beta = \gamma$ (i.e., $\delta = 0$) or $\bar{\mu}(x) = 0$, for $\gamma \leq x \leq \beta$.

To satisfy all the above cases the only condition that remains is $\beta = \gamma$ (i.e., $\delta = 0$), indicating the uniqueness of $c_r$. ■

B. Computation of $c_r$ and $c_l$ for Trapezoidal, RES, and LES FOU

Computation of $c_r$ and $c_l$ for trapezoidal [Fig. 14(a)], RES [Fig. 14(b)], and LES [Fig. 14(c)] has been performed using (13) and (12), respectively. The parameters of the resulting cubic equation of the form (17) thus obtained for all the three
FOUs mentioned above are listed in Table XI. Table XI reveals that the coefficients \( k_1 \), \( k_2 \), and \( k_3 \) for individual FOU vary widely depending on the range of \( c_r \) and \( c_l \). To obtain \( c_r \) (or \( c_l \)), we, therefore, need to solve the cubic equation by (19) for different ranges of \( c_r \) (or \( c_l \)). Following Theorem 1, it is clear that only one solution of (19) in a particular range of \( c_r \) (or \( c_l \)) would return the right value of \( c_r \) (or \( c_l \)). Thus, for each range of \( c_r \) (or \( c_l \)), the solution (19) that falls in the corresponding range of \( c_r \) (or \( c_l \)) is declared as the acceptable right or left-end centroid.

**References**


