Soft Computing (2018) 22:1361–1380
https://doi.org/10.1007/s00500-017-2938-3

METHODOLOGIES AND APPLICATION

Study on centroid type-reduction of general type-2 fuzzy logic systems with weighted enhanced Karnik–Mendel algorithms

Yang Chen[1,2] · Dazhi Wang[2]

Published online: 27 November 2017

Abstract
With the development of α-planes representation of general type-2 fuzzy sets (GT2 FSs), general type-2 fuzzy logic systems (GT2 FLSs) based on GT2 FSs have become a hot topic in academic field. While type-reduction (TR) is most critical block for a T2 FLS, generally speaking, the most popular Karnik–Mendel (KM) or enhanced KM (EKM) algorithms are used to perform the TR. The paper connects the EKM and the continuous version of EKM algorithms together and expands the EKM algorithms to three different forms of weighted EKM (WEKM) algorithms resort to the Newton–Cotes quadrature formulas of numerical integration techniques, while the EKM algorithms just become a special case of the WEKM algorithms. Four computer simulation examples are used to illustrate the performances of the WEKM algorithms. Compared with the EKM algorithms, the WEKM algorithms have smaller absolute error and faster convergence speed to compute the centroid defuzzified value of GT2 FLSs in general, which make them potentially applicable for T2 FLSs designers and adopters.

Keywords General type-2 fuzzy logic systems · α-Planes · Type-reduction · Weighted enhanced Karnik–Mendel algorithms · Continuous enhanced Karnik–Mendel algorithms · Computer simulation

1 Introduction

So far, interval type-2 fuzzy logic systems (IT2 FLSs) (Mendel 2007; Hagras and Wagner 2012; Mendel 2001; Khosravi and Nahavandi 2014; Chen et al. 2016) are still the most widely used nonlinear models for fields with high uncertainty, such as financial systems (Zarandi et al. 2009), autonomous mobile robots (Biglarbegian et al. 2011), database and information systems (Niewiadomski 2010). Takagi–Sugeno–Kang IT2 FLSs can approximate any real continuous functions on a compact set to arbitrary accuracy. The computational cost of general type-2 fuzzy logic systems (GT2 FLSs) is comparatively high. Until recent years, the α-planes representation (Mendel 2014) of general type-2 fuzzy sets (GT2 FSs) was proposed by different research groups, which tremendously decreased the computation complexity of GT2 FSs and their corresponding GT2 FLSs. In recent years, GT2 FLSs (Sanchez et al. 2015; Castillo et al. 2016a) based on the α-planes representation theories have been applied in some fields.

In general, a T2 FLS is composed of five blocks, which are fuzzifier, inference, rules, type-reduction (TR), and defuzzification. Among them, the block of TR plays the most essential role. The centroid TR (Karnik and Mendel 2001) is the most popular method in theoretical researches. Traditional Karnik–Mendel (KM) algorithms (Mendel 2013; Mendel and Liu 2007) were developed for performing the centroid TR of IT2 FLSs. It usually needs two to six iterations for the KM algorithms to converge. Then EKM algorithms (Wu and Mendel 2009) were developed by Wu and Mendel in order to reduce the computation time. And the EKM algorithms can save about two iterations compared with the KM algorithms. The continuous versions of KM and EKM (CKM and CEKM) were proposed by Mendel and Wu (2006), which were used for theoretical analysis. Liu et al. (2012) gave the theoretical explanations for the initialization of the EKM algorithms and extended the EKM algorithms to the weighted EKM (WEKM) algorithms by means of numerical integration techniques to compute the centroid left end points of

Communicated by V. Loia.

✉ Yang Chen
chenyanghanyun@163.com

[1] College of Science, Liaoning University of Technology, Jinzhou 121001, China

[2] College of Information Science and Engineering, Northeastern University, Shenyang 110819, China

IT2 FSs. All of these laid a rich theoretical foundation for applying the TR algorithms for IT2 FLSs.

In 2008, Liu (2008) introduced the α-planes representation of a GT2 FS for the first time and employed the KM algorithms to perform the centroid TR of GT2 FLSs. In the next year, Mendel et al. (2009) reviewed the α-planes representation of a GT2 FS and compared T1, IT2, and triangle quasi-GT2 FLSs to predict a chaotic Mackey–Glass time series. Similar to the α-planes representation, Wagner and Hagras (2010) used the KM algorithms to compute the centroid of the zSlices of GT2 FLSs in robot control problems. Inspired by Mendel (2014), Liu (2008), Mendel et al. (2009), Liu et al. (2012), Chen et al. (2015) and Chen and Wang (2015), this paper aims to connect the EKM and CEKM algorithms together and expands the WEKM algorithms for implementing the centroid TR of α-planes representation-based GT2 FLSs. By analyzing and comparing the sum operation in the discrete version of EKM algorithms and the integral operation in the continuous version of EKM (CEKM) algorithms, the paper employs the Newton–Cotes quadrature formulas of numerical integration techniques to expand the EKM algorithms to three different forms of WEKM algorithms. For performing the centroid TR of GT2 FLSs, the proposed WEKM algorithms have more accurate calculation results and faster convergence speed compared with the EKM algorithms.

The remainder of the paper is organized as follows: Sect. 2 briefly introduces the backgrounds of GT2 FSs and GT2 FLSs. Section 3 provides the Newton–Cotes quadrature formulas, the proposed CEKM algorithms for GT2 FLSs, and how to compute the centroid TR of GT2 FLSs according to the WEKM algorithms with three different weight assignment approaches. Section 4 gives four simulation examples, and analyzes and compares the performances of the WEKM algorithms with the EKM algorithms. Finally, conclusions are given in Sect. 5.

2 Backgrounds

2.1 General type-2 fuzzy sets

A general type-2 fuzzy set (GT2 FS) $\tilde{A}$ can be viewed as a bivariate function (Aisbett et al. 2010) on the Cartesian product, where the mapping is $\mu_{\tilde{A}} : X \times [0, 1] \to [0, 1]$ and X is the universe of the primary variable, x, of $\tilde{A}$, i.e.,

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

which is often called the point-value expression of a GT2 FS; $\mu_{\tilde{A}}(x, u)$ can also be represented as $f_x(u)$.

A vertical slice of $\mu_{\tilde{A}}(x, u)$ is a secondary membership function (MF) (Mendel and John 2002), i.e.,

$$\mu_{\tilde{A}}\left(x = x', u\right) \equiv \mu_{\tilde{A}}\left(x'\right) = \int_{u \in [0,1]} f_{x'}(u)/u, \tag{2}$$

where the sign $\int$ does not mean the integral operation, but represents union over all admissible. For simplicity, we can denote the secondary MF $\mu_{\tilde{A}}\left(x'\right)$ as $\tilde{A}(x)$ in the paper.

So Eq. (1) can be reexpressed as

$$\tilde{A} = \int_{\forall x \in X} \tilde{A}(x)/x; \tag{3}$$

this kind of vertical slices representation of GT2 FSs is very important for studying the computations of GT2 FLSs.

The two-dimensional support of $\mu_{\tilde{A}}(x, u)$ is called the footprint of uncertainty (FOU) of $\tilde{A}$, i.e.,

$$\text{FOU}\left(\tilde{A}\right) = \left\{(x, u) \in X \times [0, 1] | \mu_{\tilde{A}}(x, u) > 0\right\}, \tag{4}$$

where FOU $(\tilde{A})$ is bounded by upper MF (UMF) $\overline{\mu}_{\tilde{A}}(x)$ and lower MF (LMF) $\underline{\mu}_{\tilde{A}}(x)$.

Suppose that $\tilde{A}_\alpha(x)$ denotes the α-cut of $\tilde{A}(x), \alpha \in [0, 1]$, i.e.,

$$\tilde{A}_\alpha(y) = \{u \mid f_x(u) \geq \alpha\} = [a_\alpha(x), b_\alpha(x)]. \tag{5}$$

For any $x \in X$, we treat $\tilde{A}(x)$ as the following α-cuts decomposition, i.e.,

$$\begin{aligned} \tilde{A}(x) &= \bigcup_{\forall \alpha \in [0,1]} [\alpha/\tilde{A}_\alpha(x)] = \sup_{\forall \alpha \in [0,1]} \left[\alpha/\tilde{A}_\alpha(x)\right] \\ &= \sup_{\forall \alpha \in [0,1]} \left\{\alpha/\left[a_\alpha(x), b_\alpha(x)\right]\right\}, \end{aligned} \tag{6}$$

where $\bigcup$ represents the union operation and sup denotes the supremum. The vertical slices representation of $\tilde{A}$ could be obtained by substituting (6) into (3) as

$$\tilde{A} = \int_{\forall x \in X} \left\{ \bigcup_{\forall \alpha \in [0,1]} \left[\alpha/\tilde{A}_\alpha(x)\right] \right\} /x. \tag{7}$$

Based on the above analyses, the α-planes (horizontal slices or z-slices) (Mendel 2014; Liu 2008; Mendel et al. 2009; Wagner and Hagras 2010) representation of $\tilde{A}$ can be expressed as

$$\begin{aligned} \tilde{A} &= \bigcup_{\forall \alpha \in [0,1]} \left\{ \int_{\forall x \in X} \left[\alpha/\tilde{A}_\alpha(x)\right] /x \right\} \\ &= \bigcup_{\forall \alpha \in [0,1]} \left\{ \alpha/\left[\int_{\forall x \in X} \tilde{A}_\alpha(x)/x \right] \right\}, \end{aligned} \tag{8}$$

where the α-plane $\tilde{A}_\alpha$ is the union of primary membership functions of $\tilde{A}$ whose secondary membership grades must be greater than or equal to α, i.e.,

$$\tilde{A}_\alpha = \{(x,u), \mu_{\tilde{A}(x)}(u) \geq \alpha | \forall x \in X, \forall u \in [0,1]\}$$

or

$$\tilde{A}_\alpha = \int_{\forall x \in X} \int_{\forall u \in [0,1]} \{(x,u) | \mu_{\tilde{A}(x)}(u) \geq \alpha\}, \tag{9}$$

where Eq. (9) denotes both the point-value and continuous forms, and $\tilde{A}_\alpha$ can also be straight forwardly expressed as

$$\tilde{A}_\alpha = \int_{\forall x \in X} \tilde{A}_\alpha(x)/x = \int_{\forall x \in X} [a_\alpha(x), b_\alpha(x)]/x. \tag{10}$$

In addition, an α-plane which is raised to the α-level is usually denoted by $R_{\tilde{A}_\alpha}$, i.e.,

$$R_{\tilde{A}_\alpha} = \alpha / \tilde{A}_\alpha, \tag{11}$$

where $R_{\tilde{A}_\alpha}$ is an IT2 FS whose secondary membership grades equal to α.

Interestingly, the secondary membership grades of IT2 FSs are all equal to 1. So an IT2 FS can be completely characterized by its UMF and LMF.

2.2 General type-2 fuzzy logic systems

In general, the structure of GT2 FLSs (Mendel 2014) can be classified into two types: Mamdani and TSK. Only the Mamdani type is focused on this paper. Without loss of generality, we consider a Mamdani GT2 FLS with p inputs $x_1 \in X_1, \ldots, x_p \in X_p$ and one output $y \in Y$. It is characterized by M fuzzy rules, where the lth rule is of the form

$$\text{IF } x_1 \text{ is } \tilde{F}_1^l \text{ and } \cdots \text{ and } x_p \text{ is } \tilde{F}_p^l, \text{ THEN } y \text{ is } \tilde{G}^l \; l = 1, \ldots, M, \tag{12}$$

where $\tilde{F}_i^l (i = 1, \ldots, p; l = 1, \ldots, M)$ are the antecedent GT2 FSs and $\tilde{G}^l (l = 1, \ldots, M)$ are the consequent GT2 FSs.

In order to simplify expressions, we adopt singleton fuzzifier in the paper. When $x_i = x_i'$, only the vertical slice $\tilde{F}_i^l(x_i')$ of antecedent GT2 FSs $\tilde{F}_i^l$ is activated, whose α-cut decomposition is (see Eq. 7)

$$\tilde{F}_i^l(x_i') = \sup_{\forall \alpha \in [0,1]} \alpha / \left[a_{i,\alpha}^l(x_i'), b_{i,\alpha}^l(x_i') \right]. \tag{13}$$

For every fuzzy rule, one computes the firing interval at the α-level $F_\alpha^l(x')$ as:

$$F_\alpha : \begin{cases} F_\alpha^l(x') \equiv \left[\underline{f}_\alpha^l(x'), \overline{f}_\alpha^l(x') \right] \\ \underline{f}_\alpha^l(x') \equiv T_{i=1}^p a_{i,\alpha}^l(x_i') \\ \overline{f}_\alpha^l(x') \equiv T_{i=1}^p b_{i,\alpha}^l(x_i'), \end{cases} \tag{14}$$

where T denotes the minimum or product t-norm operation.

Suppose that the α-plane (horizontal slice) $\tilde{G}_\alpha^l$ of the consequent GT2 FS $\tilde{G}^l$ at the α-level is (see Eq. 9)

$$\tilde{G}_\alpha^l = \int_{\forall y \in Y} \tilde{G}_\alpha^l(y)/y = \int_{\forall y \in Y} \left[g_{L,\alpha}^l(y), g_{R,\alpha}^l(y) \right]/y. \tag{15}$$

Then the firing interval of each fuzzy rules is combined with the corresponding consequent α-plane $\tilde{G}_\alpha^l$ to obtain the firing rule α-plane $\tilde{B}_\alpha^l$, i.e.,

$$\tilde{B}_\alpha^l : \begin{cases} \text{FOU}\left(\tilde{B}_\alpha^l\right) = \left[\underline{\mu}_{\tilde{B}_\alpha^l}(y|x'), \overline{\mu}_{\tilde{B}_\alpha^l}(y|x') \right] \\ \underline{\mu}_{\tilde{B}_\alpha^l}(y|x') = \underline{f}_\alpha^l(x') * g_{L,\alpha}^l(y) \\ \overline{\mu}_{\tilde{B}_\alpha^l}(y|x') = \overline{f}_\alpha^l(x') * g_{R,\alpha}^l(y), \end{cases} \tag{16}$$

where $*$ represents the minimum or product operation.

Next, we aggregate all the $\tilde{B}_\alpha^l (l = 1, \ldots, M)$ to obtain the output α-plane $\tilde{B}_\alpha$, i.e.,

$$\tilde{B}_\alpha : \begin{cases} \text{FOU}\left(\tilde{B}_\alpha\right) = \left[\underline{\mu}_{\tilde{B}_\alpha}(y|x'), \overline{\mu}_{\tilde{B}_\alpha}(y|x') \right] \\ \underline{\mu}_{\tilde{B}_\alpha}(y|x') = \underline{\mu}_{\tilde{B}_\alpha^1}(y|x') \vee \cdots \vee \underline{\mu}_{\tilde{B}_\alpha^M}(y|x') \\ \overline{\mu}_{\tilde{B}_\alpha}(y|x') = \overline{\mu}_{\tilde{B}_\alpha^1}(y|x') \vee \cdots \vee \overline{\mu}_{\tilde{B}_\alpha^M}(y|x'), \end{cases} \tag{17}$$

where $\vee$ denotes the maximum operation.

Then compute the centroid of $\tilde{B}_\alpha$ to acquire the TR set $Y_{C,\alpha}(x')$ at the α-level, i.e.,

$$Y_{C,\alpha}(x') = C_{\tilde{B}_\alpha}(x') = \alpha / \left[l_{\tilde{B}_\alpha}(x'), r_{\tilde{B}_\alpha}(x') \right], \tag{18}$$

where $\alpha \in [0,1]$, and the two end points $l_{\tilde{B}_\alpha}(x')$ and $r_{\tilde{B}_\alpha}(x')$ (the centroid end points of a specific IT2 FS) can be computed by TR algorithms like EKM algorithms (Mendel 2013; Wu and Mendel 2009; Liu et al. 2012; Chen et al. 2015) as:

$$l_{\tilde{B}_\alpha}(x') = \min_{\mu_{\tilde{B}_\alpha}(y_i) \in \left[\underline{\mu}_{\tilde{B}_\alpha}(y_i), \overline{\mu}_{\tilde{B}_\alpha}(y_i) \right]} \frac{\sum_{i=1}^N y_i \mu_{R_{\tilde{B}_\alpha}}(y_i)}{\sum_{i=1}^N \mu_{R_{\tilde{B}_\alpha}}(y_i)} \tag{19}$$

and

$$r_{\tilde{B}_\alpha}(x') = \max_{\mu_{\tilde{B}_\alpha}(y_i) \in \left[\underline{\mu}_{\tilde{B}_\alpha}(y_i), \overline{\mu}_{\tilde{B}_\alpha}(y_i) \right]} \frac{\sum_{i=1}^N y_i \mu_{R_{\tilde{B}_\alpha}}(y_i)}{\sum_{i=1}^N \mu_{R_{\tilde{B}_\alpha}}(y_i)}, \tag{20}$$

Table 1 EKM algorithms to compute the centroid end points of an IT2 FS (Wu and Mendel 2009)

Step	EKM algorithm for c_l
1	Set $k = [N/2.4]$ (the nearest integer to $N/2.4$) and compute $a = \sum_{i=1}^{k} x_i \overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) + \sum_{i=k+1}^{N} x_i \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)$, $b = \sum_{i=1}^{k} \overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) + \sum_{i=k+1}^{N} \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)$, $c' = a/b$
2	Find $k' \in [1, N-1]$ such that $x_{k'} \le c' \le x_{k'+1}$
3	Check if $k' = k$. If yes, stop and set $c' = c_l$ and $k = L$. If no, go to Step 4.
4	Compute $s = \text{sign}(k' - k)$ and: $a' = a + s \sum_{i=\min(k,k')+1}^{mak(k,k')} x_i [\overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) - \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)]$ $b' = b + s \sum_{i=\min(k,k')+1}^{mak(k,k')} [\overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) - \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)]$, $c''(k') = a'/b'$
5	Set $c' = c''(k)$, $a = a'$ and $b = b'$ and go to Step 2
Step	**EKM algorithm for c_r**
1	Set $k = [N/1.7]$ (the nearest integer to $N/1.7$) and compute $a = \sum_{i=1}^{k} x_i \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) + \sum_{i=k+1}^{N} x_i \overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)$, $b = \sum_{i=1}^{k} \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) + \sum_{i=k+1}^{N} \overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)$, $c' = a/b$
2	Find $k' \in [1, N-1]$such that $x_{k'} \le c' \le x_{k'+1}$
3	Check if $k' = k$. If yes, stop and set $c' = c_r$ and $k = R$. If no, go to Step 4.
4	Compute $s = \text{sign}(k' - k)$ and: $a' = a - s \sum_{i=\min(k,k')+1}^{mak(k,k')} x_i [\overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) - \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)]$ $b' = b - s \sum_{i=\min(k,k')+1}^{mak(k,k')} [\overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) - \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)]$, $c''(k') = a'/b'$
5	Set $c' = c''(k)$, $a = a'$ and $b = b'$ and go to Step 2

where N denotes the number of discrete points of the rule consequent primary variable.

Table 1 shows the detailed steps for computing the centroid of an IT2 FS by means of the EKM algorithms, where L and R are the switching points for the lower and upper MFs, and c_l and c_r are the left and right end points of the centroid interval, respectively.

Finally, we aggregate all the α-planes $Y_{\text{EKM},\alpha}$ to constitute the T1 FS Y_{EKM}, i.e.,

$$Y_{\text{EKM}} = \sup_{\forall \alpha \in [0,1]} \alpha / Y_{\text{EKM},\alpha}\left(x'\right). \tag{21}$$

In practical calculations, one uniformly divides the value of α into n alpha-planes at $\alpha_1, \alpha_2, \ldots, \alpha_n$. Then the crisp output of a GT2 FLS is calculated as

$$y\left(x'\right) = \sum_{i=1}^{n} \alpha_i \left[\left(l_{\tilde{B}_{\alpha_i}}\left(x'\right) + r_{\tilde{B}_{\alpha_i}}\left(x'\right)\right)/2\right] / \sum_{i=1}^{n} \alpha_i. \tag{22}$$

Equation (22) is referred to as the average of end points defuzzification method, which is first proposed by Wagner and Hagras (2010). In this way, altogether n values of $Y_{C,\alpha}$ at the corresponding α-level need to be computed by the TR algorithms (Mendel 2013; Mendel and Liu 2007; Wu and Mendel 2009; Liu et al. 2012; Biglarbegian et al. 2010; Coupland and John 2007; Mendel and Liu 2013). We may use $2n$ processors to speed up the computation time, if such conditions are available.

3 WEKM algorithms

Before introducing the proposed WEKM algorithms, we must first give two parts of preliminary knowledge: Newton–Cotes quadrature formulas (Liu et al. 2012; Chen and Wang 2015) and CEKM algorithms.

3.1 Newton–Cotes quadrature formulas

The numerical integration aims to approximate the definite integral $\int_a^b f(x)\mathrm{d}x$ by the linear combination of some functional values $f(x_i)$ on the discrete points. Therefore, the calculations of definite integrals can be ascribed to computing the functional values.

Definition 1 (*Quadrature formula* Liu et al. 2012; Mathews and Fink 2004) Suppose that $a < x_0 < x_1 < \cdots < x_n = b$, for the definite integral

$$\int_a^b f(x)\mathrm{d}x = Q(f) + E(f). \tag{23}$$

If the following is true

$$Q(f) = \sum_{i=0}^{n} w_i f(x_i) = w_0 f(x_0) + \cdots + w_n f(x_n), \tag{24}$$

then Eq. (23) is referred to as the quadrature formula or numerical integration, where $\{w_i\}_{i=0}^n$ is called the weight coefficient, $\{x_i\}_{i=0}^n$ is called the integration node, and $E(f)$ is referred to as the truncation error, which is also the remainder of the quadrature formula.

Next, we employ the composite trapezoidal rule, composite Simpson rule, and composite Simpson 3/8 rule as the straight line, quadrature polynomial function, and cubic polynomial function to approximate $f(x)$, respectively.

Theorem 1 (Composite trapezoidal rule Liu et al. 2012; Mathews and Fink 2004) *Consider the function $y = f(x)$ on the closed interval $[a, b]$. Divide the interval $[a, b]$ into n subintervals $\{x_{i-1}, x_i\}_{i=1}^n$ with the equal width $h = (b - a)/n$, where the equal interval node is $x_i = x_0 + ih (i =$*

$0, 1, \ldots, n$). *Then the numerical estimation of definite integral with the composite trapezoidal rule is*

$$\int_a^b f(x)dx = \frac{h}{2}\left[f(a) + f(b) + 2\sum_{i=1}^{n-1} f(x_i)\right] + E_T(f,h). \tag{25}$$

If the function f is second-order continuous differentiable on $[a, b]$, then the error term is

$$E_T(f,h) = -\frac{(b-a)f''(\zeta)}{12}h^2, \tag{26}$$

where $a < \zeta < b$.

Theorem 2 (Composite Simpson rule Liu et al. 2012; Mathews and Fink 2004) *Consider the function $y = f(x)$ on the closed interval $[a, b]$. Divide the interval $[a, b]$ into $2n$ subintervals $\{x_{i-1}, x_i\}_{i=1}^{2n}$ with the equal width $h = (b-a)/2n$, where the equal interval node is $x_i = x_0 + ih (i = 0, 1, \ldots, 2n)$. Then the numerical estimation of definite integral with the composite Simpson rule is*

$$\int_a^b f(x)dx = \frac{h}{3}\left[f(a) + f(b) + 2\sum_{i=1}^{n-1} f(x_{2i}) + 4\sum_{i=0}^{n-1} f(x_{2i+1})\right] + E_S(f,h). \tag{27}$$

If the function f is fourth-order continuous differentiable on $[a, b]$, then the error term is

$$E_S(f,h) = -\frac{(b-a)f^{(4)}(\zeta)}{180}h^4, \tag{28}$$

where $a < \zeta < b$.

Theorem 3 (Composite Simpson 3/8 rule Liu et al. 2012; Mathews and Fink 2004) *Consider the function $y = f(x)$ on the closed interval $[a, b]$. Divide the interval $[a, b]$ into $3n$ subintervals $\{x_{i-1}, x_i\}_{i=1}^{3n}$ with the equal width $h = (b-a)/3n$, where the equal interval node is $x_i = x_0 + ih (i = 0, 1, \ldots, 3n)$. Then the numerical estimation of definite integral with the composite Simpson 3/8 rule is*

$$\int_a^b f(x)dx = \frac{3h}{8}\left[f(a) + f(b) + \sum_{i=1}^{n} 2f(x_{3i}) + \sum_{i=1}^{n} 3f(x_{3i-2}) + \sum_{i=1}^{n} 3f(x_{3i-2})\right] + E_{SC}(f,h). \tag{29}$$

If the function f is fourth-order continuous differentiable on $[a, b]$, then the error term is

$$E_{SC}(f,h) = -\frac{(b-a)f^{(4)}(\zeta)}{80}h^4, \tag{30}$$

where $a < \zeta < b$.

Suppose that Eqs. (25), (27), and (29) are measurable, that is to say, all of them have meanings in Lebesgue sense.

3.2 CEKM algorithms

CEKM algorithms are similar to the discrete version of EKM algorithms (Mendel 2013; Mendel and Liu 2007; Wu and Mendel 2009; Mendel and Wu 2006; Liu et al. 2012; Chen et al. 2015), and they can be used to investigate the theoretical property of GT2 FLSs (Mendel 2014; Liu 2008; Mendel et al. 2009) based on the α-planes representation theory.

Suppose that the centroid TR output of GT2 FLSs is the GT2 FS $\tilde{A}$, and we decompose the value of α into n values such that $\alpha = \alpha_1, \ldots, \alpha_n$. For every $\alpha_i (i = 1, \ldots, n)$, the corresponding IT2 FS $R_{\tilde{A}_{\alpha_i}} (i = 1, \ldots, n)$ is obtained (see Eq. 18). Then the EKM algorithms can be used to compute the two end points of every $R_{\tilde{A}_{\alpha_i}}$ (see Table 1). If the primary variable of $\tilde{A}$ satisfies $a = x_1 < x_2 < \cdots < x_N = b$, a and b are the left and right end points of sampling primary variable x and then the continuous version of EKM algorithms should be as follows:

$$l_{\tilde{A}_\alpha}(x') = \min_{\zeta\in[a,b]} \frac{\int_a^\zeta x\overline{\mu}_{R_{\tilde{A}_\alpha}}(x)\mathrm{d}x + \int_\zeta^b x\underline{\mu}_{R_{\tilde{A}_\alpha}}(x)\mathrm{d}x}{\int_a^\zeta \overline{\mu}_{R_{\tilde{A}_\alpha}}(x)\mathrm{d}x + \int_\zeta^b \underline{\mu}_{R_{\tilde{A}_\alpha}}(x)\mathrm{d}x} \tag{31}$$

$$r_{\tilde{A}_\alpha}(x') = \max_{\zeta\in[a,b]} \frac{\int_a^\zeta x\underline{\mu}_{R_{\tilde{A}_\alpha}}(x)\mathrm{d}x + \int_\zeta^b x\overline{\mu}_{R_{\tilde{A}_\alpha}}(x)\mathrm{d}x}{\int_a^\zeta \underline{\mu}_{R_{\tilde{A}_\alpha}}(x)\mathrm{d}x + \int_\zeta^b \overline{\mu}_{R_{\tilde{A}_\alpha}}(x)\mathrm{d}x}. \tag{32}$$

Note that $\tilde{A}$ and $R_{\tilde{A}_\alpha}$ denote the GT2 FS and the corresponding IT2 FS in Eqs. (31) and (32), while $\tilde{B}$ and $R_{\tilde{B}_\alpha}$ represent the GT2 FS and the corresponding IT2 FS in Eqs. (19) and (20). Table 2 gives the processes of computing the centroid end points of an IT2 FS by the CEKM algorithms. The sign α in Table 2 is just a representation for computing, which has different meanings with the α in the α-plane.

3.3 WEKM algorithms

The CEKM algorithms are given in order to let us better understand the EKM algorithms in theory. On the basis of Sects. 3.1 and 3.2, this section proposes the WEKM algorithms, which can be used to perform the centroid TR of GT2 FLSs in contrast to the EKM algorithms. The results in this section are adapted from Liu et al. (2012).

Table 2 CEKM algorithms to compute the centroid end points of an IT2 FS

Step	CEKM algorithm for c_l
1	Set $c = a + (b-a)/2.4$, and compute $\alpha = \int_a^c x\overline{\mu}_{\tilde{R}_{A_x}}(x)\mathrm{d}x + \int_c^b x\underline{\mu}_{\tilde{R}_{A_x}}(x)\mathrm{d}x$, $\beta = \int_a^c x\overline{\mu}_{\tilde{R}_{A_x}}(x)\mathrm{d}x + \int_c^b x\underline{\mu}_{\tilde{R}_{A_x}}(x)\mathrm{d}x, c' = \alpha/\beta$
2	Check if $\lvert c' - c\rvert < \varepsilon$($\varepsilon$ is a give bound of the algorithms). If yes, stop and set $c' = c_l$. If no, go to Step 3.
3	Compute $s = \mathrm{sign}(c' - c)$ and $\alpha' = \alpha + s\int_{\min(c,c')}^{\max(c,c')} x[\overline{\mu}_{R_{\tilde{A}_\alpha}}(x) - \underline{\mu}_{R_{\tilde{A}_\alpha}}(x)]\mathrm{d}x$ $\beta' = \beta + s\int_{\min(c,c')}^{\max(c,c')}[\overline{\mu}_{R_{\tilde{A}_\alpha}}(x) - \underline{\mu}_{R_{\tilde{A}_\alpha}}(x)]\mathrm{d}x, c'' = \alpha'/\dot{\beta}'$
4	Set $c = c', c' = c'', \alpha = \alpha', \beta = \beta'$ and go to Step 2
Step	**CEKM algorithm for c_r**
1	Set $c = a + (b-a)/1.7$, and compute $\alpha = \int_a^c x\underline{\mu}_{\tilde{R}_{A_x}}(x)\mathrm{d}x + \int_c^b x\overline{\mu}_{\tilde{R}_{A_x}}(x)\mathrm{d}x$, $\beta = \int_a^c x\underline{\mu}_{\tilde{R}_{A_x}}(x)\mathrm{d}x + \int_c^b x\overline{\mu}_{\tilde{R}_{A_x}}(x)\mathrm{d}x, c' = \alpha/\beta$
2	Check if $\lvert c' - c\rvert < \varepsilon$($\varepsilon$ is a give bound of the algorithms). If yes, stop and set $c' = c_r$. If no, go to Step 3.
3	Compute $s = \mathrm{sign}(c' - c)$ and $\alpha' = \alpha - s\int_{\min(c,c')}^{\max(c,c')} x[\overline{\mu}_{R_{\tilde{A}_\alpha}}(x) - \underline{\mu}_{R_{\tilde{A}_\alpha}}(x)]\mathrm{d}x$ $\beta' = \beta - s\int_{\min(c,c')}^{\max(c,c')}[\overline{\mu}_{R_{\tilde{A}_\alpha}}(x) - \underline{\mu}_{R_{\tilde{A}_\alpha}}(x)]\mathrm{d}x, c'' = \alpha'/\dot{\beta}'$
4	Set $c = c', c' = c'', \alpha = \alpha', \beta = \beta'$ and go to Step 2

Table 3 WEKM algorithms to compute the centroid end points of an IT2 FS

Step	WEKM algorithm for c_l
1	Set $k = [N/2.4]$ (the nearest integer to $N/2.4$) and compute $\alpha = \sum_{i=1}^{k} w_i x_i \overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) + \sum_{i=k+1}^{N} w_i x_i \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)$, $\beta = \sum_{i=1}^{k} w_i \overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) + \sum_{i=k+1}^{N} w_i \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i), c' = \alpha/\beta$
2	Find $k' \in [1, N-1]$ such that $x_{k'} \le c' \le x_{k'+1}$
3	Check if $k' = k$. If yes, stop and set $c' = c_l$ and $k = L$. If no, go to Step 4.
4	Compute $s = \mathrm{sign}(k' - k)$ and: $\alpha' = \alpha + s\sum_{i=\min(k,k')+1}^{mak(k,k')} w_i x_i[\overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) - \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)]\beta' = \beta + s\sum_{i=\min(k,k')+1}^{mak(k,k')} w_i[\overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) - \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)], c''(k') = \alpha'/\beta'$
5	Set $c' = c''(k), \alpha = \alpha'$ and $\beta = \beta'$ and go to Step 2
Step	**EKM algorithm for c_r**
1	Set $k = [N/1.7]$ (the nearest integer to $N/1.7$) and compute $\alpha = \sum_{i=1}^{k} w_i x_i \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) + \sum_{i=k+1}^{N} w_i x_i \overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)$, $\beta = \sum_{i=1}^{k} w_i \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) + \sum_{i=k+1}^{N} w_i \overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i), c' = \alpha/\beta$
2	Find $k' \in [1, N-1]$such that $x_{k'} \le c' \le x_{k'+1}$
3	Check if $k' = k$. If yes, stop and set $c' = c_r$ and $k = R$. If no, go to Step 4.
4	Compute $s = \mathrm{sign}(k' - k)$ and: $\alpha' = \alpha - s\sum_{i=\min(k,k')+1}^{mak(k,k')} w_i x_i[\overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) - \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)]\beta' = \beta - s\sum_{i=\min(k,k')+1}^{mak(k,k')} w_i[\overline{\mu}_{R_{\tilde{A}_\alpha}}(x_i) - \underline{\mu}_{R_{\tilde{A}_\alpha}}(x_i)], c''(k') = \alpha'/\beta'$
5	Set $c' = c''(k), \alpha = \alpha'$ and $\beta = \beta'$ and go to Step 2

The WEKM algorithms are numerical implementation of the CEKM algorithms. By observing and comparing the results in Tables 1 and 2, we can obtain that the CEKM algorithms are similar to the discrete version of EKM algorithms. However, the sum operations in the discrete version have been changed to the integral operations in the continuous version. It just means the sum operations for the EKM algorithms at the sampling points play the role of integrations for the corresponding function in the CEKM algorithms. According to the Newton–Cotes quadrature formulas, we can assign the corresponding weight w_i for each of the x_i, and then the more accurate results of c_l and c_r can be computed.

The EKM algorithms are just a special case of the WEKM algorithm, whose weights are all equal to 1 ($w_i = 1(i = 1, \ldots, N)$). In Table 3, according to Eq. (24) (Mathews and Fink 2004), the weights can be assigned with several methods. However, we only consider the numerical integration methods based on the Newton–Cotes quadrature formulas described in Sect. 3.1. These methods are the composite trapezoidal rule, composite Simpson rule, and composite Simpson 3/8 rule, respectively. Moreover, the corresponding WEKM algorithms can simply referred to as the TWEKM, SWEKM, and S3/8WEKM algorithms. Although in theory, we may use arbitrary order of the Newton–Cotes quadrature formulas to obtain the more generalized WEKM algorithms, as the number of orders increases, the generated Runge's phenomenon may greatly amplify the numerical integration error. And this does not conform to the requirement of estimation theory (Mathews and Fink 2004). Table 4 proposes the weights assignment methods for the TWEKM, SWEKM, and S3/8WEKM algorithms. For all the WEKM algorithms, the sampling points are equidistant distributed on $[a, b]$, i.e.,

$$x_i = a + \frac{i-1}{N-1}(b-a) \quad (i = 1, \ldots, N)\,. \tag{33}$$

To implement the proposed CEKM and WEKM algorithms to compute the centroid TR of GT2 FLSs, we can have the following specific steps:

Step 1 Union all the fired rules in the GT2 FLSs to find the output GT2 FS $\tilde{A}$.
Step2 Break α into values of $0, 1/\Delta, 2/\Delta, \ldots, (\Delta\text{–}1)/\Delta, 1$. Decompose the GT2 FS into multiple α-planes $\tilde{A}_\alpha$ with above α values.

Table 4 Weights assignments method for the WEKM algorithms (Liu et al. 2012)

Algorithms	Integration rule	Weight
EKM	_____	$w_i = 1 (i = 1, \ldots, N)$
TWEKM	Composite trapezoidal rule	$w_i = \begin{cases} 1/2, & i = 1, N, \\ 1, & i \neq 1, N. \end{cases}$
SWEKM	Composite Simpson rule	$w_i = \begin{cases} 1/2, & i = 1, N, \\ 1, & i = 1 \bmod (2), \text{ 且} i \neq 1, N, \\ 2, & i = 0 \bmod (2), \text{ 且} i \neq N. \end{cases}$
S3/8WEKM	Composite Simpson 3/8 rule	$w_i = \begin{cases} 1/3, & i = 1, N, \\ 2/3, & i = 1 \bmod (3), \text{ 且} i \neq 1, N, \\ 1, & i = 2 \bmod (3), \text{ 且} i \neq N, \\ 1, & i = 0 \bmod (2), \text{ 且} i \neq N. \end{cases}$

Step 3 Compute the centroid $\alpha/[l_{\tilde{A}_\alpha}, r_{\tilde{A}_\alpha}]$ for each of the corresponding IT2 FS $R_{\tilde{A}_\alpha}$ using the CEKM and WEKM algorithms.
Step 4 Compute the union of all of the Step 3 centroids (see Eq. 21).
Step 5 Compare the performances of the proposed WEKM algorithms with the EKM algorithms by considering the CEKM algorithms as the benchmark.

Table 3 shows WEKM algorithms to compute the centroid end points of an IT2 FS.

In Table 4, except for the EKM algorithms, the weight assignments of the three types of WEKM algorithms can be obtained from Eqs. (25), (27), and (29) according to the following steps:

1. Replace $x_i (i = 0, 1, \ldots, n)$, $x_0 = a$, $x_n = b$ in Eq. (25); and $x_i (i = 0, 1, \ldots, 2n)$, $x_0 = a$, $x_{2n} = b$ in Eq. (27); and $x_i (i = 0, 1, 2, \ldots, 3n)$ and $x_0 = a$, $x_{3n} = b$ in Eq. (29) all by $x_i (i = 1, 2, \ldots, N)$ and $x_1 = a$, $x_N = b$.

Note that, in Table 4, a sign mod represents the modular arithmetic operator. $i = j \bmod (d)$ means $i = nd + j$, where n is an integer.

2. The common coefficients of $h/2$, $h/3$, and $3h/8$ in Eqs. (25), (27), and (29) can be cancelled by the quotient of two integrals (the numerator and denominator) in Table 2.
3. In Table 4, the weight values of TWEKM and SWEKM algorithms use one half of the coefficients in the parentheses of (25) and (27), and the weight values of S3/8WEKM algorithm use one-third of the coefficients in the parentheses of (29).
4. The number of discrete points N of SWEKM and S3/8WEKM algorithms is not only restricted to $N = 2n + 1$ and $N = 3n + 1$, but as required by Eqs. (27) and (29) ($N = 1$ mod (2) and $N = 1$ mod (3)).

According to the relations between Table 4 and Eqs. (25), (27), and (29), the proposed TWEKM, SWEKM, and S3/8WEKM algorithms approximate the numerical integration MFs as first, second, and third order of polynomials, respectively. Moreover, they are only special cases of the Newton–Cotes quadrature formulas.

Furthermore, the relations between CEKM algorithms (as in Table 2) and WEKM algorithms (as in Table 3) for performing the centroid TR of GT2 FLS can be summarized as follows:

1. The WEKM algorithms perform the centroid TR based on the sum operations based on the sampling points $x_i (i = 1, \ldots, N)$. When the iterations stop, the optimal switching points can be found to approximate the centroid TR end points. But the CEKM algorithms use the integral operations to perform the centroid TR, and the accurate centroid TR end points can be calculated by the CEKM algorithms. In theory, as the sampling point $N \to +\infty$, the solutions of the WEKM algorithms approach the solutions of the CEKM algorithms.
2. For the WEKM algorithms, we may increase N to obtain more accurate results. As for the CEKM algorithms, we can set smaller error bound ε to control the difference between two adjacent iterations to improve the computational accuracy.
3. Moreover, the WEKM algorithms achieve the numerical calculation according to the sum operations, whereas the CEKM algorithms perform the calculations with the integral operations. In conclusion, the WEKM algorithms can be viewed as the numerical implementation of CEKM algorithms according to the numerical integration methods.

4 Simulation studies

Four numerical simulation examples are given in this section. Here, we suppose that the FOU of output GT2 FS of the described GT2 FLSs and the corresponding secondary MFs are known by means of merging or weighting the fuzzy rules under the guidance of inference engine (Wang et al. 2008). For the first example, the whole FOU is composed of piecewise linear function (Liu 2008; Mendel et al. 2009; Liu et al. 2012), and the corresponding vertical slices (secondary MFs) are trapezoidal MFs. For the second example, the whole FOU is composed of two different types of MFs (Mendel and Liu 2007; Liu et al. 2012; Mendel and Liu 2013), and the corresponding secondary MFs are trapezoidal MFs. For the third example, the whole FOU is completely composed of Gaussian MFs (Liu 2008; Mendel et al. 2009; Liu et al. 2012), and the corresponding secondary MFs are triangle MFs. For the last example, the whole FOU is the Gaussian T2 primary MF with uncertain standard deviations, and the corresponding secondary MFs are triangle MFs.

In the simulations, we decompose α into Δ effective values as $\alpha = 0, 1/\Delta, \ldots, (\Delta - 1)/\Delta, 1$. The value of Δ is selected to range from 1 to 100. The primary variable of the centroid TR set is uniformly sampled, where the difference between adjacent points is $x_{i+1} - x_i = 0.05$. In this paper, we plot the centroid type-reduced T1 FS with the highest Δ and compare the defuzzified outputs derived from the type-reduced T1 FS with different Δ values. For Examples 1, 3, and 4, the sampling of primary variable is $x = 0{:}0.05{:}10$, whereas for the second example, the sampling of the primary variable is $x = -5{:}0.05{:}15$.

Note that the special α-plane with α-level equal to 0 does not play a role in performing the centroid TR of GT2 FLSs (see Eq. 22).

4.1 Case 1: Piecewise linear functions with trapezoidal vertical slices (Liu 2008; Mendel et al. 2009)

As shown in Fig. 1, the upper bound of the FOU is composed of the maximum of two triangular functions, i.e.,

$$u_1(x) = \begin{cases} \frac{x-1}{2}, & 1 \le x \le 3 \\ \frac{7-x}{4}, & 3 < x \le 7 \\ 0, & \text{otherwise} \end{cases} \tag{34}$$

and

$$u_2(x) = \begin{cases} \frac{x-2}{5}, & 2 \le x \le 6 \\ \frac{16-2x}{5}, & 6 < x \le 8 \\ 0, & \text{otherwise.} \end{cases} \tag{35}$$

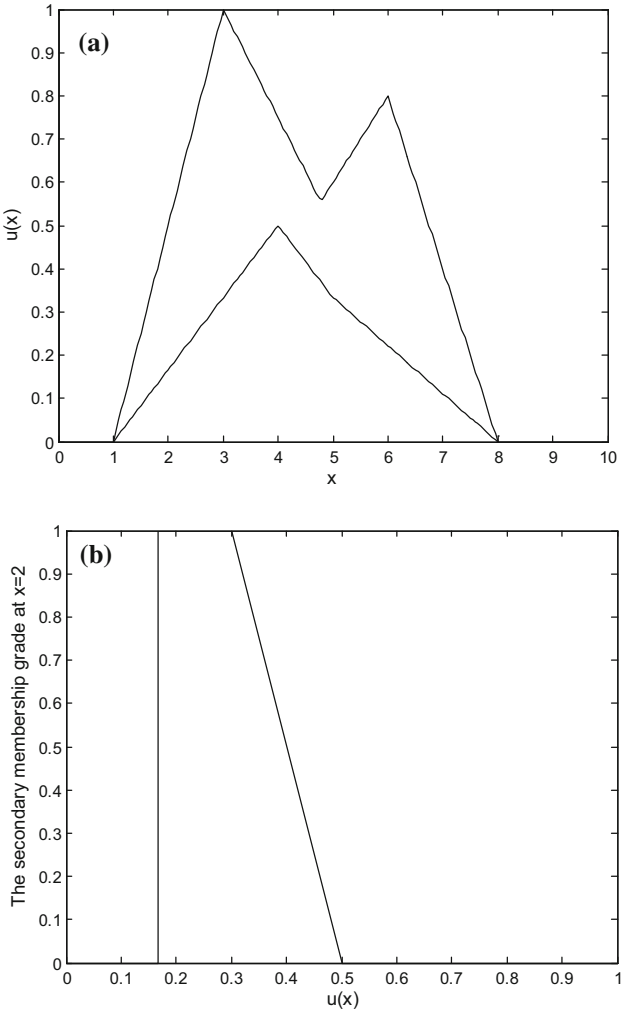

Fig. 1 **a** FOU of Case 1; **b** its corresponding vertical slice at $x = 2$

The lower bound of the FOU is composed of the maximum of two triangular functions

$$u_3(x) = \begin{cases} \frac{x-1}{6}, & 1 \le x \le 4 \\ \frac{7-x}{6}, & 4 < x \le 7 \\ 0, & \text{otherwise} \end{cases} \tag{36}$$

and

$$u_4(x) = \begin{cases} \frac{x-3}{6}, & 3 \le x \le 5 \\ \frac{8-x}{9}, & 5 < x \le 8 \\ 0, & \text{otherwise.} \end{cases} \tag{37}$$

For any value of x, the associated vertical slice is the non-symmetric trapezoid MF, whose top left and right end points are determined by

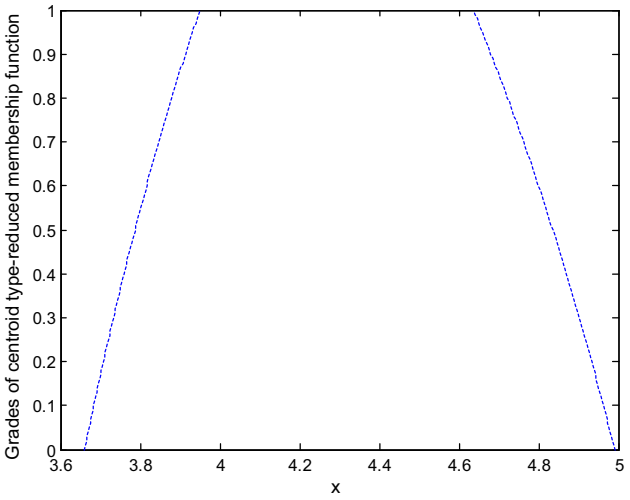

Fig. 2 The centroid type-reduced T1 FS (computed by CEKM algorithms) for $\Delta = 100$ in Case 1

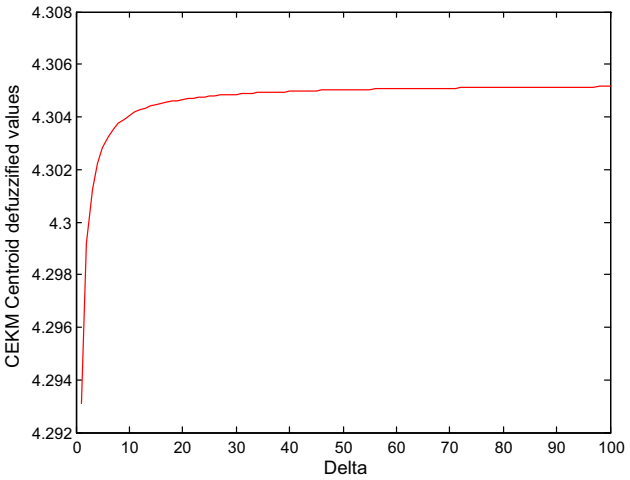

Fig. 3 The centroid defuzzified values (computed by CEKM algorithms) for Δ ranging from 1 to 100 in Case 1

$$L(x) = \underline{u}(x) + 0.6w(\overline{u}(x) - \underline{u}(x)),\ R(x) = \overline{u}(x) - 0.6(1-w)(\overline{u}(x) - \underline{u}(x)), \tag{38}$$

where $\underline{u}(x)$ and $\overline{u}(x)$ are the lower and upper bounds of the primary MF, respectively. For this case, we choose $w = 0$. In addition, the error accuracy was selected as $\varepsilon = 10^{-6}$.

First of all, we choose the CEKM algorithms as the benchmark to compute the centroid type-reduced T1 FS for $\Delta = 100$ and the centroid defuzzified values for Δ ranging from 1 to 100, which are shown in Figs. 2 and 3.

Then we study the performances of the proposed WEKM algorithms. For $\Delta = 100$, the type-reduced T1 FSs (see Eq. 21) for four types of WEKM algorithms are shown in Fig. 4a, and the absolute errors of centroid type-reduced T1 FS between the CEKM algorithms and the four types of

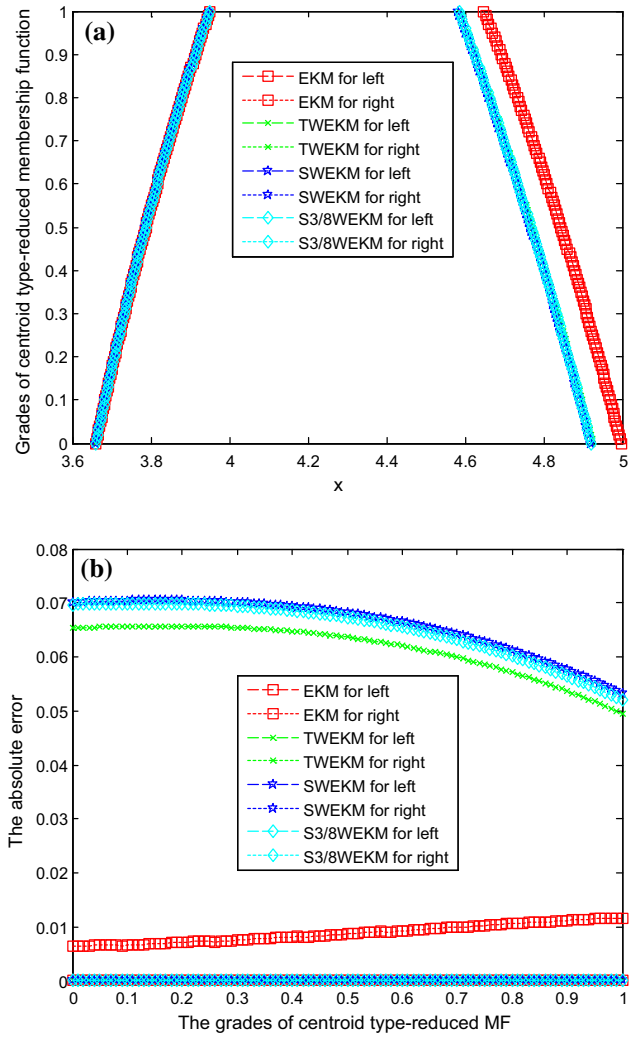

Fig. 4 **a** The centroid type-reduced T1 FSs for four types of WEKM algorithms; **b** the absolute error of centroid type-reduced T1 FS between the CEKM and WEKM algorithms for Case 1

WEKM algorithms are shown in Fig. 4b, where the grades of centroid type-reduced MF are chosen as the independent variable and the absolute error $|C_{\text{WEKM}} - C_{\text{CEKM}}|$ is the dependent variable.

Moreover, the centroid defuzzified values for four types of WEKM algorithm are shown in Fig. 5a, and the absolute error of centroid defuzzified values between the CEKM algorithms and four types of WEKM algorithms are shown in Fig. 5b, where the effective number of α-planes (Δ) is chosen as the independent variable and the absolute error $|y_{\text{WEKM}} - y_{\text{CEKM}}|$ is the dependent variable.

4.2 Case 2: Hybrid functions with trapezoidal vertical slices (Mendel et al. 2009)

As shown in Fig. 6, the upper bound of the FOU is the piecewise Gaussian function, i.e.,

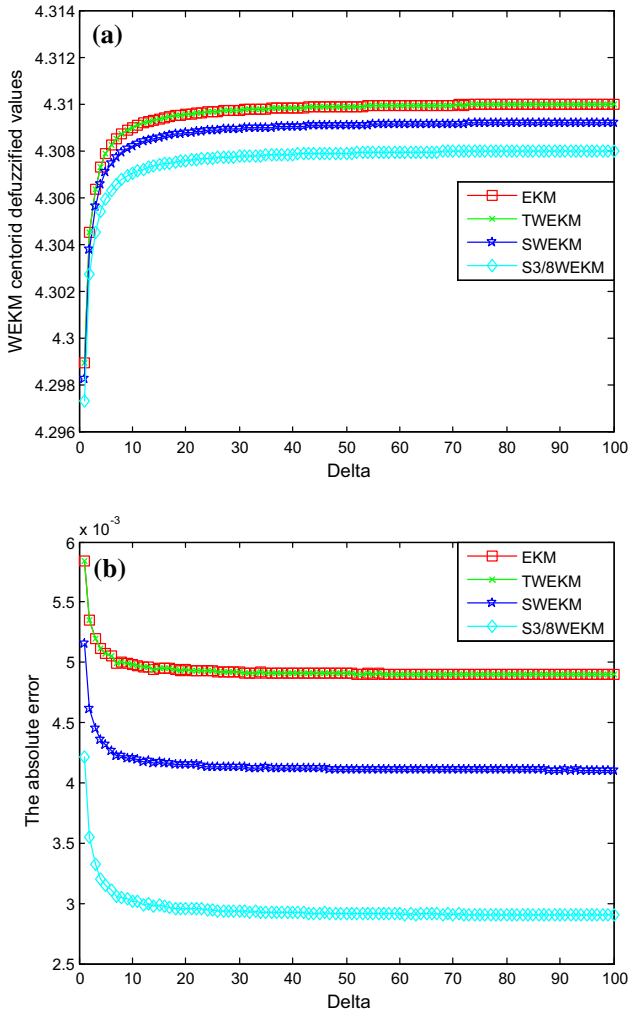

Fig. 5 **a** The centroid defuzzified values for four types of WEKM algorithms; **b** the absolute error of centroid defuzzified values between the CEKM and WEKM algorithms for Case 1

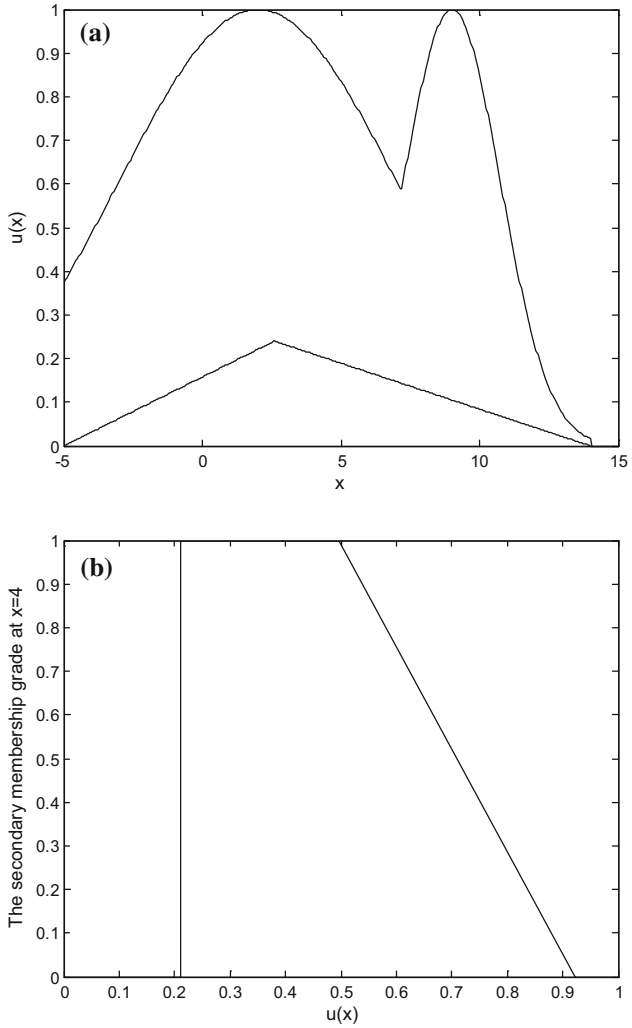

Fig. 6 **a** FOU of Case 2; **b** its corresponding vertical slice at $x = 4$

$$u_1(x) = \begin{cases} \exp\left[-\frac{1}{2}\left(\frac{x-2}{5}\right)^2\right], & -5 \le x \le 7.185 \\ \exp\left[-\frac{1}{2}\left(\frac{x-9}{1.75}\right)^2\right], & 7.185 < x \le 14. \end{cases} \tag{39}$$

The lower bound of the FOU is the piecewise linear function, i.e.,

$$u_2(x) = \begin{cases} \frac{0.6(x+5)}{19}, & -5 \le x \le 2.6 \\ \frac{0.4(14-x)}{19}, & 2.6 < x \le 14. \end{cases} \tag{40}$$

For any value of x, the corresponding vertical slice is also chosen in the form of Eq. (38), and here we still select $w = 0$ for this case.

The CEKM algorithms are used to compute the centroid type-reduced T1 FS for $\Delta = 100$ and the centroid defuzzified values for Δ ranging from 1 to 100 as shown in Figs. 7 and 8.

For $\Delta = 100$, the type-reduced T1 FSs for four types of WEKM algorithms are shown in Fig. 9a, and the absolute

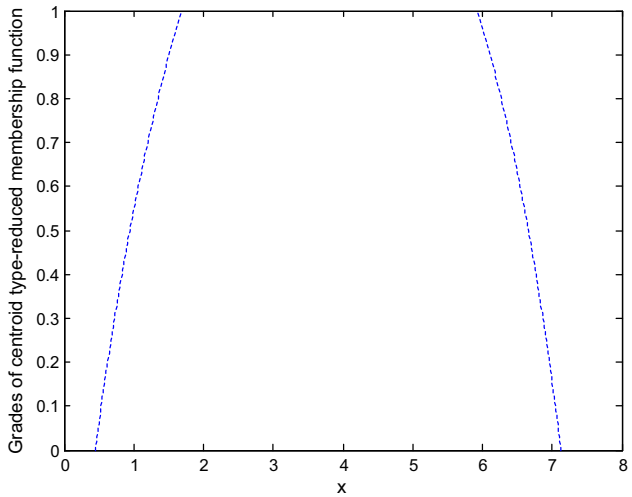

Fig. 7 The centroid type-reduced T1 FS (computed by CEKM algorithms) for $\Delta = 100$ in Case 2

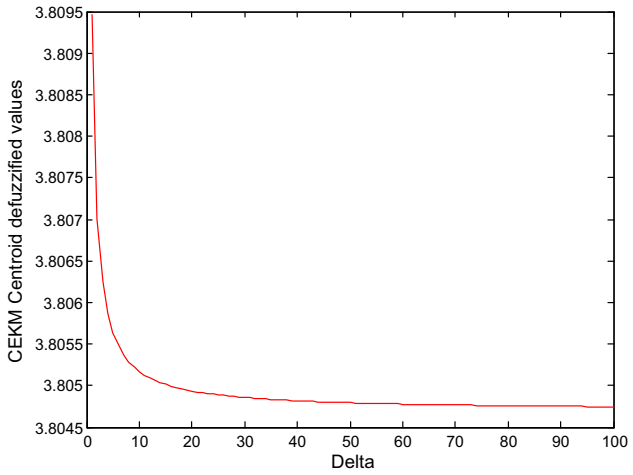

Fig. 8 The centroid defuzzified values (computed by CEKM algorithms) for Δ ranging from 1 to 100 in Case 2

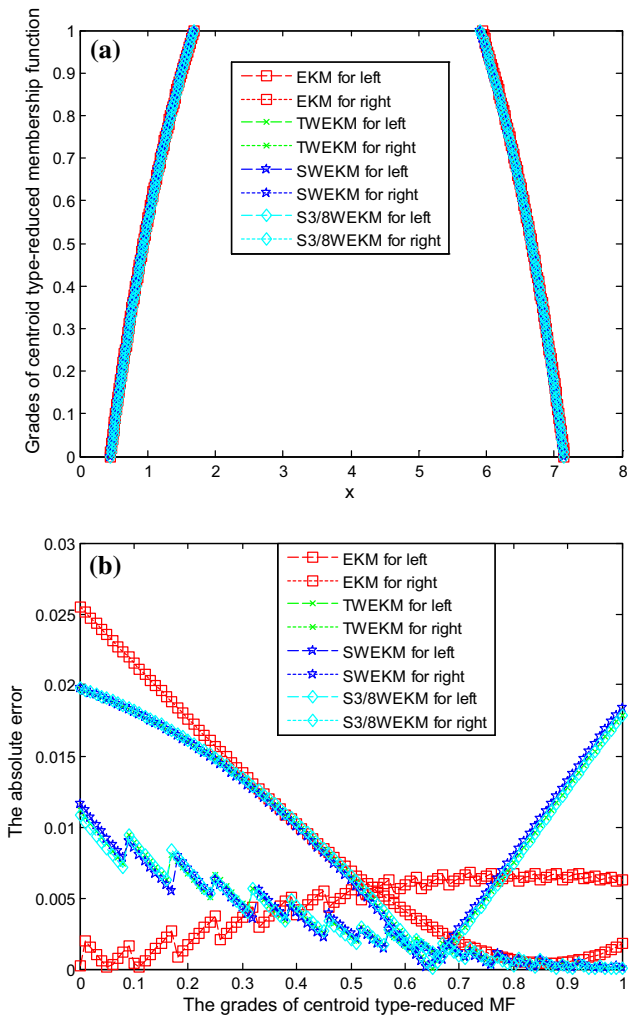

Fig. 9 **a** The centroid type-reduced T1 FSs for four types of WEKM algorithms; **b** the absolute error of centroid type-reduced T1 FS between the CEKM and WEKM algorithms for Case 2

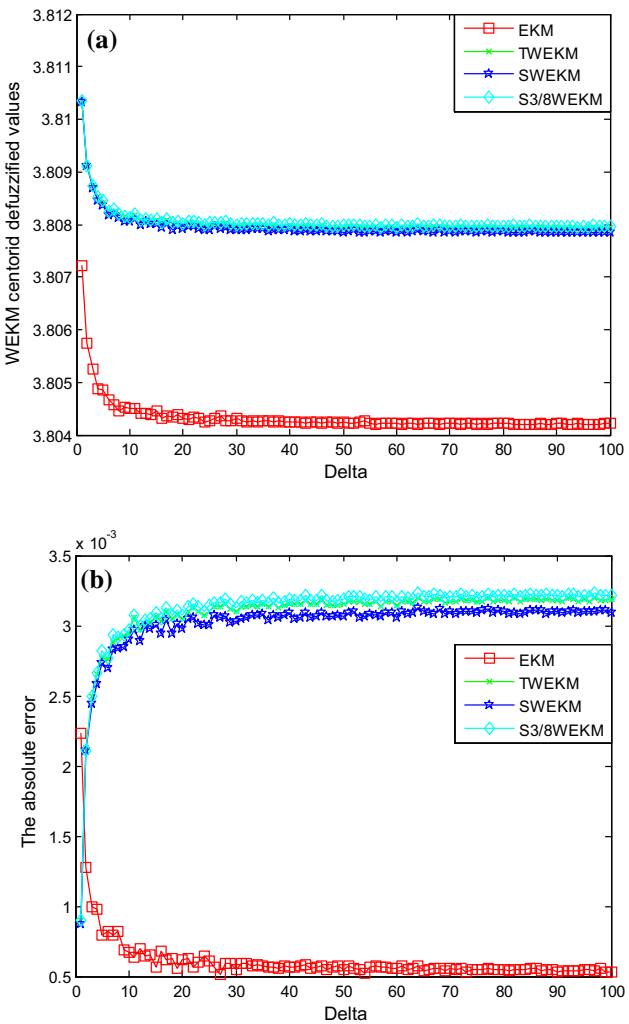

Fig. 10 **a** The centroid defuzzified values for four types of WEKM algorithms; **b** the absolute errors of centroid defuzzified values between the CEKM and WEKM algorithms for Case 2

errors of the centroid type-reduced T1 FS between the CEKM algorithms and four types of WEKM algorithms are shown in Fig. 9b.

The centroid defuzzified values for four types of WEKM algorithm are shown in Fig. 10a, and the absolute errors of centroid defuzzified values between the CEKM algorithms and the four types of WEKM algorithms are shown in Fig. 10b.

4.3 Case 3: Piecewise Gaussian functions with triangle vertical slices (Liu 2008; Mendel et al. 2009)

See Fig. 11, the upper bound of the FOU is the maximum of two Gaussian functions, i.e.,

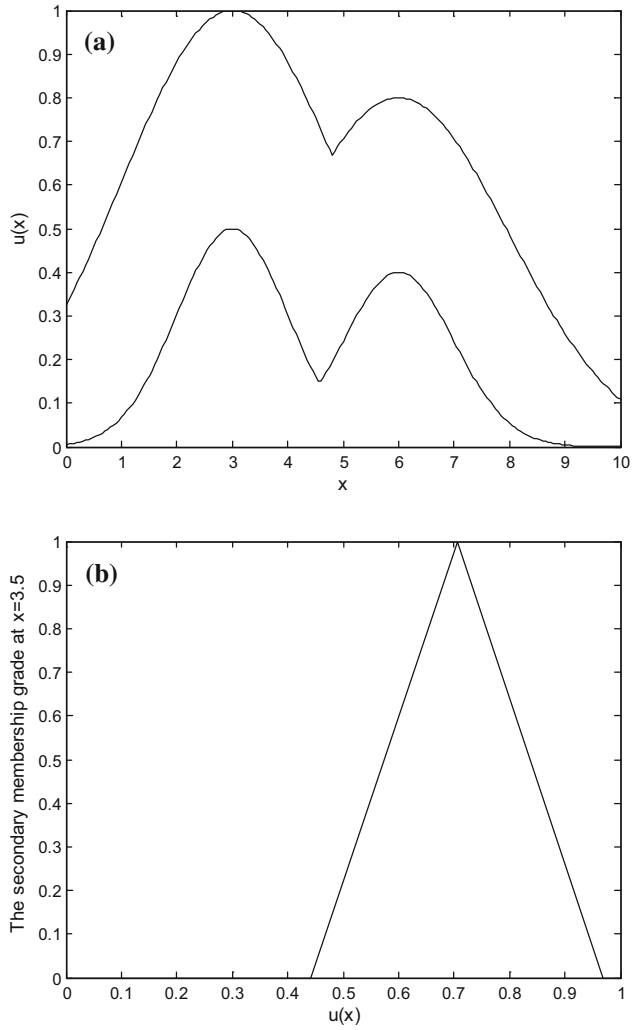

Fig. 11 **a** FOU of Case 3; **b** its corresponding vertical slice at $x = 3.5$

$$u_1(x) = \exp\left[-\frac{1}{2}\left(\frac{x-3}{2}\right)^2\right] \tag{41}$$

and

$$u_2(x) = 0.8\exp\left[-\frac{1}{2}\left(\frac{x-6}{2}\right)^2\right]. \tag{42}$$

The lower bound of the FOU is the maximum of another two Gaussian functions, i.e.,

$$u_3(x) = 0.5\exp\left[-\frac{(x-3)^2}{2}\right] \tag{43}$$

and

$$u_4(x) = 0.4\exp\left[-\frac{(x-6)^2}{2}\right]. \tag{44}$$

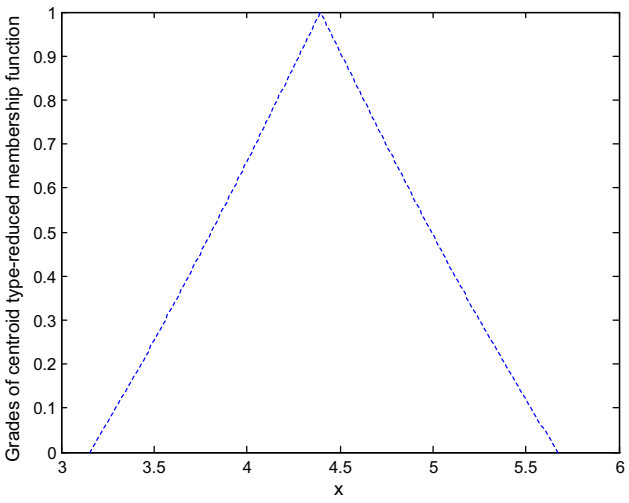

Fig. 12 The centroid type-reduced T1 FS (computed by CEKM algorithms) for $\Delta = 100$ in Case 3

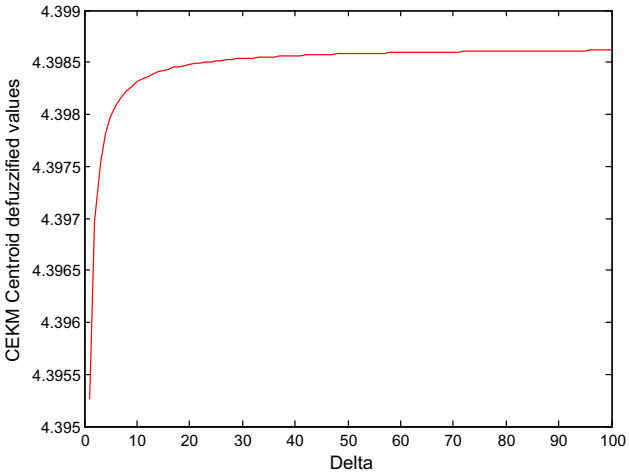

Fig. 13 The centroid defuzzified values (computed by CEKM algorithms) for Δ ranging from 1 to 100 in Case 3

For any value of x, the corresponding vertical slice is the triangle MF, whose apex is determined by

$$\text{Apex} = \underline{u}(x) + w(\overline{u}(x) - \underline{u}(x)), \tag{45}$$

where $\underline{u}(x)$ and $\overline{u}(x)$ are the lower and upper bounds of the primary MF, respectively. In this test, we choose $w = 0.5$.

We choose the CEKM algorithms as the benchmark to compute the centroid type-reduced T1 FS for $\Delta = 100$ and the centroid defuzzified values for Δ ranging from 1 to 100, which are shown in Figs. 12 and 13.

As $\Delta = 100$, the type-reduced T1 FSs for four types of WEKM algorithms are shown in Fig. 14a, and the absolute errors of centroid type-reduced T1 FS between the CEKM algorithms and the four types of WEKM algorithms are shown in Fig. 14b.

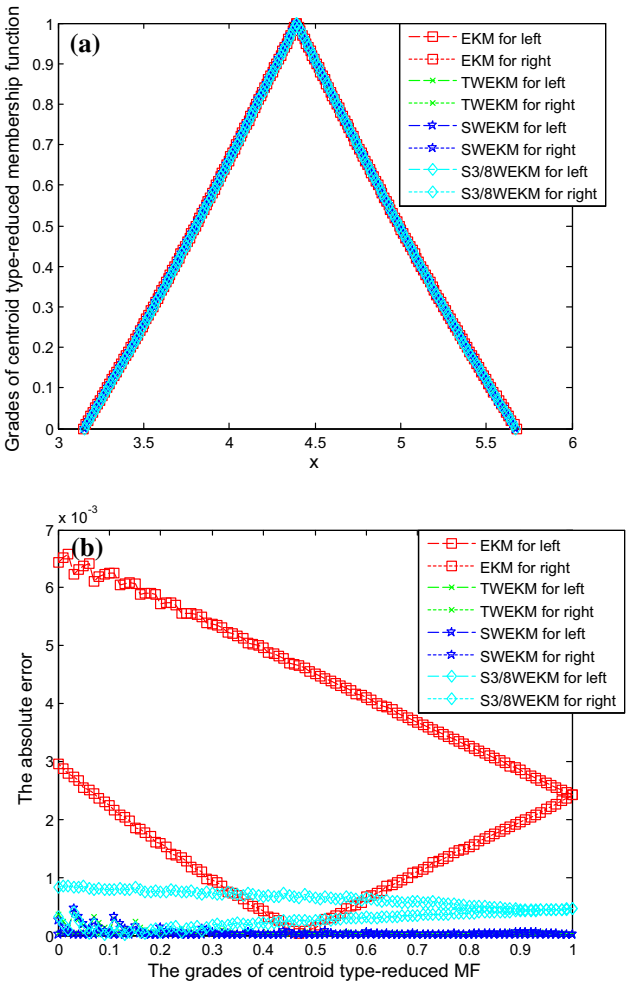

Fig. 14 **a** The centroid type-reduced T1 FSs for four types of WEKM algorithms; **b** the absolute errors of centroid type-reduced T1 FS between the CEKM and WEKM algorithms for Case 3

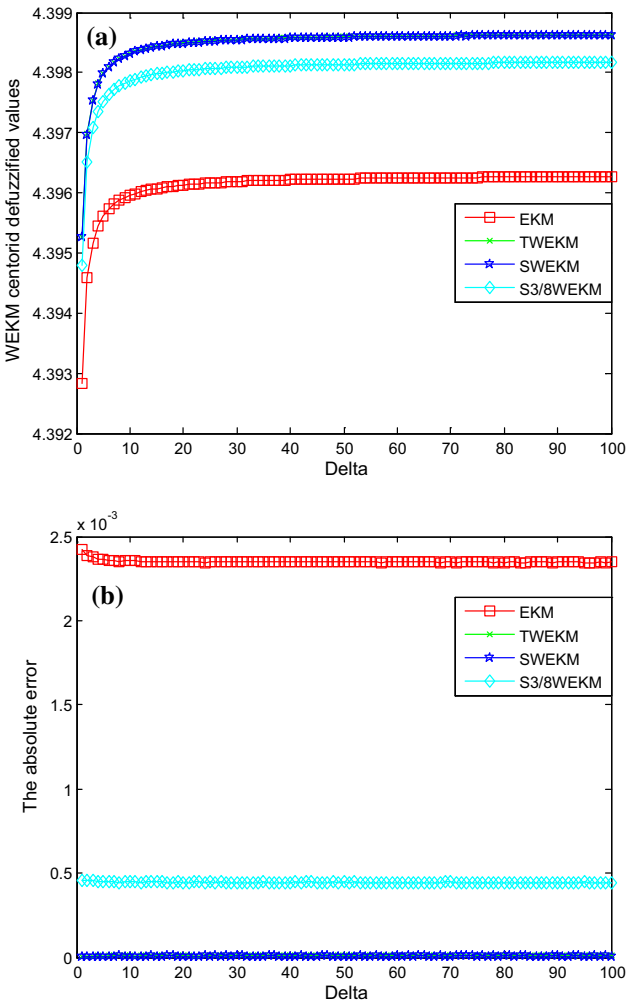

Fig. 15 **a** The centroid defuzzified values for four types of WEKM algorithms; **b** the absolute errors of centroid defuzzified values between the CEKM and WEKM algorithms for Case 3

Moreover, the centroid defuzzified values for four types of WEKM algorithm are shown in Fig. 15a, and the absolute errors of centroid defuzzified values between the CEKM algorithms and four types of WEKM algorithms are shown in Fig. 15b.

4.4 Case 4: Gaussian T2 primary MF (with fixed mean and uncertain standard deviations) with triangle vertical slices (Mendel et al. 2009; Liu et al. 2012)

As shown in Fig. 16, the upper bound of the FOU is a Gaussian MF, i.e.,

$$u_1(x) = \exp\left[-\frac{1}{2}\left(\frac{x-3}{1.75}\right)^2\right]. \tag{46}$$

The lower bound of the FOU is another Gaussian function, i.e.,

$$u_2(x) = \exp\left[-\frac{1}{2}\left(\frac{x-3}{0.25}\right)^2\right]. \tag{47}$$

For any value of x, the corresponding vertical slice is chosen in the form of Eq. (45). In this test, we choose $w = 0.75$.

The CEKM algorithms are still chosen as the benchmark to compute the centroid type-reduced T1 FS for $\Delta = 100$ and the centroid defuzzified values for Δ ranging from 1 to 100 as shown in Figs. 17 and 18.

For $\Delta = 100$, the type-reduced T1 FSs for four types of WEKM algorithms are shown in Fig. 19a, and the absolute errors of centroid type-reduced T1 FS between the CEKM algorithms and four types of WEKM algorithms are shown in Fig. 19b.

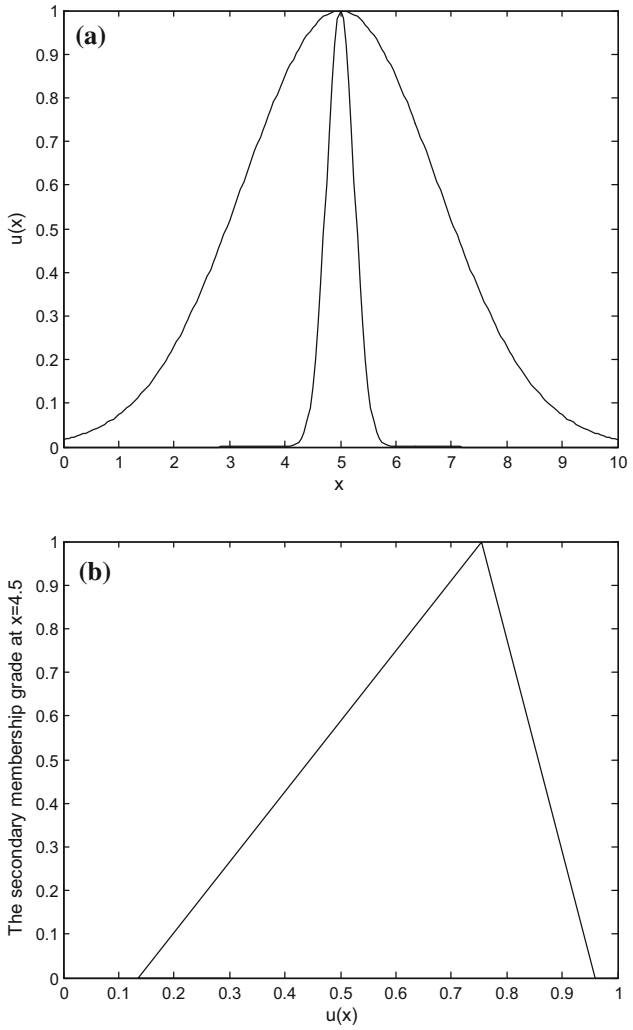

Fig. 16 **a** FOU of Case 4; **b** its corresponding vertical slice at $x = 4.5$

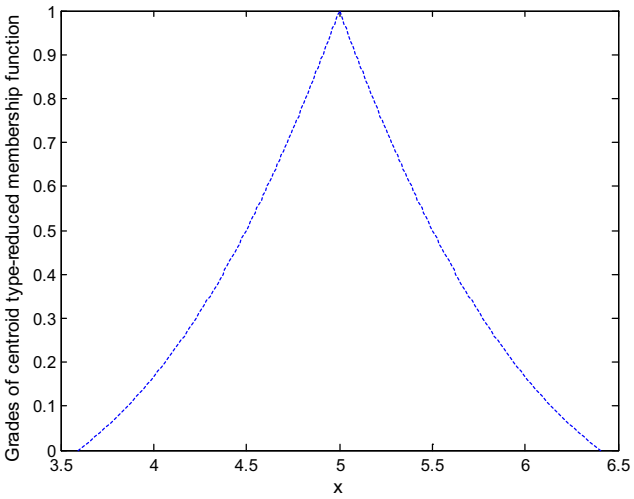

Fig. 17 The centroid type-reduced T1 FS (computed by CEKM algorithms) for $\Delta = 100$ in Case 4

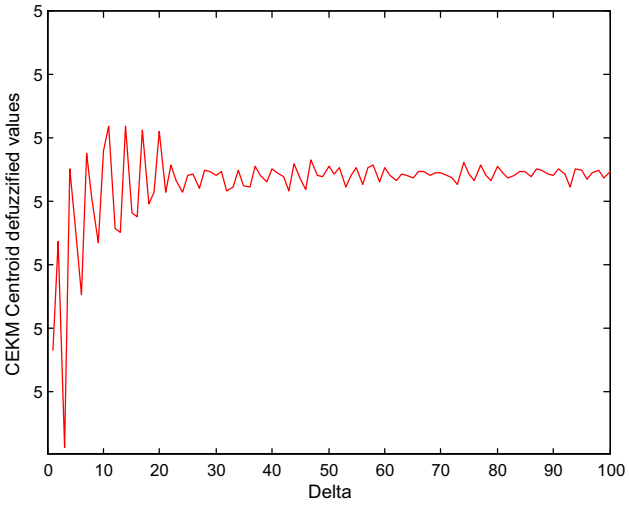

Fig. 18 The centroid defuzzified values (computed by CEKM algorithms) for Δ ranging from 1 to 100 in Case 4

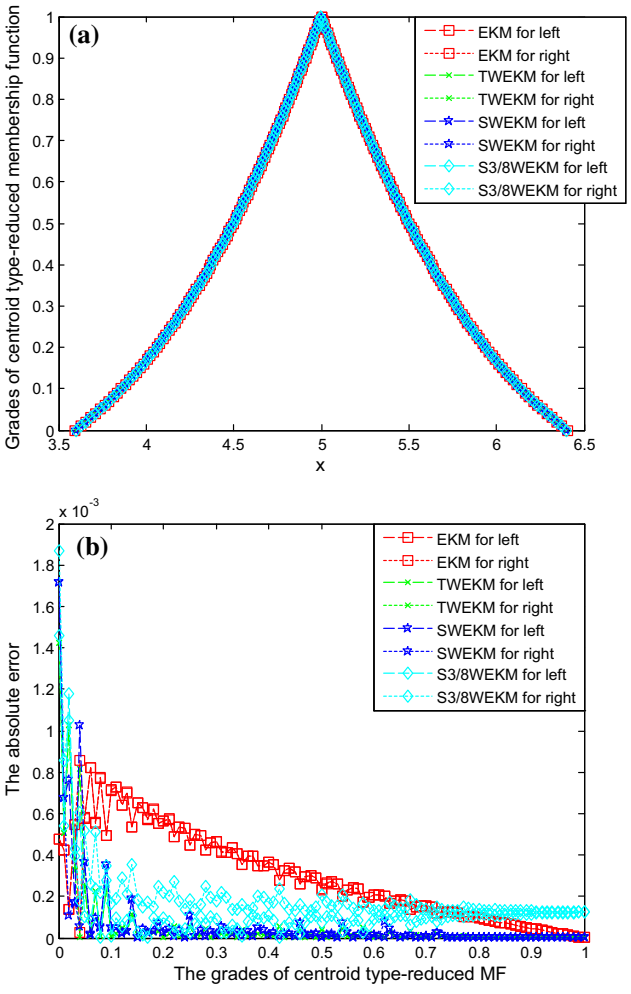

Fig. 19 **a** The centroid type-reduced T1 FSs for four types of WEKM algorithms; **b** the absolute errors of centroid type-reduced T1 FS between the CEKM and WEKM algorithms for Case 4

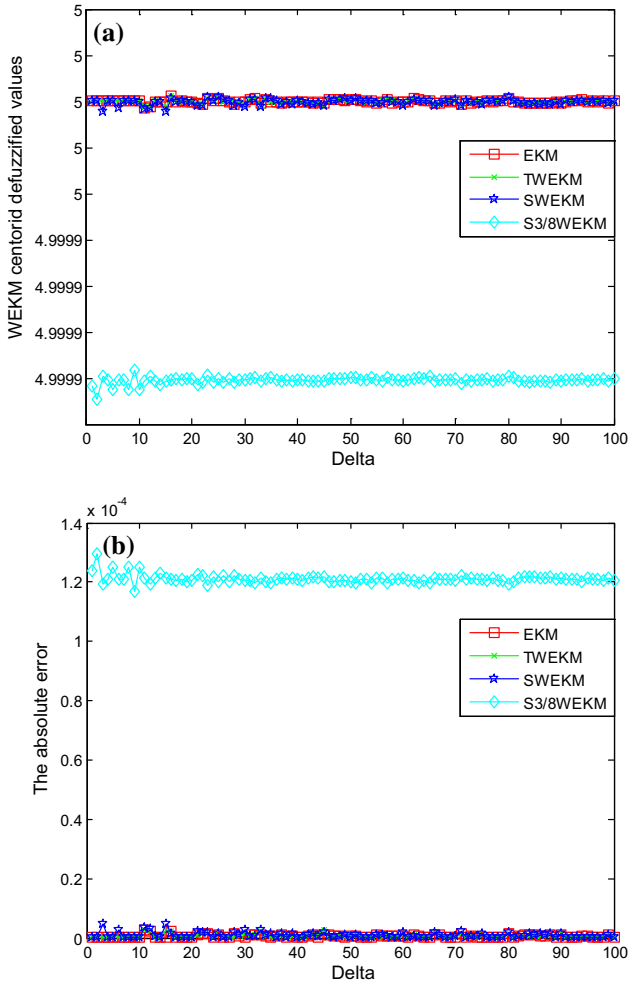

Fig. 20 **a** The centroid defuzzified values for four types of WEKM algorithms; **b** the absolute errors of centroid defuzzified values between the CEKM and WEKM algorithms for Case 4

Finally, the centroid defuzzified values for four types of WEKM algorithm are shown in Fig. 20a, and the absolute errors of centroid defuzzified values between the CEKM algorithms and four types of WEKM algorithms are shown in Fig. 20b.

4.5 Discussion

For all of the above four cases, we first discuss the performances for performing the centroid type-reduced T1 FSs (both left side and right side of the four types of WEKM algorithms). After observing Figs. 4b, 9b, 14b, and 19b, we obtain the conclusion that the proposed three types of WEKM algorithms outperform the special WEKM (EKM) algorithms in Cases 3 and 4, the proposed WEKM algorithms and EKM algorithms both have their own advantages on the particular ranges in Case 2, and the EKM algorithms outperform the proposed WEKM algorithms in Case 1.

For the four cases, the more important thing is to discuss computing the centroid defuzzified values for the GT2 FLSs. In order to measure the performances the EKM, TWEKM, SWEKM, and S3/8WEKM algorithms further, we define and compute the relative error $|y_{\mathrm{WEKM}_i} - y_{\mathrm{CEKM}_I}|/|y_{\mathrm{CEKM}_i}|(i = 1, \ldots, 4)$ for all four cases for Δ ranging from 1 to 100. Table 5 gives the average relative error of the centroid defuzzified values as Δ ranges from 1 to 100, where the last row in Table 5 is the total mean average relative error for the four types of WEKM algorithms.

From Table 5 and Figs. 5b, 10b, 15b, and 20b, the following conclusions can be obtained:

1. From Figs. 5b, 10b, 15b, and 20b, we find that the absolute errors of four types of WEKM algorithms all converge, in the four cases. In Case 1, the S3/8WEKM algorithms obtain the minimum absolute errors, the EKM and TWEKM almost obtain the maximum absolute errors whose values are almost the same, and the SWEKM algorithms obtain the medium absolute errors. In Case 2, the absolute errors obtained by the EKM algorithms are less than that by the proposed three types of WEKM algorithms at most values of Δ. However, at several initial Δ values, the corresponding absolute errors obtained by the EKM algorithms are greater than that by the proposed WEKM algorithms. In Case 3, the absolute errors obtained by the proposed WEKM algorithms are all less than that by the EKM algorithms, whereas the TWEKM and SWEKM algorithms obtain the minimum absolute errors, and the S3/8WEKM algorithms obtain the medium absolute error. In Case 4, the S3/8WEKM algorithms obtain the maximum absolute error and the other three types of WEKM algorithms obtain the relatively small absolute errors whose values are almost the same.
2. From Table 5, we find that the maximum average relative errors of the EKM and TWEKM algorithms are both 0.114651%, the maximum average relative error of the SWEKM algorithms is 0.096386%, and the maximum average relative error of the S3/8WEKM algorithms is 0.081919%.
3. According to items 1 and 2, we can see that the appropriate WEKM algorithms may be selected to obtain better computational accuracy than the EKM algorithms.

In the above analysis, we choose $w = 0$ for Cases 1 and 2, $w = 0$ for Cases 1 and 2, $w = 0.5$ for Case 3, and $w = 0.75$ for Case 4. Next, we select different values of w for four cases. In addition, we summarize the results of average relative error of the centroid defuzzified values as Δ ranges from 1 to 100 in Table 6. Here we choose $w = 1$ for Case 1, $w = 0.25$ for Case 2, $w = 0.75$ for Case 3, and $w = 0.5$ for Case 4.

Table 5 Average relative error of the centroid defuzzified values as Δ ranges from 1 to 100

Algorithm	EKM	TWEKM	SWEKM	S3/8WEKM
Case 1	0.00114651	0.00114651	0.00096386	0.00068703
Case 2	0.00016067	0.00081159	0.00079131	0.00081919
Case 3	0.00053427	0.00000234	0.00000152	0.00010139
Case 4	0.00000011	0.00000012	0.00000017	0.00002422
Total mean	0.00046039	0.00049014	0.00043922	0.00040796

Table 6 Average relative error of the centroid defuzzified values as Δ ranges from 1 to 100 (with different values of w)

Algorithm	EKM	TWEKM	SWEKM	S3/8WEKM
Case 1	0.00707651	0.00707651	0.00696286	0.00685563
Case 2	0.01061840	0.00942086	0.00944295	0.00940927
Case 3	0.00060771	0.00000218	0.00000142	0.00011553
Case 4	0.00000015	0.00000014	0.00000019	0.00002265
Total mean	0.00457569	0.00412492	0.00410186	0.00410077

In Table 6, we can also find that the proposed WEKM algorithms still have better performances than the EKM algorithms for performing the centroid type-reduction and defuzzification of GT2 FLSs.

In order to apply these algorithms in real-time applications, we must study the computation times. In Cases 1, 3, and 4, the number of sampling points of the primary variable is fixed as $x = 0{:}0.05{:}10$, whereas for Case 2, the number of sampling points of the primary variable is fixed as $x = -5{:}0.05{:}15$. Here we only choose $w = 0$ for Case 1, $w = 0$ for Case 2 $w = 0.5$ for Case 3, and $w = 0.75$ for Case 4. However, for studying the computation times of centroid defuzzified values, in the following we choose the number of the primary variable to be $N = 100{:}50{:}2000$. The specific computation times of the algorithms depend on the environments of both software and hardware, whose computational results are unrepeatable. In this paper, we choose the simulation platform as the Microsoft Windows XP Professional system, and the dual-core CPU Dell desktop with E5300@2.6 GHz and 2.00 GB memory. All the algorithms are programmed using the MATLAB 2013a. The number of effective α-planes (Δ) is fixed at 200, and the number of sampling points of the primary variable is selected as the independent variable. The computation times are shown in Figs. 21, 22, 23 and 24.

If we do not consider the fluctuations in computation time for the number of sampling points N, all four WEKM algorithms emerge in a linear way with respect to the N. So we adopt the least square regression model $t = a + bN$ for all algorithms, where t is the computation time, and the regression coefficients are given in Table 7. Moreover, we define the computation time difference rate for four types of algorithms as

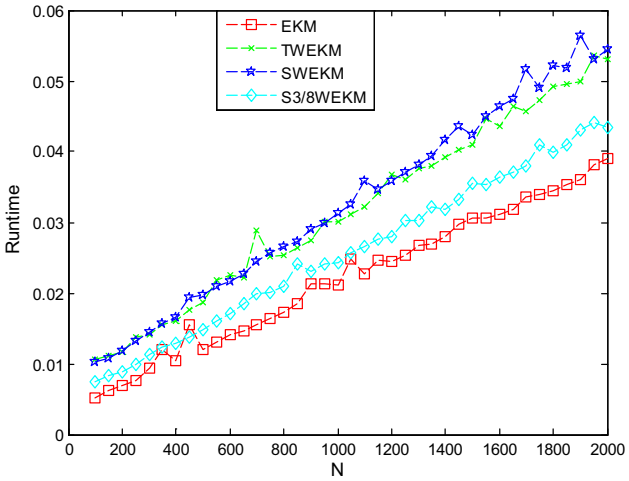

Fig. 21 Comparisons of runtime for Case 1 (with $N = 100{:}50{:}2000$)

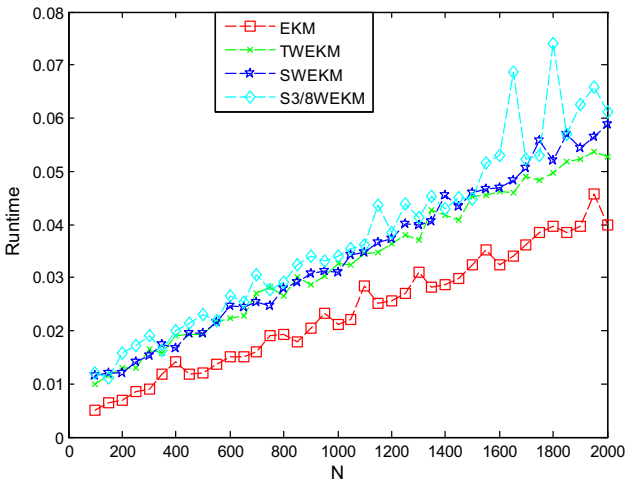

Fig. 22 Comparisons of runtime for Case 2 (with $N = 100{:}50{:}2000$)

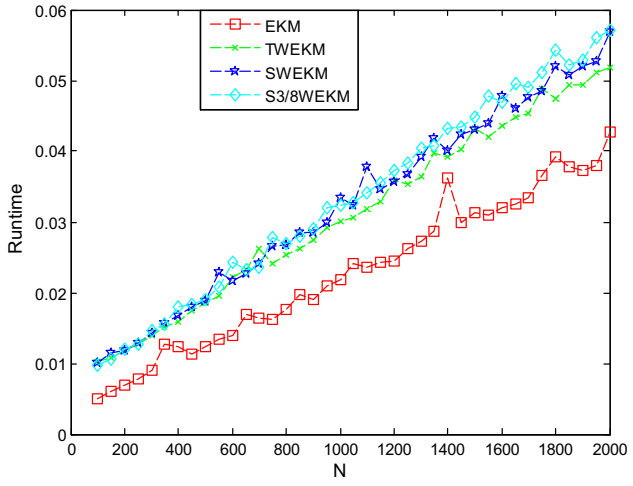

Fig. 23 Comparisons of runtime for Case 3 (with $N = 100{:}50{:}2000$)

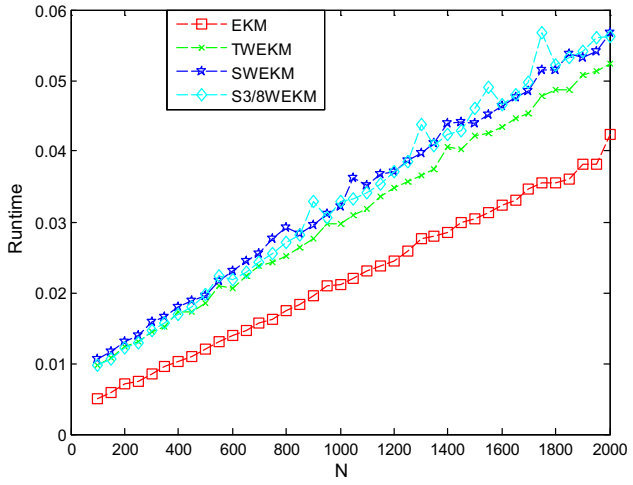

Fig. 24 Comparisons of runtime for Case 4 (with $N = 100{:}50{:}2000$)

$$\left(\max_{i=1,\ldots,4}\{t_i\} - \min_{i=1,\ldots,4}\{t_i\}\right) \Big/ \max_{i=1,\ldots,4}\{t_i\}, \tag{48}$$

where $t_i\,(i = 1, \ldots, 4)$ is the computation time for four types of algorithms.

From Table 7 and Figs. 21, 22, 23 and 24, as for different numbers of sampling points, we observe that the convergence speed of the proposed WEKM algorithms is faster than that

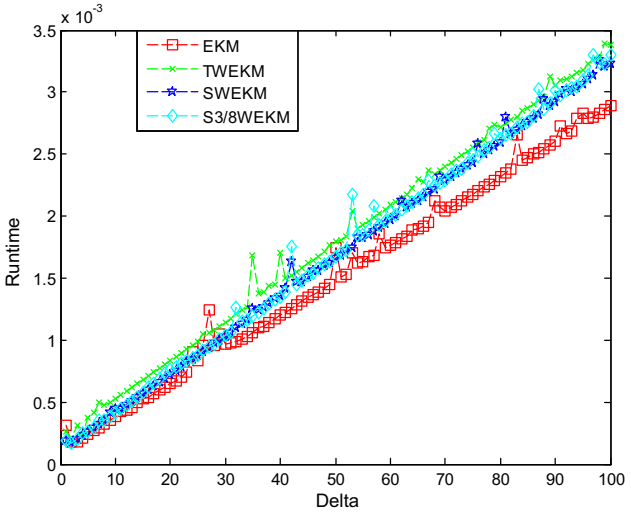

Fig. 25 Comparisons of runtime for Case 1 (with $\Delta = 1{:}1 : 100$)

of the EKM algorithms. In other words, the computation time of EKM algorithm is less than that of the WEKM algorithms. That is because the weights assignment for the proposed WEKM algorithms is more complex than for the EKM algorithms. However, for the proposed WEKM algorithms, we may use less number of sampling points to obtain the same computation time as the EKM algorithms. When the number of sampling points is chosen as $N = 100{:}50{:}2000$, the computation time difference rate for the four WEKM algorithms for the four cases is between 16.38 and 58.28%.

Next we study the computation times for four cases, but choose the number of effective α-plane as the independent variable, where that number varies from 1 to 100. The computation times are shown in Figs. 25, 26, 27 and 28. Observing these figures, we can find that the convergence speed of the proposed WEKM algorithms is faster than that of the EKM algorithms for performing the centroid type-reduction of GT2 FLSs.

The proposed WEKM algorithms can be used for studying the TR of IT2 or GT2 FLSs. If only the computation accuracy is taken into account, observe from Table 5 that S3/8WEKM algorithm is the best choice. In the practical design and application of T2 FLSs, real-time computations are needed and the sampling rate $(1/N)$ is fixed. Considering both Table 5 and

Table 7 Compute the regression model by least square for four types of algorithms

Regression coefficient	EKM $a/10^{-3}$	$b/10^{-3}$	TWEKM $a/10^{-3}$	$b/10^{-3}$	SWEKM $a/10^{-3}$	$b/10^{-3}$	S3/8WEKM $a/10^{-3}$	$b/10^{-3}$
Case 1	0.017	4.246	0.023	8.004	0.024	7.247	0.019	5.421
Case 2	0.019	3.503	0.023	8.655	0.025	7.716	0.028	8.051
Case 3	0.018	3.578	0.023	7.622	0.024	7.683	0.025	7.143
Case 4	0.018	2.859	0.023	7.521	0.024	8.481	0.025	6.963
Total mean	0.018	3.547	0.023	7.951	0.024	7.782	0.024	6.894

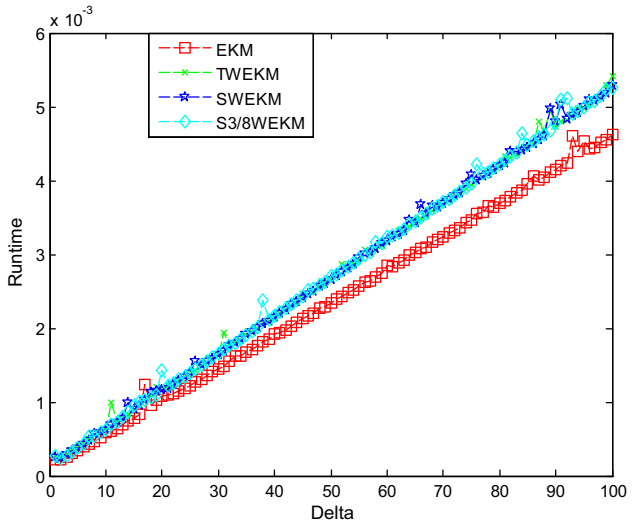

Fig. 26 Comparisons of runtime for Case 2 (with $\Delta = 1{:}1 : 100$)

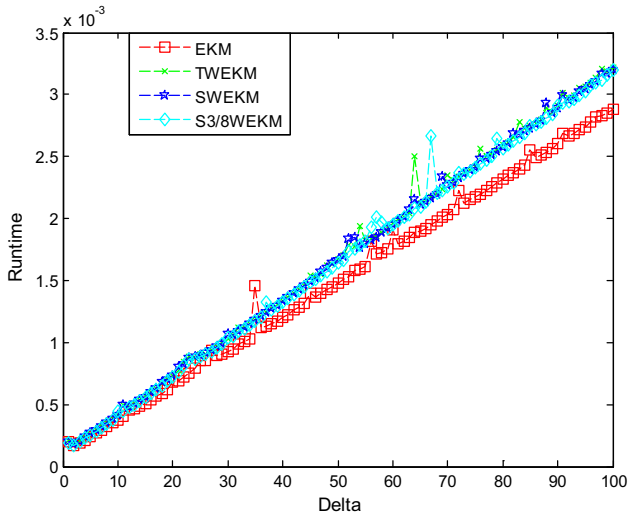

Fig. 27 Comparisons of runtime for Case 3 (with $\Delta = 1{:}1 : 100$)

Figs. 21, 22, 23 and 24 simultaneously, we suggest one share the S3/8WEKM algorithms for performing centroid TR of GT2 FLSs as the piecewise linear functions with trapezoidal vertical slices in Case 1, adopt the EKM algorithms for computing the centroid TR of GT2 FLSs as the hybrid functions with trapezoidal vertical slices in Case 2, use the TWEKM or SWEKM algorithms for performing the centroid TR of GT2 FLSs as the piecewise Gaussian functions with triangle vertical slices in Case 3, and adopt the EKM or TWEKM algorithms for computing the centroid TR of GT2 FLSs as the Gaussian T2 primary MF with triangle vertical slices in Case 4.

Finally, it should be pointed out that this paper only focuses on the comparison of EKM and WEKM algorithms and their performance in theory. When the number of sampling points is fixed, the proposed WEKM algorithms can improve the computational accuracy compared with the EKM algorithms. However, if the requirement of computation accuracy is not very high, the proposed WEKM algorithms may not show their advantages over the EKM algorithms. Simple EKM algorithms will do. In addition, the WEKM algorithms cannot be applied to problems in which there is no numerical integration.

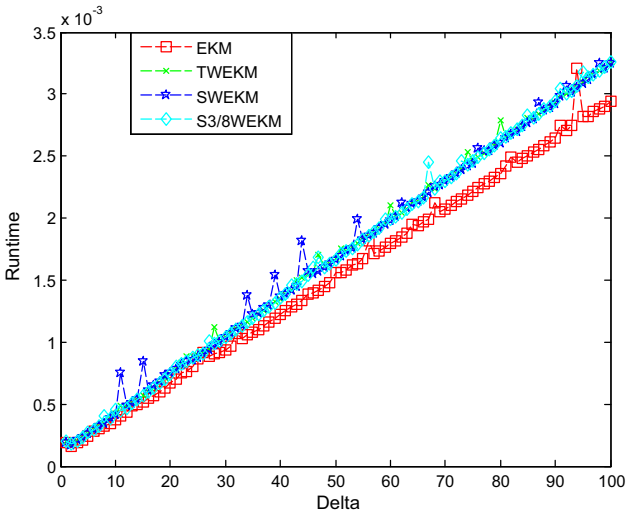

Fig. 28 Comparisons of runtime for Case 4 (with $\Delta = 1{:}1 : 100$)

5 Conclusions

In this paper, the EKM algorithms are extended to three different forms of WEKM algorithms according to the Newton–Cotes quadrature formulas in numerical integration. The CEKM algorithms are chosen as the benchmark for performing centroid TR and defuzzification of GT2 FLSs. The proposed WEKM algorithms are then employed to compute both the centroid type-reduced T1 FSs and their defuzzified values, and are compared with the EKM algorithms. This paper applies the WEKM algorithms that are derived and given in Liu et al. (2012) for an IT2 FS at each alpha-plane. Four simulation examples are provided to show that the proposed WEKM algorithms can obtain better absolute error and faster convergence speed than the EKM algorithms.

There are much interesting works that lie ahead, including the study of center-of-sets type-reduction of GT2 FLSs (Mendel 2014; Wagner and Hagras 2010), and making use of intelligent optimization algorithms (Chen et al. 2013; Hidalgo et al. 2012; Zhai et al. 2012; Hsu and Juang 2013; Olivas et al. 2016; Juang and Chang 2010) to design and apply the IT2 or GT2 FLSs (Chen and Wang 2015; Bilgin et al. 2013; Gonzalez et al. 2017; Caraveo et al. 2017; Castillo et al. 2016b; Gonzalez et al. 2016; Melin et al. 2014; Linda and Manic 2012). Future studies will be focused on T2 FLSs

design and applications based on Mendel (2001), Khosravi and Nahavandi (2014), Chen et al. (2016), Mendel (2014), Sanchez et al. (2015), Castillo et al. (2016a) and Mendel (2013) and this paper.

Acknowledgements This paper is partially sponsored by the Natural Science Foundation of China (No. 61374113) and Fundamental Research Funds for Liaoning's Universities (No. JL201615410). The author is so thankful to Prof. J. M. Mendel, who has offered the author some worthy suggestions.

Compliance with ethical standards

Conflict of interest All authors declare that they have no conflict of interest.

References

Aisbett J, Rickard JT, Morgenthaler DG (2010) Type-2 fuzzy sets as functions on spaces. IEEE Trans Fuzzy Syst 18(4):841–844

Biglarbegian M, Melek WW, Mendel JM (2010) On the stability of interval type-2 TSK fuzzy logic systems. IEEE Trans Syst Man Cybern B Cybern 40(3):798–818

Biglarbegian M, Melek WW, Mendel JM (2011) Design of novel interval type-2 fuzzy controllers for modular and reconfigurable robots: theory and experiments. IEEE Trans Ind Electron 58(4):1371–1384

Bilgin A, Hagras H, Malibari A et al (2013) Towards a linear general type-2 fuzzy logic based approach for computing with words. Soft Comput 17(12):2203–2222

Caraveo C, Valdez F, Castillo O (2017) Nature-inspired design of hybrid intelligent systems. Studies in computational intelligence

Castillo O, Amador-Angulo L, Castro JR, Garcia-Valdez M (2016a) A comparative study of type-1 fuzzy logic systems, interval type-2 fuzzy logic systems and generalized type-2 fuzzy logic systems in control problems. Inf Sci 354(Part C):257–274

Castillo O, Cervantes L, Soria J et al (2016b) A generalized type-2 fuzzy granular approach with applications to aerospace. Inf Sci 354(C):165–177

Chen Y, Wang DZ (2015) Studies on centroid type-reduction algorithms for interval type-2 fuzzy logic systems. In: IEEE international conference on big data and cloud computing, pp 344–349

Chen S, Chang Y, Pan J (2013) Fuzzy rules interpolation for sparse fuzzy rule-based systems based on interval type-2 gaussian fuzzy sets and genetic algorithms. IEEE Trans Fuzzy Syst 21(3):412–425

Chen Y, Wang DZ, Ning W (2015) Studies on centroid type-reduction algorithms for general type-2 fuzzy logic systems. Int J Innov Comput Inf Control 11(6):1987–2000

Chen Y, Wang DZ, Tong SC (2016) Forecasting studies by designing Mamdani interval type-2 fuzzy logic systems: with the combination of BP algorithms and KM algorithms. Neurocomputing 174(Part B):1133–1146

Coupland S, John R (2007) Geometric type-1 and type-2 fuzzy logic systems. IEEE Trans Fuzzy Syst 15(1):3–15

Gonzalez C, Castro JR, Melin P, Castillo O (2016) An edge detection method based on generalized type-2 fuzzy logic. Soft Comput 20(2):773–784

Gonzalez CI, Melin P, Castro JR, Mendoza O, Castillo O (2017) General type-2 fuzzy edge detection in the preprocessing of a face recognition system. Springer, Berlin

Hagras H, Wagner C (2012) Towards the wide spread use of type-2 fuzzy logic systems in real world applications. IEEE Comput Intell Mag 7(3):14–24

Hidalgo D, Melin P, Castillo O (2012) An optimization method for designing type-2 fuzzy inference systems based on the footprint of uncertainty using genetic algorithms. Expert Syst Appl 39(4):4590–4598

Hsu CH, Juang CF (2013) Evolutionary robot wall-following control using type-2 fuzzy controller with species-de-activated continuous ACO. IEEE Trans Fuzzy Syst 21(1):100–112

Juang CF, Chang PH (2010) Designing fuzzy-rule-based systems using continuous ant-colony optimization. IEEE Trans Fuzzy Syst 18(1):138–149

Karnik NN, Mendel JM (2001) Centroid of a type-2 fuzzy set. Inf Sci 132(1–4):195–220

Khosravi A, Nahavandi S (2014) Load forecasting using interval type-2 fuzzy logic systems: optimal type reduction. IEEE Trans Ind Inf 10(2):1055–1063

Linda O, Manic M (2012) Monotone centroid flow algorithm for type reduction of general type-2 fuzzy sets. IEEE Trans Fuzzy Syst 20(5):805–819

Liu FL (2008) An efficient centroid type reduction strategy for general type-2 fuzzy logic system. Inf Sci 178(9):2224–2236

Liu X, Mendel JM, Wu DR (2012) Study on enhanced Karnik–Mendel algorithms: initialization explanations and computation improvements. Inf Sci 184(1):75–91

Mathews JH, Fink KD (2004) Numerical methods using matlab. Prentice-Hall Inc., Upper Saddle River

Melin P, Gonzalez CI, Castro JR, Mendoza O, Castillo O (2014) Edge-detection method for image processing based on generalized type-2 fuzzy logic. IEEE Trans Fuzzy Syst 22(6):1515–1525

Mendel JM (2001) Uncertain rule-based fuzzy logic systems: introduction and new directions. Prentice-Hall, Englewood Cliffs

Mendel JM (2007) Type-2 fuzzy sets and systems: an overview. IEEE Comput Intell Mag 2007 2(2):20–29

Mendel JM (2013) On KM algorithms for solving type-2 fuzzy set problems. IEEE Trans Fuzzy Syst 21(3):426–446

Mendel JM (2014) General type-2 fuzzy logic systems made simple: a tutorial. IEEE Trans Fuzzy Syst 22(5):1162–1182

Mendel JM, John RB (2002) Type-2 fuzzy sets made simple. IEEE Trans Fuzzy Syst 10(2):117–127

Mendel JM, Liu F (2007) Super-exponential convergence of the Karnik–Mendel algorithms for computing the centroid of an interval type-2 fuzzy set. IEEE Trans Fuzzy Syst 15(2):309–320

Mendel JM, Liu X (2013) Simplified interval type-2 fuzzy logic systems. IEEE Trans Fuzzy Syst 21(6):1056–1069

Mendel JM, Wu HW (2006) Type-2 fuzzistics for symmetric interval type-2 fuzzy sets: part 1, forward problems. IEEE Trans Fuzzy Syst 14(6):781–792

Mendel JM, Liu FL, Zhai DY (2009) α-Plane representation for type-2 fuzzy sets: theory and applications: theory and applications. IEEE Trans Fuzzy Syst 17(5):1189–1207

Niewiadomski A (2010) On finity, countability, cardinalities, and cylindric extensions of type-2 fuzzy sets in linguistic summarization of databases. IEEE Trans Fuzzy Syst 18(3):532–545

Olivas F, Valdez F, Castillo O et al (2016) Dynamic parameter adaptation in particle swarm optimization using interval type-2 fuzzy logic. Soft Comput 20(3):1057–1070

Sanchez MA, Castillo O, Castro JR (2015) Generalized type-2 fuzzy systems for controlling a mobile robot and a performance comparison with interval type-2 and type-1 fuzzy systems. Expert Syst Appl 42(14):5904–5914

Wagner C, Hagras H (2010) Toward general type-2 fuzzy logic systems based on zSlices. IEEE Trans Fuzzy Syst 18(4):637–660

Wang T, Chen Y, Tong SC (2008) Fuzzy reasoning models and algorithms on type-2 fuzzy sets. Int J Innov Comput Inf Control 4(10):2451–2460

Wu DR, Mendel JM (2009) Enhanced Karnik–Mendel algorithms. IEEE Trans Fuzzy Syst 17(4):923–934

Zarandi MHF, Rezaee B, Turksen IB, Neshat E (2009) A type-2 fuzzy rule-based expert system model for stock price analysis. Expert Syst Appl 36(1):139–154

Zhai DY, Hao MS, Mendel JM (2012) Universal image noise removal filter based type-2 fuzzy logic system and QPSO. Int J Uncertain Fuzziness Knowl Based Syst 20(supp02):207–232