ARTICLE IN PRESS FSS:7869

Available online at www.sciencedirect.com

ScienceDirect

Fuzzy Sets and Systems ••• (••••) •••–•••

FUZZY sets and systems

www.elsevier.com/locate/fss

Alpha-cut representation used for defuzzification in rule-based systems

Amir Pourabdollah [a,*], Jerry M. Mendel [b], Robert I. John [c]

[a] *Nottingham Trent University, UK* [1]
[b] *University of Southern California, USA* [2]
[c] *University of Nottingham, UK* [3]

Received 27 September 2019; received in revised form 13 May 2020; accepted 15 May 2020

This paper is dedicated to the memory of Professor Robert I. John

Abstract

Alpha-cut representation of fuzzy sets has been used as a basis for fuzzy numbers ranking in some applications but rarely used for defuzzification of rule-based systems or fuzzy controllers. Moreover, such alpha-cut defuzzification (called ACD here) is not yet formally linked to the membership function (MF) or to the common MF-based defuzzification methods, namely the centroid. The ACD can be considered as a generalisation of the similar algorithms in fuzzy numbers to any fuzzy set. A close-form formula for ACD is developed that involves both MF and its derivative, which shows that ACD reflects both static and dynamic aspects of a fuzzy set. Moreover, formal links between ACD and some MF-based defuzzification methods are shown. Through two groups of experiments, the utility of the new method is compared with centroid defuzzification. Particularly, we examined how the ACD significantly outperforms the centroid for noisy time-series prediction. Finally, the computation complexity of ACD is shown to be about the same as the centroid method, for convex fuzzy sets. Our tests suggest that ACD can be considered as a viable alternative defuzzification method for fuzzy system designers.

Keywords: Alpha-cut; Defuzzification; Fuzzy sets

1. Introduction

In a fuzzy system, it is the defuzzifier, which is a mapping from a fuzzy set (FS) into a real number, that produces its crisp output [1]. Many such mappings have been reviewed and/or proposed in the literature (e.g., in [2–5]), the

[*] Corresponding author.
E-mail address: amir.pourabdollah@ntu.ac.uk (A. Pourabdollah).
[1] Department of Computing and Technology, Clifton Campus, Nottingham, NG11 8NS, UK.
[2] University of Southern California, Los Angeles, CA 90089-2564, USA.
[3] School of Computer Science, Jubilee Campus, Nottingham, NG8 1BB, UK.

https://doi.org/10.1016/j.fss.2020.05.008
0165-0114/© 2020 Elsevier B.V. All rights reserved.

most popular of which arguably involve some sort of weighted averaging over their membership function (MF). The centre of gravity (COG), arguably the most widely-used defuzzification method for fuzzy rule-based systems [6,2], is an example of defuzzification methods based on representing a FS by its MF. However, a FS can also be represented by its α-cuts, the importance of which is that it is directly linked to confidence levels, i.e., each α-cut indicates a confidence interval at its corresponding certitude level α.

The two algorithmic approaches to defuzzification, i.e., based on MF representation and α-cut representation, are already known in the context of fuzzy numbers as support-based and level-based methods respectively [7], where they are mainly used for ranking the fuzzy numbers. However, to the best of our knowledge, the similar level-based algorithms have not been applied out of the fuzzy numbers context, e.g. for rule-based fuzzy controllers, in which non-convex, multi-modes and non-normal fuzzy sets at the system's output are to be defuzzified.[4] This motivates working on several topics: 1) formulating a generalised *α-cut defuzzification* (called ACD here) applicable for any type of fuzzy sets based on their MFs; 2) Linking the ACD formulation to some known MF-based defuzzification methods; 3) Developing a minimal computational models for ACD comparable to those of the common methods; and 4) Investigating the performance of fuzzy rule-based systems if ACD is used. All of these topics are to be addressed in this paper.

The provided close-form formula for ACD involves both MF and its derivative. This shows that ACD represents both static and dynamic aspects of a fuzzy set. Even though their underlying representations are different, it will be shown that there are formal links between ACD and the MF-based methods. It is also shown that the new method and COG are special cases of a general framework already known as Weighted-Function COG, introduced in [2]. Since MF is the basis for the support-based methods (such as COG), this means that a formal gap between the two defuzzification approaches is bridged. Moreover, a simple computational model is developed for ACD that is comparable to that of the COG.

COG has been chosen as a benchmark support-based method in this paper, so that the properties, formulations and the performance of the level-based ACD are compared to it. The rationale for this selection is not only for being the most widely used method, but also because ACD can be formally compared to a wider group of methods based on MF averaging. As will be shown in Section 5, COG, along with some other methods such as FOM, LOM, MOM, MOS, FOS, LOS and BADD[5] can be generalised as averaging methods over the MF, but with different weighting factors [2].

Introducing another defuzzification method may seem unnecessary to the reader, but the outcomes of the experiments described in sections 4 and 5, as well as an axiomatic study provided in section 3, support the development of ACD. In particular, we will show by means of some examples, that significantly better time-series forecasting results are obtained when ACD is used versus when COG is used. During a formal analysis in Section 3, we show that the derivative of MF appears in the ACD formulation as a weighting function. This means that ACD reflects both the static and the dynamic properties of the MF, whereas other MF-based methods (e.g., COG) solely focuses on the static characteristics. This may explain why the ACD outperforms in our experiments, and suggests that ACD should be considered as a viable alternative defuzzification method for fuzzy system designers.

2. A generalised alpha-cut defuzzification of fuzzy sets

2.1. Definitions

We start from defining COG and α-cut before explaining ACD. ACD will be introduced for general fuzzy sets, i.e. by dropping the fuzzy number-specific requirements such as normality, convexity and uni-modality.

2.1.1. Centroid, or Centre of Gravity Defuzzification (COG)

The centroid of a FS with a MF represented as $\mu(x)$ along its x-axis is defined in continuous and discrete modes, respectively as:

[4] This will be detailed in Section 2.

[5] Other defuzzification approaches found in the literature [3] are not considered here in order to limit the scope of this paper.

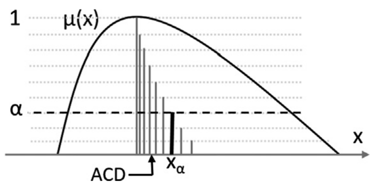

Fig. 1. The process of calculating the ACD: The spike generated for each α, of height α, is located at the average value of its corresponding α-cut.

$$COG = \frac{\int x\mu(x)dx}{\int \mu(x)dx}; \tag{1}$$

$$COG = \frac{\sum_i x_i \mu(x_i)}{\sum_i \mu(x_i)} \tag{2}$$

2.1.2. α-Cut

An α-cut is a subset of the universe whose membership grades are greater than or equal to $\alpha \in [0, 1]$ [8]. A FS can be represented as the fuzzy union of its α-cuts multiplied by the respective levels.

2.1.3. α-Cut defuzzification (ACD)

It is intuitive to consider the individual contributions of each α-cut, in which the x-values that appear in more α-cuts should have a greater contribution in defuzzification than the others. Based on this idea, the ACD of a FS with a MF $\mu(x)$ is defined here as a two-step process (see Fig. 1): (1) for each α-level, the average of all x-values that belong to the corresponding α-cut is calculated (x_α); then, a spike of height α is placed at x_α. (2) A weighted average of all these spikes is calculated, in which the x-value of each spike is weighted by its corresponding α-levels. In other words, the x_α values with higher α have a greater contribution to the final defuzzification result than those with lower α. Formally, if μ is the MF and each α-cut is a set of points along x-axis called $[\mu]_\alpha$, then the ACD in discrete mode is defined as:

$$ACD = \frac{\sum_i \alpha_i \overline{[\mu]_{\alpha_i}}}{\sum_i \alpha_i} \quad i = 1 \ldots N \tag{3}$$

where $\overline{[\mu]_{\alpha_i}}$ represents the average of the ith α-cut over all its values along the x-axis, and N is the number of discretisation levels along the vertical axis. It is notable that (3) is directly taken from a FS representation.

We also notice that the peaks of the produced spikes in step (1) may not collectively show a single-valued continuous function of x, since different spikes may overlap on a single x-value while other x-values may not have any correspondent spike. Thus, step (2) should not be taken as an equivalent to calculating a form of COG over a function produced in step (1).

For any non-flat area of a convex[6] set's MF, x_α is the average of two crossing points of the MF with each α level, i.e., the centre of the corresponding α-cut (see Fig. 1). Additionally, if the FS is non-convex, it is possible that some of the generated α-cuts are not continuous along the x-axis; however, (3) still holds, i.e. x_α is still the average of all x values that belong to the corresponding α-cut. In this case, it is even possible that x_α does not belong to the corresponding α-cut. For the special case of a flat area (plateau) of a MF, x_α is still the centre of the plateau.

Finally, for non-normal FSs, the α levels have a maximum less than 1, however this does not change (3), because the ACD process stops when the maximum α is reached (i.e., the empty α-cuts produce zero-height spikes). If α_m is the maximum α level, the generalisation of (3) to a continuous universe becomes:

[6] We use the FS convexity definition as provided in [9,10]. Convexity always occurs when the FS's MF is first monotonically non-decreasing and then monotonically non-increasing [11].

$$ACD = \frac{\int_0^{\alpha_m} \alpha \overline{[\mu]_\alpha} d\alpha}{\int_0^{\alpha_m} \alpha d\alpha} = \frac{2}{\alpha_m^2} \int_0^{\alpha_m} \alpha \overline{[\mu]_\alpha} d\alpha \tag{4}$$

2.2. *ACD and fuzzy numbers*

Fuzzy numbers, as defined in [12] is a generalisation of ordinary numbers to uncertain values by representing the number as a fuzzy set, in which the fuzzy set is restricted to be normal, convex and uni-modal.[7] Defuzzification in this context is the process of choosing a representative ordinary value, and is mostly used for ranking the fuzzy numbers. Many support-based and level-based defuzzification algorithms are developed for fuzzy numbers, some of which reviewed here are closely related to the ACD.

As shown in ACD definition, averaging weights are given to the α-cuts, so that more contributions are given to the higher αs. A flat averaging of all midpoints of the alpha-cuts, i.e., without weighting, is also introduced as a defuzzification method by Yager in [14], called ALC (Averaging Level Cuts) in [15] too. The ALC algorithm is also equivalent to calculating the *Expected Value (EV)* of a fuzzy number as introduced by Oussalah in [15] and Heilpern in [16]. The *value* of a fuzzy number is defined by Delgado et al. in [13], based on using α-levels as weighting factors in averaging the α-cut midpoints. One can show that this approach is mathematically equivalent to ACD. However, the justification provided in [13] for choosing such weights is limited to comparing the graphical shape of the MFs when ranking different fuzzy numbers.

Giving flexible (e.g., non-linear) weighs to α-levels is another known method for fuzzy number defuzzification. This flexibility enables the designers to emphasise or de-emphasise certain α levels according to application-dependent purposes or subjective interpretation of those levels. In [17], subjective and application-dependent weights for α levels are introduced. A more generalised formulation is introduced in [18], later called WABL (Weighted Averaging Based on Levels) in [7]. WABL considers a more flexible averaging weights for different α levels by introducing a probability density function (PDF) along α levels called $p(\alpha)$ for $\alpha \in [0, 1]$. By this generalised algorithm, EV (or equivalently, ACD) is a special case of the WABL when $p(\alpha) = 2\alpha$ (i.e., a linear PDF). ALC [15] can also be considered as another special case of the WABL where $p(\alpha) = 1$ (i.e., a constant PDF). We notice that in [7], a formal link between WABL and COG is provided but only for the cases of triangular and trapezoidal MFs. In Section 5, formal links between ACD, MF and COG will be developed for general fuzzy sets.

Within WABL framework, it is arguable why a linear PDF, among all other possible PDFs, is used in this paper for ACD algorithm. We notice that in many fuzzy rule-based systems such as fuzzy controllers, the output fuzzy set is a result of a highly non-linear inference algorithm such as Mamdani's [19], so that the output fuzzy sets may have a very irregular MF shape, not comparable to that of a fuzzy number. These MFs are usually non-convex, multi-modes and non-normal. Particularly, they are not usually meaningful or interpretable by a human. Even if they satisfy the conditions of being fuzzy numbers, it is unlikely that a designer can treat them as fuzzy numbers, e.g., rank them or assign weights to them subjectively. That is why an intuitive way of linear weighting is used for ACD rather than fine tuning the parameters of a generalised WABL. While linking the tuned parameters to subjective human-interpretable concepts in many irregular fuzzy controller outputs seems impractical, it is intuitive to assign the higher weights to the higher α levels linearly.

Similarly, the ideas of interval-valued defuzzification and the mean of fuzzy numbers initially suggested by Dubois and Prade [20] are another areas of research for defuzzifying a fuzzy number having a level-based approach. Having a gradual number approach to fuzzy interval (instead of a fuzzy set approach), as proposed by Fortin, Dubois and Fargie in [21] also leads to a similar two-step level-based defuzzification to what Yager suggested for fuzzy numbers [14], and to what we propose for general fuzzy sets.

To the best of our knowledge, linear level-based defuzzifications have not been utilised out of the fuzzy numbers or fuzzy interval contexts, nor have been formally linked to other defuzzification algorithms that were discussed. The

[7] The uni-modality condition is lifted in some papers, such as in [13].

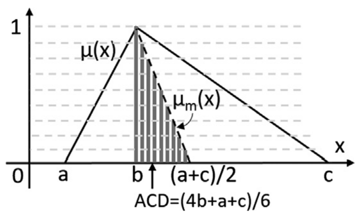

Fig. 2. Calculating ACD for a triangle MF. The generated spikes collectively define $\mu_m(x)$, so that the ACD is the centroid of the shaded triangle.

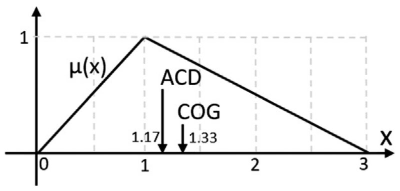

Fig. 3. An example of a triangle MF with calculated ACD and COG.

ACD can be considered as a generalisation of the mathematically-equivalent algorithms from fuzzy numbers to any FS, in which the fuzzy number-specific requirements on the MF, such as normality, convexity and uni-modality, are eliminated.

2.3. Example: ACD and COG of triangle and trapezoidal MFs

As an example, we consider the triangle MF shown in Fig. 2, in which the spikes produced by averaging the α-cuts are shown. Given the fact that the spikes are evenly distributed, it is observed that the spikes collectively form a right triangle, in which the ACD is its centroid along the x-axis. Consequently,[8]

$$ACD = b + [(a+c)/2 - b]/3 = (4b + a + c)/6 \tag{5}$$

Using (1), the COG of the same triangle MF is:

$$COG = (a + b + c)/3 \tag{6}$$

The difference between the two defuzzified values is $(a - 2b + c)/6$ or $ACD - b$. For example, if a=0, b=1 and c=3, then COG=1.33 and ACD=1.17 (see Fig. 3). The difference between the two values is 0.17 (a 12.5% change).

A trapezoidal MF is represented by four parameters a, b, c and d along x-axis. It can be shown using (1) and (3), that:

$$COG = \frac{c^2 + d^2 - a^2 - b^2 - ab + cd}{3(c + d - a - b)} \tag{7}$$

$$ACD = (a + 2b + 2c + d)/6 \tag{8}$$

E.g., for $(a, b, c, d) = (0, 1, 2, 4)$, $COG = 1.66$ and $ACD = 1.80$.

2.4. Another example: a double-singleton fuzzy set

The differences between COG and ACD can be further demonstrated by examining the simple case of a double-singleton fuzzy set, i.e., the union of two fuzzy singletons. Let us consider a set that contains two unequal singletons on points a and b having membership grades s_1 and s_2 respectively ($s_1 \leq s_2$), as shown in Fig. 4.

According to (1), the COG of the MF shown in Fig. 4 is:

$$COG = \frac{as_1 + bs_2}{s_1 + s_2} \tag{9}$$

[8] The centroid of a right triangle is one-third of its leg away from the right angle (proof is left to the reader).

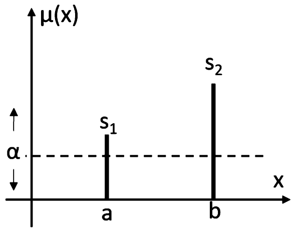

Fig. 4. A double-singleton fuzzy set, i.e., a union of two fuzzy singletons.

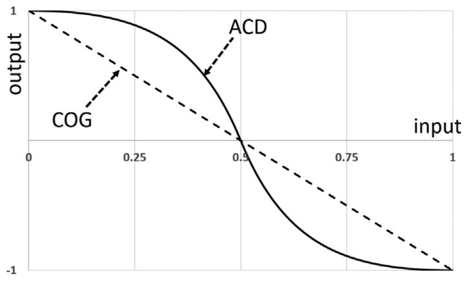

Fig. 5. The input-output mapping of the designed simplistic FLS, comparing the results when COG and ACD are employed alternatively.

For calculating the ACD of this MF, we notice that for $0 \le \alpha \le s_1$, the midpoint of all α-cuts is $(a+b)/2$. For $s_1 < \alpha \le s_2$, the midpoint is b since the singleton located on point a is discontinued. Finally for $s_2 < \alpha \le 1$, the ACD calculation process stops. According to (4), the ACD of this MF becomes:

$$ACD = \frac{\int_0^{s_1} \frac{a+b}{2} \alpha d\alpha + \int_{s_1}^{s_2} b\alpha d\alpha}{\int_0^{s_2} \alpha d\alpha} = \frac{\frac{a+b}{2} s_1^2 + b(s_2^2 - s_1^2)}{s_2^2} = \frac{a-b}{2}\left(\frac{s_1}{s_2}\right)^2 + b \tag{10}$$

As mentioned earlier, ACD has a potential to be used as the defuzzifier of a fuzzy controller. In order to practically realise the difference between the above two formulas, consider a very simplistic fuzzy controller that produces an output fuzzy set in the form of Fig. 4, then we will compare how the FLS's outputs using the two defuzzification methods vary in respect to the FLS's input. Moreover, although FLSs generally produce a highly non-linear input-output mapping, such simplistic FLS can be exceptionally designed in a way that produces a linear map if COG is employed.

The designed FLS is a single-input single-output system with input $x \in [0, 1]$, and output $y \in [-1, 1]$. The antecedent sets are two triangular MFs evenly located around 0 and 1 (labelled "low" and "high"), and the single consequent set consists of two normal singletons located at -1 and 1 (labelled "low" and "high" too). The rules are: if x is low then y is low; and if x is high then y is high. Using min-max Mamdani inference, simple calculations show that the output fuzzy set before defuzzification is a union of two singletons located at -1 and 1. This becomes similar to Fig. 4, with $a = -1$, $b = 1$, $s_1 = x$ and $s_2 = 1 - x$.

Replacing the parameter values in (9) yields to $COG = 1 - 2x$. This is a linear mapping function due to the over-simplicity of the system and the fact that $(a+b)/2$ factor in (10) eventually becomes 0 in this example. For ACD, replacing a, b, s_1 and s_2 in (10)[9] yields to a non-linear mapping, as:

$$ACD = \begin{cases} \frac{1-2x}{(1-x)^2} & for \quad 0 \le x \le 0.5 \\ \frac{1-2x}{x^2} & for \quad 0.5 \le x \le 1 \end{cases} \tag{11}$$

Fig. 5 shows the input-output mapping functions of the designed FLS when COG and ACD are applied alternatively. The two algorithms produce similar results when the input is at the middle or at the endpoints of its range. Compared to COG, ACD shows a more tendency towards producing values near 1 and -1 and a faster transition over

[9] Consider that (10) was developed assuming $s_1 \le s_2$.

producing values near 0.5. Notably, in the case that COG produces a linear function due to the system's simplicity, the system is complex enough to make a non-linear mapping by ACD. This can be explained by knowing the fact that by definition, ACD is a two-pass algorithm whereas COG is a one-pass. The ACD algorithm first applies an averaging along each α cut, then it uses the results of the first pass for applying another weighted averaging over different α-levels. COG on the other hand, involves a single round of weighted averaging.

Although the above observation is limited to a specific FLS, one may conclude that ACD is computationally more complex. We will analyse this in the next sub-section and in sub-section 3.7.

2.5. A computational model

As (3) suggests, an algorithm for ACD computation needs nested loops, where the outer and inner loops increment the α level and the x value, respectively. According to (3), the average of each α-cut is used to locate a spike of height α. These spikes are then weighted-averaged over the x-axis. This algorithm does not require the FS to be convex.

Listing 1 shows a sample program implemented for ACD computation. Two arrays $x[]$ and $mf[]$ collectively represent the FS's MF, and the parameter dis_levels indicates the number of discretisation levels. In Listing 1, averaging each α-cut and calculating the spikes' weighted average are both implemented within a single loop, so that the numerator and the denumerator in (3) are computed at the same time. Using this technique, the algorithm has become relatively simple.

```
double AlphaCutDefuzzification
(double x[], double mf[], int disc_levels){
        double numerator = 0.0, denumerator = 0.0;
        double x_Sum = 0.0, x_Count =0.0;
        for (double alpha = 0 ; alpha <= 1 ;
                alpha += 1.0/disc_levels){
                for (int i=1; i<=disc_levels; i++){
                        if (mf[i] >= alpha){
                        xSum ++ x[i];
                        xCount++;
                }
                numerator += alpha * xSum / xCount;
                denumerator += alpha;
        }
        return(numerator / denumerator);
}
```

Listing 1: General ACD Algorithm.

For implementing COG, the most straight-forward algorithm is discretisation where the MF is discretised equally along x and y-axes. There are however, some other algorithms which are optimised for computation as proposed in [22]. In this paper, we focus on the discretisation method for its simplicity. For comparison, the algorithm needed for the COG computation is provided in Listing 2.

```
double CentroidDefuzzification
(double x[], double mf[], int disc_levels){
        double numerator = 0.0, denumerator = 0.0;
        double x_Sum = 0.0, x_Count =0.0;
        for (int i=1; i<=disc_levels; i++){
                numerator += mf[i] * x[i];
                denumerator += mf[i];
        }
        return(numerator / denumerator);
}
```

Listing 2: COG Algorithm.

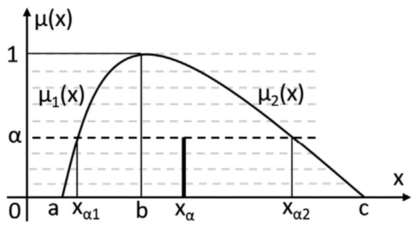

Fig. 6. A normal convex set's MF (solid curve) crossed with different α levels (dashed line). The membership function μ is divided into two monotonic functions μ_1 and μ_2. At each α level, x_α is the average of the x value of the two crossing points $x_{\alpha 1}$ and $x_{\alpha 2}$.

Comparing Listings 1 and 2, observe that the existence of the nested loops in the ACD algorithm makes it more complex than the COG algorithm. However, in the experiments described in the next section, the run-time difference is shown to be not more that 5% of the total time needed for generating the FLS outputs. In the next section, a simpler computational model for convex FSs is presented too.

3. A formal approach

In this section we develop a closed-form formula for ACD, and then check if there is any formal relation between ACD and COG. Additionally, some related challenges are discussed.

3.1. Developing a formula for ACD

Although (3) can be used to develop an algorithm for calculating ACD, it does not provide a closed-form for ACD, because the averaging of each α-cut is not yet formulated based on $\mu(x)$. In this section we obtain such a formula. The analysis provided here is limited to convex FSs.

Assume a normal convex FS defined in a continuous universe (Fig. 6), expressed as:

$$\mu(x) = \begin{cases} \mu_1(x), & for \quad a \le x \le b \\ \mu_2(x), & for \quad b \le x \le c \end{cases} \tag{12}$$

where in (12), $\mu_1(x)$ is monotonically increasing and $\mu_2(x)$ is monotonically decreasing.

As α takes different values between 0 and 1 along the vertical axis, $\mu_1^{-1}(\alpha)$ and $\mu_2^{-1}(\alpha)$ map a single α into two values along the x-axis, namely $x_{\alpha 1}$ and $x_{\alpha 2}$. The average of the two values is calculated as:

$$x_\alpha = \frac{x_{\alpha 1} + x_{\alpha 2}}{2} = \frac{\mu_1^{-1}(\alpha) + \mu_2^{-1}(\alpha)}{2} \tag{13}$$

From the ACD definition in (4) it is known that:

$$ACD = \int_{\alpha=0}^{1} \alpha x_\alpha d\alpha \Bigg/ \int_{\alpha=0}^{1} \alpha d\alpha \tag{14}$$

We know that $\int_0^1 \alpha d\alpha = [\alpha^2/2]_0^1 = \frac{1}{2}$. From (13) and (14):

$$ACD = \int_{\alpha=0}^{1} \alpha \mu_1^{-1}(\alpha) d\alpha + \int_{\alpha=0}^{1} \alpha \mu_2^{-1}(\alpha) d\alpha \tag{15}$$

Next, we prefer to change the integral variable from α to x, so that the formulation becomes free of α and any reverse functions, as well as being comparable to COG in (1). We know that:

$$x = \begin{cases} \mu_1^{-1}(\alpha) & for \quad a \le x \le b \\ \mu_2^{-1}(\alpha) & for \quad b \le x \le c \end{cases} \tag{16}$$

Notice that:

$$\alpha = \mu(x); \quad d\alpha = \begin{cases} d[\mu_1(x)] = \mu_1'(x)dx & for \quad a \le x \le b \\ d[\mu_2(x)] = \mu_2'(x)dx & for \quad b \le x \le c \end{cases} \tag{17}$$

where the prime sign denotes the derivative.[10] Note also, that in the two terms of (15), changing α from 0 to 1 in the first term corresponds to changing x from a to b, but in the second term, this corresponds to changing x reversely from c to b (see Fig. 6).

(16) and (17) can then be used to change the integral variable in (15) from α to x, i.e.:

$$ACD = \int_{x=a}^{b} x\mu_1(x)\mu_1'(x)dx + \int_{x=c}^{b} x\mu_2(x)\mu_2'(x)dx \tag{18}$$

Because $\mu_1(x)$ in the range $[a, b]$ is equivalent to $\mu(x)$ in this range, and the same for $\mu_2(x)$ in the range $[b, c]$, then:

$$ACD = \int_{x=a}^{b} x\mu(x)\mu'(x)dx + \int_{x=c}^{b} x\mu(x)\mu'(x)dx \tag{19}$$

The two integral terms in (19) cannot be merged since the second integral range would need to be from b to c (see Fig. 6). However by noticing that $\mu(x)$ is a decreasing function over $[b, c]$, we know that $\mu'(x)$ is always less than or equal to zero. Using the absolute value function, we are then able to replace $\mu'(x)$ in the first integral by $|\mu'(x)|$, and in the second integral by $-|\mu'(x)|$. Thus (19) can be expressed as:

$$ACD = \int_{x=a}^{b} x\mu(x)|\mu'(x)|dx + \int_{x=b}^{c} x\mu(x)|\mu'(x)|dx \tag{20}$$

$$ACD = \int_{x=a}^{c} x\mu(x)|\mu'(x)|dx \tag{21}$$

Because $\mu(x) = 0$ for all $x < a$ and $x > c$, then:

$$ACD = \int x\mu(x)|\mu'(x)|dx \tag{22}$$

(22) is the desired closed-form formula for ACD.

Although the above approach is not directly extendible to non-convex FSs, we are able to directly extend it to non-normal MFs. Let the MF's maximum be $\alpha_m < 1$, then all the previous integrals in the α-domain formulations become limited to $[0, \alpha_m]$. The denominator of (14) is then equal to $\alpha_m^2/2$ instead of $1/2$. The mapping from the α-domain to the x-domain is similar to the normal MF case, i.e. changing α from 0 to α_m corresponds to changing x from a to b in $\mu_1(x)$, and from c to b in $\mu_2(x)$. Thus, the only effect is introducing a constant *non-normality factor* $(1/\alpha_m^2)$ into (22), i.e.:

$$ACD = \frac{1}{\alpha_m^2} \int x\mu(x)|\mu'(x)|dx \tag{23}$$

3.2. Example: triangle MF

Consider the triangle MF discussed in Subsection 2.3 that is shown in Fig. 7 as a combination of two increasing and decreasing MFs. We recalculate ACD using (22), as follows:

[10] Extending (17) to the discrete mode can be challenging since there could be variable discretisation levels along the two domains, leaving the mathematical relation between $\Delta\alpha$ and Δx generally undefined. See also Section 3.7.

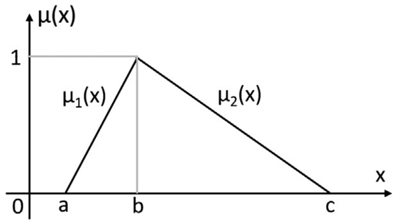

Fig. 7. The triangle MF shown as two joint MFs.

$$\mu(x)=\begin{cases}\mu_1(x)=(x-a)/(b-a) & for \quad a\le x\le b\\ \mu_2(x)=(c-x)/(c-b) & for \quad b\le x\le c\end{cases} \tag{24}$$

$$\mu'(x)=\begin{cases}\mu_1'(x)=1/(b-a) & for \quad a\le x\le b\\ \mu_2'(x)=1/(b-c) & for \quad b\le x\le c\end{cases} \tag{25}$$

$$\begin{aligned}ACD&=\int x\mu(x)|\mu'(x)|dx=\int_a^b \frac{x(x-a)}{(b-a)^2}dx-\int_b^c\frac{x(x-c)}{(c-b)^2}dx\\ &=\frac{2b^3-3ab^2+a^3}{6(b-a)^2}+\frac{c^3-3cb^2-2b^3}{6(c-b)^2}=\frac{a+2b}{6}+\frac{c+2b}{6}=(a+4b+c)/6\end{aligned} \tag{26}$$

(26) matches the result already derived in (5).

The matching of the results in this example shows that for some MF types, there is actually no need to use (22) to obtain ACD. Note that, if the tops of the produced spikes collectively produce a "middle curve" that can be mathematically expressed as a function of x, then ACD can be directly calculated as the centroid of that curve. For the discussed triangle MF example, the middle curve (called $\mu_m(x)$ in Fig. 2) represents a mathematical function that can be expressed using a, b and c, so that ACD value, located at the centroid of $\mu_m(x)$, can also be expressed using the same parameters.

3.3. α-Domain and x-domain interchange

Reviewing the formal approach presented above, it is questionable why the ACD formulation is initially started in α-domain and then changed to x-domain.

By definition, in α-cut representation, a FS is decomposed according to its α levels, thus it is intuitive that the calculation of ACD consists of integrals on α-domain, not initially on x-domain. However, the integrals on two domains are interchangeable, helped by a mathematical relation between $d\alpha$ and dx (see (17)). In discrete mode however, this can be challenging since there could be different discretisation levels along the two domains, leaving the mathematical relation between $\Delta\alpha$ and Δx undefined.

For some well-defined and/or monotonic MFs, there is actually no need to change the domain for the sake of AD calculation, as discussed for the example in a triangular MF. In general case however, it is quite possible that the produced middle curve cannot be represented by a well-defined single-valued function - see Fig. 8). In such cases, the proposed formal approach in changing between α and x domains would be necessary. On the other hand, it will not be possible to always start the COG formulation from α-domain since the MF is not a single-valued function from the α-axis view (unless MF is monotonic).

3.4. A link between ACD and COG formulations

It is of interest to find a mathematical relation between COG in (1) and ACD in (22). Although for a symmetric MF, the values of ACD and COG are equal,[11] and for special MF geometries such as triangular MFs the difference between the two can be mathematically expressed (see Subsection 2.3), in general no direct mathematical link is apparent

[11] In a FS with symmetric MF, both ACD and COG formulations locate the middle of the FS's support.

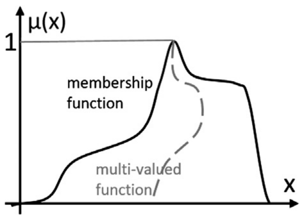

Fig. 8. An example of a MF whose middle-curve is a multi-valued function.

between (1) and (22).[12] Interestingly, as we show next, ACD and COG are special cases of a general defuzzification framework, namely the *Weighted Function COG (WFCOG)*, introduced in [2].

WFCOG of a fuzzy set A with MF $\mu(x) > 0$ is defined as:

$$V_f(A) = \frac{\int x\mu(x)f(x)dx}{\int \mu(x)f(x)dx}; \tag{27}$$

where $f(x) \geq 0$ is a weighting function and all the integrals are over the MF's support. Function f represents the relative weights that a decision maker may put on the value x, or a *utility* value for x. In other words, both $\mu(x)$ and $f(x)$ identify the contribution of x in calculating the defuzzified value. A constant $f(x)$ means that all x have the same utility, leading to COG as defined in (1).

Comparing $V_f(A)$ in (27) with ACD in (22) for a normal convex FS suggests that $|\mu'(x)|$ can be considered to be a weighting function $f(x)$, however the denominator of (27) needs to be worked out in order to validate this suggestion.

The denominator of (27), after replacing $f(x)$ with $|\mu'(x)|$ yields (see Fig. 6):

$$\int \mu(x)f(x)dx = \int_a^c \mu(x)|\mu'(x)|dx = \int_a^b \mu_1(x)\mu_1'(x)dx + \int_c^b \mu_2(x)\mu_2'(x)dx \tag{28}$$

Using (16) and (17), (28) can be expressed as:

$$\int_a^c \mu(x)f(x)dx = \int_{\alpha=0}^{1} \alpha d\alpha + \int_{\alpha=0}^{1} \alpha d\alpha = 2 \times \frac{1}{2} = 1 \tag{29}$$

Since the denominator of (27) is 1, we conclude that:

$$ACD = V_f(A); \qquad f(x) = |\mu'(x)| \tag{30}$$

which means that ACD is a special case of WFCOG where $f(x) = |\mu'(x)|$.

We already know that COG is another special case of WFCOG where $f(x) = 1$. In [2] it is shown that some other known defuzzification methods are also special cases of WFCOG.

For the case of a non-normal MF with a maximum value $\alpha_m < 1$, it can be directly concluded, from (23) and (27), that:

$$ACD = V_f(A); \qquad f(x) = \frac{|\mu'(x)|}{\alpha_m^2} \tag{31}$$

This shows that the only effect of non-normality is a constant factor of $1/\alpha_m^2$ in the weighting function $f(x)$.

Equations (30) and (31) mean that for a convex FS, ACD is a WFCOG in which the *sharpness* (i.e., the absolute derivative) of the MF at x is used as weights, whereas in COG, the weights are equally 1. Fig. 9 illustrates the effect of considering the MF's sharpness as weights. In this figure, the black curve is a sample MF $\mu(x)$. While the weighting function used for COG calculation is $f_1(x) = 1$, the dashed curve $f_2(x) = |\mu'(x)|$ (drawn on a different vertical

[12] Note that unlike (1), (22) contains a derivative form of the MF.

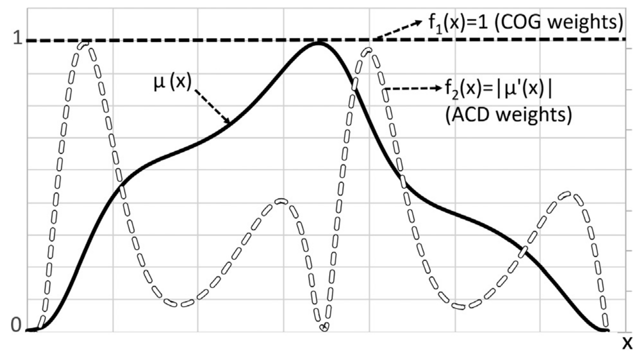

Fig. 9. An example of COG and ACD calculations based on different weighting functions. The function f_1 shows the constant weights (=1) used in COG, whereas the function f_2 (normalised to [0,1] for a better visibility) shows the ACD's dynamic weights changing by the sharpness of the corresponding membership grade.

scale for a better visibility) shows the weighting function used for calculating ACD (30). The dashed curve indicates where along the x-axis the membership grades have more or less contributions towards calculating ACD; the sharper areas of $\mu(x)$ have more weights than the flatter areas. In the standard COG method, all of the MF values have equal contributions towards its calculation.

In summary, we began with a definition of ACD using the α-cut representation, in order to define a method within a FS context. We then concluded that for a convex FS, expressing ACD using its MF leads to considering the MF's *sharpness* as the weighting factor, whose values would have equally been 1 if COG was used. This means that, in the ACD method, in order to know the weighting factor at point x, one must look at the value of the MF at x *and* its values at neighbouring x's. In other words, ACD reflects the static *and* dynamic properties of the MF at a certain x, whereas COG only reflects the static characteristics of the MF.

3.5. Linking ACD and BADD

BADD (Basic Defuzzification Distribution) [23] is an important related defuzzification approach, defined as:

$$BADD = \frac{\int x\mu(x)^{\gamma}dx}{\int \mu(x)^{\gamma}dx} \tag{32}$$

Similar to WFCOG, BADD is a generalised method that can adapt itself to the designer's attitude in weighting different parts of the MF. However BADD applies the adaptation by changing a free parameter $\gamma \in [0, \infty)$ that adjusts the shape and the non-linearity of the MF accordingly. Although BADD is developed as a compromise between probabilistic and possibilistic approaches in fuzzy sets operations [23], it is still considered as a support-based method and therefore cannot be directly represented in a level-based form, since it applies a form of weighed averaging over the MF. Evidently, it can be reduced to some support-based methods such as COG (γ=1), MOM ($\gamma \to \infty$) and MOS (γ=0).

Notably, in [2] where WFCOG was introduced, a more generalised formula was also presented that integrates both the weighting function of WFCOG and the γ-adjustment of BADD [23]. It was called Maximum Entropy Weighting Function BADD (MEWFBADD), defined as:

$$V_{f,\gamma}(A) = \frac{\int x\mu(x)^{\gamma} f(x)dx}{\int \mu(x)^{\gamma} f(x)dx} \tag{33}$$

Since it was shown in this paper that ACD and COG are special cases of WFCOG, it can also be concluded that BADD, ACD and COG are all special cases of MEWFBADD. Practically, we notice that the bigger the γ, the more importance is given to the values with bigger MF, which is similar to the more weights given by ACD to the bigger

α-values. Therefore, BADD with $\gamma > 1$ is expected to demonstrate some similar behaviours to ACD. Moreover, it can be expected that as γ moves from 1 to ∞, BADD is likely to meet ACD at some γ values when travelling from COG to MOM. This can be the subject of a future work.

3.6. An axiomatic study

Ban and Coroianu [24] provide a number of axioms to characterise the ranking methods used for fuzzy numbers. Although this approach is focused on the context of fuzzy numbers, the same approach can also be extended to evaluate any defuzzification method. They have provided the conditions that satisfy their axioms. Without reviewing the analytical details, some of those results can be directly used for evaluating the ACD. Interestingly, in the case of trapezoidal MFs, the *value* of a fuzzy number, defined as the average of α-cut means weighted by their corresponding α, is shown to satisfy *all* the axioms. This result can be readily extended to the case of ACD, since calculating the value of a fuzzy number is mathematically equivalent to calculating the ACD of a FS (see Subsection 2.2).

ACD can also be evaluated within the axiomatic approach of Runkler and Glenser [25]. They have developed a set of 13 axioms as quality criteria for *rational* defuzzification algorithms. Among the 13 axioms, it has been shown that the COG satisfies nine of them: *Zero-element, One-Element, Monotony, Symmetry, x-Translation, x-Scaling, μ-Translation (weak version), μ-Scaling and Hedges*. Similarly, it can be shown that ACD satisfies the same nine axioms. The proofs are straightforward, since the given proofs for COG can be extended to ACD by applying the WFCOG weights (9). The other axioms not satisfied by COG are still not satisfied by ACD. This can be shown by means of counterexamples, e.g., neither COG nor ACD necessarily represent the mean value of a fuzzy number (i.e., the x value corresponding to the single maximum of $\mu(x)$). Importantly, their axioms serve as a set of desirable properties not as an axiomatic basis for a defuzzifier (which is very different from, e.g., the axiomatic basis for t-norm, t-conorm and complement). To the best of our knowledge, there is no theory that indicates that satisfying a subset of these properties is good or optimal, nor is there a universal agreement on why a defuzzifier has to satisfy a certain number of them.

3.7. A simpler computation model

In Subsection 2.4, we provided a general algorithm for implementing ACD based on its original definition (3). ACD can also be implemented more simply for convex FSs, by means of the following discrete version of (22):

$$ACD = \sum_{i=1}^{N} (x_i \mu(x_i) |\Delta\mu(x_i)|) \tag{34}$$

where N is the number of discretisation levels along the x-axis.

Comparing the computation complexities of implementing COG (2) and ACD (34), observe that in (2), two summations are computed over the x range, in addition to a final division, whereas in (34), one summation as well as the absolute change of the MF in each step are computed. This shows that the computation complexity of implementing an algorithm for ACD should be about the same as for COG.

The following shows a sample ACD implementation based on (34). In Listing 3, two arrays $x[]$ and $mf[]$ collectively represent the fuzzy set's MF, and the parameter *dis_levels* indicates the number of discretisation levels.

```
double AlphaCutDefuzzification_Convex
(double x[], double mf[], int disc_levels){
        double x_Sum = 0.0, delta_mf=0.0;
        for (int i=2; i<=disc_levels; i++){
                abs_delta_mf=mf[i]>mf[i-1] ? mf[i] : mf[i-1];
                xSum += x[i] * mf[i] * abs_delta_mf ;
        }
        return xSum;
}
```

Listing 3: ACD Algorithm for convex FSs.

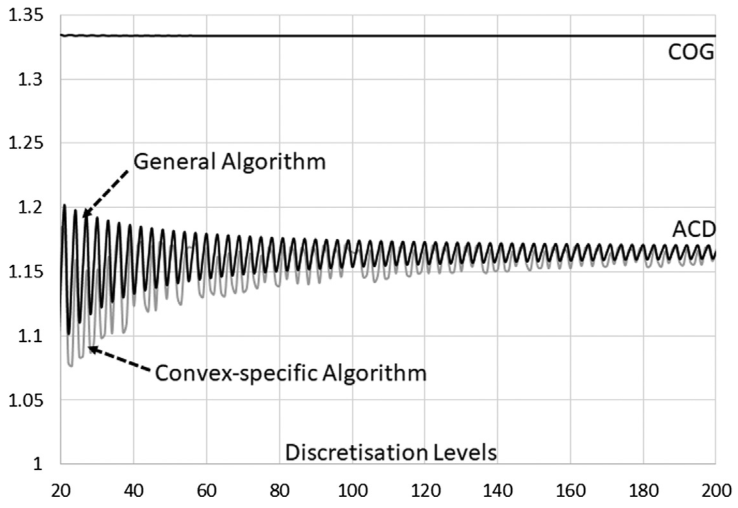

Fig. 10. Convergence patterns of two ACD computation algorithms for the triangle MF of Fig. 3, when discretisation levels increase, together with the COG calculation results.

Comparing the general and convex-specific algorithms developed for ACD (Listings 1 and 3 respectively), it is initially observed that, since Listing 3 has a single loop it is much simpler and runs faster than Listing 1. Moreover, the complexity of Listing 3 is about the same as the complexity of the COG algorithm (Listing 2). We ran a timing test to compare the running times of Listings 2 and 3. For a sample triangular MF shown in Fig. 3, and 100 discretisation levels, a Java program took 153 ms for COG and 160 ms for ACD computation on a laptop with an Intel Core-i5 8 GB RAM.

Note that, there is a discretisation issue associated with the ACD computation using (34). According to (3), ACD is defined based on having fixed α intervals, $\Delta\alpha$, but selecting fixed $\Delta\alpha$ along the vertical axis corresponds to variable Δx along the x-axis, since the relation between $\Delta\alpha$ and Δx is generally not proportional. Thus, we predict that low discretisation levels may produce more discretisation errors using (34) in Listing 3, than those produced using (3) in Listing 1. The outcome of the two algorithms should theoretically converge for high discretisation levels, but with possibly different convergence patterns.

In order to study this effect, we compared the convergence patterns of the two ACD algorithms using an example. We consider the same triangle MF example of Fig. 3. Since this MF is not symmetric, the fixed α intervals along the vertical axis correspond to two different intervals along the x-axis, so that in low discretisation levels, we expect more discretisation errors. Because the FS is convex, both the general algorithm in Listing 1 and the convex-specific algorithm in Listing 3 can be utilised in order to calculate the ACD value. For the case of COG computations (Listing 2), equal intervals over the x-axis are used, so that we expect lower discretisation errors than ACD. In this example, the linearity of the MF shape causes the COG computations rapidly approach its final value. Letting the discretisation levels range from 20 to 200 over the x-axis between x=0 to x=3, the converging patterns of the two ACD algorithms are shown and compared in Fig. 10, along with the COG convergence pattern. It is observed that although both ACD algorithms approach their final value of 1.17 in periodic patterns, the convergence pattern of the general ACD algorithm has a relatively less average error for lower discretisation levels than does the convex-specific ACD algorithm.

4. Experiment 1: performance test in FLS for time-series prediction

In this section, we compare the performances of two FLSs designed for the prediction of two time series (Mackey-Glass and Lorenz), where all FLS parameters are the same, but the defuzzification methods are different. Two levels of measurement noise are used. The research question for comparing the utility of the ACD and COG defuzzification methods in this experiment is: In an FLS designed for noisy time-series prediction, can replacing the COG defuzzifier with the ACD defuzzifier generate less prediction error?

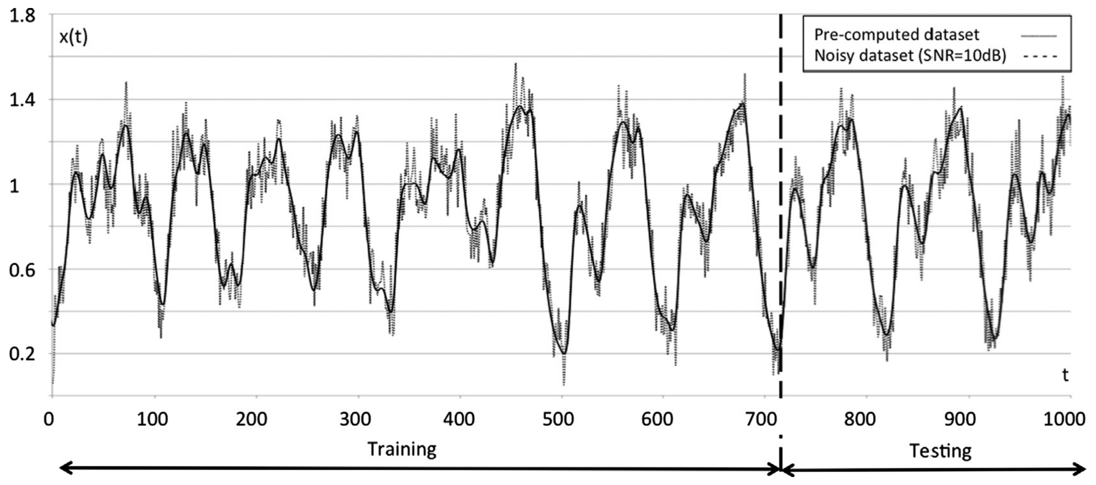

Fig. 11. Illustration of the pre-computed and noisy (SNR=10 dB) MG time series. A FS is trained from $t = 1$ to $t = 700$ and tested from $t = 701$ to $t = 1000$.

4.1. *Mackey-glass time series prediction*

4.1.1. *Methodology*

The Mackey-Glass (MG) chaotic time-series has been widely used to evaluate the prediction performance of intelligent systems (e.g., [26–29]). It is described by the following differential equation [30]:

$$\frac{dx}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{35}$$

For $\tau > 17$, (35) is known to demonstrate a chaotic behaviour; we selected $\tau = 30$. Using (35), $x(t)$ is calculated for 2000 consecutive points, i.e., $t = -999$ to $t = 1000$ (Fig. 11). The first 1000 points are to let the initial transients die out. For $t = 1$ to $t = 700$ the noisy time series data are used to develop a set of fuzzy rules according to the one-pass method described in [31]. The number of generated rules varies from 184 for noise-free data to 570 for high noise levels.

The last 300 points from $t = 701$ to $t = 1000$ are used for testing the FLS. Nine of the past points in the time series are employed as inputs to generate a predicted value. Seven equally-distributed triangular MFs are used to model the input domains (from about 0.2 to 1.4). For all FLSs, Mamdani inference is used with min and max operators for the t-norm and t-conorm respectively. The same 100 discretisation levels are used for all FLS computations.

Two noise conditions are used to test the FLS's performance, namely, signal-to-noise ratios (SNRs) of 10 dB and 20 dB. White Gaussian noise is added to the input data, in which the standard deviation (STD) of the noise is derived from the system's SNR value. If the STD of the signal and the STD of the noise are σ_s and σ_n, respectively, then SNR can be written as:

$$SNR = 10\log(\frac{\sigma_s^2}{\sigma_n^2}) \quad ; or \quad \sigma_n = \frac{SNR}{10^{(\sigma_s/20)}} \tag{36}$$

Each experiment generates 300 output fuzzy sets that are defuzzified in order to generate a crisp output using both COG and ACD. The outputs are then compared with the pre-computed values generated by (35). The difference between each FLS output and the pre-computed value is then calculated, which leads to calculating MSE (Mean Square Error) for all 300 points. Finally, the calculated MSE$_{ACD}$ and MSE$_{COG}$ are compared. In order to mitigate the effect of randomness, each result for each noise level is averaged over 30 repetitions of the entire experiment.

4.1.2. *Results*

Fig. 12 illustrates sample ACD and COG results for the 10 dB and 20 dB noise levels, as well as the pre-computed time series. Average and STD of the calculated MSEs over the 30 experiments are summarized in Table 1. Observe from Table 1, that for lower noise levels (SNR=20 dB) on average, using ACD leads to about 35% less MSE than

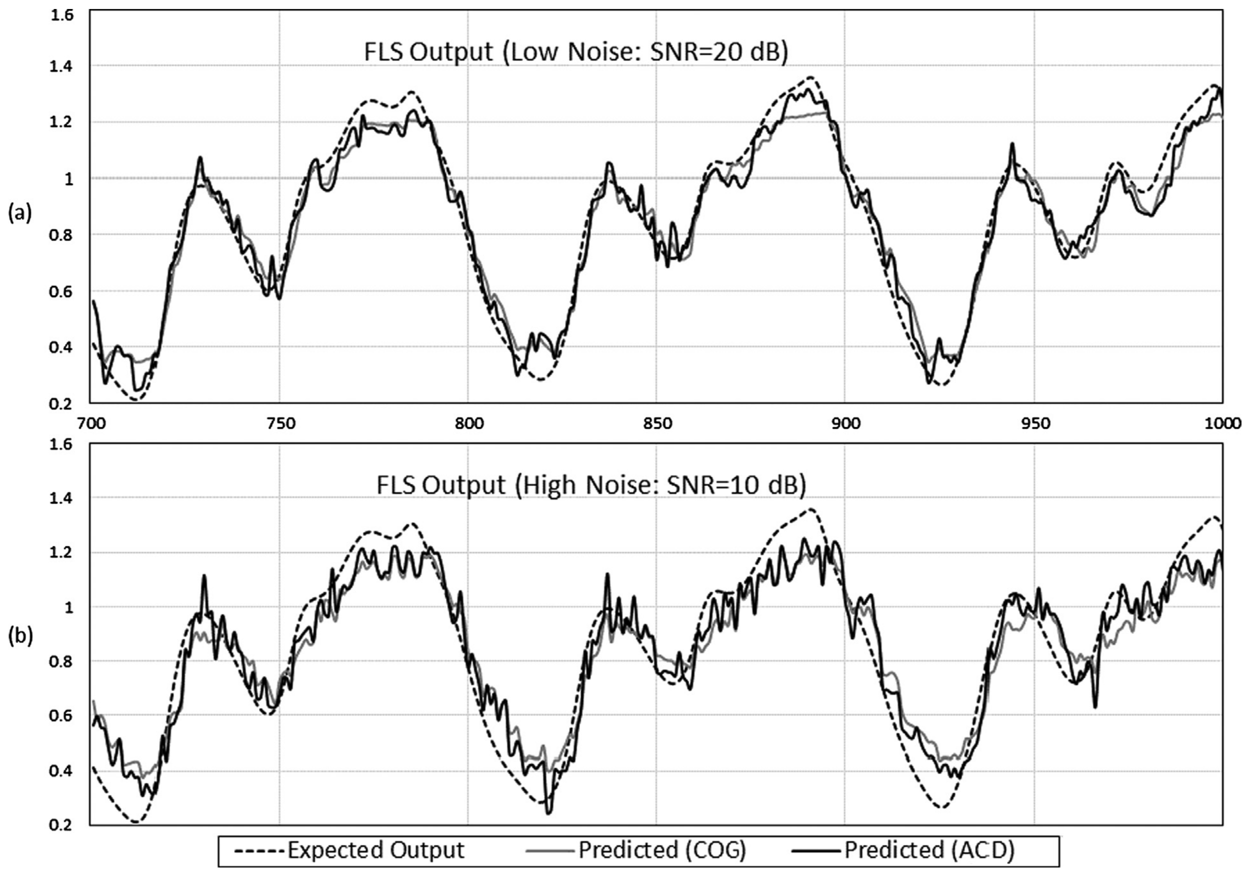

Fig. 12. Sample FLS prediction results of MG time series using ACD and COG methods, when SNR is (a) 20 dB and (b) 10 dB.

Table 1
Comparative results of applying COG and ACD, for prediction of MG time series, under two input noise levels.

Method	SNR (dB)	MSE (Mean)	Change in MSE (Mean)	MSE (STD)	Change in MSE (STD)
COG	20	0.00455		0.00064	
ACD	20	0.00296	−34.95%	0.00042	−34.28%
COG	10	0.01324		0.00183	
ACD	10	0.00999	−24.50%	0.00111	−39.09%

COG. In higher noise levels (SNR=10 dB), ACD leads to about 25% less MSE than COG. The STDs of the MSEs are significantly reduced in all cases when ACD is utilised, indicating more reliable results.

In order to compare the ACD results with one other defuzzification method than COG, we repeated the MG times series experiment using MOM (mean of maxima) defuzzification and compared the results between MOM and ACD. This comparison is summarised in Table 2, which indicates that for SNR=20 dB (10 dB) using ACD leads to 47.29% (31.31%) less average MSE than using MOM. A similar improvement is also observed between the standard deviation of the measured MSEs, in which ACD method gains 42.10% (34.43%) less MSE standard deviation than MOM in SNR=20 dB (10 dB).

In terms of computational complexity, a timing test was performed to measure the time needed to compute the FLS output for each run. On a laptop PC with an Intel Core-i5 8 GB RAM using Java, on average each output was computed in 44.67 ms when ACD is utilised, compared to 46.59 ms for COG. As discussed in Section 2, a longer time is expected for the ACD algorithm, however since the time needed for defuzzification is a small part of the total time

Table 2
Comparative results of applying MOM and ACD, for prediction of MG time series, under two input noise levels.

Method	SNR (dB)	MSE (Mean)	Change in MSE (Mean)	MSE (STD)	Change in MSE (STD)
MOM	20	0.00436		0.00064	
ACD	20	0.00296	−47.29%	0.00042	−42.10%
MOM	10	0.01312		0.00169	
ACD	10	0.00999	−31.31%	0.00111	−34.43%

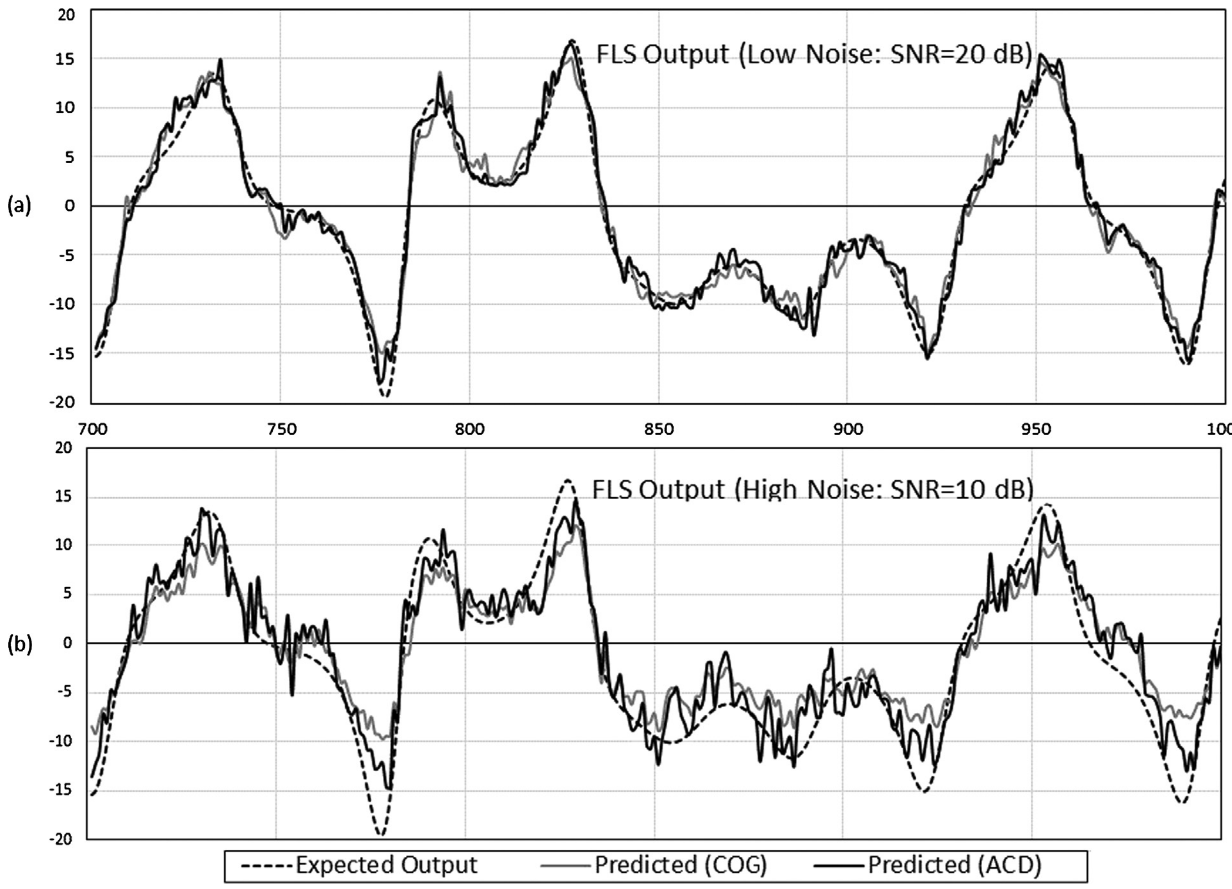

Fig. 13. Sample FLS prediction results of Lorenz time series using ACD and COG methods, when SNR is (a) 20 dB and (b) 10 dB.

needed for each FLS run, the measured FLS run time showed about 4% increase for the FLS that uses ACD compared to the FLS that uses COG.

4.2. Lorenz time series prediction

4.2.1. Methodology

Next, we compare the designed FLS's performances using another well-known time series in hydrodynamics and meteorology, namely Lorenz [32], whose differential equations are:

$$\frac{dx}{dt} = \sigma(y - x); \quad \frac{dy}{dt} = rx - y - xz; \quad \frac{dz}{dt} = xy - bz. \tag{37}$$

This is a 3D time series but our focus is on $x(t)$ only. To demonstrate a chaotic behaviour, the parameters σ, b and r are respectively set to 10, $\frac{8}{3}$ and 28, as suggested in [32]. The procedures of generating rules, adding noise and

Table 3
Results of applying COG and ACD, for prediction of Lorenz time series, under two input noise levels.

Method	SNR (dB)	MSE (Mean)	Change in MSE (Mean)	MSE (STD)	Change in MSE (STD)
COG	20	3.20448		0.78748	
ACD	20	2.17572	−32.10%	0.41960	−46.72%
COG	10	11.0502		1.93773	
ACD	10	7.68346	−30.47%	1.19482	−38.34%

Table 4
Results of applying MOM and ACD, for prediction of Lorenz time series, under two input noise levels.

Method	SNR (dB)	MSE (Mean)	Change in MSE (Mean)	MSE (STD)	Change in MSE (STD)
MOM	20	3.80812		0.67136	
ACD	20	2.17572	−42.88%	0.41960	−37.50%
MOM	10	10.6833		1.77462	
ACD	10	7.68346	−28.08%	1.19482	−32.74%

averaging the results for 30 random additive noise scenarios are similar to those described in the first subsection, and are therefore not repeated.

4.2.2. Results

It is observable from the results that when the ACD method is utilised for defuzzification, the predicted output more closely follows the expected output, than when the classical COG method is used (Fig. 13). This is very similar to what we already observed for MG time series in the previous subsection. The statistical results of the experiment with Lorenz time series also, demonstrates some similar patterns of MSE reduction (both average and STD) when the new defuzzification algorithm is applied.

Table 3 shows that for 30 repetitive runs, in lower noise levels (SNR=20 dB), there is about 32% reduction in the predicted MSE when the ACD method is utilised. In higher noise level (SNR=10 dB), the reduction is about 30%. MSE STDs over the 30 runs are reduced by about 46% and 38% in the lower and the higher noise levels, respectively. Regarding computation time, a test was performed on the same hardware and software settings used for MG time series. Each FLS run using ACD took 36.21 ms on average, compared to 34.44 ms required for each run using COG (about 5% increase).

We also repeated the Lorenz times series experiment using MOM and compared the results with ACD, as summarised in Table 4. For SNR=20 dB (10 dB) using ACD leads to 42.88% (28.08%) less average MSE than using MOM. A similar improvement is also observed between the standard deviation of the measured MSEs, in which ACD method gains 37.50% (32.74%) less MSE standard deviation than MOM in SNR=20 dB (10 dB).

4.3. Discussion

The potential for using the ACD method for improving FLS performance, compared to the standard COG method, has been demonstrated. It has been observed from the designed FLS for predicting two noisy times series, that when the ACD method is utilised for defuzzification, the predicted output more closely follows the expected output than when the classical COG method is used. The statistical results for the two experiments show similar patterns of MSE reduction (for the both average and STD) when ACD algorithm is utilised. Although the ACD algorithm has an additional complexity over the COG algorithm, its observed effect on the total time required for running each FLS is about 5% increase.

In order to investigate the conditions in which ACD, COG and MOM compete in producing closer predictions to the expected values in the studied time series, we picked three sample points in each time series, in which a) ACD prediction is closer; b) COG prediction is closer; or c) ACD and COG make about the same prediction. We will compare the results of MOM with the other two methods too. For each sample point, the fuzzy output set before

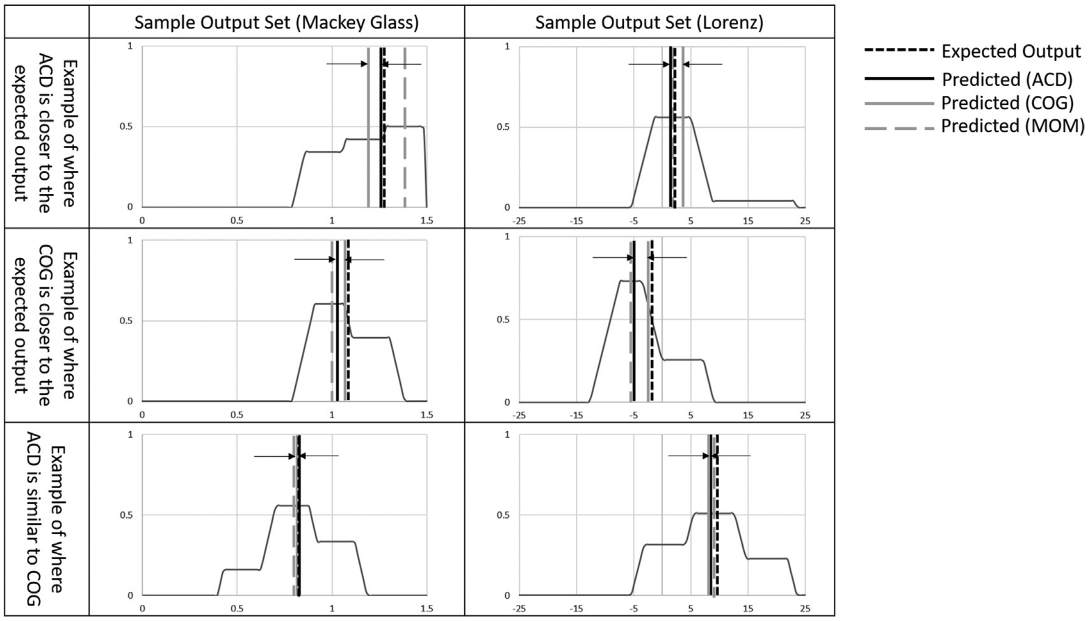

Fig. 14. Three sample extreme points in each time series (MG and Lorenz), in which ACD prediction is closer (top), COG prediction is closer (middle) and in which ACD and COG make about the same predictions as the expected value (bottom).

defuzzification is observed. It is noticeable that the three defuzzification methods are applied to the same output fuzzy set.

Fig. 14 shows the results of the investigation for the three sample points in each time series. At each point, the output fuzzy set under defuzzification is shown, together with its COG, ACD, MOM, and the expected value (the ground truth coming from the time series formula). Observing Fig. 14 can show some heuristic facts behind the differences between the results of ACD, COG and MOM.

First, as expected, the three methods do similarly regarding the nearly symmetric MFs, as well as being similarly close to the expected value. This can be observed in the bottom row of Fig. 14. Secondly, with more non-symmetric MFs, the fact that ACD gives more weight to the upper alpha levels makes its result more differentiated from COG. On the other hand, MOM is only dependent on the maximum points of the MF, so it is independent of the other irregularities out of the maximum plateau. So it can be roughly concluded that ACD has a tendency towards the top MF values (so has to MOM) whereas COG represents the geometrical centre of the MF. This can be observed in the top two rows of Fig. 14. The locations of the expected value within the more irregular MFs in this figure seem more unpredictable, as the generation of the MF is a consequence of a highly complex process in both training and inference stages. COG makes a closer prediction only where the expected value is eventually towards the geometric centre. By all means, an average over all the points, of which only three extreme samples are investigated, shows a general tendency of the ACD output to the expected value compared to COG and MOM, as shown earlier in this section.

An interesting case which demonstrates the ACD formal link to MF and its derivative in (22), is the top-right MF of Fig. 14. The long narrow channel on the right side of the MF has drawn the geometrical centre, thus the COG, to the right making it distant from the expected value. The channel, however, has almost no effect on ACD according to (22) since along the flat channel the derivative of the MF is zero. In general, it can be roughly concluded that since ACD ignores any flat areas, the irregular MFs with large flat areas can showcase the differences between ACD and COG.

Making direct judgements on why each defuzzification method may make a closer prediction than the other, based on the picked sample points and the look of the fuzzy sets in Fig. 14, is not straight-forward. In fact, there is no reason to label a particular defuzzification method "better" or "worse" than the other based on making a closer prediction to the expected value in a specific example. The time series being chaotic together with many other involved imperfections and uncertainties collectively affect the prediction quality.

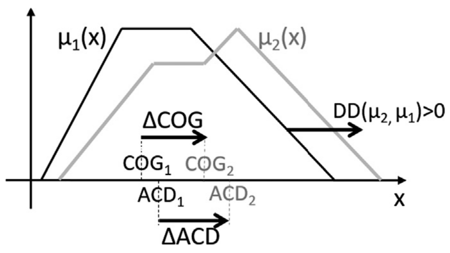

Fig. 15. Two FSs are shown together with their directional distance (DD) as well as their defuzzified values using ACD and COG methods.

Additionally, there are some contextual differences between MOM and ACD/COG which should be carefully considered in discussing the results of this experiment. MOM is a method that is designed for implicative fuzzy systems [33], where the output fuzzy sets can have very wide or unlimited supports. In this case, MOM has natural advantages over COG (and ACD), not only due to its computational simplicity but also because the averaging basis of COG or ACD makes them unsuitable for such large universes of discourse. This showcases why the results of this experiment should not be directly generalised for all types of fuzzy systems.

Finally, it is noticeable that the experiments in this section use the one-pass method in rule training, which is simple but leads to a relatively large number of rules. The rule sets could be optimised using more advanced methods such as Derivative-Based Design [11]) leading to a much smaller rule set and an optimized MF parameters. As a future work, it would be interesting to learn if ACD still outperforms COG for such a smaller but optimized set of rules.

5. Experiment 2: correlation test for directional distance

In this experiment we are interested to compare ACD and COG using a pairwise distance measure. Many such methods exist for measuring the distance between two FSs (e.g., [34–36]. We focus on the Directional Distance (DD) measure [37], which is able to measure the distance between two FSs as a signed value (in two directions), and has the same units as the x-axis of the MFs. An extended experiment of this type in comparing ACD with COG is reported in [38].

The definition of the DD between two FSs is based on the distance between their α-cuts. According to [37], if the α-cuts of two arbitrary FSs X and Y are defined as intervals $[\mu_X]_\alpha = [\overline{x}_\alpha, \underline{x}_\alpha]$ and $[\mu_Y]_\alpha = [\overline{y}_\alpha, \underline{y}_\alpha]$ respectively, their DD is defined as:

$$DD(X,Y) = \frac{\sum \left(\alpha . h([\mu_X]_\alpha, [\mu_Y]_\alpha)\right)}{\sum \alpha}; \quad \alpha \in [0,1] \tag{38}$$

$$h([\mu_X]_\alpha, [\mu_Y]_\alpha) = \begin{cases} \overline{x}_\alpha - \overline{y}_\alpha, & if \quad |\overline{x}_\alpha - \overline{y}_\alpha| > |\underline{x}_\alpha - \underline{y}_\alpha| \\ \underline{x}_\alpha - \underline{y}_\alpha, & otherwise, \end{cases} \tag{39}$$

where h is a modified version of Hausdorff distance [36] between two intervals. A positive DD indicates how much Y's MF is to the right side of X's MF, and vice-versa. For any FS pair, the numeric difference between the defuzzified values should closely follow the DD between the two sets, so we will compare this behaviour between ACD and COG (see Fig. 15) over a large number of FS pairs.

5.1. Method

The first step in this experiment is to synthetically produce a large number of FS pairs. We used an already existing FLS and produced 50 uniformly distributed random input values. The generated 50 output FSs led to 1225 possible pairings. Secondly, the DD were calculated for each of the 1225 FS pairs. Then each of the 50 sets was defuzzified using COG and ACD methods (using 100 levels of discretisation for x and α) based on the algorithms explained in Listings 1 and 2. Finally, for each of the 1225 pairs, the differences between the two defuzzified values were calculated (called ΔCOG and ΔACD). To compare the utility of ACD and COG, the correlation coefficient between ΔCOG and DD was calculated and compared with the correlation coefficient between ΔACD and DD, using bivariate correlation [39]:

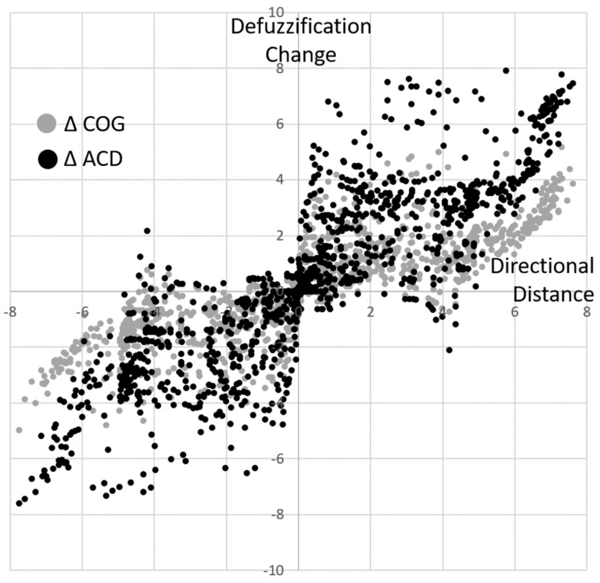

Fig. 16. The relation between the DDs of two FSs vs. the signed difference in their defuzzified values, tested for COG and ACD, over a large number of FS pairs.

$$Correlation\,(X,Y) = \frac{\sum_{i=0}^{n}[(x_i-\overline{x})(y_i-\overline{y})]}{\sqrt{\sum_{i=1}^{n}[(x_i-\overline{x})^2(y_i-\overline{y})^2]}}; \qquad (40)$$

where $X = \{x_1 ... x_n\}$ and $Y = \{y_1 ... y_n\}$ are two sample sets of n points with average values $\overline{x}$ and $\overline{y}$. In this experiment, X is the set of either ΔCOG_i's or ΔACD_i's, whereas Y is the set of DD_i's. A closer correlation coefficient to 1 for a FS pair and for a particular defuzzification method, shows that the observed change in the defuzzified value is more closely bound to the directional distance between the two sets, i.e., a closer behaviour to what is intuitively expected.

5.2. Results

Fig. 16 depicts the experiment results. For each examined pair of FSs, their DDs are shown along the horizontal axis (ranging from negative to positive values), and the calculated ΔCOG and ΔACD are shown along the vertical axis. Both methods generally demonstrate strong correlations between DDs and changes in the defuzzified values over the 1225 pairs of random FSs, however, the average correlation coefficient between DD and ΔCOG is 0.82 whereas the average correlation coefficient between DD and ΔACD is 0.86. This shows about 4.88% more correlation when ACD is applied.

6. Conclusions and future work

In this paper, we defined a new defuzzification method of generalised fuzzy sets, called α-cut defuzzification (ACD). Since the output fuzzy sets of rule-based systems can be in any shape, the ACD utility has been examined in this paper for such systems. Our approach, similar to a number of reviewed approaches in fuzzy number ranking, is inspired by the FS concept of the α-cut, rather than MF-based, statistical/physical concepts such as the standard COG. For continuous convex FSs (normal or non-normal) in continuous mode, we developed a closed-form formula for ACD, and demonstrated that COG and ACD are both special cases of the Weighted Function COG, introduced in [2]. Whereas those weights for COG are constant, for ACD they are the absolute derivative of the MF, which lets the dynamic properties of the MF be included in ACD. As expected, for symmetric MFs, ACD and COG values are equal. For discrete FSs, computation models were provided for both convex and non-convex FSs, and it was shown that the ACD algorithm for convex FSs has about the same complexity as COG.

Two series of experiments were performed to compare the relative utilities of the COG and ACD methods. For predicting noisy time series (Mackey-Glass and Lorenz) using a rule-based FLS, ACD performed much better than

COG, having closer and less variant predictions in two different noise conditions. We also statistically showed in another experiment, that the ACD outcomes more closely follow the MF changes over a large number of FS pairs, as compared to COG, in which the changes are measured by each pair's directional distance. In summary, the potential for ACD to outperform COG looks very promising.

For future works, more real-world scenarios and experiments have to be used for testing the utility and performance of ACD, especially in FLSs where MF parameters are optimised as part of the FLS design process.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors wish to thank the respective anonymous reviewers for their time, insightful reviews and accurate comments for improving this paper.

Very sadly, while the revision of this paper was being done, our respective author Professor Robert I. John passed away. This paper is dedicated to his memory.

References

[1] R.R. Yager, D. Filev, On the issue of defuzzification and selection based on a fuzzy set, Fuzzy Sets Syst. 55 (3) (1993) 255–271.
[2] X. Liu, Parameterized defuzzification with maximum entropy weighting function - another view of the weighting function expectation method, Math. Comput. Model. 45 (1–2) (2007) 177–188.
[3] S. Roychowdhury, W. Pedrycz, A survey of defuzzification strategies, Int. J. Intell. Syst. 16 (6) (2001) 679–695.
[4] J.J. Saade, H.B. Diab, Defuzzification techniques for fuzzy controllers, IEEE Trans. Syst. Man Cybern., Part B, Cybern. 30 (1) (2000) 223–229.
[5] T.A. Runkler, Selection of appropriate defuzzification methods using application specific properties, IEEE Trans. Fuzzy Syst. 5 (1) (1997) 72–79.
[6] W. Van Leekwijck, E.E. Kerre, Defuzzification: criteria and classification, Fuzzy Sets Syst. 108 (2) (1999) 159–178.
[7] E.N. Nasibov, A. Mert, On methods of defuzzification of parametrically represented fuzzy numbers, Autom. Control Comput. Sci. 41 (5) (2007) 265–273.
[8] L. Wang, A Course in Fuzzy Systems and Control, Prentice Hall Press, USA, 1999.
[9] L.A. Zadeh, Fuzzy sets, Inf. Control 8 (3) (1965) 338–353.
[10] G. Klir, B. Yuan, Fuzzy Sets and Fuzzy Logic, Vol. 4, Prentice Hall, New Jersey, 1995.
[11] J.M. Mendel, Uncertain Rule-Based Fuzzy Systems: Introduction and New Directions, 2nd edition, Springer, Cham, Switzerland, 2017.
[12] D. Dubois, H. Prade, Operations on fuzzy numbers, Int. J. Syst. Sci. 9 (6) (1978) 613–626.
[13] M. Delgado, M.A. Vila, W. Voxman, On a canonical representation of fuzzy numbers, Fuzzy Sets Syst. 93 (1) (1998) 125–135.
[14] R.R. Yager, A procedure for ordering fuzzy subsets of the unit interval, Inf. Sci. 24 (2) (1981) 143–161.
[15] M. Oussalah, On the compatibility between defuzzification and fuzzy arithmetic operations, Fuzzy Sets Syst. 128 (2) (2002) 247–260.
[16] S. Heilpern, The expected value of a fuzzy number, Fuzzy Sets Syst. 47 (1) (1992) 81–86.
[17] L.M. de Campos Ibáñez, A.G. Muñoz, A subjective approach for ranking fuzzy numbers, Fuzzy Sets Syst. 29 (2) (1989) 145–153.
[18] E.N. Nasibov, Certain integral characteristics of fuzzy numbers and a visual interactive method for choosing the strategy of their calculation, J. Comput. Syst. Sci. Int. 41 (4) (2002) 584–590.
[19] E.H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, Int. J. Man-Mach. Stud. 7 (1) (1975) 1–13.
[20] D. Dubois, H. Prade, The mean value of a fuzzy number, Fuzzy Sets Syst. 24 (3) (1987) 279–300.
[21] J. Fortin, D. Dubois, H. Fargier, Gradual numbers and their application to fuzzy interval analysis, IEEE Trans. Fuzzy Syst. 16 (2) (2008) 388–402.
[22] E. Van Broekhoven, B. De Baets, Fast and accurate center of gravity defuzzification of fuzzy system outputs defined on trapezoidal fuzzy partitions, Fuzzy Sets Syst. 157 (7) (2006) 904–918.
[23] D.P. Filev, R.R. Yager, A generalized defuzzification method via bad distributions, Int. J. Intell. Syst. 6 (7) (1991) 687–697.
[24] A.I. Ban, L.C. Coroianu, Simplifying the search for effective ranking of fuzzy numbers, IEEE Trans. Fuzzy Syst. 23 (2) (2015) 327–339.
[25] T. Runkler, M. Glesner, A set of axioms for defuzzification strategies towards a theory of rational defuzzification operators, in: IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 1993, IEEE, 1993, pp. 1161–1166.
[26] A. Lapedes, R. Farber, Nonlinear signal processing using neural networks, Tech. Rep. LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM, 1987.
[27] R.S. Crowder, Predicting the Mackey-Glass time series with Cascade-correlation learning, in: Proceedings of Connectionist Models Summer School, Carnegie Mellon Univ., 1990, pp. 117–123.
[28] J. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cybern. 23 (3) (1993) 665–685.

[29] R.D. Jones, Y. Lee, C. Barnes, G. Flake, K. Lee, P. Lewis, S. Qian, Function approximation and time series prediction with neural networks, in: IJCNN International Joint Conference on Neural Networks, IEEE, 1990, pp. 649–665.

[30] M.C. Mackey, L. Glass, et al., Oscillation and chaos in physiological control systems, Science 197 (4300) (1977) 287–289.

[31] L. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Syst. Man Cybern. 22 (6) (1992) 1414–1427.

[32] E.N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. 20 (2) (1963) 130–141.

[33] O. Cordón, F. Herrera, A. Peregrín, Looking for the best defuzzification method features for each implication operator to design accurate fuzzy models, Technical Report, Department of Computer Science and Artificial Intelligent, University of Granada, Spain, 1999.

[34] X. Liu, Entropy, distance measure and similarity measure of fuzzy sets and their relations, Fuzzy Sets Syst. 52 (3) (1992) 305–318.

[35] B. Chaudhur, A. Rosenfeld, On a metric distance between fuzzy sets, Pattern Recognit. Lett. 17 (11) (1996) 1157–1160.

[36] R. Zwick, E. Carlstein, D.V. Budescu, Measures of similarity among fuzzy concepts: a comparative analysis, Int. J. Approx. Reason. 1 (2) (1987) 221–242.

[37] J. McCulloch, C. Wagner, U. Aickelin, Measuring the directional distance between fuzzy sets, in: Computational Intelligence (UKCI), 2013 13th UK Workshop on, IEEE, 2013, pp. 38–45.

[38] A. Pourabdollah, Fuzzy number value or deffuzified value; which one does it better?, in: IEEE International Conference on Fuzzy Systems, IEEE, 2020.

[39] T.W. Anderson, An Introduction to Multivariate Statistical Analysis, Vol. 2, Wiley, New York, 1958.