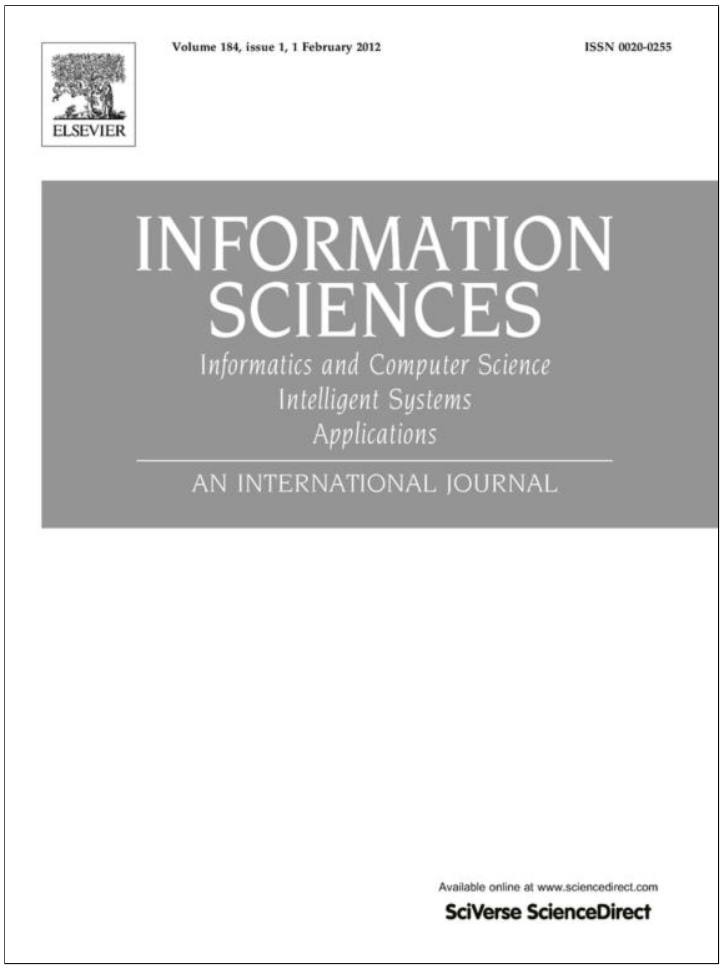


Information Sciences 184 (2012) 75–91

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Study on enhanced Karnik–Mendel algorithms: Initialization explanations and computation improvements ☆

Xinwang Liu [a,b,*], Jerry M. Mendel [b], Dongrui Wu [b,c]

[a] School of Economics and Management, Southeast University, Nanjing, Jiangsu 210096, China
[b] Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2564, USA
[c] Industrial Artificial Intelligence Lab, GE Global Research, Niskayuna, NY 12309, USA

ARTICLE INFO

Article history:
Received 24 August 2010
Received in revised form 13 July 2011
Accepted 24 July 2011
Available online 11 August 2011

Keywords:
Enhanced Karnik–Mendel (EKM) algorithms
Weighted EKM (WEKM) algorithms
Interval type-2 fuzzy set (IT2 FS)
Numerical integration
Centroid computation

ABSTRACT

Computing the centroid of an interval type-2 fuzzy set is an important operation in a type-2 fuzzy logic system, and is usually implemented by Karnik–Mendel (KM) iterative algorithms. By connecting KM algorithms and continuous KM algorithms together, this paper gives theoretical explanations on the initialization methods of KM and Enhanced Karnik–Mendel (EKM) algorithms, proposes exact methods for centroid computation of an interval type-2 fuzzy set, and extends the Enhanced Karnik–Mendel (EKM) algorithms to three different forms of weighted EKM (WEKM) algorithms. It shows that EKM algorithms become a special case of the WEKM algorithms when the weights of the latter are constant value. It also shows that, in general, the weighted EKM algorithms have smaller absolute error and faster convergence speed than the EKM algorithms which make them very attractive for real-time applications of fuzzy logic system. Four numerical examples are used to illustrate and analyze the performance of WEKM algorithms.

1. Introduction

Interval type-2 fuzzy sets (IT2 FSs) are the most widely used type-2 fuzzy sets because they are computationally simple to use. Karnik–Mendel (KM) algorithms originated for computing the centroid of IT2 FSs [5], have been used in many applications of T2 FSs, and play an important role in type-2 fuzzy logic systems [4,10,13,17].

Mendel [11] used KM algorithms to compute the derivatives in interval type-2 fuzzy logic systems. Wang et al. [21] used KM algorithms for optimal training of interval type-2 fuzzy neural networks. Wu and Mendel [23,24] applied them to compute uncertainty measures, such as rank, similarity, variance and skewness of IT2 FSs. Zhai and Mendel [28] use KM algorithms to compute the uncertainty measures of general type-2 fuzzy sets. Liu and Mendel [8] proposed new α-cut algorithms for solving the fuzzy weighted average (FWA) problem using KM algorithms. Compared with other FWA algorithms [1–3,6], they showed that the KM algorithms are the fastest to date. Mendel and Wu [18] proposed the continuous form of KM algorithms. Mendel and Liu [15] also used continuous KM algorithms for theoretical analyses, and proved monotonicity and super-exponential convergence of the algorithms. This work laid a theoretical foundation for the application of KM algorithms. Mendel [12] applied the KM algorithms to type-2 based computing with words. Wu and Mendel [22] used KM algorithms to compute the linguistic weighted average (LWA) of IT2 FSs, and this has been integrated into perceptual computing [17,25,26]. Liu [7] used KM algorithms for computing the centroid of a general type-2 fuzzy set based on the

☆ The work is supported by the National Natural Science Foundation of China (NSFC) under Projects 70771025 and 71171048.
* Corresponding author at: School of Economics and Management, Southeast University, Nanjing, Jiangsu 210096, China.
E-mail addresses: xwliu@seu.edu.cn (X. Liu), mendel@sipi.usc.edu (J.M. Mendel), drwu09@gmail.com (D. Wu).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.07.042

α-plane decomposition of such a fuzzy set, and Mendel et al. [16] used such KM algorithms to design a triangle quasi-type-2 fuzzy logic system. Similar to the α-plane method, Wagner and Hagras [20] used KM algorithms to compute the centroid of the zSlices of general T2 FLSs in robot control problems. Finally, Wu and Mendel [24] proposed Enhanced KM (EKM) algorithms to reduce the computational cost of the standard KM algorithms. Yeh et al. [27] give an extension of EKM to the general type-reduction problem.

In this paper, we propose continuous KM and EKM algorithms for computing the centroid of an interval type-2 fuzzy set. By comparing the sum operation of KM algorithms and the integral operation in the continuous KM algorithms, EKM algorithms are extended to weighted EKM (WEKM) algorithms that use numerical integration techniques. The new WEKM algorithms are more precise and converge faster to the exact centroid values of the IT2 Fs sets than the EKM algorithms, and they include the EKM algorithms as a special case.

The organization of the paper is as follows. Section 2 gives preliminaries about numerical integration in numerical analysis, and KM and EKM algorithms for centroid computation of IT2 FSs. Section 3 provides theoretical explanations on the initialization methods of KM and EKM algorithms. Section 4 proposes a method to compute the exact value of the centroid of an IT2 FS, and three new WEKM algorithms with three different weight assignment methods. Section 5 compares the performances of our four specific algorithms using four numerical examples. Section 6 summarizes the main results and draws conclusions.

2. Preliminaries

2.1. Numerical integration

The goal of numerical integration is to approximate the definite integral of $f(x)$ over the interval $[a,b]$ by evaluating $f(x)$ at a finite number of sample points.[1]

Definition 1 (*Quadrature formula*). Suppose that $a = x_0 < x_1 < \cdots < x_m = b$. A formula of the form

$$Q(f) = \sum_{k=0}^{m} w_k f(x_k) = w_0 f(x_0) + w_1 f(x_1) + w_2 f(x_2) + \cdots + w_m f(x_m) \tag{1}$$

with the property that

$$\int_a^b f(x)dx = Q(f) + E(f) \tag{2}$$

is called a *numerical integration* or *quadrature formula*. The term $E[f]$ is called the *truncation error* for integration. The values $\{x_k\}_{k=0}^m$ are called the *quadrature nodes* and $\{w_k\}_{k=0}^m$ are called the *weights*.

For all applications, it is necessary to know something about the accuracy of the numerical solution. This leads us to:

Definition 2. Degree of precisionThe degree of precision of a quadrature formula is the positive integer n such that $E(P_i) = 0$ for all polynomials $P_i(x)$ of degree $i \leqslant n$, but for which $E(P_{i+1}) \neq 0$ for some polynomial $P_{i+1}(x)$ of degree $n+1$, that is, $\int_a^b P_i(x)dx = Q(P_i)$ when degree $i \leqslant n$, and $\int_a^b P_{i+1}(x)dx \neq Q(P_{i+1})$ when degree $i = n+1$.

When the polynomial $P_m(x)$ of degree m is used to approximate $f(x)$, the integral of $f(x)$ is approximated by the integral of $P_m(x)$, and the resulting formula is called a *Newton–Cotes quadrature formula*. For approximating polynomials of degree $m = 1$, 2, 3, this formula is called *Trapezoidal Rule*, *Simpson's Rule*, and *Simpson* 3/8 *Rule*, respectively. Because of the non-smooth or oscillatory nature of the function $f(x)$ in $[a,b]$, one usually splits $[a,b]$ with quadrature nodes, and applies the composite Newton–Cotes quadrature formula. For these three rules, the quadrature nodes $\{x_k\}_{k=0}^m$ are chosen to be equally spaced.

The following composite Trapezoidal Rule approximates $f(x)$ using straight lines.

Theorem 1 (*Composite Trapezoidal Rule*). *Consider $y = f(x)$ over $[a,b]$. Suppose that the interval $[a,b]$ is subdivided into m subintervals $\{x_{k-1}, x_k\}_{k=1}^m$ of equal width $h = \frac{b-a}{m}$ by using the equally spaced nodes $x_k = x_0 + kh$ for $k = 0,1,2,\ldots,m$. The numerical approximation to the integral of $f(x)$ with the composite Trapezoidal Rule is*

$$\int_a^b f(x)dx = \frac{h}{2}\left(f(a) + f(b) + 2\sum_{k=1}^{m-1} f(x_k)\right) + E_T(f,h). \tag{3}$$

If f is second-order continuous differentiable on $[a,b]$, i.e. $f(x) \in C^2[a,b]$, the error term $E_T(f,h) = -\frac{(b-a)f''(\xi)}{12}h^2 = O(h^2)$, where $a < \xi < b$, and $O(h^2)$ means that when the step size is reduced by a factor of 1/2 the error term $E_T(f,h)$ should be reduced by approximately $\left(\frac{1}{2}\right)^2 = 0.25$.

Simpson's Rule approximates $f(x)$ using quadratic polynomial functions.

[1] All the material in this section has been adapted from [9].

Theorem 2 (*Composite Simpson's Rule*). *Consider $y = f(x)$ over $[a,b]$. Suppose that the interval $[a,b]$ is subdivided into $2m$ subintervals $\{x_{k-1}, x_k\}_{k=1}^{2m}$ of equal width $h = \frac{b-a}{2m}$ by using the equally spaced nodes $x_k = x_0 + kh$ for $k = 0,1,2,\ldots,2m$. The numerical approximation to the integral of $f(x)$ with the composite Simpson's Rule is*

$$\int_a^b f(x)dx = \frac{h}{3}\left(f(a) + f(b) + 2\sum_{k=1}^{m-1} f(x_{2k}) + 4\sum_{k=1}^{m} f(x_{2k-1})\right) + E_S(f,h). \tag{4}$$

If f is fourth-order continuous differentiable on $[a,b]$, i.e. $f(x) \in C^4[a,b]$, the error term $E_S(f,h) = -\frac{(b-a)f^{(4)}(\xi)}{180}h^4 = O(h^4)$, where $a < \xi < b$, and $O(h^4)$ means that when the step size is reduced by a factor of 1/2 the error term $E_T(f,h)$ should be reduced by approximately $\left(\frac{1}{2}\right)^4 = 0.0625$.

Simpson's 3/8 Rule approximates $f(x)$ using cubic polynomial functions.

Theorem 3 (*Composite Simpson's 3/8 Rule*). *Consider $y = f(x)$ over $[a,b]$. Suppose that the interval $[a,b]$ is subdivided into $3m$ subintervals $\{x_{k-1}, x_k\}_{k=1}^{3m}$ of equal width $h = \frac{b-a}{3m}$ by using the equally spaced sample points $x_k = x_0 + kh$ for $k = 0,1,2,\ldots,3m$. The numerical approximation to the integral of $f(x)$ with the composite Simpson's 3/8 Rule is*

$$\int_a^b f(x)dx = \frac{3h}{8}\sum_{k=1}^{m}(f(x_{3k-3}) + 3f(x_{3k-2}) + 3f(x_{3k-1}) + f(x_{3k})) + E_{SC}(f,h), \tag{5}$$

that is

$$\int_a^b f(x)dx = \frac{3h}{8}\left(f(a) + f(b) + \sum_{k=1}^{m-1} 2f(x_{3k}) + \sum_{k=1}^{m} 3f(x_{3k-2}) + \sum_{k=1}^{m} 3f(x_{3k-1})\right) + E_{SC}(f,h). \tag{6}$$

If f is fourth-order continuous differentiable on $[a,b]$, i.e. $f(x) \in C^4[a,b]$, the error term $E_{SC}(f,h) = -\frac{(b-a)f^{(4)}(\xi)}{80}h^4 = O(h^4)$, where $a < \xi < b$.

It should be noted that we have assumed all the integral functions in (4)–(6) are measurable, i.e. the integrals make sense in a in Lebesgue sense.

2.2. Karnik–Mendel algorithms for computing the centroid of an IT2 FS

The Karnik–Mendel (KM) algorithms were developed to compute the centroid of an IT2 FS [5]. Let $x_i (i = 1,2\ldots,N)$ represent the discretiztion of IT2 FS $\widetilde{A}$; $\underline{\mu}_{\widetilde{A}}(x_i)$ and $\bar{\mu}_{\widetilde{A}}(x_i)$ are the lower and upper membership functions that are associated with $\widetilde{A}$ respectively. Using the wavy-slice representation theorem for a type-2 fuzzy set [14], the centroid of $\widetilde{A}$, $c_{\widetilde{A}} = [c_l, c_r]$, can be computed as the optimal solutions of the following interval weighted average problems [5,15]:

$$c_l = \min_{\forall\theta_i \in [\underline{\mu}_{\widetilde{A}}(x_i), \bar{\mu}_{\widetilde{A}}(x_i)]} \frac{\sum_{i=1}^N x_i\theta_i}{\sum_{i=1}^N \theta_i}, \tag{7}$$

$$c_r = \max_{\forall\theta_i \in [\underline{\mu}_{\widetilde{A}}(x_i), \bar{\mu}_{\widetilde{A}}(x_i)]} \frac{\sum_{i=1}^N x_i\theta_i}{\sum_{i=1}^N \theta_i}. \tag{8}$$

It is well known that c_l and c_r can be expressed, as:

Table 1
KM algorithms for computing the centroid end-points of an IT2 FS, $\widetilde{A}$ [5,13,17].

Step[a]	KM algorithm for c_l $c_l = \min_{\forall\theta_i \in \left[\underline{\mu}_{\widetilde{A}}(x_i), \bar{\mu}_{\widetilde{A}}(x_i)\right]} \left(\sum_{i=1}^N x_i\theta_i / \sum_{i=1}^N \theta_i\right)$	KM algorithm for c_r $c_r = \max_{\forall\theta_i \in \left[\underline{\mu}_{\widetilde{A}}(x_i), \bar{\mu}_{\widetilde{A}}(x_i)\right]} \left(\sum_{i=1}^N x_i\theta_i / \sum_{i=1}^N \theta_i\right)$
1	Initialize θ_i by setting $\theta_i = \left[\underline{\mu}_{\widetilde{A}}(x_i) + \bar{\mu}_{\widetilde{A}}(x_i)\right]/2, i = 1,2\ldots,N$ and then compute $c' = c(\theta_1,\theta_2,\ldots,\theta_N) = \sum_{i=1}^N x_i\theta_i / \sum_{i=1}^N \theta_i$.	
2	Find $k' (1 \leqslant k' \leqslant N-1)$ such that $x_{k'} \leqslant c' \leqslant x_{k'+1}$.	
3	Set $\theta_i = \bar{\mu}_{\widetilde{A}}(x_i)$ when $i \leqslant k'$, and $\theta_i = \underline{\mu}_{\widetilde{A}}(x_i)$ when $i \geqslant k'+1$, and then compute $c_l(k') = \frac{\sum_{i=1}^N x_i\theta_i}{\sum_{i=1}^N \theta_i}$.	Set $\theta_i = \underline{\mu}_{\widetilde{A}}(x_i)$ when $i \leqslant k'$, and $\theta_i = \bar{\mu}_{\widetilde{A}}(x_i)$ when $i \geqslant k'+1$, and then compute $c_r(k') = \frac{\sum_{i=1}^N x_i\theta_i}{\sum_{i=1}^N \theta_i}$.
4	Check if $c_l(k') = c'$. If yes, stop and set $c_l(k') = c_l$ and $k' = L$. If no, go to Step 5.	Check if $c_r(k') = c'$. If yes, stop and set $c_r(k') = c_r$ and $k' = R$. If no, go to Step 5.
5	Set $c' = c_l(k')$ and go to Step 2.	Set $c' = c_r(k')$ and go to Step 2.

[a] Note that $x_1 \leqslant x_2 \leqslant \cdots \leqslant x_N$.

Table 2
EKM algorithms for computing the centroid end-points of an IT2 FS, $\widetilde{A}$ [17,24].

Step [a]	EKM algorithm for c_l $c_l = \min_{\forall\theta_i \in \left[\underline{\mu}_{\widetilde{A}}(x_i), \bar{\mu}_{\widetilde{A}}(x_i)\right]} \left(\sum_{i=1}^{N} x_i\theta_i / \sum_{i=1}^{N}\theta_i\right)$	EKM algorithm for c_r $c_r = \max_{\forall\theta_i \in \left[\underline{\mu}_{\widetilde{A}}(x_i), \bar{\mu}_{\widetilde{A}}(x_i)\right]} \left(\sum_{i=1}^{N} x_i\theta_i / \sum_{i=1}^{N}\theta_i\right)$
1	Set $k = [N/2.4]$(the nearest integer to $N/2.4$) and compute: $\alpha = \sum_{i=1}^{k} x_i \bar{\mu}_{\widetilde{A}}(x_i) + \sum_{i=k+1}^{N} x_i \underline{\mu}_{\widetilde{A}}(x_i)$, $\beta = \sum_{i=1}^{k} \bar{\mu}_{\widetilde{A}}(x_i) + \sum_{i=k+1}^{N} \underline{\mu}_{\widetilde{A}}(x_i)$. Compute $c' = \alpha/\beta$.	Set $k = [N/1.7]$(the nearest integer to $N/1.7$) and compute: $\alpha = \sum_{i=1}^{k} x_i \underline{\mu}_{\widetilde{A}}(x_i) + \sum_{i=k+1}^{N} x_i \bar{\mu}_{\widetilde{A}}(x_i)$, $\beta = \sum_{i=1}^{k} \underline{\mu}_{\widetilde{A}}(x_i) + \sum_{i=k+1}^{N} \bar{\mu}_{\widetilde{A}}(x_i)$.
2	Find $k' \in [1, N-1]$ such that $x_{k'} \leqslant c' \leqslant x_{k'+1}$.	
3	Check if $k' = k$. If yes, stop and set $c' = c_l$ and $k = L$. If no, go to Step 4.	Check if $k' = k$. If yes, stop and set $c' = c_r$ and $k = R$. If no, go to Step 4.
4	Compute $s = sign(k' - k)$ and: $\alpha' = \alpha + s\sum_{i=\min(k,k')+1}^{\max(k,k')} x_i[\bar{\mu}_{\widetilde{A}}(x_i) - \underline{\mu}_{\widetilde{A}}(x_i)]$, $\beta' = \beta + s\sum_{i=\min(k,k')+1}^{\max(k,k')} [\bar{\mu}_{\widetilde{A}}(x_i) - \underline{\mu}_{\widetilde{A}}(x_i)]$. Compute $c''(k') = \alpha'/\beta'$.	Compute $s = sign(k' - k)$ and: $\alpha' = \alpha - s\sum_{i=\min(k,k')+1}^{\max(k,k')} x_i[\bar{\mu}_{\widetilde{A}}(x_i) - \underline{\mu}_{\widetilde{A}}(x_i)]$. $\beta' = \beta - s\sum_{i=\min(k,k')+1}^{\max(k,k')} [\bar{\mu}_{\widetilde{A}}(x_i) - \underline{\mu}_{\widetilde{A}}(x_i)]$.
5	Set $c' = c''(k'), \alpha = \alpha', \beta = \beta'$ and $k = k'$ and go to Step 2.	

[a] Note that $x_1 \leqslant x_2 \leqslant \cdots \leqslant x_N$.

$$c_l = \frac{\sum_{i=1}^{L} x_i \bar{\mu}_{\widetilde{A}}(x_i) + \sum_{i=L+1}^{N} x_i \underline{\mu}_{\widetilde{A}}(x_i)}{\sum_{i=1}^{L} \bar{\mu}_{\widetilde{A}}(x_i) + \sum_{i=L+1}^{N} \underline{\mu}_{\widetilde{A}}(x_i)}, \tag{9}$$

$$c_r = \frac{\sum_{i=1}^{R} x_i \underline{\mu}_{\widetilde{A}}(x_i) + \sum_{i=R+1}^{N} x_i \bar{\mu}_{\widetilde{A}}(x_i)}{\sum_{i=1}^{R} \bar{\mu}_{\widetilde{A}}(x_i) + \sum_{i=L+1}^{N} \underline{\mu}_{\widetilde{A}}(x_i)}, \tag{10}$$

where L and R are called *switch points* with $x_L \leqslant c_l \leqslant x_{L+1}$ and $x_R \leqslant c_r \leqslant x_{R+1}$. The determination of L and R are performed by using KM algorithms that are summarized in Table 1 [5,13,17].

Mendel and Liu [15] proved that the KM algorithms converge monotonically and super-exponentially fast. Recently, Wu and Mendel [24] proposed Enhanced KM (EKM) algorithms, which are summarized in Table 2. The EKM algorithms improve the KM algorithms in the following three ways [24]:

1. A better initialization method $k = [N/2.4]$ for c_l and $k = [N/1.7]$ for c_r is used to reduce the number of iterations;
2. A subtle computing techniques is used to reduce the computational cost of each of the algorithm's iterations by using intermediate values a, b, in which only the differences of the sum operator are computed in every new iteration; and,
3. The termination condition of the iterations is changed from $c(k) = c'$ in Step 4 to $k' = k$ in Step 3, which saves the computation of the last iteration.

3. On the Initializations of KM and EKM Algorithms

3.1. Continuous KM and EKM Algorithms

Continuous KM algorithms [18,19] were proposed for studying the theoretical properties of IT2 FS centroid computations. They can also be found in [15].

Observing the relationships between (7) and (9), and (8) and (10), c_l and c_r can be expressed as:

$$c_l = \min_{k=0,1,2,\ldots,N} c_l(k) = \min_{k=0,1,2,\ldots,N} \frac{\sum_{i=1}^{k} x_i \bar{\mu}_{\widetilde{A}}(x_i) + \sum_{i=k+1}^{N} x_i \underline{\mu}_{\widetilde{A}}(x_i)}{\sum_{i=1}^{l} \bar{\mu}_{\widetilde{A}}(x_i) + \sum_{i=k+1}^{N} \underline{\mu}_{\widetilde{A}}(x_i)}, \tag{11}$$

$$c_r = \max_{k=0,1,2,\ldots,N} c_r(k) = \max_{k=0,1,2,\ldots,N} \frac{\sum_{i=1}^{k} x_i \underline{\mu}_{\widetilde{A}}(x_i) + \sum_{i=k+1}^{N} x_i \bar{\mu}_{\widetilde{A}}(x_i)}{\sum_{i=1}^{l} \underline{\mu}_{\widetilde{A}}(x_i) + \sum_{i=k+1}^{N} \bar{\mu}_{\widetilde{A}}(x_i)}. \tag{12}$$

We assume all the x_is are different, and $a = x_1 < x_2 < \cdots < x_N = b$, where a, b are the left-hand (smallest) and the right-hand (largest) sampled values, respectively.[2] Then, the continuous versions of (11) and (12) are

[2] As noted in [19, p. 363], if Gaussian MFs are used, the theoretical results can be extended to $a \to -\infty$, $b \to +\infty$; but, in practice and in algorithm design, when truncations are used, a and b are again finite numbers.

$$c_l = \min_{\xi\in[a,b]} f_l(\xi) = \min_{\xi\in[a,b]} \frac{\int_a^\xi x\bar{\mu}_{\tilde{A}}(x)dx + \int_\xi^b x\underline{\mu}_{\tilde{A}}(x)dx}{\int_a^\xi \bar{\mu}_{\tilde{A}}(x)dx + \int_\xi^b \underline{\mu}_{\tilde{A}}(x)dx}, \tag{13}$$

$$c_r = \max_{\xi\in[a,b]} f_r(\xi) = \max_{\xi\in[a,b]} \frac{\int_a^\xi x\underline{\mu}_{\tilde{A}}(x)dx + \int_\xi^b x\bar{\mu}_{\tilde{A}}(x)dx}{\int_a^\xi \underline{\mu}_{\tilde{A}}(x)dx + \int_\xi^b \bar{\mu}_{\tilde{A}}(x)dx}. \tag{14}$$

From Table 1, continuous versions of the KM algorithms for c_l and c_r, which give the optimal solution of (13) and (14), can be expressed as in Table 3. Similar expressions can also be found in [15,18,19] with infinite integral domain $(-\infty, +\infty)$ instead of $[a,b]$.

Using the notations of f_l in (13) and f_r in (14), from Steps 2 and 4 of Table 3, one can observe that

$$\xi_l = f_l(\xi') \quad \text{and } \xi_l = \xi', \tag{15}$$

$$\xi_r = f_r(\xi') \quad \text{and } \xi_r = \xi'. \tag{16}$$

These are *fixed point iteration formulas*, i.e. when iterations terminate at Step 3, $c_l = \xi_l$ and $c_r = \xi_r$, so that

$$c_l = f_l(c_l), \quad c_r = f_r(c_r). \tag{17}$$

Note that c_l and c_r are the fixed points of $f_l(\xi)$ and $f_r(\xi)$, respectively.

In the same way, continuous versions of EKM algorithms are given Table 4, for which the relationships of (17) still hold.

3.2. Theoretical interpretations of KM algorithm and EKM algorithm initialization methods

In this subsection, new interpretations are provided for the initialization methods of the KM and EKM algorithms that are given in Tables 1 and 2, whose continuous forms are in Tables 3 and 4, respectively. In addition, a new initialization method

Table 3
Continuous KM (CKM) algorithms for computing the centroid end-points of an IT2 FS, $\tilde{A}$.

Step	CKM algorithm for c_l	CKM algorithm for c_r
	$c_l = \min_{\forall\theta(x)\in[\underline{\mu}_{\tilde{A}}(x),\bar{\mu}_{\tilde{A}}(x)]} \frac{\int_a^b x\theta(x)dx}{\int_a^b \theta(x)dx}$	$c_r = \max_{\forall\theta(x)\in[\underline{\mu}_{\tilde{A}}(x),\bar{\mu}_{\tilde{A}}(x)]} \frac{\int_a^b x\theta(x)dx}{\int_a^b \theta(x)dx}$
1	Let $\theta(x) = (\underline{\mu}_{\tilde{A}}(x) + \bar{\mu}_{\tilde{A}}(x))/2$, and compute the initial value ξ', as $\xi' = \frac{\int_a^b x\theta(x)dx}{\int_a^b \theta(x)dx}$.	
2	Set $\theta(x) = \bar{\mu}_{\tilde{A}}(x)$ when $x \leqslant \xi'$, and $\theta(x) = \underline{\mu}_{\tilde{A}}(x)$ when $x \geqslant \xi'$, and then compute $\xi_l = \frac{\int_a^b x\theta(x)dx}{\int_a^b \theta(x)dx}$.	Set $\theta(x) = \underline{\mu}_{\tilde{A}}(x)$ when $x \leqslant \xi'$, and $\theta(x) = \bar{\mu}_{\tilde{A}}(x)$ when $x \geqslant \xi'$, and then compute $\xi_r = \frac{\int_a^b x\theta(x)dx}{\int_a^b \theta(x)dx}$.
3	Check if $\lvert\xi' - \xi_l\rvert \leqslant \varepsilon$ (ε is a given error bound of the algorithms). If yes, stop and set $c_l = \xi_l$. If no, go to Step 4.	Check if $\lvert\xi' - \xi_r\rvert \leqslant \varepsilon$ (ε is a given error bound of the algorithms). If yes, stop and set $c_r = \xi_r$. If no, go to Step 4.
4	Set $\xi' = \xi_l$ and go to Step 2.	Set $\xi' = \xi_r$ and go to Step 2.

Table 4
Continuous EKM (CEKM) algorithms for computing the centroid end-points of an IT2 FS, $\tilde{A}$.

Step	CEKM algorithm for c_l	CEKM algorithm for c_r
	$c_l = \min_{\forall\theta(x)\in[\underline{\mu}_{\tilde{A}}(x),\bar{\mu}_{\tilde{A}}(x)]} \frac{\int_a^b x\theta(x)dx}{\int_a^b \theta(x)dx}$	$c_r = \max_{\forall\theta(x)\in[\underline{\mu}_{\tilde{A}}(x),\bar{\mu}_{\tilde{A}}(x)]} \frac{\int_a^b x\theta(x)dx}{\int_a^b \theta(x)dx}$
1[a]	Set $c = a + (b-a)/2.4$, and compute $\alpha = \int_a^c x\bar{\mu}_{\tilde{A}}(x)dx + \int_c^b x\underline{\mu}_{\tilde{A}}(x)dx$, $\beta = \int_a^c \bar{\mu}_{\tilde{A}}(x)dx + \int_c^b \underline{\mu}_{\tilde{A}}(x)dx$. Compute $c' = \alpha/\beta$.	Set $c = a + (b-a)/1.7$, and compute $\alpha = \int_a^c x\underline{\mu}_{\tilde{A}}(x)dx + \int_c^b x\bar{\mu}_{\tilde{A}}(x)dx$, $\beta = \int_a^c x\underline{\mu}_{\tilde{A}}(x)dx + \int_c^b x\bar{\mu}_{\tilde{A}}(x)dx$.
2	Check if $\lvert c' - c\rvert \leqslant \varepsilon$ (ε is a given error bound of the algorithms). If yes, stop and set $c' = c_l$. If no, go to Step 4.	Check if $\lvert c' - c\rvert \leqslant \varepsilon$ (ε is a given error bound of the algorithms). If yes, stop and set $c' = c_r$. If no, go to Step 4.
3	Compute $s = sign(c' - c)$ and: $\alpha' = \alpha + s\int_{\min(c,c')}^{\max(c,c')} x\left[\bar{\mu}_{\tilde{A}}(x) - \underline{\mu}_{\tilde{A}}(x)\right]dx$, $\beta' = \beta + s\int_{\min(c,c')}^{\max(c,c')} [\bar{\mu}_{\tilde{A}}(x) - \underline{\mu}_{\tilde{A}}(x)]dx$. Compute $c'' = \alpha'/\beta'$.	Compute $s = sign(c' - c)$ and: $\alpha' = \alpha - s\int_{\min(c,c')}^{\max(c,c')} x[\bar{\mu}_{\tilde{A}}(x) - \underline{\mu}_{\tilde{A}}(x)]dx$, $\beta' = \beta - s\int_{\min(c,c')}^{\max(c,c')} [\bar{\mu}_{\tilde{A}}(x) - \underline{\mu}_{\tilde{A}}(x)]dx$.
4	Set $c = c'$, $c' = c''$, $\alpha = \alpha'$, $\beta = \beta'$ and go to Step 2.	

[a] The initialization step utilizes the shift-invariant property of computing the centroid of an IT2 FS [19], i. e. one can always set $a = 0$, so that the total sample number N corresponds to the integral length $b - a$.

is provided which combines the initialization methods of these algorithms, and includes the Tables 1 and 2 initialization methods as special cases.

For the purpose of algorithm initialization, suppose $\underline{\mu}_{\tilde{A}}(x) = \bar{\mu}_{\tilde{A}}(x) = \theta(x)$ for all $x \in [a,b]$; then, observe that $\theta(x) = (\underline{\mu}_{\tilde{A}}(x) + \bar{\mu}_{\tilde{A}}(x))/2$, which agrees with Step 1 in Table 3. In this case, (13) and (14) become the same, so that

$$c_l = c_r = \frac{\int_a^b x\theta(x)dx}{\int_a^b \theta(x)dx} = \frac{\int_a^b x\left(\frac{\underline{\mu}_{\tilde{A}}(x)+\bar{\mu}_{\tilde{A}}(x)}{2}\right)dx}{\int_a^b \frac{\underline{\mu}_{\tilde{A}}(x)+\bar{\mu}_{\tilde{A}}(x)}{2}dx}. \tag{18}$$

(18) is the continuous KM algorithm's initialization method given in Table 3, denoted here as $\xi^{(1)}$:

$$\xi^{(1)} = \frac{\int_a^b x\theta(x)dx}{\int_a^b \theta(x)dx}, \tag{19}$$

where $\theta(x) = (\underline{\mu}_{\tilde{A}}(x) + \bar{\mu}_{\tilde{A}}(x))/2$.

The discrete form of $\xi^{(1)}$ is found in Steps 1 and 2 in Table 1, and is

$$k^{(1)} = \left\{ k | x_k \leqslant \frac{\sum_{i=1}^N x_i(\underline{\mu}_{\tilde{A}}(x_i) + \bar{\mu}_{\tilde{A}}(x_i))}{\sum_{i=1}^N (\underline{\mu}_{\tilde{A}}(x_i) + \bar{\mu}_{\tilde{A}}(x_i))} \leqslant x_{k+1}, 1 \leqslant k \leqslant N-1 \right\}. \tag{20}$$

This KM algorithm initialization method should provide good results when $\underline{\mu}_{\tilde{A}}(x)$ and $\bar{\mu}_{\tilde{A}}(x)$ are very close to each other, because we have just shown that it becomes the exact optimal solution of (13) or (14) when $\underline{\mu}_{\tilde{A}}(x) = \bar{\mu}_{\tilde{A}}(x)$.

Next, we give interpretations of the initialization methods for EKM algorithms, and show that its initialization methods are based on the difference between $\underline{\mu}_{\tilde{A}}(x)$ and $\bar{\mu}_{\tilde{A}}(x)$. Suppose now that

$$\frac{\int_a^b \bar{\mu}_{\tilde{A}}(x)dx}{\int_a^b \underline{\mu}_{\tilde{A}}(x)dx} \triangleq \rho. \tag{21}$$

Note that $\rho \geqslant 1$, because $\underline{\mu}_{\tilde{A}}(x) \leqslant \bar{\mu}_{\tilde{A}}(x)$ for all $x \in [a,b]$. Suppose, also, for the purpose of initializing the EKM algorithms, it is assumed that $\underline{\mu}_{\tilde{A}}(x)$ and $\bar{\mu}_{\tilde{A}}(x)$ are constants for all $x \in [a,b]$, namely $\underline{\mu}_{\tilde{A}}(x) \equiv m > 0$ and $\bar{\mu}_{\tilde{A}}(x) \equiv \rho m$, so that (21) is satisfied. Then, from (13),

$$f_l(\xi) = \frac{\int_a^\xi x\bar{\mu}_{\tilde{A}}(x)dx + \int_\xi^b x\underline{\mu}_{\tilde{A}}(x)dx}{\int_a^\xi \bar{\mu}_{\tilde{A}}(x)dx + \int_\xi^b \underline{\mu}_{\tilde{A}}(x)dx} = \frac{\int_a^\xi \rho m x dx + \int_\xi^b m x dx}{\int_a^\xi \rho m dx + \int_\xi^b m dx} = \frac{\int_a^\xi \rho x dx + \int_\xi^b x dx}{\int_a^\xi \rho dx + \int_\xi^b dx} = \frac{\rho(\xi^2 - a^2) + (b^2 - \xi^2)}{2(\rho(\xi - a) + (b - \xi))},$$

so that

$$f_l'(\xi) = \frac{(\rho-1)(\rho(\xi-a)^2 - (b-\xi)^2)}{2(\rho(\xi-a) + (b-\xi))^2}. \tag{22}$$

Setting $f_l'(\xi) = 0$, it follows that:

$$\frac{(b-\xi)^2}{(\xi-a)^2} = \rho,$$
$$\frac{b-\xi}{\xi-a} = \sqrt{\rho}. \tag{23}$$

Solving (23) for ξ, one obtains

$$\xi_l = \frac{b + a\sqrt{\rho}}{1+\sqrt{\rho}} = a + \frac{b-a}{1+\sqrt{\rho}}. \tag{24}$$

From (22), it can be verified that for all $\xi \in [a, \xi_l), f_l'(\xi) < 0$, and for all $\xi \in (\xi_l, b), f_l'(\xi) > 0$, so ξ_l is the minimum value of $f_l(\xi)$ for $\xi \in [a,b]$. It can also be verified that $f_l(\xi_l) = \xi_l$, which means that $c_l = f_l(\xi_l) = \xi_l$, confirming the relationship for c_l in (17).

Proceeding in a similar manner in (14), it follows, that

$$f_r(\xi) = \frac{(\xi^2 - a^2) + \rho(b^2 - \xi^2)}{2((\xi - a) + \rho(b - \xi))},$$

so that

$$f_r'(\xi) = -\frac{(\rho-1)((\xi-a)^2 - \rho(b-\xi)^2)}{2((\xi-a) + \rho(b-\xi))^2}. \tag{25}$$

Setting $f_r'(\xi) = 0$, it follows that:

$$\frac{(b-\xi)^2}{(\xi-a)^2} = \frac{1}{\rho},$$
$$\frac{b-\xi}{\xi-a} = \sqrt{1/\rho}. \tag{26}$$

Solving (26) for ξ, one obtains

$$\xi_r = \frac{b + a\sqrt{1/\rho}}{1+\sqrt{1/\rho}} = a + \frac{b-a}{1+\sqrt{1/\rho}}. \tag{27}$$

From (25), it can be verified that for for all $\xi \in [a, \xi_r), f_r'(\xi) > 0$, and for all $\xi \in (\xi_r, b), f_r'(\xi) < 0$, so ξ_r is the maximum value of $f_r(\xi)$ for $\xi \in [a,b]$. It can also be verified that $f_r(\xi_r) = \xi_r$, which means $c_r = f_r(\xi_r) = \xi_r$, confirming the relationship for c_r in (17).

Combining (24) and (27) together, our new initialization method for ξ, denoted $\xi^{(2)}$, is:

$$\xi^{(2)} = \begin{cases} a + \frac{b-a}{1+\sqrt{\rho}} & \text{for } c_l, \\ a + \frac{b-a}{1+\sqrt{1/\rho}} & \text{for } c_r. \end{cases} \tag{28}$$

Because $\rho \geqslant 1$, it follows that $\xi^{(2)} \leqslant a + \frac{1}{2}(b-a)$ for c_l, and $\xi^{(2)} \geqslant a + \frac{1}{2}(b-a)$ for c_r.

Based on the footnote of Table 4, letting $a = 0$ (or by comparing the initialization expressions in Table 2 for the EKM algorithms and in Table 4 for the CEKM algorithms, the discrete form of (28) is:

$$k^{(2)} = \begin{cases} [N/(1+\sqrt{\rho})] & \text{for } c_l, \\ [N/(1+\sqrt{1/\rho})] & \text{for } c_r. \end{cases} \tag{29}$$

where

$$\rho = \frac{\sum_{i=1}^N \bar{\mu}_{\tilde{A}}(x_i)}{\sum_{i=1}^N \underline{\mu}_{\tilde{A}}(x_i)}. \tag{30}$$

When $\rho = 2$,

$$1 + \sqrt{p} = 1 + \sqrt{2} \approx 2.4, \tag{31}$$
$$1 + \sqrt{1/p} = 1 + \sqrt{1/2} = 1 + \sqrt{2}/2 \approx 1.7. \tag{32}$$

so that (28) and (29) become

$$\xi_2^{(2)} = \begin{cases} a + (b-a)/2.4 & \text{for } c_l, \\ a + (b-a)/1.7 & \text{for } c_r. \end{cases} \tag{33}$$

$$k_2^{(2)} = \begin{cases} [N/2.4] & \text{for } c_l, \\ [N/1.7] & \text{for } c_r. \end{cases} \tag{34}$$

These are the initialization methods for the CEKM and EKM algorithms that are given in Tables 4 and 2, respectively.

Wu and Mendel [24] determined EKM algorithm initialization parameters $L_0 = [N/2.4]$ and $R_0 = [N/1.7]$ from empirical simulations. Observe that their results coincides with (29) when $\rho = 2$, which means that their EKM algorithm initialization parameter can be seen as a special case of our initialization method.

From the above analyses, observe that the KM algorithm initialization method $\xi_0^{(1)}$ and $k^{(1)}$ in (19) and (20) should be suitable when $\underline{\mu}_{\tilde{A}}(x)$ and $\bar{\mu}_{\tilde{A}}(x)$ are very close; and, our just proposed new initialization method $\xi_0^{(2)}$ and $k^{(2)}$ in (28) and (29) should be suitable when the differences between $\underline{\mu}_{\tilde{A}}(x)$ and $\underline{\mu}_{\tilde{A}}(x)$ are large. In practice, one can take a combination of both initialization algorithms by using a parameter α, i.e.:

$$\begin{cases} \xi^{(0)} = (1-\alpha)\xi^{(1)} + \alpha\xi^{(2)} & \text{for continuous algorithms}, \\ k^{(0)} = [(1-\alpha)k^{(1)} + \alpha k^{(2)}] & \text{for discrete algorithms}. \end{cases} \tag{35}$$

where $\alpha \in [0,1]$. In general, α weights the difference between $\underline{\mu}_{\tilde{A}}(x)$ and $\bar{\mu}_{\tilde{A}}(x)$. One can choose $\alpha = 1/2$ to simultaneously take equal advantage of these two initialization methods. Of course, with some additional data, it may be possible to optimize α in order to obtain a better initialization.

We have tested the new initialization method (35) with many examples. Although our results show that using (35) can improve the initiation to the optimal solution, (35) is not as simple as the EKM algorithm's initialization method. Furthermore, we have observed that (35) can only improve the total computation cost for problems whose total iteration number is greater than 6. Consequently, in the rest of this paper, we continue to use the EKM algorithm's initialization method.

Based on this decision, one may view the results in this section as providing a mathematical derivation of the EKM algorithm's empirical initialization results that were observed by Wu and Mendel [24].

4. Weighted EKM algorithms

To date, the CKM algorithms in Table 3 (or their somewhat different forms in [15,18,19]) have only been used to obtain a better theoretical understanding of KM (EKM) algorithms. In this section, we obtain a new class of EKM algorithms, called weighted EKM algorithms–WEKM algorithms, whose results can be compared with the EKM algorithms.

WEKM algorithms are numerical implementations of CEKM algorithms. Comparing the KM algorithms in Table 1 and the continuous KM algorithms in Table 3, and the corresponding EKM algorithms in Tables 2 and 4, observe that the sum operations in the KM (EKM) algorithms at the sampling points x_i play the role of integration of the corresponding functions. Using the general quadrature formula (1), one can assign weights w_i to the membership function values at sampling points x_i, and obtain more accurate values of c_l and c_r. In this way, we extended the EKM algorithms to weighted EKM (WEKM) algorithms, as given in Table 5.

The EKM algorithms are a special case of WEKM algorithms when $w_i = 1 (i = 1,2,\ldots,N)$. In Table 5, there are various weight assignment methods according to the quadrature formula (1) [9]. Here, we only give the weight assignment methods for the three numerical integration methods described in Section 2: Trapezoidal Rule, Simpson's Rule and Simpson's 3/8 Rule, whose quadrature formulas are (3), (4) and (6), respectively. The corresponding WEKM algorithms are called TWEKM, SWEKM and S3/8WEKM algorithms, respectively. Table 6 gives the weight assignment methods on the EKM, TWEKM, SWEKM and S3/8WEKM algorithms. For all three WEKM algorithms, the sampling points are distributed evenly in $[a,b]$ with $x_i = a + \frac{i-1}{N-1}(b-a), i = 1,2,\ldots,N$, according to Theorems 1–3.

In Table 6, except for the weight assignment method of the EKM, the three WEKM algorithm's weight assignment methods are obtained from (3)–(6) using the following steps:

1. $x_k (k = 0,1,\ldots,m)$ and $x_0 = a$, $x_m = b$ in (3); $x_k (k = 0,1,\ldots,2m)$ and $x_0 = a$, $x_{2m} = b$ in (4); and $x_k (k = 0,1,\ldots,3m)$ and $x_0 = a$, $x_{3m} = b$ in (6), are each replaced by $x_i (i = 1,2,\ldots,N)$ and $x_1 = a$, $x_N = b$.

Table 5
Weighted EKM (WEKM) algorithms for computing the centroid end-points of IT2 FS, $\tilde{A}$,.

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

[a] Note that $x_1 \leqslant x_2 \leqslant \cdots \leqslant x_N$.

Table 6
Weight assignment methods of WEKM (EKM) algorithms.

Algorithms	Integration Rule	Weight value
EKM	–	$w_i = 1 (i = 1,2,\ldots,N)$
TWEKM	Trapezoidal Rule	$w_i = \begin{cases} 1/2 & \text{if } i = 1, N, \\ 1 & \text{if } i \neq 1, N. \end{cases}$
SWEKM	Simpson's Rule	$w_i = \begin{cases} 1/2 & \text{if } i = 1, N \\ 1 & \text{if } i = 1 \bmod {}^{a}(2) \text{ and } i \neq 1, N, \\ 2 & \text{if } i = 0 \bmod (2) \text{ and } i \neq N. \end{cases}$
S3/8WEKM	Simpson's 3/8 Rule	$w_i = \begin{cases} 1/3 & \text{if } i = 1, N \\ 2/3 & \text{if } i = 1 \bmod(3) \text{ and } i \neq 1, N, \\ 1 & \text{if } i = 2 \bmod(3) \text{ and } i \neq N, \\ 1 & \text{if } i = 0 \bmod(3) \text{ and } i \neq N. \end{cases}$

[a] mod is modular arithmetic operator. $i = j \bmod (d)$ means $i = nd + j$, where n is an integer.

2. The common coefficients of h (i. e., $h/2, h/3, 3h/8$) in (3), (4), and (6) are neglected because they cancel in the quotient of two integrals in Table 4.
3. The TWEKM and SWEKM algorithm weight values in Table 6 also use one-half of the coefficients in the parentheses of (3) and (4), and the S3/8WEKM algorithm weight values use one-third of the coefficients in the parentheses of (6).
4. The number of points N of SWEKM and S3/8WEKM algorithms are not restricted to $N = 2m + 1$ and $N = 3m + 1$ (i. e., $N = 1 \bmod (2)$ and $N = 1 \bmod (3)$), as (4) and (6) require.

According to the relationships between Table 6 and (3)–(6), the TWEKM, SWEKM and S3/8WEKM algorithms approximate the numerical integrations of the membership functions with polynomials of order $n = 1, 2, 3$, respectively, which are special cases of the Newton–Cotes formula, as explained in Section 2.1.

Remark 1. In theory, any degree n Newton–Cotes formula could be used to obtain more general WEKM algorithms; however, for large n, numerical integration can sometimes suffer from catastrophic Runge's phenomenon, where the error grows when n increases, which is contrary to the goal of approximation theory [9].

Regarding which of the WEKM algorithms to use, one should first observe the nonlinear nature of the upper and lower membership functions, and then select approximate numerical integration rules for the WEKM algorithms. Because of the piecewise integrals of the LMF and UMF in Tables 3 and 4, one could also mix more than one Table 6 weight assignment method depending upon the natures of the LMF and UMF.

The relationships between the CEKM algorithms in Table 4 and WEKM algorithms in Table 5 for computing the centroid of IT2 FS are that:

1. WEKM algorithms compute the centroid value based on the addition operations applied to sample data $x_i (i = 1, 2, \ldots, N)$, and find the optimal switch points as the approximation of the centroid value when the iteration terminates. CEKM algorithms compute the centroid value using integral operations and obtain the exact centroid values of an IT2 FS. The solution of WEKM algorithms approaches to that of CEKM algorithms when the sample size $N \to +\infty$.
2. For WEKM algorithms, more accurate results are obtained by increasing the sample size N. For CEKM algorithms, more accurate results are obtained by setting a smaller accuracy level ε to control the differences between two adjacent iteration values.
3. WEKM algorithms are computed numerically with addition operations, whereas CEKM algorithms are computed symbolically with integral operations. WEKM algorithms are numerical implementations of CEKM algorithms using numerical integration methods.

5. Experimental evaluation of the proposed methods

5.1. Introduction

In this section, centroid computations are considered for the four IT2 FSs $\widetilde{A}_1 \sim \widetilde{A}_4$ in Table 7, that correspond to the FOUs used in the examples in [15,16].

For $\widetilde{A}_1 \sim \widetilde{A}_4$, if the FOUs of these IT2 FSs are approximated with different order polynomials, and the order of polynomials is used to describe the linear-nonlinear properties of these FOUs, it can be observed intuitively that the nonlinear property becomes greater from $\widetilde{A}_1$ to $\widetilde{A}_4$, which means the higher order polynomials are required to approximate the FOUs in the sequence of $\widetilde{A}_1$ to $\widetilde{A}_4$. This is used to explain the performances of the three kinds of WEKM algorithms and EKM algorithms below.

As in [15], the algorithms are only illustrated for computing c_l, because the results for c_r are very similar to those for c_l.

5.2. Accuracy performance comparison and analysis

From footnote 2 on page 5, the centroid computation of IT2 FS can always be limited to a finite domain $[a, b]$; the continuous EKM (CEKM) algorithms in Table 4 can be used to compute the exact centroid value of an IT2 FS. The exact centroid computations for $\widetilde{A}_1 \sim \widetilde{A}_4$ using CEKM algorithms are given in Table 8. The same value in the last two iterations corresponds to the termination conditions of the algorithms in Table 4.

The exact left centroid values of these IT2 FSs are:

$$c_l^*(\widetilde{A}_1) = 3.660534, \quad c_l^*(\widetilde{A}_2) = 0.445924, \quad c_l^*(\widetilde{A}_3) = 3.155741, \quad c_l^*(\widetilde{A}_4) = 3.595542.$$

The performances of WEKM algorithms are studied next.

When the number of samples, N is fixed, $[x_1, x_N] \equiv [a, b]$ where $x_i = x_1 + \frac{i-1}{N-1}(b - a), i = 1, 2, \ldots, N$. N ranges from 100 to 2000 with step length equal to 100.

Computation results are given in Figs. 1–4. Part (a) of each figure gives c_l versus N for EKM algorithms and the three WEKM algorithms (TWEKM, SWEKM and S3/8WEKM), so that one can observe the convergence process to c_l^* as N increases. Part (b) of each figure gives absolute error $|c_l - c_l^*|$ for each algorithm as a function of N, so that the performance of each

Table 7
IT2 FS examples.

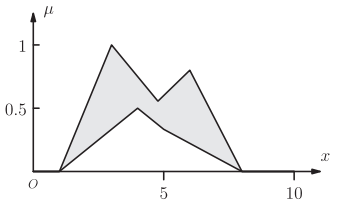
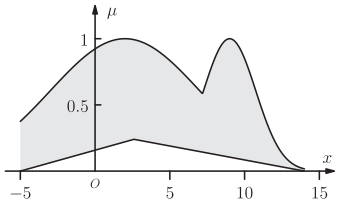
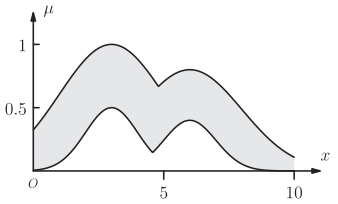
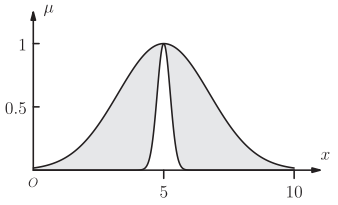
MF Type	MF Expression	MF Plot
(1) Piecewise linear MFs [16]	$\underline{\mu}_{\widetilde{A}_1}(x) = \max\left\{\begin{bmatrix}(x-1)/2 & \text{if } 1\leqslant x\leqslant 3,\\ (7-x)/4 & \text{if } 4\leqslant x\leqslant 7,\\ 0 & \text{otherwise.}\end{bmatrix}, \begin{bmatrix}(x-2)/5 & \text{if } 2\leqslant x\leqslant 6,\\ (16-2x)/5 & \text{if } 6\leqslant x\leqslant 8,\\ 0 & \text{otherwise.}\end{bmatrix}\right\}$ $\bar{\mu}_{\widetilde{A}_1}(x) = \max\left\{\begin{bmatrix}(x-1)/6 & \text{if } 1\leqslant x\leqslant 3,\\ (7-x)/6 & \text{if } 3\leqslant x\leqslant 7,\\ 0 & \text{otherwise.}\end{bmatrix}, \begin{bmatrix}(x-3)/6 & \text{if } 2\leqslant x\leqslant 5,\\ (8-x)/9 & \text{if } 5\leqslant x\leqslant 8,\\ 0 & \text{otherwise.}\end{bmatrix}\right\}$	
(2) Triangular LMF and Gaussian UMF [15]	$\underline{\mu}_{\widetilde{A}_2}(x) = \begin{cases}\frac{0.6(x+5)}{19} & \text{if } x\in[-5,2.6],\\ \frac{0.4(14-x)}{19} & \text{if } x\in[2.6,14].\end{cases}$ $\bar{\mu}_{\widetilde{A}_2}(x) = \begin{cases}\exp\left[-\frac{1}{2}\left(\frac{x-2}{5}\right)^2\right] & \text{if } x\in[-5,7.185],\\ \exp\left[-\frac{1}{2}\left(\frac{x-9}{1.75}\right)^2\right] & \text{if } x\in[7.185,14].\end{cases}$	
(3) Piecewise Gaussian MFs [16]	$\underline{\mu}_{\widetilde{A}_3}(x) = \max\left\{\exp\left[-\frac{(x-3)^2}{8}\right], 0.8\exp\left[-\frac{(x-6)^2}{8}\right]\right\}$ $\bar{\mu}_{\widetilde{A}_3}(x) = \max\left\{0.5\exp\left[-\frac{(x-3)^2}{2}\right], 0.4\exp\left[-\frac{(x-6)^2}{2}\right]\right\}$ $x\in[0,10].$	
(4) Gaussian MFs with uncertain deviation [15]	$\underline{\mu}_{\widetilde{A}_4}(x) = \exp\left[-\frac{1}{2}\left(\frac{x-5}{0.25}\right)^2\right]$ $\bar{\mu}_{\widetilde{A}_4}(x) = \exp\left[-\frac{1}{2}\left(\frac{x-5}{1.75}\right)^2\right]$ $x\in[0,10].$	

Table 8
CEKM algorithms iterations for c_l with $\varepsilon = 10^{-6}$.

t [a]	0	1	2	3	4	5
$\widetilde{A}_1$	3.916667	3.664327	3.660535	3.660534	3.660534	
$\widetilde{A}_2$	2.916667	0.761481	0.452688	0.445927	0.445924	0.445924
$\widetilde{A}_3$	4.166667	3.222779	3.156043	3.155741	3.155741	
$\widetilde{A}_4$	4.166667	3.663813	3.596623	3.595542	3.595542	

[a] t is the iteration number. $t = 0$ corresponds to the initialization step.

algorithm can be observed more clearly. Part (c) of each figure gives c_l for the three WEKM algorithms in the same way as part (a), but with a magnified scale. Part (d) of each figure gives absolute error $|c_l - c_l^*|$ for the three WEKM algorithms as in part (b) also with a magnified scale. Parts (a) and (b) are mainly used to compare the performances of EKM algorithms and the three WEKM algorithms, whereas parts (c) and (d) are used to compare the performances of the three WEKM algorithms.

To measure the performance of each algorithm for $\widetilde{A}_i (i = 1,\ldots,4)$, for each sample size N the relative error $|c_l - c_l^*|/|c_l^*|$ was computed. The average of the relative errors $|c_l - c_l^*|/|c_l^*|$ over the values of N for each $\widetilde{A}_i (i = 1,\ldots,4)$ is given in Table 9. The last line of Table 9 gives the overall average of the relative errors, averaged over the four $\widetilde{A}_i$ for each algorithm.

From Figs. 1–4 and Table 9, observe that:

1. From parts (a) and (b) of Figs. 1–4, except for $\widetilde{A}_1$, *all WEKM algorithms lead to smaller absolute errors than the EKM algorithms, and they also converge to the accurate values faster than do the EKM algorithms*. The reason for both of these results is that the EKM algorithms only use a simple sum. Both EKM and WEKM algorithms for $\widetilde{A}_1$ are very good due to the linear property of the FOU of $\widetilde{A}_1$(this is discussed further in Item 2(d)). From Table 9, the biggest average relative error for EKM algorithms is **1.4141%**, while the biggest average relative error for WEKM algorithms is **0.0038%**. In a similar way, the

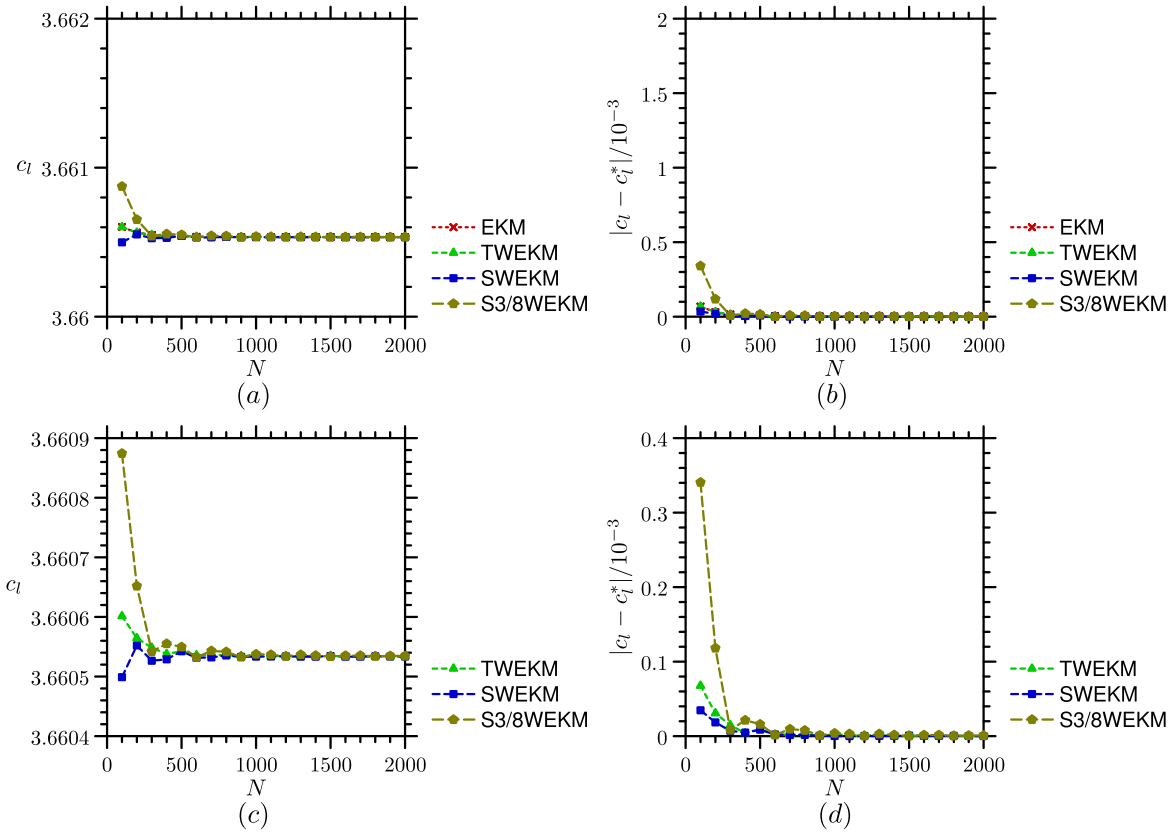


Fig. 1. Computation results for $\widetilde{A}_1$ (EKM and TWEKM lie on top of each other in (a) and (b)).

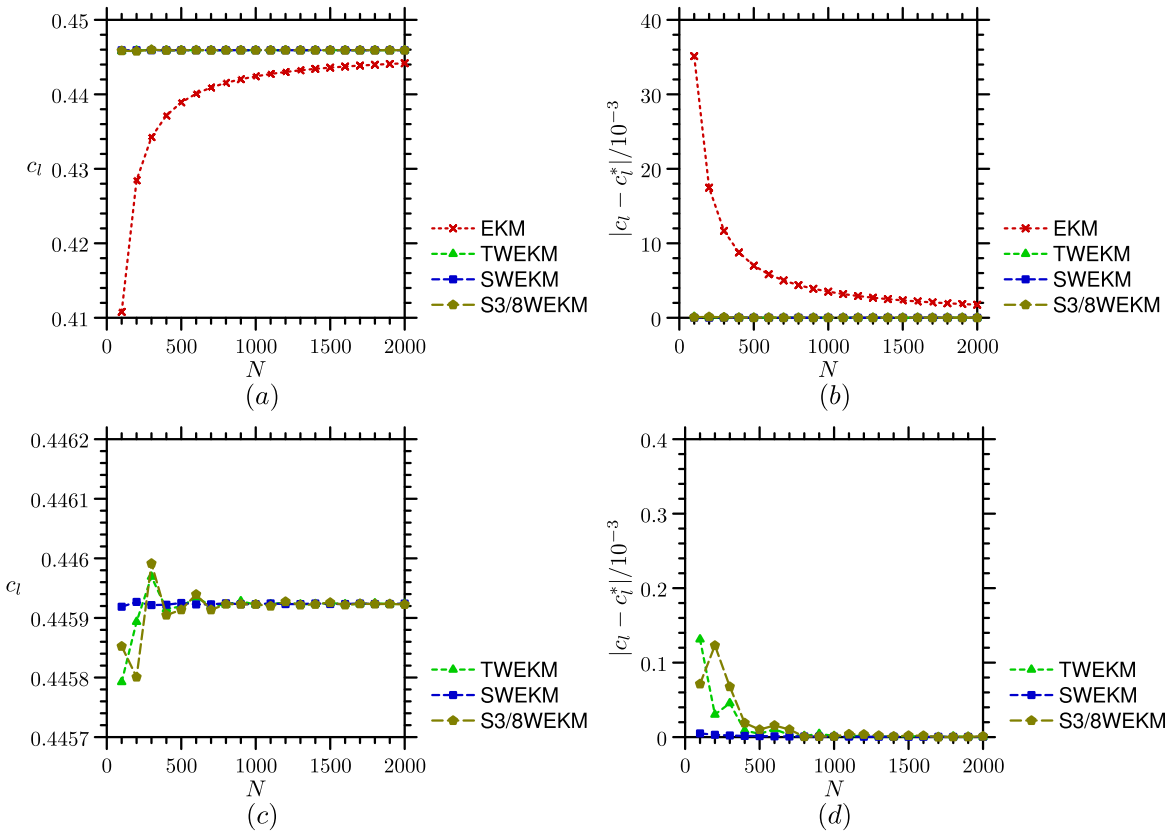


Fig. 2. Computation results for $\widetilde{A}_2$ (TWEKM, SWEKM and S3/8WEKM lie on top of each other in (a) and (b)).

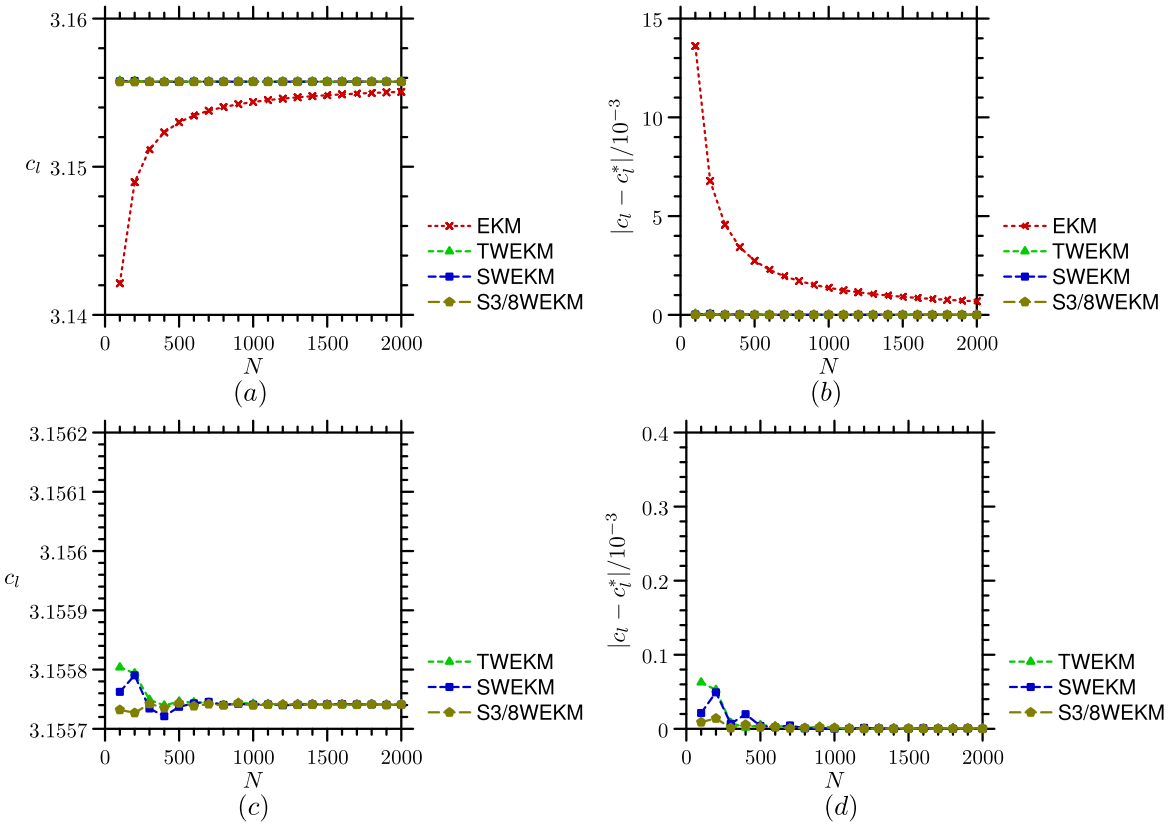


Fig. 3. Computation results for $\widetilde{A}_3$ (TWEKM, SWEKM and S3/8WEKM lie on top of each other in (a) and (b)).

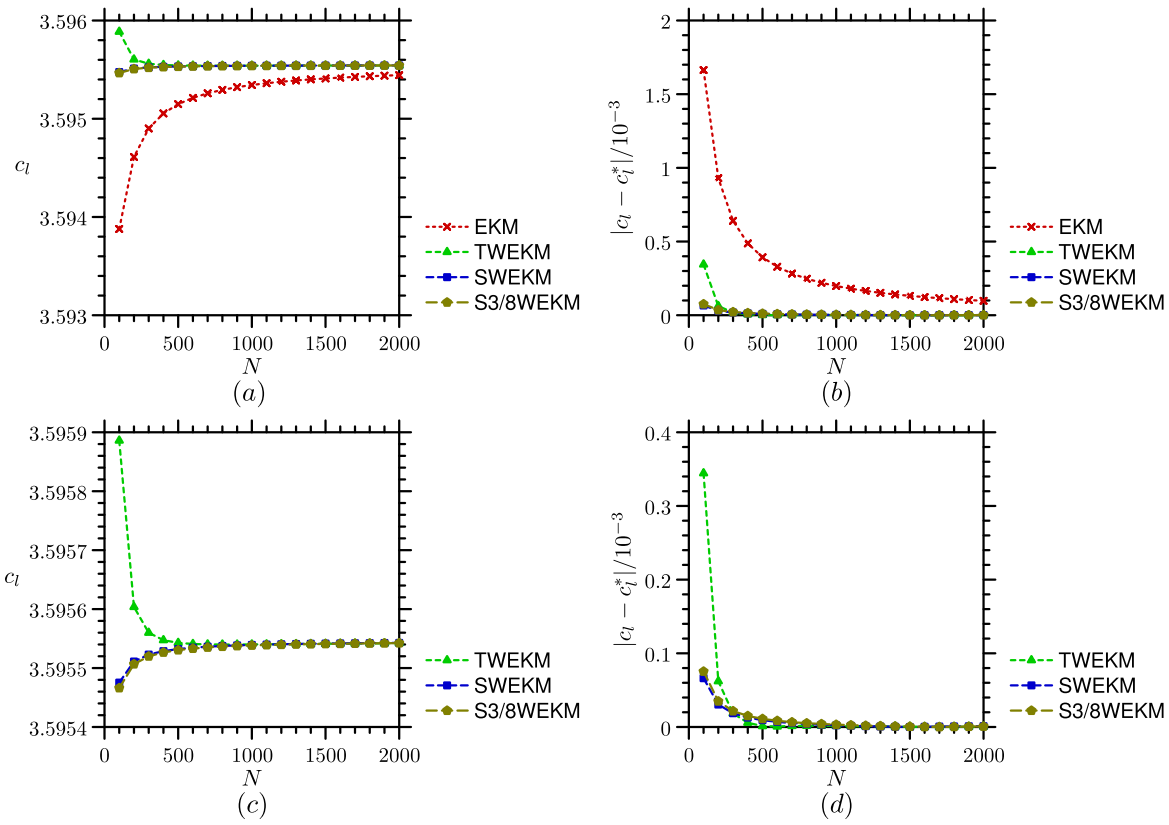


Fig. 4. Computation results for $\widetilde{A}_4$ (SWEKM and S3/8WEKM lie on top of each other in (a)–(d)).

Table 9
Average of the relative errors $|c_l - c_l^*|/|c_l^*|$, for $N = 100$ to 2000.

Algorithms	EKM	TWEKM	SWEKM	S3/8WEKM
$\widetilde{A}_1$	0.000002	0.000002	0.000001	0.000007
$\widetilde{A}_2$	**0.014141**[a]	0.000028	0.000002	**0.000038**
$\widetilde{A}_3$	0.000779	0.000002	0.000002	0.000001
$\widetilde{A}_4$	0.000093	0.000006	0.000002	0.000003
Overall average	**0.003754**	0.000010	0.000002	**0.000012**

[a] Bold-faced numbers are explained in the text.

overall average relative error of EKM algorithms for all IT2 FSs is **0.3754%**, whereas the overall average relative error of WEKM algorithms for all IT2 FSs is less than **0.0012%**. This implies that the computational accuracy of WEKM algorithms is greatly improved over the EKM algorithms. Comparing the vertical axis scales of Figs. 1–4, from parts (a) and (b), one can observe that the error bounds of EKM algorithms are different for different IT2 FSs, but from parts (c) and (d), the error bounds of WEKM algorithms for different IT2 FSs are on the same scale. *This means that WEKM algorithms also have better error stability than EKM algorithms.*

2. From parts (c) and (d) of Figs. 1–4, the statistics of Table 9, and recalling that TWEKM, SWEKM and S3/8WEKM algorithms approximate membership functions with polynomial orders 1(linear), 2 (quadratic) and 3 (cubic), respectively:
   (a) For $\widetilde{A}_1$ in Fig. 1, the results of linear polynomial approximation TWEKM algorithms and quadratic polynomial approximation SWEKM algorithms are very close, and both are better than that of S3/8WEKM algorithms. The cubic polynomial approximation of S3/8WEKM algorithms seems unsuitable because of $\widetilde{A}_1$'s piecewise linear lower and upper membership functions. The linearity of the membership functions can also be used to explain why the non-weighted EKM algorithms for $\widetilde{A}_1$ has almost the same good results as the TWEKM algorithms in Fig. 1 (a)and (b).
   (b) For $\widetilde{A}_2$ in Fig. 2, all three WEKM algorithms have very good performances (Fig. 2 (a), (b)), and SWEKM algorithms performs the best among them (Fig. 2 (c), (d)). Because of its mixed linear LMF and nonlinear UMF, both the linear approximation of TWEKM algorithms and cubic polynomial approximation of S3/8WEKM algorithm are not the best choices.
   (c) For $\widetilde{A}_3$ in Fig. 3, all three WEKM algorithms have very good performances (Fig. 3(a) and (b)), but the S3/8WEKM algorithms performs the best among them (Fig. 3(c) and (d)), which is different from the case of $\widetilde{A}_2$. The nonlinear nature of $\widetilde{A}_3$ is greater than $\widetilde{A}_2$, and that of $\widetilde{A}_2$ is greater than $\widetilde{A}_1$ (see Table 7). This can be used to explain why the SWEKM algorithms is the best choice for $\widetilde{A}_2$, and the S3/8WEKM algorithm is the best choice for $\widetilde{A}_3$. On the other hand, for $\widetilde{A}_1$, the S3/8WEKM algorithm is the worst choice because of the Runge's phenomenon described in Remark 1.
   (d) For $\widetilde{A}_4$ in Fig. 4, the three WEKM algorithms still greatly outperform the EKM algorithms. The results of SWEKM and S3/8WEKM algorithms are very good and similar, but the linear polynomial approximation of TWEKM algorithms become the worst of all the WEKM algorithms. This is contrary to the case of $\widetilde{A}_1$, where the cubic polynomial approximation of S3/8WEKM algorithms performs the worst. The reason for this is the cubic polynomial approximation of S3/8WEKM algorithm over-fits for the linear FOU of $\widetilde{A}_1$, and the linear approximation of TWEKM algorithms under-fit the nonlinear FOU of $\widetilde{A}_4$.

In summary, from $\widetilde{A}_1$ to $\widetilde{A}_4$, *as the nonlinear nature of the FOU increases, the WEKM algorithms also perform better and better consistent with the nature of the higher order of polynomials in the sequence of EKM, TWEKM, SWEKM, S3/8WEKM.*

In general, there are two ways to view these centroid computations:

1. Approximation theory, where one wants to obtain high-precision results. To accomplish this, one may have to adjust the sample rate ($1/N$), and choose the best algorithm.
2. Fuzzy logic system (FLS) applications, where an FOU is created prior to type reduction. Now, one must fix the sampling rate ($1/N$) in advance because of design constraints, and has to then choose the best algorithm.

In the former case, our examples indicate that WEKM algorithms at $N = 300$ have smaller errors than the results of EKM algorithms even at $N = 10000$. This means that WEKM algorithms may obtain nearly exact centroid values with much less

Table 10
Average of the relative errors $|c_l - c_l^*|/|c_l^*|$ for $N = 100$ to 500.

Algorithms	EKM	TWEKM	SWEKM	S3/8WEKM
$\widetilde{A}_1$	0.000007	0.000007	0.000004	0.000028
$\widetilde{A}_2$	**0.035899**[a]	0.000099	0.000006	**0.000131**
$\widetilde{A}_3$	0.001973	0.000008	0.000006	0.000002
$\widetilde{A}_4$	0.000229	0.000024	0.000008	0.000009
Overall average	**0.009527**	0.000034	0.000006	**0.000042**

[a] Bold-faced numbers are explained in the text.

data sampling than required by the EKM algorithms, and with a precision the latter can not attain. In the latter case, the WEKM algorithms appear to be vastly superior to the EKM algorithms, especially when N is small, e.g. $N \in [100,500]$, which can be observed clearly from Figs. 2–4(a) and (b), whose statistics are summarized in Table 10.

From Table 10, the biggest average relative error for EKM algorithms is **3.5899%**, whereas the biggest average relative error for WEKM algorithms is **0.0131%**. In a similar way, the overall average relative error of EKM algorithms for all IT2 FSs is **0.9527%**, whereas the overall average relative error of WEKM algorithms for all IT2 FSs is less than **0.0042%**.

Note, finally, that if one has a FLS for which accuracy may not be important, then EKM algorithms will give acceptable results.

5.3. Computation time comparisons and analysis

Next, we compare the computation times for these algorithms, something that is useful in applications, especially for those that have real-time requirements. Unlike the computation of c_l in Figs. 1–4, where one always gets the same results that are independent of the hardware and software environments, the computing times of the algorithms are not exactly repeatable. Simulations were conducted to examine the performance of our algorithms. The platform used is Microsoft

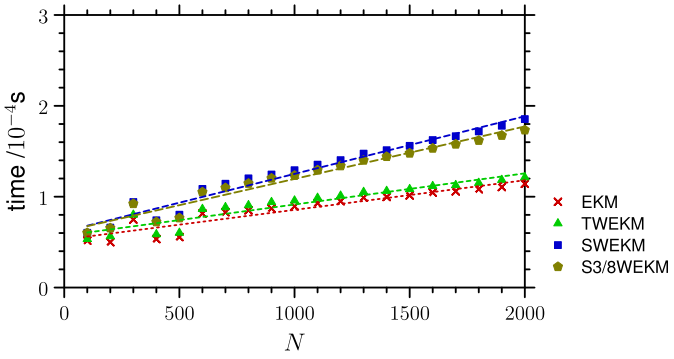


Fig. 5. Computation time comparisons for $\widetilde{A}_1$.

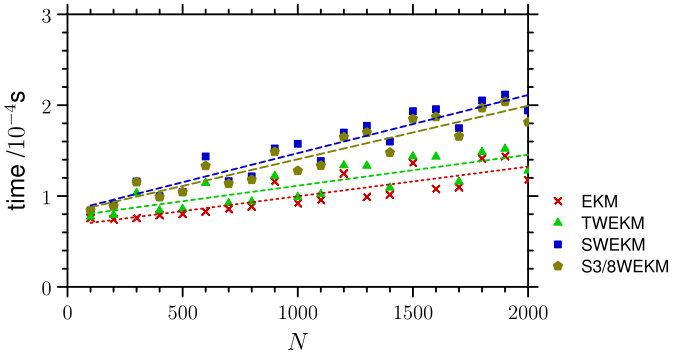


Fig. 6. Computation time comparisons for $\widetilde{A}_2$.

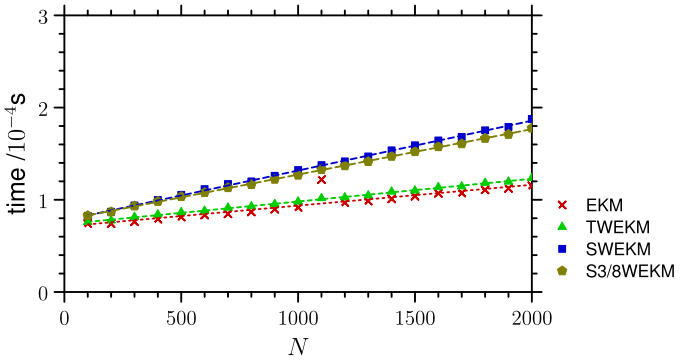


Fig. 7. Computation time comparisons for $\widetilde{A}_3$.

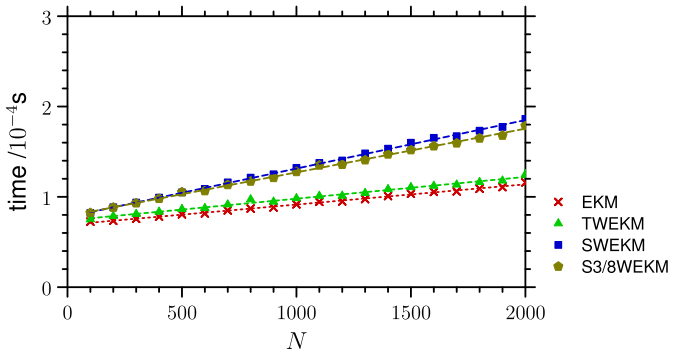


Fig. 8. Computation time comparisons for $\widetilde{A}_4$.

Table 11
Least squares computation time regression model coefficients for the four algorithms.

Regression	EKM		TWEKM		SWEKM		S3/8WEKM	
Coefficients	$c/10^{-5}$	$\alpha/10^{-8}$	$c/10^{-5}$	$\alpha/10^{-8}$	$c/10^{-5}$	$\alpha/10^{-8}$	$c/10^{-5}$	$\alpha/10^{-8}$
$\widetilde{A}_1$	5.291	3.261	5.713	3.428	6.172	6.339	6.181	5.775
$\widetilde{A}_2$	6.714	3.261	7.727	3.409	8.317	6.403	8.172	5.882
$\widetilde{A}_3$	7.106	2.261	7.353	2.463	7.766	5.406	7.804	4.928
$\widetilde{A}_4$	6.909	2.239	7.389	2.409	7.765	5.368	7.870	4.839
Average	6.505	2.756	7.046	2.927	7.505	5.879	7.057	5.356

Table 12
Recommended algorithms for different application purposes.

Case	Approximation theory	FLS
Linear MFs	(TW) EKM	(TW) EKM
Nonlinear MFs	S (3/8) WEKM	S (3/8) WEKM
Mixture MFs	SWEKM	TWEKM

(TW) EKM means EKM or TWEKM; S (3/8) WEKM means SWEKM or S3/8WEKM.

Windows XP Professional, Version2002, Service Pack 3, Intel (R) Core (2) DUO CPU, E8400@3.00 GHz, 2.99 GHz, 3.25 Gb RAM. The algorithms were programmed with Matlab 2009b. Figs. 5–8 show the total computation times for $N = 100, 200, \ldots, 2000$, which were obtained using Matlab time functions *tic* and *toc* in seconds.

Observe that, if the fluctuations of computation times for different N are not considered, the computation times of these algorithms approximately change with the sampling size N in a linear way. The least squares regression models $t = c + \alpha N$ for these algorithms can be obtained, with the coefficients listed in Table 11.

From Figs. 5–8 and Table 11, one observes that EKM and TWEKM algorithms are very similar, and those of SWEKM and S3/8WEKM algorithms are also very similar because of their similar weight assignment methods in Table 6. The computation time of EKM algorithms is slightly better than TWEKM algorithms because of the weight assignment values at the two end points. The computation time of S3/8EKM algorithms is also slightly better than SWEKM algorithms because the former has a relatively simpler weight assignment method than the latter. If these four kinds of algorithms are classified into two families, the computation times of the SWEKM and S3/8WEKM algorithms are less than double those of the EKM and TWEKM algorithms. The difference ratios of these four algorithms [3] for $\widetilde{A}_1, \widetilde{A}_2, \widetilde{A}_3, \widetilde{A}_4$ range between 18% ~ 27% at $N = 100$ and 70% ~ 72% at $N = 2000$.

5.4. Algorithm recommendations for applications

From the computational accuracy statistics in Tables 9 and 10, one observes that, the SWEKM algorithms are the best choice if only the accuracy requirement is considered. Considering computational accuracy statistics in Tables 9 and 10 and computational times in Figs. 5–8 together, the recommended centroid IT2 FS computation algorithms for approximation and FLS application purposes are given in Table 12. The former requires high-precision and the sampling rate $(1/N)$ is

[3] The difference ratios are computed as $(\max_{i=1,\ldots,4}\{t_i\} - \min_{i=1,\ldots,4}\{t_i\})/\min_{i=1,\ldots,4}\{t_i\}$, where $t_i (i = 1, \ldots, 4)$ are the computation time of these four algorithms.

changeable, and the latter may require real-time computation and the sampling rate (1/N) is usually fixed (during the design of the FLS).

The linear membership function case of Table 12 is $\widetilde{A}_1$, whose computation results are shown in Figs. 1 and 5 for such an FOU. The non-weighted EKM algorithms or linear weighted TWEKM algorithms are recommended; both have similar computation results and computation time. The nonlinear membership function cases of Table 12 are $\widetilde{A}_3$ and $\widetilde{A}_4$, whose computation results are shown in Figs. 3, 4 and 7, 8. The quadratic weighted SWEKM algorithms or cubic weighted S3/8WEKM algorithms are recommended, also because of their similar computation results and computation time. The mixed linear and nonlinear membership function case of Table 12 is $\widetilde{A}_2$, whose computing results are shown in Figs. 2 and 6. SWEKM algorithms are recommended for best approximation results, but TWEKM algorithms are recommended for FLS applications where the approximation accuracy and computation time must be considered simultaneously.

From Figs. 5–8, the centroid computation of IT2 FSs is very fast, i.e. the times are within 3×10^{-4} seconds. This strongly suggests that our algorithms can be used in fuzzy logic control systems, where centroid type reduction is usually avoided because of its iterative nature.

Finally, it should be noted that, in comparisons between EKM and WEKM algorithms, we have only concentrated on the theoretical performances of the algorithms. From the numerical examples, for very high accuracy, we can see that the family of WEKM algorithms can greatly improve the accuracy over EKM algorithms using the same number of sample points; or, the samples points can be greatly reduced for the WEKM algorithms over the EKM algorithms to obtain the same accuracy. However, if the accuracy requirement is not very high, such as two significant figures, then the simple EKM algorithms provide good enough results, i.e. WEKM algorithms and EKM algorithms return the same results. WEKM maybe not always needed and their advantages can not be demonstrated.

6. Conclusions

After comparing discrete and continuous KM (EKM) algorithms, a theoretical explanation for the initialization methods of both KM and EKM algorithms has been provided. Continuous KM and EKM algorithms have been used to compute the exact centroid values of an IT2 FS. EKM algorithms have been extended to weighted EKM (WEKM) algorithms using quadrature formulas from numerical integration. Three weight assignment methods for WEKM algorithms have been proposed. Four numerical examples have shown that, although the EKM and WEKM algorithms return the same values at low accuracy requirements, if very high accuracy is needed, the WEKM algorithms have smaller absolute error and converge faster to the exact centroid values of an IT2 FS than do EKM algorithms, for the same sampling rates. Recommended choices of a proper centroid algorithm have been given at the end of Section 5, and depend on the nature of the LMF and UMF as well as whether the centroid is to be used for approximation, where high accuracy is required, or for a FLS, where computations are usually constrained by a pre-specified sampling rate.

Finally, it is important to understand that KM and EKM algorithms are also used for many other kinds of problems in which the x_is are not naturally ordered and do not correspond to the sampled values of a continuous variable, e.g. interval weighted average, fuzzy weighted average, linguistic weighted average and generalized centroid [8,17,22]. For such applications, there is no numerical integral integration for the KM (EKM) algorithms; hence the results of this paper can not be used for such problems.

Acknowledgment

The authors would like to acknowledge Editor in Chief and the reviewers of this paper who made very useful suggestions that have improved the presentation of our results.

References

[1] P.T. Chang, K.C. Hung, K.P. Lin, C.H. Chang, A comparison of discrete algorithms for fuzzy weighted average, IEEE Transactions on Fuzzy Systems 14 (5) (2006) 663–675.
[2] Y.Y. Guh, C.C. Hon, K.M. Wang, E.S. Lee, Fuzzy weighted average: a max-min paired elimination method, Computers & Mathematics with Applications 32 (8) (1996) 115–123.
[3] S.M. Guu, Fuzzy weighted averages revisited, Fuzzy Sets and Systems 126 (3) (2002) 411–414.
[4] R.I. John, S. Coupland, Type-2 fuzzy logic: a historical view, IEEE Computational Intelligence Magazine 2 (1) (2007) 57–62.
[5] N.N. Karnik, J.M. Mendel, Centroid of a type-2 fuzzy set, Information Sciences 132 (1-4) (2001) 195–220.
[6] D.H. Lee, D.H. Park, An efficient algorithm for fuzzy weighted average, Fuzzy Sets and Systems 87 (1) (1997) 39–45.
[7] F. Liu, An efficient centroid type-reduction strategy for general type-2 fuzzy logic system, Information Sciences 178 (9) (2008) 2224–2236.
[8] F. Liu, J.M. Mendel, Aggregation using the fuzzy weighted average as computed by the Karnik–Mendel algorithms, IEEE Transactions on Fuzzy Systems 16 (1) (2008) 1–12.
[9] J.H. Mathews, K.K. Fink, Numerical Methods Using Matlab, Prentice-Hall Inc, Upper Saddle River, NJ, 2004.
[10] J.M. Mendel, Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions, Prentice-Hall, Upper Saddle River, NJ, 2001.
[11] J.M. Mendel, Computing derivatives in interval type-2 fuzzy logic systems, IEEE Transactions on Fuzzy Systems 12 (1) (2004) 84–98.
[12] J.M. Mendel, Computing with words and its relationships with fuzzistics, Information Sciences 177 (4) (2007) 988–1006.
[13] J.M. Mendel, On answering the question 'where do I start in order to solve a new problem involving interval type-2 fuzzy sets?', Information Sciences 179 (19) (2009) 3418–3431.
[14] J.M. Mendel, R.I. John, Type-2 fuzzy sets made simple, IEEE Transactions on Fuzzy Systems 10 (2) (2002) 117–127.

[15] J.M. Mendel, F. Liu, Super-exponential convergence of the Karnik–Mendel algorithms for computing the centroid of an interval type-2 fuzzy set, IEEE Transactions on Fuzzy Systems 15 (2) (2007) 309–320.
[16] J.M. Mendel, F. Liu, D. Zhai, α-plane representation for type-2 fuzzy sets: theory and applications, IEEE Transactions on Fuzzy Systems 17 (5) (2009) 1189–1207.
[17] J.M. Mendel, D. Wu, Perceptual Computing: Aiding People in Making Subjective Judgments, Wiley-IEEE Press, Hoboken, NJ, 2010.
[18] J.M. Mendel, H. Wu, Type-2 fuzzistics for symmetric interval type-2 fuzzy sets: Part 1, forward problems, IEEE Transactions on Fuzzy Systems 14 (6) (2006) 781–792.
[19] J.M. Mendel, H. Wu, New results about the centroid of an interval type-2 fuzzy set, including the centroid of a fuzzy granule, Information Sciences 177 (2) (2007) 360–377.
[20] C. Wagner, H. Hagras, Toward general type-2 fuzzy logic systems based on zslices, IEEE Transactions on Fuzzy Systems 18 (4) (2010) 637–660.
[21] C.H. Wang, C.S. Cheng, T.T. Lee, Dynamical optimal training for interval type-2 fuzzy neural network (T2FNN), IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics 34 (3) (2004) 1462–1477.
[22] D. Wu, J.M. Mendel, Aggregation using the linguistic weighted average and interval type-2 fuzzy sets, IEEE Transactions on Fuzzy Systems 15 (6) (2007) 1145–1161.
[23] D. Wu, J.M. Mendel, Uncertainty measures for interval type-2 fuzzy sets, Information Sciences 177 (23) (2007) 5378–5393.
[24] D. Wu, J.M. Mendel, Enhanced Karnik–Mendel algorithms, IEEE Transactions on Fuzzy Systems 17 (4) (2009) 923–934.
[25] D. Wu, J.M. Mendel, Perceptual reasoning for perceptual computing: a similarity-based approach, IEEE Transactions on Fuzzy Systems 17 (6) (2009) 1397–1411.
[26] D.R. Wu, J.M. Mendel, Computing with words for hierarchical decision making applied to evaluating a weapon system, IEEE Transactions On Fuzzy Systems 18 (3) (2010) 441–460.
[27] C.Y. Yeh, W.H.R. Jeng, S.J. Lee, An enhanced type-reduction algorithm for type-2 fuzzy sets, IEEE Transactions on Fuzzy Systems 19 (2) (2011) 227–240.
[28] D.Y. Zhai, J.M. Mendel, Uncertainty measures for general type-2 fuzzy sets, Information Sciences 181 (3) (2011) 503–518.