Enhanced Karnik-Mendel Algorithms for Interval Type-2 Fuzzy Sets and Systems

Dongrui Wu and Jerry M. Mendel

Signal and Image Processing Institute, Ming Hsieh Department of Electrical Engineering,
University of Southern California, 3740 McClintock Ave., Los Angeles, CA 90089-2564
Email: dongruiw@usc.edu; mendel@sipi.usc.edu.

***Abstract*— The Karnik-Mendel (KM) algorithms are iterative procedures widely used in fuzzy logic theory. They are known to converge monotonically and super-exponentially fast; however, several (usually two to six) iterations are still needed before convergence occurs. Methods to reduce their computational cost are proposed in this paper. Extensive simulations show that on average the enhanced KM algorithms can save about two iterations, which corresponds to more than a 39% reduction in computation time.**

***Index Terms*— Karnik-Mendel algorithms, interval type-2 fuzzy sets, type-reduction, centroid computation**

I. Introduction

The following problem is frequently met in (but not limited to) fuzzy logic theory:

Given

$$x_i \in X_i \equiv [\underline{x}_i, \overline{x}_i], \qquad i = 1, 2, \ldots, N \quad (1)$$

$$w_i \in W_i \equiv [\underline{w}_i, \overline{w}_i], \qquad i = 1, 2, \ldots, N \quad (2)$$

where

$$\underline{x}_i \leq \overline{x}_i, \qquad i = 1, 2, \ldots, N \quad (3)$$

$$\underline{w}_i \leq \overline{w}_i, \qquad i = 1, 2, \ldots, N \quad (4)$$

compute

$$Y = \frac{\sum_{i=1}^{N} X_i W_i}{\sum_{i=1}^{N} W_i} \equiv [y_l, y_r], \quad (5)$$

where

$$y_l = \min_{\substack{\forall x_i \in [\underline{x}_i, \overline{x}_i] \\ \forall w_i \in [\underline{w}_i, \overline{w}_i]}} \frac{\sum_{i=1}^{N} x_i w_i}{\sum_{i=1}^{N} w_i} \quad (6)$$

$$y_r = \max_{\substack{\forall x_i \in [\underline{x}_i, \overline{x}_i] \\ \forall w_i \in [\underline{w}_i, \overline{w}_i]}} \frac{\sum_{i=1}^{N} x_i w_i}{\sum_{i=1}^{N} w_i} \quad (7)$$

Places where this problem occurs are:

1. *Computing uncertainty measures for interval type-2 fuzzy sets (IT2 FSs):*
   - 1.1) In computing the centroid of an IT2 FS $\tilde{A}$ [1], [3], [5], $\underline{x}_i = \overline{x}_i = z_i$ represent discretizations of the primary variable z, and interval $[\underline{w}_i, \overline{w}_i]$ is the membership grade[1] of z_i, as shown in Fig. 1. Y is the centroid of $\tilde{A}$.
   - 1.2) In computing the variance of an IT2 FS $\tilde{A}$ [9], $\underline{x}_i = \overline{x}_i = [z_i - c(\tilde{A})]^2$, where $c(\tilde{A})$ is the center of the centroid of $\tilde{A}$, and $[\underline{w}_i, \overline{w}_i]$ is the membership grade of z_i. Y is the variance of $\tilde{A}$.
   - 1.3) In computing the skewness of an IT2 FS $\tilde{A}$ [9], $\underline{x}_i = \overline{x}_i = [z_i - c(\tilde{A})]^3$, and $[\underline{w}_i, \overline{w}_i]$ is the membership grade of z_i. Y is the skewness of $\tilde{A}$.
2. *Type-reduction:*
   - 2.1) In centroid and center-of-sums type-reduction of IT2 fuzzy logic systems (FLSs) [1], an IT2 FS is first obtained by combining the output sets for fired rules, after which computing the type-reduced set is equivalent to computing the centroid of that IT2 FS, as in Item 1.1. Y is the type-reduced set.
   - 2.2) In center-of-sets type-reduction of IT2 FLSs [1], X_i represents the centroid of the consequent IT2 FS of the ith rule, and W_i is the firing level of that rule. Y is the type-reduced set.
   - 2.3) In height type-reduction of IT2 FLSs [1], $\underline{x}_i = \overline{x}_i$ represents the point having maximum membership in the consequent type-1 FS of the ith rule, and W_i is the firing level of that rule. Y is the type-reduced set. The operations in modified height type-reduction [1] are quite similar, except that W_i is multiplied by a scale factor.
3. *Computing novel weighted averages*[2] *(NWAs):*
   - 3.1) In computing the interval weighted average (IWA) [6], X_i are input signals and W_i are their associated weights, both of which are interval sets. Y, which is also an interval set, is the IWA.
   - 3.2) In computing the fuzzy weighted average (FWA) [4], [6], X_i and W_i are α-cuts on the input signals and the weights, both of which are type-1 FSs. Y is the corresponding α-cut on the FWA.
   - 3.3) In computing the linguistic weighted average (LWA) [6], [7], X_i and W_i are α-cuts of the upper membership functions–UMFs (lower membership functions–LMFs) of the inputs signals and the weights, both of which are IT2 FSs. Y is the

[1]The lower and upper memberships of z_i are usually denoted as $\underline{\mu}(z_i)$ and $\overline{\mu}(z_i)$, respectively [1]. To be consistent with (6) and (7), in this paper we denote them as $\underline{w}_i$ and $\overline{w}_i$, respectively.

[2]NWAs are weighted averages in which at least one of the weights are novel models [6], i.e., intervals, type-1 FSs, or IT2 FSs.

corresponding α-cut on the UMF (LMF) of the LWA, which is also an IT2 FS.

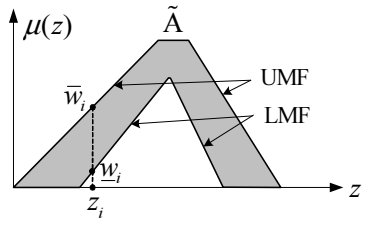

Fig. 1. An IT2 FS. UMF: upper membership function; LMF: lower membership function.

It is well-known that y_l and y_r can be expressed as [1]

$$y_l = \min_{w_i \in [\underline{w}_i, \overline{w}_i]} \frac{\sum_{i=1}^{N} \underline{x}_i w_i}{\sum_{i=1}^{N} w_i} = \frac{\sum_{i=1}^{L} \underline{x}_i \overline{w}_i + \sum_{i=L+1}^{N} \underline{x}_i \underline{w}_i}{\sum_{i=1}^{L} \overline{w}_i + \sum_{i=L+1}^{N} \underline{w}_i} \tag{8}$$

$$y_r = \max_{w_i \in [\underline{w}_i, \overline{w}_i]} \frac{\sum_{i=1}^{N} \overline{x}_i w_i}{\sum_{i=1}^{N} w_i} = \frac{\sum_{i=1}^{R} \overline{x}_i \underline{w}_i + \sum_{i=R+1}^{N} \overline{x}_i \overline{w}_i}{\sum_{i=1}^{R} \underline{w}_i + \sum_{i=R+1}^{N} \overline{w}_i} \tag{9}$$

where L and R are switch points. There is no closed-form solution for L and R, and hence for y_l and y_r. KM algorithms [1] are used to compute them iteratively. Because the computational burden and iterative nature of KM algorithms may hinder them from some real-time applications, a reduction in the computational cost is desired, and this is the focus of our paper.

This paper is organized as follows: Section II briefly introduces the original KM algorithms; Section III proposes the enhanced KM (EKM) algorithms; Section IV presents simulation results to verify the effectiveness of the proposed algorithms; and, finally, Section V draws conclusions.

II. The Original KM Algorithms

A. KM Algorithm for Computing y_l [1], [3]

1) Sort $\underline{x}_i$ $(i = 1, 2, \ldots, N)$ in increasing order and call the sorted $\underline{x}_i$ by the same name, but now $\underline{x}_1 < \underline{x}_2 < \cdots < \underline{x}_N$. Match the weights w_i with their respective $\underline{x}_i$ and renumber them so that their index corresponds to the renumbered $\underline{x}_i$.
2) Initialize w_i by setting

$$w_i = \frac{\underline{w}_i + \overline{w}_i}{2} \qquad i = 1, 2, \ldots, N \tag{10}$$

and then compute

$$y = \frac{\sum_{i=1}^{N} \underline{x}_i w_i}{\sum_{i=1}^{N} w_i} \tag{11}$$

3) Find switch point k $(1 \le k \le N-1)$ such that

$$\underline{x}_k \le y \le \underline{x}_{k+1} \tag{12}$$

4) Set

$$w_i = \begin{cases} \overline{w}_i, & i \le k \\ \underline{w}_i, & i > k \end{cases} \tag{13}$$

and compute

$$y' = \frac{\sum_{i=1}^{N} \underline{x}_i w_i}{\sum_{i=1}^{N} w_i} \tag{14}$$

5) Check if $y' = y$. If yes, stop, set $y_l = y$ and call k L. If no, go to Step 6.
6) Set $y = y'$ and go to Step 3.

B. KM Algorithm for Computing y_r [1], [3]

1) Sort $\overline{x}_i$ $(i = 1, 2, \ldots, N)$ in increasing order and call the sorted $\overline{x}_i$ by the same name, but now $\overline{x}_1 < \overline{x}_2 < \cdots < \overline{x}_N$. Match the weights w_i with their respective $\overline{x}_i$ and renumber them so that their index corresponds to the renumbered $\overline{x}_i$.
2) Initialize w_i by setting

$$w_i = \frac{\underline{w}_i + \overline{w}_i}{2} \qquad i = 1, 2, \ldots, N \tag{15}$$

and then compute

$$y = \frac{\sum_{i=1}^{N} \overline{x}_i w_i}{\sum_{i=1}^{N} w_i} \tag{16}$$

3) Find switch point k $(1 \le k \le N-1)$ such that

$$\overline{x}_k \le y \le \overline{x}_{k+1} \tag{17}$$

4) Set

$$w_i = \begin{cases} \underline{w}_i, & i \le k \\ \overline{w}_i, & i > k \end{cases} \tag{18}$$

and compute

$$y' = \frac{\sum_{i=1}^{N} \overline{x}_i w_i}{\sum_{i=1}^{N} w_i} \tag{19}$$

5) Check if $y' = y$. If yes, stop, set $y_r = y$ and call k R. If no, go to Step 6.
6) Set $y = y'$ and go to Step 3.

The KM algorithms have been proven to converge monotonically and super-exponentially fast [5]; however, several (usually two to six) iterations are still needed before convergence occurs. As pointed out in [5], "*an open problem is to find an optimal way to initialize the KM algorithm*," optimal in the sense that the super-exponential convergence factor δ defined below in (37) is minimized.

III. EKM Algorithms

This section presents EKM algorithms to reduce the computational cost of the original ones. Similar to the original KM algorithms, the EKM algorithms also consist of two parts, one for computing y_l and the other for computing y_r. Because the two parts are quite similar, we focus on the EKM algorithm for computing y_l in this section.

A. Optimal Initial Switch Point

When we use (10) to initialize the KM algorithm, y_l in the first iteration can be expressed as

$$y_l = \frac{\sum_{i=1}^{N} \underline{x}_i \frac{\overline{w}_i + \underline{w}_i}{2}}{\sum_{i=1}^{N} \frac{\overline{w}_i + \underline{w}_i}{2}} \tag{20}$$

which looks quite different from (8), and suggests that better choices for the initialization of the KM algorithm in line with (8) should be possible.

Observe that (8) shows that when $i \leq L$, $\overline{w}_i$ is used to compute y_l; and, when $i > L$, $\underline{w}_i$ is used to compute y_l. This implies that a better initialization of y_l is to find a good guess of L, L_0. Because y_l is the smallest value of Y, we conjecture that very probably it is smaller than[3] $\underline{x}_{[N/2]}$, the center element of $\underline{x}_i$; consequently, L_0 should also be smaller than $[N/2]$. We performed extensive simulations by initializing $L_0 = \{[N/2], [N/2.1], \ldots, [N/2.6]\}$ and comparing the number of iterations for the algorithms to converge for uniformly and independently distributed $\underline{w}_i$, $\overline{w}_i$ and $\underline{x}_i$, and found that $L_0 = [N/2.4]$ gave the fewest number of iterations (more details on the simulations and comparison are given in Section IV). We performed similar simulations for y_r, and found that the optimal initial switch point is $R_0 = [N/1.7]$.

B. The Termination Test

Observe from Step 5 in Section II-A that the test $y' = y$ is performed to determine whether the iterations should stop or continue. When the iterations stop, $y' = y$, and because y' is obtained during the present iteration and y was obtained from the previous iteration, $y' = y$ means the present iteration makes no contribution to minimizing y_l, consequently, it can be deleted without changing y_l.

Denote the switch points for y' and y as k' and k, respectively. Then,

$$y' = \frac{\sum_{i=1}^{k'} \underline{x}_i \overline{w}_i + \sum_{i=k'+1}^{N} \underline{x}_i \underline{w}_i}{\sum_{i=1}^{k'} \overline{w}_i + \sum_{i=k'+1}^{N} \underline{w}_i} \tag{21}$$

$$y = \frac{\sum_{i=1}^{k} \underline{x}_i \overline{w}_i + \sum_{i=k+1}^{N} \underline{x}_i \underline{w}_i}{\sum_{i=1}^{k} \overline{w}_i + \sum_{i=k+1}^{N} \underline{w}_i} \tag{22}$$

Obviously, $y' = y$ is equivalent to $k' = k$. So, by changing the termination condition from $y' = y$ to $k' = k$, we have the same y_l but save one iteration. How to do this is shown in Section III-D.

C. Further Computational Cost Reduction

In the original KM algorithm for computing y_l, in each iteration we compute $\sum_{i=1}^{N} w_i$ and $\sum_{i=1}^{N} \underline{x}_i w_i$ in entirety and then compute y' in (14). This is a waste of computational power because results from the previous iteration are not utilized.

After the jth iteration, let switch point k, $\sum_{i=1}^{N} w_i$ and $\sum_{i=1}^{N} \underline{x}_i w_i$ be denoted as k_j, $(\sum_{i=1}^{N} w_i)_j$ and $(\sum_{i=1}^{N} \underline{x}_i w_i)_j$, respectively. Usually k_j and k_{j+1} are quite close to each other. Consequently, the w_i in the $(j+1)$th iteration share lots of common terms with the w_i from the jth iteration. This means $(\sum_{i=1}^{N} w_i)_j$ and $(\sum_{i=1}^{N} \underline{x}_i w_i)_j$ can be used to compute $(\sum_{i=1}^{N} w_i)_{j+1}$ and $(\sum_{i=1}^{N} \underline{x}_i w_i)_{j+1}$, i.e., only the differences between $(\sum_{i=1}^{N} w_i)_{j+1}$ and $(\sum_{i=1}^{N} w_i)_j$, as well as $(\sum_{i=1}^{N} \underline{x}_i w_i)_{j+1}$ and $(\sum_{i=1}^{N} \underline{x}_i w_i)_j$, need to be computed, after which these differences are added to $(\sum_{i=1}^{N} w_i)_j$ and $(\sum_{i=1}^{N} \underline{x}_i w_i)_j$ [as shown in (27) and (28)]. A similar technique was already used in [2].

D. EKM Algorithms

As a summary, the complete EKM algorithms are presented.

The **EKM algorithm for computing** y_l is:

1) Sort $\underline{x}_i$ $(i = 1, 2, \ldots, N)$ in increasing order and call the sorted $\underline{x}_i$ by the same name, but now $\underline{x}_1 \leq \underline{x}_2 \leq \cdots \leq \underline{x}_N$. Match the weights w_i with their respective $\underline{x}_i$ and renumber them so that their index corresponds to the renumbered $\underline{x}_i$.
2) Set $k = [N/2.4]$ (the nearest integer to $N/2.4$), and compute

$$a = \sum_{i=1}^{k} \underline{x}_i \overline{w}_i + \sum_{i=k+1}^{N} \underline{x}_i \underline{w}_i \tag{23}$$

$$b = \sum_{i=1}^{k} \overline{w}_i + \sum_{i=k+1}^{N} \underline{w}_i \tag{24}$$

and

$$y = a/b \tag{25}$$

3) Find $k' \in [1, N-1]$ such that

$$\underline{x}_{k'} \leq y \leq \underline{x}_{k'+1} \tag{26}$$

4) Check if $k' = k$. If yes, stop, set $y_l = y$ and call k L. If no, continue.
5) Compute $s = \text{sign}(k' - k)$, and

$$a' = a + s \sum_{i=\min(k,k')+1}^{\max(k,k')} \underline{x}_i (\overline{w}_i - \underline{w}_i) \tag{27}$$

$$b' = b + s \sum_{i=\min(k,k')+1}^{\max(k,k')} (\overline{w}_i - \underline{w}_i) \tag{28}$$

$$y' = a'/b' \tag{29}$$

6) Set $y = y'$, $a = a'$, $b = b'$ and $k = k'$. Go to Step 3.

The **EKM algorithm for computing** y_r is:

1) Sort $\overline{x}_i$ $(i = 1, 2, \ldots, N)$ in increasing order and call the sorted $\overline{x}_i$ by the same name, but now $\overline{x}_1 \leq \overline{x}_2 \leq \cdots \leq \overline{x}_N$. Match the weights w_i with their respective $\overline{x}_i$ and renumber them so that their index corresponds to the renumbered $\overline{x}_i$.
2) Set $k = [N/1.7]$ (the nearest integer to $N/1.7$), and compute

$$a = \sum_{i=1}^{k} \overline{x}_i \underline{w}_i + \sum_{i=k+1}^{N} \overline{x}_i \overline{w}_i \tag{30}$$

[3] $[N/2]$ denotes the nearest integer number that $N/2$ can be rounded to. This conversion is needed because L_0 (R_0) must be an integer number whereas $N/2$ is not necessarily an integer.

$$b = \sum_{i=1}^{k} \underline{w}_i + \sum_{i=k+1}^{N} \overline{w}_i \tag{31}$$

and

$$y = a/b \tag{32}$$

3) Find $k' \in [1, N-1]$ such that

$$\overline{x}_{k'} \le y \le \overline{x}_{k'+1} \tag{33}$$

4) Check if $k' = k$. If yes, stop, set $y_r = y$ and call k R. If no, continue.
5) Compute $s = \text{sign}(k' - k)$, and

$$a' = a - s \sum_{i=\min(k,k')+1}^{\max(k,k')} \overline{x}_i(\overline{w}_i - \underline{w}_i) \tag{34}$$

$$b' = b - s \sum_{i=\min(k,k')+1}^{\max(k,k')} (\overline{w}_i - \underline{w}_i) \tag{35}$$

$$y' = a'/b' \tag{36}$$

6) Set $y = y'$, $a = a'$, $b = b'$ and $k = k'$. Go to Step 3.

IV. Comparative Studies

Extensive simulations have been conducted to verify the performance of the EKM algorithms. The platform was a Dell Precision 690 Workstation running Windows XP x64 Edition and Matlab 7.3.0 with two Intel Xeon 2.66GHZ processors and 2GB RAM. Because the results for computing y_r are quite similar to those for computing y_l, only the comparative studies for computing y_l are presented in this section.

In the simulations, we increased N by one from 3 to 20 (i.e., $N = 3, 4, \ldots, 20$), and then increased it by five from 20 to 100 (i.e., $N = 25, 30, \ldots, 100$). For each N, 10,000 Monte Carlo simulations were used to compute y_l, i.e., for each N, 10,000 $\underline{x}_i$ were generated using Matlab function *rand(10000,1)*, and 10,000 pairs of $\{\underline{w}_i, \overline{w}_i\}$ were generated by using Matlab function *rand(10000,2)*. Observe that all $\underline{x}_i$, $\underline{w}_i$ and $\overline{w}_i$ were constrained in $[0, 1]$, and $\underline{x}_i$ were independent of $\underline{w}_i$ and $\overline{w}_i$. To make sure $\underline{w}_i \le \overline{w}_i$, we checked each pair of $\{\underline{w}_i, \overline{w}_i\}$ and assigned the smaller value to $\underline{w}_i$ and the larger one to $\overline{w}_i$.

A. Performance Measures and Observations

The performance measures used in the comparative studies and the corresponding observations are:

1) *Mean and standard deviation of super-exponential convergence factor* δ (Fig. 2), which is defined in (30) of [5] as an indicator of the super-exponential convergence speed for computing y_l. Because δ in [5] is defined for the continuous version of KM algorithms, and in this study we used the discrete version of KM algorithms, we used the following discrete version of δ:

$$\delta \equiv \frac{y_{l1} - y_l}{y_{l0} - y_l} \tag{37}$$

In (37), y_{l0} is the initial value of y_l (i.e., the y computed by (11) for the original KM algorithm, or the y computed by (25) for the EKM algorithm), and y_{l1} is the y' computed from the first iteration of each algorithm.

Observe from Fig. 2 that both the mean and the standard deviation of δ for the EKM algorithm are smaller than those of the original KM algorithm. Most impressively, the mean of δ calculated from the EKM algorithm is about $1/100$ of that from the original KM algorithm, which suggests that the EKM algorithm converges much faster than the original one.

2) *Average number of iterations obtained from different algorithms* (Fig. 3). For both the original and EKM algorithms the number of iterations is defined as the times the loop consisting of Steps (3)-(6) in Sections II-A and III-D are executed. These definitions are consistent with those used in [5].

From Fig. 3, observe that the average number of iterations for the EKM algorithm is smaller than that for the original KM algorithm. More interestingly, the average number of iterations for the EKM algorithm is less than one. This is because uniformly and independently distributed $\underline{w}_i$, $\overline{w}_i$ and $\underline{x}_i$ were used in the simulation, and hence $L_0 = [N/2.4]$ has a good chance to be the final switch point, especially when N is small, as confirmed by Fig. 4 and explained in Observation (3). When the distributions of $\underline{w}_i$, $\overline{w}_i$ and $\underline{x}_i$ are not uniform and independent, the average number of iterations for the EKM algorithm will increase, as discussed in [8]. Observe also that as N increases, the average number of iterations in the original KM algorithm also increases; however, the increase becomes much slower as N gets larger. This coincides with the conclusion that KM algorithms converge monotonically and super-exponentially fast [5].

3) *Histograms of the number of iterations obtained from different algorithms.*

Results for $N = \{5, 10, 100\}$ are shown in Fig. 4, where $\{5, 10\}$ represent the typical N used in type-reduction and in computing the IWA, FWA and LWA, and 100 represents the typical N used in computing the centroid, variance and skewness of IT2 FSs. Observe that when $N \le 100$, the original KM algorithm converges in 2-4 iterations whereas the EKM algorithm converges in 0-2 iterations. Observe also that when $N = 5$, there is more than 50% probability that the EKM algorithm converges in zero iterations, i.e. the initial switch point L_0 equals the final switch point L. This result seems surprising, but it can be easily explained. When $N = 5$, there are only four possible switch points $L = \{1, 2, 3, 4\}$. The initial switch point $L_0 = [N/2.4] = 2$ has a good chance to be the final switch point, i.e., at least $1/4$ in probability. As N increases, the number of possible switch points also increases, and hence the probability that the EKM converges in zero iterations decreases, as confirmed

by Figs. 4(d) and 4(f).

4) *The average reduced number of iterations.*

   Observe from Fig. 5 that the EKM algorithm eliminates about two iterations, and that the reduced number of iterations increases on average as N gets larger.

5) *Probability that the algorithm converges in one iteration.*

   As shown in Fig. 6, the EKM algorithm converges in one iteration with a probability of 0.97, and that probability is almost a constant for all N. On the other hand, the original KM algorithm almost surely needs more than one iteration to converge. These results are also verified by Fig. 4.

6) *The average computation time for different algorithm.*

   Observe from Fig. 7 that the average computation time for both algorithms increases as N increases; however, the EKM algorithm is much faster than the original KM algorithm. Observe also that the standard deviation of the computation time for the EKM algorithm is slightly larger than that for the original KM algorithm.

7) *The percentage of computation time reduction over the original KM algorithm.*

   Observe from Fig. 8 that the percentage of computation time reduction of the EKM algorithm over the original KM algorithm decreases as N increases. For $N \in [3, 100]$, there is more than a 39% computation time reduction.

Finally, we wish to emphasize that the above results were obtained only for uniformly and independently distributed $\underline{w}_i$, $\overline{w}_i$ and $\underline{x}_i$, and the performance of the EKM algorithms may deteriorate for other distributions, as discussed in a journal version of this paper [8].

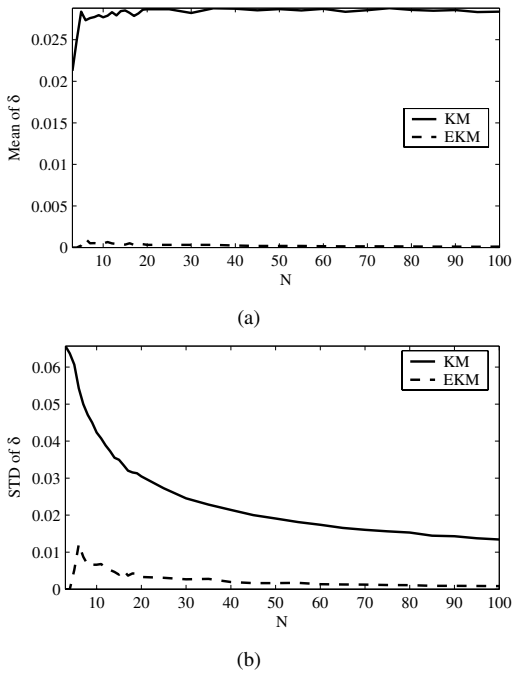

Fig. 2. (a) Mean and (b) standard deviation of δ when computing y_l.

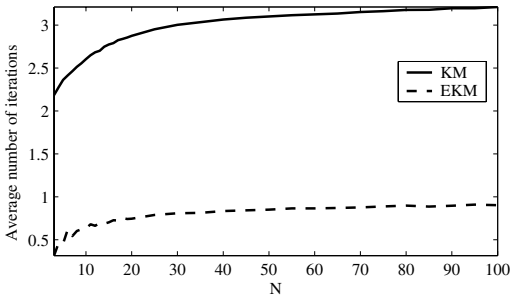

Fig. 3. Average number of iterations when computing y_l.

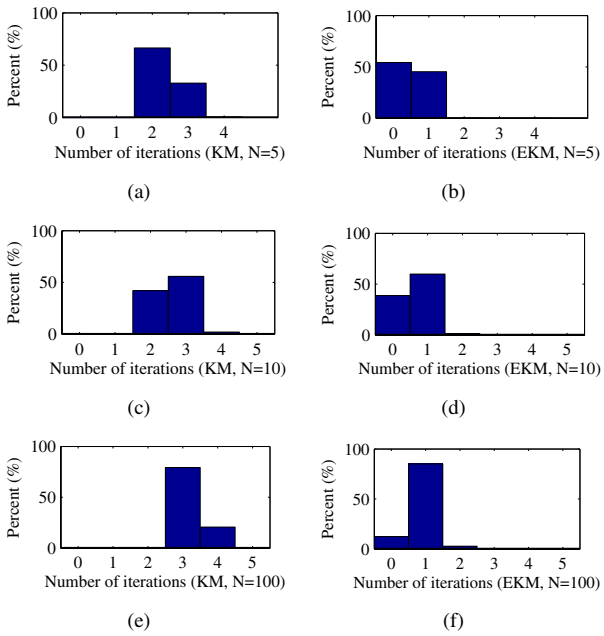

Fig. 4. Histograms of the number of iterations for different KM algorithms. (a)-(b): $N = 5$; (c)-(d): $N = 10$; (e)-(f): $N = 100$.

B. Discussions

One of the most time-consuming steps in the EKM algorithms is the first one, i.e., to sort $\underline{x}_i$ ($\overline{x}_i$) in ascending order; however, frequently this step can be skipped, e.g.:

1) In computing the centroid of an IT2 FS, $\underline{x}_i = \overline{x}_i = z_i$, where z_i are the samples of the primary variable (Fig. 1), and they are already in ascending order.
2) In computing the skewness of an IT2 FS, $\underline{x}_i = \overline{x}_i = [z_i - c(\tilde{A})]^3$, because z_i are already in ascending order, $[z_i - c(\tilde{A})]^3$ are also in ascending order.
3) For center-of-sets type-reduction, once the design of an IT2 FLS is completed, the centroids of the consequent IT2 FSs can be computed and sorted off-line, and hence they can be used directly in on-line computations without further sorting.

Monte Carlo simulations were also used to study the further computation time reduction for the EKM algorithms when sorting is not needed. The same $\underline{w}_i$, $\overline{w}_i$ and $\underline{x}_i$ as those in Section IV were used. Denote the time to finish Step 1 of the EKM algorithms as t_1, and the time to finish Steps 2-6 as t_2. Then, the ratio $t_1/(t_1 + t_2)$ was used as an

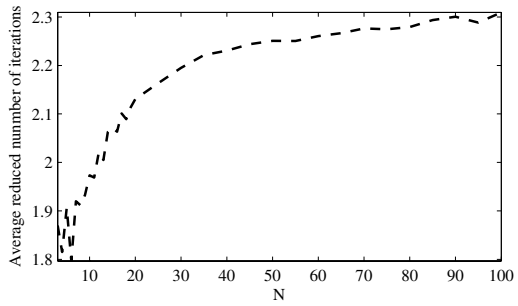

Fig. 5. Average reduced number of iterations of the EKM algorithm over the original KM algorithm.

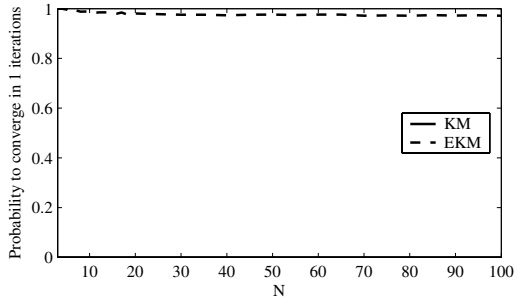

Fig. 6. The probability of each algorithm converging in one iteration. Note that the probability corresponding to the original KM algorithm is 0.

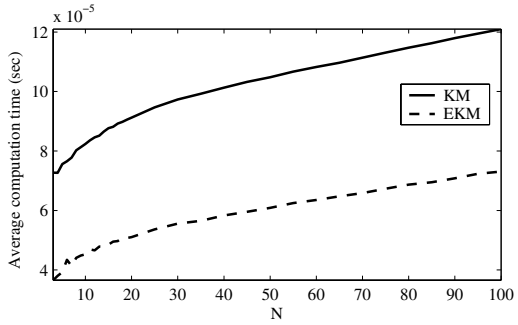

Fig. 7. Average computation time for different KM algorithms.

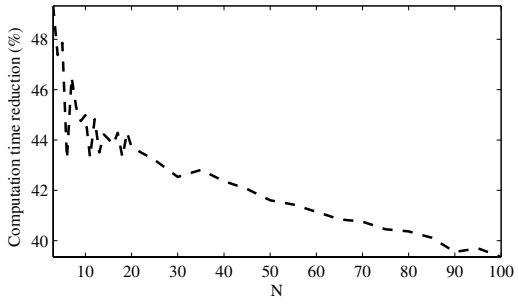

Fig. 8. Percentage of computation time reduction of the EKM algorithm over the original KM algorithm.

indicator of how much computation time can be saved if no sorting is needed. The results are shown in Fig. 9. Observe that when $N \in [3, 11]$, the percentage of computation time reduction decreases as N increases; on the other hand, when $N \in [11, 100]$, the percentage of computation time reduction increases as N increases. Fig. 9 also shows that there is at least 23% computation time reduction. This will make the EKM algorithms more suitable for real-time applications.

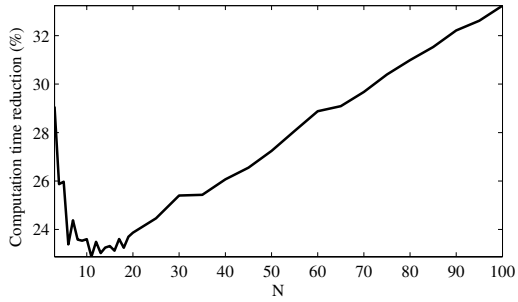

Fig. 9. Percentage of computation time reduction for the EKM algorithm if sorting of $\underline{x}_i$ ($\overline{x}_i$) is not needed.

V. Conclusions

KM algorithms are iterative procedures widely used in centroid, variance and skewness computations of IT2 FSs, type-reduction of IT2 FLSs, and for computing the IWA, FWA and LWA. They have been proven to converge monotonically and super-exponentially fast; however, several (usually two to six) iterations are still needed before convergence occurs. In this paper EKM algorithms have been proposed to reduce the computational cost. Extensive simulations show that on average the EKM algorithms can save about two iterations, which corresponds to a more than 39% reduction in computation time. An additional (at least) 23% computational cost can be saved if no sorting of $\underline{x}_i$ ($\overline{x}_i$) is needed.

References

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

[2] S.-M. Guu, "Fuzzy weighted averages revisited," *Fuzzy Sets and Systems*, vol. 126, pp. 411–414, 2002.

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

[4] F. Liu and J. M. Mendel, "Aggregation using the fuzzy weighted average, as computed using the Karnik-Mendel algorithms," *IEEE Trans. on Fuzzy Systems*, 2007, in press.

[5] J. M. Mendel and F. Liu, "Super-exponential convergence of the Karnik-Mendel algorithms for computing the centroid of an interval type-2 fuzzy set," *IEEE Trans. on Fuzzy Systems*, vol. 15, no. 2, pp. 309–320, 2007.

[6] J. M. Mendel and D. Wu, "Signal fusion using novel weighted averages," submitted to *Proc. IEEE*, Feb 2007.

[7] D. Wu and J. M. Mendel, "Aggregation using the linguistic weighted average and interval type-2 fuzzy sets," accepted by *IEEE Trans. on Fuzzy Systems*, 2007.

[8] ——, "Enhanced Karnik-Mendel Algorithms," submitted to *IEEE Trans. on Fuzzy Systems*, April 2007.

[9] ——, "Uncertainty measures for interval type-2 fuzzy sets," submitted to *Information Sciences*, March 2007.