Aggregation Using the Fuzzy Weighted Average as Computed by the Karnik–Mendel Algorithms

Feilong Liu, *Student Member, IEEE*, and Jerry M. Mendel, *Life Fellow, IEEE*

***Abstract*—By connecting work from two different problems—the fuzzy weighted average (FWA) and the generalized centroid of an interval type-2 fuzzy set—a new α-cut algorithm for solving the FWA problem has been obtained, one that is monotonically and superexponentially convergent. This new algorithm uses the Karnik–Mendel (KM) algorithms to compute the FWA α-cut end-points. It appears that the KM α-cut algorithms approach for computing the FWA requires the fewest iterations to date, and may therefore be the fastest available FWA algorithm to date.**

***Index Terms*—Centroid, fuzzy weighted average, interval type-2 fuzzy set, Karnik–Mendel (KM) algorithms.**

I. Introduction

SOMETIMES the same or similar problems are solved in different settings. This is a paper about such a situation, namely, computing the *fuzzy weighted average* (FWA), which is a problem that has been studied in multiple criteria decision making [2]–[5], [8], [9] and computing the *generalized centroid of an interval type-2 fuzzy set* [herein referred to as the generalized centroid (GC)], which is a problem that has been studied in rule-based interval type-2 fuzzy logic systems [6], [10], [11]. Here it is demonstrated how very significant computational advantages can be obtained for solving the FWA problem by using the GC algorithms developed by Karnik and Mendel.[1]

The FWA, which is a weighted average involving type-1 (T1) fuzzy sets[2] and is explained mathematically below, is useful as an aggregation (fusion) method in situations where decisions (x_i) and expert weights (w_i) are modeled as T1 fuzzy sets, thereby allowing some uncertainties about either the decisions, or weights, or both to be incorporated into the average.

To begin, consider the following *weighted average*:

$$y = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} = f(w_1, \ldots, w_n, x_1, \ldots, x_n). \quad (1)$$

In (1), w_i are *weights* that act upon *decisions* (or, attributes, indicators, features, etc.) x_i. While it is always true that the sum of the normalized weights that act upon each x_i add to one, it is not a requirement that the sum of the unnormalized weights must add to one. In many situations requiring $\sum_{i=1}^{n} w_i = 1$ is too restrictive, especially when w_i are no longer crisp numbers [as in items 3)–5) below]; so, such a requirement is not imposed. It is the normalization that makes the calculation of the FWA very challenging.

There is a hierarchy of averages that can be associated with (1). They are enumerated next so that it will be clear where the FWA stands in this hierarchy.

1) $\forall w_i$ and $\forall x_i$ are crisp numbers: In this case, y is a crisp number, the commonly used *arithmetic weighted average,* a number that is easily computed using arithmetic.
2) $\forall w_i$ are crisp numbers, and $\forall x_i$ are interval numbers, i.e., $x_i = [a_i, b_i]$ where interval end-points a_i and b_i are prespecified: In this case, y is an interval number (a *weighted average of intervals*), i.e., $y = [y_l, y_r]$, where y_l and y_r are easily computed [because interval sets only appear in the numerator of (1)] using interval arithmetic.
3) $\forall x_i$ are crisp numbers and $\forall w_i$ are interval numbers, i.e., $w_i = [c_i, d_i]$ where interval end-points c_i and d_i are prespecified: This is a special case of the FWA that also corresponds to the so-called *centroid* of an interval type-2 fuzzy set [6], [10]. In this case, y is also an interval number (a *normalized interval-weighted average of crisp numbers*), i.e., $y = [y_l, y_r]$, but there are no known closed-form formulas for computing y_l and y_r. What makes these a challenging set of computations is the appearance of interval sets in both the numerator and the denominator of (1). The KM iterative algorithms [6], [10] have been used to compute y_l and y_r. These algorithms converge to exact solutions monotonically and superexponentially fast [10], so it takes very few iterations for them to converge to the actual values of y_l and y_r. They are explained in Section III.
4) $\forall x_i$ are interval numbers, i.e., $x_i = [a_i, b_i]$ where interval end-points a_i and b_i are prespecified and $\forall w_i$ are interval numbers, i.e., $w_i = [c_i, d_i]$ where interval end-points c_i and d_i are prespecified: This is another special case of the FWA that also corresponds to the so-called generalized centroid *of interval type-2 fuzzy sets.* As in case 3), y is also an interval number (a *normalized interval-weighted average of interval numbers*), i.e., $y = [y_l, y_r]$, but again there are no known closed-form formulas for computing y_l and y_r. The KM iterative algorithms have also been used to compute y_l and y_r. In this case, theory shows (as explained more fully in Sections II and III) that y_l is obtained by replacing each x_i in (1) with its associated interval left-end point a_i and y_r is obtained by replacing each x_i in (1) with its associated interval right-end point, b_i.
5) $\forall x_i$ are type-1 fuzzy numbers, i.e., each x_i is described by the membership function (MF) of a type-1 fuzzy set

Manuscript received November 6, 2005; revised April 8, 2006, August 7, 2006, and October 30, 2006.

The authors are with the Signal and Image Processing Institute, Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2564 USA (e-mail: feilongl@usc.edu; mendel@sipi.usc.edu).

Digital Object Identifier 10.1109/TFUZZ.2007.896229

[1]These algorithms are often referred to in the T2 fuzzy logic system (FLS) literature as the "KM algorithms" and are also referred to herein in this way.

[2]This includes the special case when the fuzzy sets are interval sets.

$\mu_{X_i}(x_i)$, where this MF must be prespecified, and $\forall w_i$ are also type-1 fuzzy numbers, i.e., each w_i is described by the MF of a type-1 fuzzy set $\mu_{W_i}(w_i)$, where this MF must also be prespecified. Of course, there could be special subcases of this case, e.g., only the weights are type-1 fuzzy numbers. This case is the FWA, and now y is a type-1 fuzzy set, with MF $\mu_Y(y)$, but there is no known closed-form formula for computing $\mu_Y(y)$. α-cuts, an α-cut decomposition theorem [7] of a type-1 fuzzy set, and a variety of algorithms can be used to compute $\mu_Y(y)$, as will be described in Sections II–IV. This is the case that is of immediate interest and importance and is the one on which this paper focuses.

The rest of this paper is organized as follows. Section II provides discussions about previous α-cut approaches for computing the FWA. Section III provides a lot of information about the KM algorithms including their derivations, statements for solving the FWA problem, and convergence properties. Section IV provides summaries of Lee and Park's [8] efficient FWA algorithms because they are the computationally most efficient ones that have appeared in the literature prior to this paper and have a structure similar to the KM algorithms. Section V presents experimental results in which the KM and efficient (EFWA) algorithms are compared. Section VI draws conclusions. Appendixes A and B provide important properties about the FWA; they should be read when the reader reaches Comment 3 in Section III-B.

II. Previous Approaches for Computing the FWA

According to [3], "The fuzzy weighted average (FWA) problem is, given n fuzzy weights W_i and n fuzzy X_i, how to obtain the fuzzy weighted average of the variable in (1)." Beginning in 1987, various solutions to this problem have been proposed. Dong and Wong [2] presented the first FWA algorithm; Liou and Wang [9] presented an improved FWA (IFWA) algorithm; and Lee and Park [8] presented an EFWA. All three algorithms are based on α-cuts.

A. On Dong and Wong's FWA Algorithm

Dong and Wong [2] were apparently the first to develop a method for computing the FWA. Although their algorithm is based on α-cuts and an α-cut decomposition theorem [7], it is very computationally inefficient because it uses an exhaustive search. Their algorithm is [5] the following.

1) Discretize the complete range of the membership [0, 1] of the fuzzy numbers into the following finite number of m α-cuts, $\alpha_1,\ldots,\alpha_m$, where the degree of accuracy depends on the number of α-cuts, i.e., m.
2) For each α_j, find the corresponding intervals for X_i in x_i and W_i in w_i. Denote the end-points of the intervals of x_i and $w_i (i = 1,\ldots,n)$ by $[a_i(\alpha_j), b_i(\alpha_j)]$ and $[c_i(\alpha_j), d_i(\alpha_j)]$, respectively.
3) Construct the 2^{2n} distinct permutations[3] of the $2n$ array $(w_1,\ldots,w_n; x_1,\ldots,x_n|\alpha_j)$ that involve just the interval end-points of $a_i(\alpha_j)$ and $b_i(\alpha_j)$; and $c_i(\alpha_j)$ and $d_i(\alpha_j)$.
4) Compute $y_k(\alpha_j) = f(w_{k1},\ldots,w_{kn}; x_{k1},\ldots,x_{kn} \mid \alpha_j)$, where $(w_{k1},\ldots,w_{kn}; x_{k1},\ldots,x_{kn} \mid \alpha_j)$ is the kth permutation of the 2^{2n} distinct permutations, and $k = 1,\ldots,2^{2n}$. Then the desired interval for $y(\alpha_j)$ is
$$y(\alpha_j) = \forall y \ni y \in [\min_k y_k(\alpha_j), \max_k y_k(\alpha_j)]. \qquad (2)$$
5) Repeat steps 2)–4) for every $\alpha_j, j = 1,\ldots,m$. Compute $\mu_Y(y)$ using $y(\alpha_1),\ldots,y(\alpha_m)$ and an α-cut decomposition theorem [7], i.e., let
$$I_{\alpha_j Y}(y) = \begin{cases} 1 & \forall y \in [\min_k y_k(\alpha_j), \max_k y_k(\alpha_j)] \\ 0 & \forall y \notin [\min_k y_k(\alpha_j), \max_k y_k(\alpha_j)] \end{cases} \qquad (3a)$$
so that
$$\mu_Y(y) = \sup_{\forall \alpha_j (j=1,\ldots,m)} \alpha_j I_{\alpha_{j_Y}}(y). \qquad (3b)$$

Since there are 2^{2n} permutations; this algorithm requires 2^{2n} iterations[4] for each α-cut, or a total of $m2^{2n}$ iterations; hence, the complete algorithm is intractable and therefore is not a practical FWA algorithm, even for moderate values of n.

B. On Liou and Wang's Improved FWA (IFWA) Algorithm

Liou and Wang [9] did some important analyses that led to an algorithm that drastically reduced the computational complexity of Dong and Wong's algorithm. They were the first to observe that since the x_i appear only in the numerator of (1), only the smallest values of the x_i are used to find the smallest value of (1), and only the largest values of the x_i are used to find the largest value of (1); hence
$$\begin{aligned} &\min_{\substack{\forall x_i \in [a_i(\alpha_j), b_i(\alpha_j)] \\ \forall w_i \in [c_i(\alpha_j), d_i(\alpha_j)]}} f(w_1,\ldots,w_n, x_1,\ldots,x_n|\alpha_j) \\ &= \min_{\forall w_i \in [c_i(\alpha_j), d_i(\alpha_j)]} f(w_1,\ldots,w_n, a_1,\ldots,a_n \mid \alpha_j) \\ &\triangleq \min_{\forall w_i \in [c_i(\alpha_j), d_i(\alpha_j)]} f_L(w_1,\ldots,w_n \mid \alpha_j) = f_L^*(\alpha_j) \end{aligned} \qquad (4)$$
where
$$f_L(w_1,\ldots,w_n|\alpha_j) \triangleq \frac{\sum_{i=1}^n w_i(\alpha_j) a_i(\alpha_j)}{\sum_{i=1}^n w_i(\alpha_j)} \qquad (5)$$
and
$$\begin{aligned} &\max_{\substack{\forall x_i \in [a_i(\alpha_j), b_i(\alpha_j)] \\ \forall w_i \in [c_i(\alpha_j), d_i(\alpha_j)]}} f(w_1,\ldots,w_n, x_1,\ldots,x_n \mid \alpha_j) \\ &= \max_{\forall w_i \in [c_i(\alpha_j), d_i(\alpha_j)]} f(w_1,\ldots,w_n, b_1,\ldots,b_n|\alpha_j) \\ &\triangleq \max_{\forall w_i \in [c_i(\alpha_j), d_i(\alpha_j)]} f_U(w_1,\ldots,w_n|\alpha_j) = f_U^*(\alpha_j) \end{aligned} \qquad (6)$$
where
$$f_U(w_1,\ldots,w_n \mid \alpha_j) \triangleq \frac{\sum_{i=1}^n w_i(\alpha_j) b_i(\alpha_j)}{\sum_{i=1}^n w_i(\alpha_j)}. \qquad (7)$$
Note that $f_L(w_1,\ldots,w_n \mid \alpha_j)$ and $f_U(w_1,\ldots,w_n \mid \alpha_j)$, $\forall w_i(\alpha_j) \in [c_i(\alpha_j), d_i(\alpha_j)]$ are interval sets and f_L^* and f_U^* are the left-end and right-end values of $f_L(w_1,\ldots,w_n \mid \alpha_j)$ and $f_U(w_1,\ldots,w_n \mid \alpha_j)$, respectively.

Based on some theorems and a lemma, Liou and Wang developed the first *iterative* algorithms for computing $f_L^*(\alpha_j)$ and

[3]In obtaining the number 2^{2n}, the fact that each of the nw_i and nx_i can take on its respective two end-point values was used.

[4]In some papers (e.g., [9]) the term "evaluation" is used instead of "iteration." Note also that number of arithmetic operations, sorts, and searches are not considered herein.

$f_U^*(\alpha_j)$. The maximum number of iterations for the two IFWA algorithms, as reported in [9], is $n(n+1)+2$ for each α-cut, or at most $m[n(n+1)+2]$ iterations; hence, the IFWA algorithms have significantly fewer iterations than the $m2^{2n}$ iterations of Dong and Wong's FWA.

C. On Lee and Park's Efficient FWA (EFWA) Algorithm

Lee and Park [8] also provided iterative search algorithms for computing $f_L^*(\alpha_j)$ and $f_U^*(\alpha_j)$. Their algorithms build upon the observations of Liou and Wang that were discussed above, as summarized in (4) and (6). A statement of the EFWA is provided in Section IV because it will be easier to do this after the KM algorithms have been stated. The maximum number of iterations for the two EFWA algorithms, as reported in [8], is $2n \ln n$ for each α-cut, or at most $m(2n \ln n)$ iterations, which is significantly less than the at most mn^2 iterations of the IFWA algorithms.

D. Comments

These are not the only methods that have been published. Others are in [3]–[5]. Except for [3], all of these methods are based on α-cuts and an α-cut decomposition theorem. Of the existing algorithms that use α-cuts, the EFWA represents the most computationally efficient one to compute the FWA (prior to the KM algorithms approach for computing the FWA) and will be compared with the KM algorithm approach in Section V.

III. The KM Algorithms

A. Derivation

Regardless of whether $f_L^*(\alpha_j)$ in (4) or $f_U^*(\alpha_j)$ in (6) are computed, it is necessary to minimize or maximize the function[5]

$$f(w_1,\ldots,w_n) \equiv \frac{\sum_{i=1}^n x_i w_i}{\sum_{i=1}^n w_i} \tag{8}$$

where for the purposes of this section the notation is simplified by not showing the explicit dependencies of the x_i and w_i on the α-cut, α_j. Differentiating $f(w_1,\ldots,w_n)$ with respect to w_k, observe that

$$\frac{\partial f(w_1,\ldots,w_n)}{\partial w_k} = \frac{x_k - f(w_1,\ldots,w_n)}{\sum_{i=1}^n w_i} \quad k=1,\ldots,n. \tag{9}$$

As noted by Karnik and Mendel [6], equating $\partial f/\partial w_k$ to zero does not give us any information about the value of w_k that optimizes $f(w_1,\ldots,w_n)$, i.e.,

$$\begin{aligned} f(w_1,\ldots,w_n) = x_k &\Rightarrow \frac{\sum_{i=1}^n x_i w_i}{\sum_{i=1}^n w_i} = x_k \\ &\Rightarrow \frac{\sum_{i\neq k}^n x_i w_i}{\sum_{i\neq k}^n w_i} = x_k. \end{aligned} \tag{10}$$

[5]In order to compute $f_L^*(\alpha_j)$, set $x_i = a_i$ in (8), so that (8) reduces to (5), and to compute $f_U^*(\alpha_j)$ set $x_i = b_i$, so that (8) reduces to (7).

Observe that w_k no longer appears in the final expression in (10), so that the direct calculus approach does not work. Returning to (9), because $\sum_{i=1}^n w_i > 0$, it is true that

$$\frac{\partial f(w_1,\ldots,w_n)}{\partial w_k} \begin{cases} \geq 0 & \text{if } x_k \geq f(w_1,\ldots,w_n) \\ < 0 & \text{if } x_k < f(w_1,\ldots,w_n) \end{cases}. \tag{11}$$

This equation gives us the direction in which w_k should be changed in order to increase or decrease $f(w_1,\ldots,w_n)$, i.e.,

$$\begin{aligned} &\text{If } x_k > f(w_1,\ldots,w_n) \\ &\quad \begin{cases} f(w_1,\ldots,w_n) \text{ increases as } w_k \text{ increases} \\ f(w_1,\ldots,w_n) \text{ decreases as } w_k \text{ decreases} \end{cases} \\ &\text{If } x_k < f(w_1,\ldots,w_n) \\ &\quad \begin{cases} f(w_1,\ldots,w_n) \text{ increases as } w_k \text{ decreases} \\ f(w_1,\ldots,w_n) \text{ decreases as } w_k \text{ increases} \end{cases}. \end{aligned} \tag{12}$$

Because $w_k \in [c_k, d_k]$, the maximum value w_k can attain is d_k and the minimum value it can attain is c_k. Equation (12) therefore implies that $f(w_1,\ldots,w_n)$ attains its *minimum value* f_L^* if 1) for those values of k for which[6] $x_k < f(w_1,\ldots,w_n)$, set $w_k = d_k$ and 2) for those values of k for which $x_k > f(w_1,\ldots,w_n)$, set $w_k = c_k$. Similarly, deduce from (12) that $f(w_1,\ldots,w_n)$ attains its *maximum value* f_U^* if 1) for those values of k for which $x_k < f(w_1,\ldots,w_n)$, set $w_k = c_k$ and 2) for those values of k for which $x_k > f(w_1,\ldots,w_n)$, set $w_k = d_k$. Consequently, to compute f_L^* or $f_U^* w_k$ *switches only one time* between d_k and c_k, or between c_k and d_k, respectively. The KM algorithms (described below) locate each switch point and, in general, the switch point for f_U^*, k_U, is different from the switch point for f_L^*, k_L.

Putting all of these facts together, f_L^* in (4) and f_U^* in (6) can be expressed as (see footnote [5])

$$f_L^* = \min_{\forall w_i \in [c_i,d_i]} \frac{\sum_{i=1}^n a_i w_i}{\sum_{i=1}^n w_i} = \frac{\sum_{i=1}^{k_L} a_i d_i + \sum_{i=k_L+1}^n a_i c_i}{\sum_{i=1}^{k_L} d_i + \sum_{i=k_L+1}^n c_i} \tag{13}$$

$$f_U^* = \max_{\forall w_i \in [c_i,d_i]} \frac{\sum_{i=1}^n b_i w_i}{\sum_{i=1}^n w_i} = \frac{\sum_{i=1}^{k_U} b_i c_i + \sum_{i=k_U+1}^n b_i d_i}{\sum_{i=1}^{k_U} c_i + \sum_{i=k_U+1}^n d_i} \tag{14}$$

where k_U and k_L are computed using the KM algorithms, which are described next.

B. KM Algorithms

The algorithms for finding k_U and f_U^*, and k_L and f_L^*, are very similar [6], [10].

KM algorithm for k_U and f_U^*:

0) *Rearrangement for the f_U^* calculations:* In (8), x_i are replaced by the b_i, where b_i are arranged in ascending order, i.e., $b_1 \leq b_2 \leq \cdots \leq b_n$. Associate the respective w_i with its (possibly) relabeled b_i. In the remaining steps it is assumed that the notation used in (8) is for the reordered (if necessary) w_i and b_i.

[6]These results are stated in the order $x_k < f(w_1,\ldots,w_n)$ and then $x_k > f(w_1,\ldots,w_n)$ because x_k in (8) are naturally ordered, i.e., $x_1 \leq x_2 \leq \cdots \leq x_n$. If x_k are not so ordered, then they must first be so reordered.

1) Initialize w_i for $i = 1, 2, \ldots, n$ and then compute

$$f'_U = \frac{\sum_{i=1}^n b_i w_i}{\sum_{i=1}^n w_i}. \quad (15)$$

Two ways for initializing w_i are:

a) $w_i = (c_i + d_i)/2$ for $i = 1, 2, \ldots, n$;

b) $w_i = c_i, i \le \lfloor (n+1)/2 \rfloor$ and $w_i = d_i, i > \lfloor (n+1)/2 \rfloor$, where $\lfloor \bullet \rfloor$ denotes the first integer equal to or smaller than $\bullet$.

2) Find $k (1 \le k \le n-1)$ such that $b_k \le f'_U < b_{k+1}$.

3) Set $w_i = c_i$ for $i \le k$ and $w_i = d_i$ for $i \ge k+1$ and compute

$$f''_U(b_k) \equiv f''_U(k) = \frac{\sum_{i=1}^k b_i c_i + \sum_{i=k+1}^n b_i d_i}{\sum_{i=1}^k c_i + \sum_{i=k+1}^n d_i}. \quad (16)$$

4) Check if $f''_U(k) = f'_U$. If yes, then stop. $f''_U(k)$ is the maximum value of $\sum_{i=1}^n b_i w_i / \sum_{i=1}^n w_i$ which equals f^*_U, and k equals the switch point k_U If no, go to step 5)

5) Set f'_U equal to $f''_U(k)$ and go to step 2).

KM algorithm for k_L and f^*_L:

Rearrangement for the f^*_L calculations: In (8), x_i are replaced by the a_i, where a_i are arranged in ascending order, i.e., $a_1 \le a_2 \le \cdots \le a_n$. Associate the respective w_i with its (possibly) relabeled a_i. In the remaining steps it is assumed that the notation used in (8) is for the reordered (if necessary) w_i and a_i.

1) Initialize w_i [in either of the two ways listed after (15)] and then compute

$$f'_L = \frac{\sum_{i=1}^n a_i w_i}{\sum_{i=1}^n w_i}. \quad (17)$$

2) Find $k (1 \le k \le n-1)$ such that $a_k \le f'_L < a_{k+1}$.

3) Set $w_i = d_i$ for $i \le k$ and $w_i = c_i$ for $i \ge k+1$, and compute

$$f''_L(a_k) \equiv f''_L(k) = \frac{\sum_{i=1}^k a_i d_i + \sum_{i=k+1}^n a_i c_i}{\sum_{i=1}^k d_i + \sum_{i=k+1}^n c_i}. \quad (18)$$

4) Check if $f''_L(k) = f'_L$. If yes, then stop. $f''_L(k)$ is the minimum value of $\sum_{i=1}^n a_i w_i / \sum_{i=1}^n w_i$ and equals f^*_L, and k is the switch point k_L. If no, go to step 5).

5) Set f'_L equal to $f''_L(k)$ and go to step 2).

Computing the FWA: In order to compute the FWA using the KM algorithms, steps 1), 2), 5), and 6) in Dong and Wong's FWA algorithm remain unchanged (see Section II-A). Steps 3) and 4) are replaced by:

3. Compute $\max_k y_k(\alpha_j) = f^*_U(\alpha_j)$ using the KM algorithm for f^*_U.

4. Compute $\min_k y_k(\alpha_j) = f^*_L(\alpha_j)$ using the KM algorithm for f^*_L.

Comments:

1) Recently, we came across the very interesting paper by Auephanwiriyakul and Keller [1], in which they have proposed using the KM algorithms and α-cuts for computing cluster centers, which, as they point out, is an FWA. Consequently, we acknowledge that they are the first to do this. Although they refer to Dong and Wong's [2] original FWA algorithm, which as pointed out above is terribly inefficient, they do not mention the other algorithms, most notably the EFWA [8].

2) It is easy to modify the KM algorithms so that their stopping rules occur in step 2) instead of in step 4). Extensive simulations have shown that doing this does not reduce the number of iterations required by the algorithms; hence, the algorithms have been presented above as they have appeared previously in the literature.

3) Fig. 1 provides a graphical interpretation of the KM algorithm that computes f^*_L. Justification for the shape and location of $f''_L(k)$ (in relation to $y = a_k$) is provided in Appendix A. The large dots are plots of $f''_L(k)$ for $k = 1, \ldots, 9$; note that k is associated with the subscript of a_k. The 45° line $y = a_k$ is shown because of the computations in step 2) of the KM algorithm. f'_L has been chosen according to method (b) stated below (15). Because $n = 9$, $\lfloor (n+1)/2 \rfloor = 5$, which is why f'_L is located at a_5. After f'_L has been computed, then a horizontal line is drawn until it intersects $y = a_k$. By virtue of step 2) of the algorithm, $a_4 \le f'_L < a_5$, and the 45° line $y = a_k$ is slid down until a_4 is reached, at which point $f''_L(4)$ is computed. This is the vertical line from a_4 that intersects a large dot. Because $f''_L(4) \ne f'_L$, the algorithm then goes through another complete iteration before it stops, at which time $f''_L(4)$ has been determined to be f^*_L. For this example, the KM algorithm converges in two iterations.

C. *Properties of the KM Algorithms*

Auephanwiriyakul and Keller [1] provide an equation for the computational complexity per α-cut for each of the KM algorithms and then for the two algorithms. The maximum number of iterations for each KM algorithm is n [6] for each α-cut, or at most mn iterations per KM algorithm. Below it is explained why this upper bound, which is extremely conservative, is incorrect as a greater upper bound and can be vastly improved upon by an approximate least upper bound.

Recently, the following have been proved [11].

1) The KM algorithms are monotonically convergent.

2) Within the quadratic domain of convergence, the KM algorithms are superexponentially convergent. Let j denote an iteration counter ($j = 1, 2, \ldots$). A formula for j as a function of prescribed bits of accuracy ε is derived in [11]. Because this formula is used in Section V to support our simulations, it is stated next for f^*_L. A comparable formula exists for f^*_U.

Let $f'_L(0)$ denote the first value of f'_L as given in (17) and[7]

$$\delta \equiv \frac{f''_L(1) - f^*_L}{f'_L(0) - f^*_L} \le 1. \quad (19)$$

[7]$\delta \le 1$ is a result of the monotonic convergence of the KM algorithms.

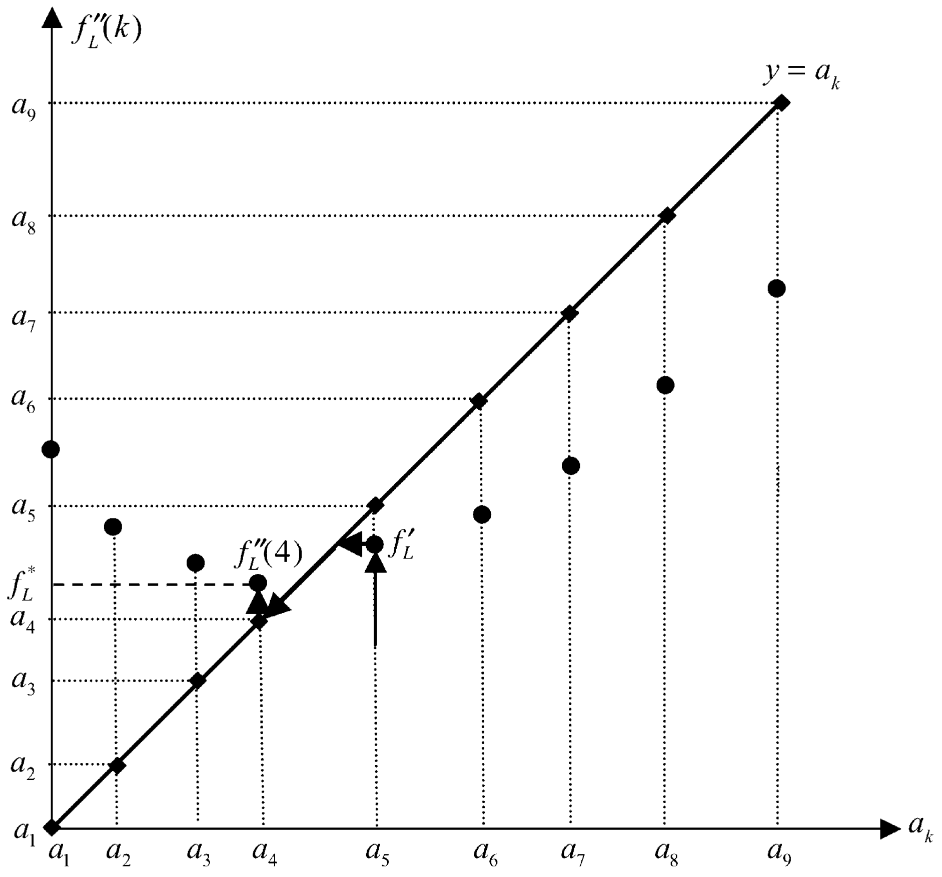


Fig. 1. A graphical interpretation of the KM algorithm that computes f^*_L. The solid line shown for $y = a_k$ only has values at $a_1, a_2, \ldots, a_9$. The large dots are $f''_L(k), k = 1, \ldots, 9$.

Superexponential convergence[8] of the KM algorithm occurs to within ε bits of accuracy when

$$|f''_L(j) - f''_L(j-1)| \leq \varepsilon \tag{20}$$

which, as proven in [11], is satisfied by the first integer $j_\varepsilon \geq 2$ for which

$$\left|\delta^{2^j} - \delta^{2^{(j-1)}}\right| \leq \frac{\varepsilon\delta}{f'_L(0) - f^*_L}. \tag{21}$$

In general, this equation has no closed-form solution for j; however, for small values of δ, the term δ^{2^j} (which is much smaller than the term $\delta^{2^{(j-1)}}$) can be dropped, and the resulting equation can be solved for j_ε as

$$j' = 1 + \frac{1}{\ln 2} \times \ln\left\{\frac{1}{\ln\delta} \times \ln\left[\frac{\varepsilon\delta}{(f'_L(0) - f^*_L)}\right]\right\} \tag{22}$$
$$= 1 + \frac{1}{\ln 2} \times \ln\left\{1 + \frac{1}{\ln\delta} \times \ln\left[\frac{\varepsilon}{(f'_L(0) - f^*_L)}\right]\right\}$$
$$j_\varepsilon = \text{first integer larger than } j'. \tag{23}$$

3) The use of (22) and (23) requires knowing the answer f^*_L as well as $f''_L(1)$, and that one is in the quadratic domain of convergence. The latter also depends on knowing f^*_L, whereas $f''_L(1)$ is only available after the first iteration of the KM algorithm; hence, the use of these equations is limited. However, these equations can be used after the fact (as done in Section V) to confirm the very rapid convergence of the KM algorithms.

Our many simulations have revealed that, for two significant figures of accuracy, the convergence of the KM algorithms occurs in two to six iterations regardless of n. Discussions about this are provided in Section V.

IV. EFWA Algorithms

In Section V convergence results for the KM algorithms are compared with the EFWA algorithms. Based on the derivation of the KM algorithms, it is relatively easy to understand the following.

EFWA Algorithm for f^*_U:

0) *Rearrangement for the f^*_U calculations*: Same as Step 0) in the KM Algorithm.

1) Initialize first $= 1$ and last $= n$.

2) Let $k = \lfloor (1/2)(\text{first}+\text{last}) \rfloor$ and compute $f''_U(k)$ in (16).

3) If $b_k \leq f''_U(k) < b_{k+1}$, stop and set $f^*_U = f''_U(k)$ and $k = k_U$; otherwise, go to step 4).

4) If $f''_U(k) \geq b_k$, set first $= k + 1$; otherwise, set last $= k$. Go to step 2).

EFWA Algorithm for f^*_L:

0) *Rearrangement for the f^*_L calculations:* Same as step 0) in the KM algorithm.

[8]Why this is "superexponential convergence" rather than "convergence" is explained fully in [11]. It is evidenced by the appearance of δ^{2^j} in (21), the amplitude of whose logarithm is not linear but is concave upwards, which is indicative of a superexponential convergence factor.

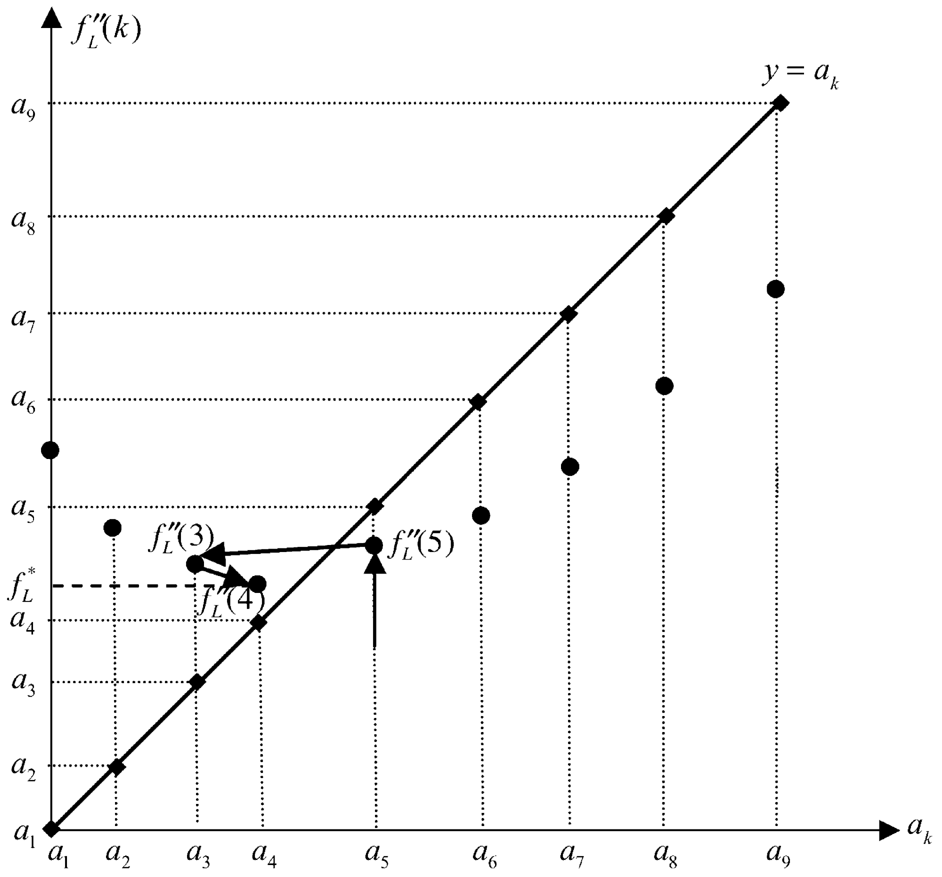
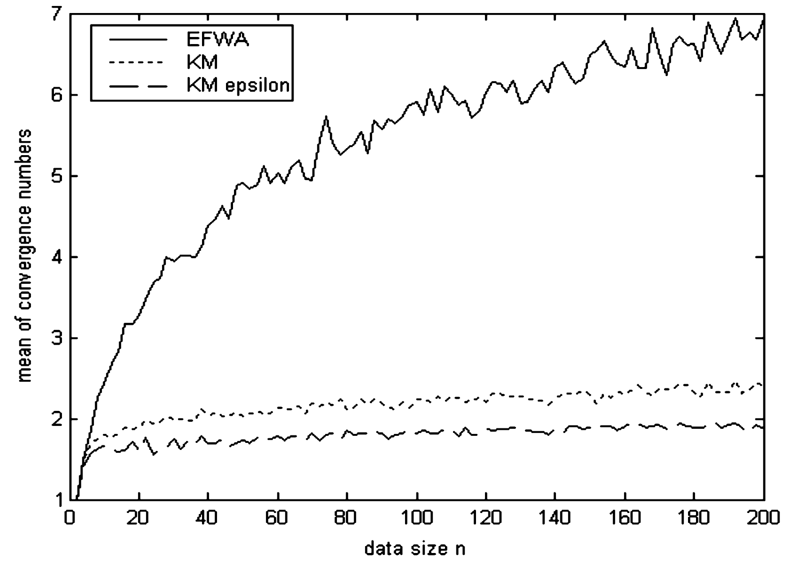


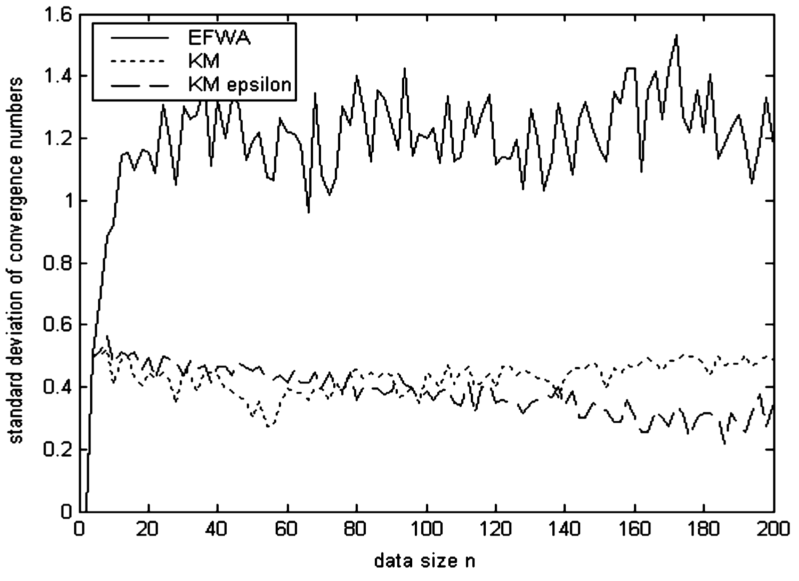
Fig. 2. A graphical interpretation of the EFWA algorithm that computes f^*_L. The solid line shown for $y = a_k$ only has values at $a_1, a_2, \ldots, a_9$. The large dots are $f''_L(k), k = 1, \ldots, 9$.

TABLE I
MEAN AND ± ONE STANDARD DEVIATION OF CONVERGENCE NUMBERS FOR THREE FWA ALGORITHMS AND FIVE PARAMETER DISTRIBUTIONS

Algorithm	Data Dimension n for *Uniformly Distributed* Parameters					
	200	500	1000	5000	10000	20000
EFWA	6.72±1.30	7.96±1.43	9.00±1.32	11.53±1.18	12.44±1.27	13.32±1.60
KM	2.36±0.50	2.69±0.46	2.91±0.29	3.01±0.10	3.02±0.14	3.00±0.00
KM-epsilon	1.93±0.26	1.98±0.14	2.00±0.00	2.00±0.00	2.00±0.00	2.00±0.00
Algorithm	Data Dimension n for *Gaussian Distributed* Parameters					
	200	500	1000	5000	10000	20000
EFWA	6.97±1.60	8.07±1.23	9.07±1.27	11.53±1.26	12.32±1.29	13.36±1.20
KM	2.79±0.43	2.97±0.33	3.02±0.14	3.18±0.39	3.25±0.44	3.36±0.48
KM-epsilon	1.90±0.00	1.98±0.14	2.00±0.00	2.00±0.00	2.00±0.00	2.00±0.00
Algorithm	Data Dimension n for *Non-Central T Distributed* Parameters					
	200	500	1000	5000	10000	20000
EFWA	6.56±1.60	8.02±1.16	9.04±1.22	11.46±1.40	12.48±1.62	13.55±1.27
KM	2.82±0.39	3.00±0.20	3.08±0.27	3.18±0.39	3.38±0.49	3.63±0.49
KM-epsilon	1.91±0.29	1.99±0.10	2.00±0.00	2.00±0.00	2.00±0.00	2.00±0.00
Algorithm	Data Dimension n for *Exponentially Distributed* Parameters					
	200	500	1000	5000	10000	20000
EFWA	6.72±1.34	7.94±1.49	9.06±1.45	11.39±1.58	12.40±1.61	13.21±1.51
KM	2.16±0.44	2.13±0.39	2.28±0.47	2.68±0.47	2.88±0.33	2.96±0.20
KM-epsilon	1.01±0.10	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00
Algorithm	Data Dimension n for *Chi-Squared Distributed* Parameters					
	200	500	1000	5000	10000	20000
EFWA	6.64±1.61	7.90±1.72	8.95±1.60	11.33±1.80	12.28±1.58	13.17±1.78
KM	2.69±0.46	2.95±0.30	2.99±0.10	3.10±0.30	3.20±0.40	3.28±0.45
KM-epsilon	1.92±0.27	1.98±0.14	2.00±0.00	2.00±0.00	2.00±0.00	2.00±0.00
Algorithm	Data Dimension n for Parameters: Average Mean and Average ± One Standard Deviation Across the Five Distributions					
	200	500	1000	5000	10000	20000
EFWA	6.72±1.50	7.97±1.42	9.02±1.38	11.45±1.46	12.38±1.48	13.32±1.49
KM	2.56±0.52	2.75±0.48	2.86±0.41	3.03±0.40	3.15±0.42	3.25±0.45
KM-epsilon	1.73±0.42	1.79±0.41	1.80±0.40	1.80±0.40	1.80±0.40	1.80±0.40



(a)



(b)

Fig. 3. (a) Mean and (b) standard deviation of the convergence numbers when parameters are uniformly distributed.

1) Initialize first $= 1$ and last $= n$.

2) Let $k = \lfloor (1/2)(\text{first}+\text{last}) \rfloor$ and compute $f''_L(k)$ in (18).

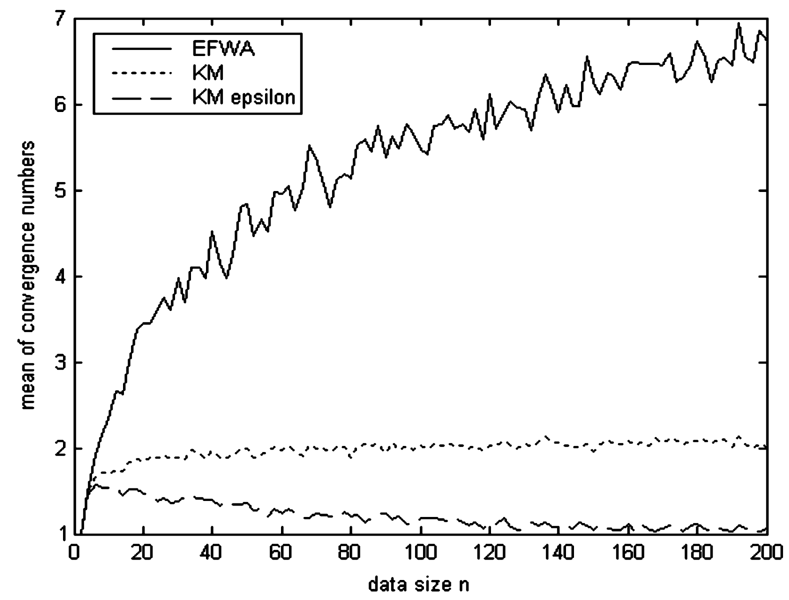
3) If $a_k \leq f'''_L(k) < a_{k+1}$, stop and set $f^*_L = f''_L(k)$; otherwise, go to step 4).

4) If $f''_L(k) \geq a_k$, set first $= k+1$; otherwise, set $last = k$. Go to step 2).

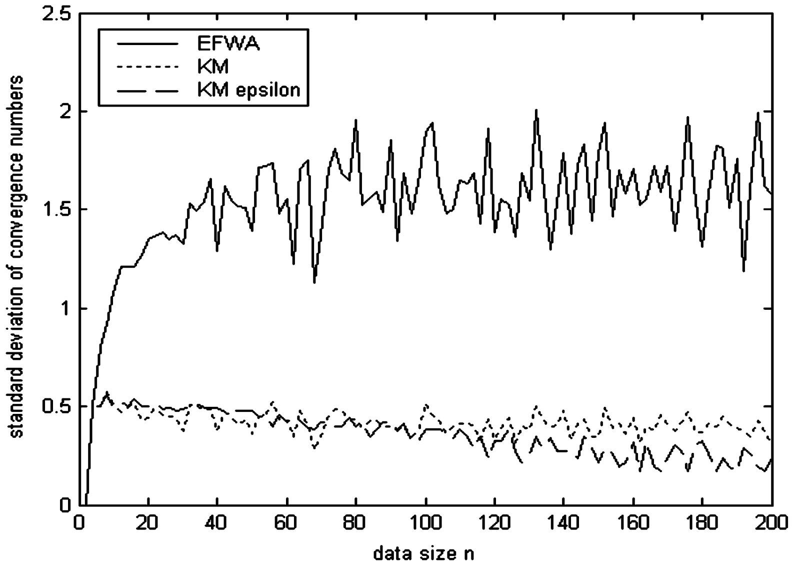
Comment: Fig. 2 provides a graphical interpretation of the EFWA algorithm that computes the same f^*_L as in Fig. 1. As in Fig. 1, the large dots are plots of $f''_L(k)$ for $k = 1, \ldots, 9$. The 45° line $y = a_k$ is shown because of the computations in step 3) of the EFWA algorithm. Since $n = 9$, $\lfloor (n+1)/2 \rfloor = 5$, so that $k = 5$, which is why $f''_L(5)$ has been located at a_5. For this value of $f''_L(5)$, the test in step 3) fails [i.e., $f''_L(5)$ is not between a_5 and a_6] and as a result of step 4) [i.e., $f''_L(5) < a_5$] set $last = 5$. Returning to step 2), $k = 3$ and then $f''_L(3)$ is computed, which is why there is a line with an arrow on it from $f''_L(5)$ to $f''_L(a_3) = f''_L(3)$. For this value of $f''_L(3)$, the test in step 3) again fails [i.e., $f''_L(3)$ is not between a_3 and a_4] and as a result of step 4) [i.e., $f''_L(3) > a_3$] set first $= 4$. Returning to step 2), $k = 4$ and then $f''_L(4)$ is computed, which is why there is an arrow on it from $f''_L(3)$ to $f''_L(a_4) = f''_L(4)$. For this value of $f''_L(4)$ the test in step 3) passes [i.e., $f''_L(4)$ is between a_4 and a_5], the algorithm stops, and $f^*_L = f''_L(4)$. This example, which is the same as that in Fig. 1, has taken the EFWA three iterations to converge, whereas it took the KM algorithm two iterations to converge.

V. Experimental Results

In this section, simulation results are presented in which convergence numbers are compared for the KM and EFWA algorithms. Because, for each α-cut, both algorithms consist of two parts—one for computing f^*_U and one for computing f^*_L—and each part is essentially the same, this simulation study was only performed for f^*_L.



(a)



(b)

Fig. 4. (a) Mean and (b) standard deviation of the convergence numbers when parameters are exponentially distributed.

There are many ways in which algorithm comparison experiments could be designed. In our simulations, *random designs* were used in which, for a fixed value of n (the number of terms in the FWA) and α, each a_i and the interval end-points for each w_i (i.e., c_i and d_i) $(i = 1, 2, \ldots, n)$ were chosen randomly according to a prescribed distribution. This was done for $n = 2, 4, 6, \ldots, 198, 200, 500, 1000, 5000, 10000, 20000$ and for uniform, Gaussian, noncentral T, exponential, and chi-squared distributions. The three random sequences a_i, c_i, and d_i were generated using the same distribution but with different seed numbers and for all the distributions $a_i \in [0, 10], c_i \in [0, 1]$, and $d_i \in [0, 1]$. The a_i were then sorted in increasing order, and the corresponding c_i and d_i always satisfied the requirements that $c_i \geq 0, d_i \geq 0$ and $d_i \geq c_i$. For each value of n and each distribution, 100 Monte Carlo simulations were performed and the numbers of iterations for both the KM and EFWA to converge to f_L^* were recorded. The mean and standard deviation of the convergence numbers were then computed.

Results for $n = 200, 500, 1000, 5000, 10\,000, 20\,000$ are summarized in Table I. Because the results from these experiments looked very similar regardless of a specific distribution, the results for $n = 2, 4, \ldots, 198, 200$ are only presented in Figs. 3 and 4 for the uniform and exponential distributions. The parameters used for these distributions are the following.

- For the uniform distribution, a_i, c_i, and d_i were generated by Matlab function $rand()$ using different seeds, respectively. All a_i values were scaled by ten, so that $a_i \in [0, 10]$ for $\forall i$.

  For the exponential distribution, a_i, c_i, and d_i were generated by Matlab function *exprnd()* with parameter 50, 0.5, and 0.6 with different seeds, respectively, after which they were scaled so that $a_i \in [0, 10], c_i \in [0, 1]$, and $d_i \in [0, 1]$, e.g., the a_i were scaled to $10a_i / \max_{\forall i} a_i$.

Note that "convergence number" refers to the number of iterations it takes for an algorithm to STOP. Note, also, that "KM epsilon" refers to the KM algorithm that uses the approximate stopping rule (20) for which $\varepsilon = 0.01$. This stopping rule is different from the exact stopping rule given in step 4) of the KM algorithm.

Observe from Figs. 3 and 4 the following.

1) The mean values of the convergence numbers for each algorithm are similar for both distributions, e.g., the mean convergence number for the KM algorithm is approximately two, and it is $\lceil (\ln n) - 1 \rceil$ for the EFWA algorithm, where $\lceil \bullet \rceil$ denotes the first integer equal to or larger than $\bullet$. The standard deviation of the convergence number is similar for both distributions for the KM algorithm, e.g., approximately 0.5, but it is distribution-dependent for the EFWA algorithm, e.g., approximately 1.2 for the uniform distribution and 1.5 for the exponential distribution. Consequently, for $n \leq 200$, the KM algorithm appears to be more robust to the distribution than is the EFWA algorithm.
   According to central limit theory, the distribution of average convergence numbers (for each n) is approximately normal with mean and standard deviation equal to the sample mean and standard deviation shown in Figs. 3 and 4. The sample mean + two times the sample standard deviation can be used to evaluate the convergence numbers with 97.5% confidence.[9] Consequently the KM algorithm converges in approximately three iterations, and the EFWA algorithm converges in approximately $\ln n + 1.4$ to $\ln n + 2$ iterations (to within 97.5% confidence); hence, in terms of number of iterations to converge, the KM algorithm is computationally more efficient than the EFWA algorithm.[10]
   When $n \geq 3$, the KM algorithm needs a smaller number of iterations than does the EFWA algorithm. When $n \leq 2$, both algorithms need the same number of iterations.
   The KM epsilon convergence numbers depicted in Figs. 3 and 4 agree with those obtained from (22) and (23), e.g., for data generated using a uniform distribution and $n = 200$, the KM algorithm leads to $f'_L(0) = 4.4077, f^*_L = 4.3783, f''_L(1) = 4.378361494$. Consequently, $\delta \equiv [f''_L(1) - f^*_L]/(f'_L(0) - f^*_L) = 2.09 \times 10^{-3}$ and $\ln \delta = -6.1706$. For $\varepsilon = 0.01, \ln\{\varepsilon/[f'_L(0) - f^*_L]\} = -1.0784$, so that
$$\begin{aligned} j &= 1 + \ln(1 + \ln\{\varepsilon/[f'_L(0) - f^*_L]\}/\ln\delta)/\ln 2 \\ &= 1 + \ln(1 + [-1.0784/-6.1706)/\ln 2 \\ &= 1.23 \end{aligned}$$
   and therefore $j_\varepsilon = 2$, which equals the true mean convergence number of $k = 2$.

Observe from Table I the following.

1) For each n, because simulations are independent of each other, convergence number can be considered as an independent identically distributed random variable; hence, according to central limit theory, the distribution of average convergence number (across the different distributions) is approximately normal with mean and standard deviation equal to the sample mean and standard deviation (across the different distributions). These statistics are provided in the last (bottom) section of the table.
2) The average convergence numbers (across the different distributions) of the KM-epsilon algorithm seems to be independent of the FWA parameter n, whereas the convergence number of the EFWA seems to be close to $\log n$ and the convergence number of the KM algorithm varies much less than does the EFWA with n.
3) For the EFWA, the theoretical convergence number for the worst case is $\log n$ (see Comment 2 in Section IV), which agrees with observations.
4) For the KM algorithm, the upper bound of its convergence number is a small number that is slightly dependent on n (to within 97.5% confidence). For $n = 200, 500, 1000, 5000, 10\,000$, that number (which is the integer just larger than sample mean + 2 sample standard deviations) is four, whereas for $n = 20\,000$ that number is five. This result suggests very strongly that the theoretical worst case bound for the convergence number, provided by Karnik and Mendel [6], can be greatly improved upon. At present, this is an open research issue. See Appendix C for a proof that the Karnik–Mendel worst case bound is too conservative.

VI. Conclusion

By connecting solutions from two different problems—the fuzzy weighted average and the generalized centroid of an IT2 FS—a new α-cut algorithm for solving the FWA problem has been obtained, one that converges monotonically and superexponentially fast. Simulations demonstrate that for each α-cut, convergence occurs (to within a 97.5% confidence interval) in three to five iterations. It appears, therefore, that the KM α-cut algorithms approach for computing the FWA requires the fewest iterations to date, and may therefore be the fastest available FWA algorithm to date.

If $2m$ parallel processors are available, the new KM α-cut algorithms can be used to compute the FWA in three to five iterations (to within 97.5% confidence) because the calculations of f^*_L and f^*_U are totally independent, and all $m\alpha$-cut calculations are also independent.

Research is under way to extend the FWA from type-1 fuzzy sets to interval type-2 fuzzy sets, something called the *linguistic weighted average* (LWA) [13]. The FWA plays a major role in computing the LWA.

Finally, finding a (theoretical) least upper bound for the KM algorithms is still an open research problem.

Appendix A
Properties of $f''_L(k)$ and Corresponding Proofs

For the following properties, it is assumed that a_i and b_i have been sorted in increasing order so that $a_1 \leq a_2 \leq \cdots \leq a_n$ and $b_1 \leq b_2 \leq \cdots \leq b_n$. Additionally, recall that the minimum of

[9]Because iterations must be positive, this is a one-sided confidence interval.

[10]The two algorithms use the same number of sorts, searches, and arithmetic operations; hence, their total numbers of computations can be compared solely on the basis of iterations to converge.

$f''_L(k)$ occurs at $k = k_L$; hence, $f^*_L = f''_L(k_L)$. k_L is the *switch point* for the minimum calculation.

Property 1 (Location of the Minimum): When $k = k_L$, then

$$a_{k_L} \le f''_L(k_L) = f^*_L < a_{k_L+1} \tag{A-1}$$

where

$$f''_L(k_L) = \frac{\sum_{i=1}^{k_L} a_i d_i + \sum_{i=k_L+1}^{n} a_i c_i}{\sum_{i=1}^{k_L} d_i + \sum_{i=k_L+1}^{n} c_i} \tag{A-2}$$

and [see (4)][11]

$$\begin{aligned} f^*_L &= \min_{\forall w_i \in [c_i, d_i]} f_L(w_1, \dots, w_n \,|\, \alpha_j) \\ &= \min_{k=1,\dots,n} (f''_L(k)) = f''_L(k_L). \end{aligned} \tag{A-3}$$

■

This property locates the minimum f^*_L of $f_L(w_1, \dots, w_n \,|\, \alpha_j)$ between two values of a_i. For example, in Fig. 1, the minimum of $f''_L(k) (k = 1, \dots, n)$ is $f''_L(4)$, so that $k_L = 4$. Observe that $a_4 \le f''_L(4) < a_5$.

Proof: That (A-1) is true follows directly from step 2) of the KM algorithm, when $k = k_L$; (A-2) follows from (18) when $k = k_L$; and (A-3) is obviously true because it is the problem that is solved by the KM algorithm.

Property 2 (Location of $f''_L(k)$ in Relation to the Line $y = a_k$): $f''_L(k)$ lies above the line $y = a_k$ when a_k is to the left of f^*_L and lies below the line $y = a_k$ when a_k is to the right of f^*_L, i.e.,

$$f''_L(k) \begin{cases} > a_k, & \text{when } a_k < f^*_L \\ < a_k, & \text{when } a_k > f^*_L \end{cases}. \tag{A-4}$$

■

This property provides interesting relations between a_k and $f''_L(k)$ on both sides of the minimum f^*_L, e.g., in Fig. 1, $f''_L(k)$ lies above a_k to the left of $f^*_L \equiv f''_L(4)$ and lies below a_k to the right of $f^*_L \equiv f''_L(4)$.

This property does not imply $f''_L(k)$ is monotonic on either side of f^*_L; but, it does demonstrate that $f''_L(k)$ cannot be above the line $y = a_k$ to the right of f^*_L. Property 3 is about the monotonicity of $f''_L(k)$; it will show, e.g., that $f''_L(k)$ is monotonically nondecreasing to the right of f^*_L. This could occur in two very different ways, namely, $f''_L(k)$ could be above the line $y = a_k$ or below that line to the right of f^*_L. Property 2 rules out the former.

Proof: Because f^*_L is the minimum of $f_L(w_1, \dots, w_n \,|\, \alpha_j)$, it must be true that

$$f''_L(k) \ge f^*_L, \text{for } k = 1, \dots, n. \tag{A-5}$$

Consequently, when $f^*_L > a_k$

$$f''_L(k) \ge f^*_L > a_k \tag{A-6}$$

which completes the proof of the first row of (A-4).

[11]Recall, also, that k is associated with the subscript of a_k.

From (A-2), observe that

$$f''_L(k_L) \left(\sum_{i=1}^{k_L} d_i + \sum_{i=k_L+1}^{n} c_i \right) = \sum_{i=1}^{k_L} a_i d_i + \sum_{i=k_L+1}^{n} a_i c_i. \tag{A-7}$$

Let $j \ge k_L + 1$ and add $\sum_{i=k_L+1}^{j} a_i(d_i - c_i)$ to both sides of (A-7), in which (via Property 1) $f''_L(k_L) = f^*_L$, to see that

$$\begin{aligned} & f^*_L \left(\sum_{i=1}^{k_L} d_i + \sum_{i=k_L+1}^{n} c_i \right) + \sum_{i=k_L+1}^{j} a_i(d_i - c_i) \\ &= \sum_{i=1}^{k_L} a_i d_i + \sum_{i=k_L+1}^{n} a_i c_i + \sum_{i=k_L+1}^{j} a_i(d_i - c_i) \\ &= \sum_{i=1}^{j} a_i d_i + \sum_{i=j+1}^{n} a_i c_i. \end{aligned} \tag{A-8}$$

Because $j \ge k_L + 1$ and (Property 1) $a_{k_L} \le f''_L(k_L) = f^*_L < a_{k_L+1}$, it must be true that

$$f^*_L < a_j. \tag{A-9}$$

Applying (A-9) and the fact that $d_i - c_i \ge 0$ to (A-8), observe that

$$\begin{aligned} & \sum_{i=1}^{j} a_i d_i + \sum_{i=j+1}^{n} a_i c_i \\ & < a_j \left(\sum_{i=1}^{k_L} d_i + \sum_{i=k_L+1}^{n} c_i \right) + a_j \sum_{i=k_L+1}^{j} (d_i - c_i) \\ & = a_j \left(\sum_{i=1}^{j} d_i + \sum_{i=j+1}^{n} c_i \right). \end{aligned} \tag{A-10}$$

Because $\sum_{i=1}^{j} d_i + \sum_{i=j+1}^{n} c_i > 0$, (A-10) can also be expressed as

$$a_j > \frac{\sum_{i=1}^{j} a_i d_i + \sum_{i=j+1}^{n} a_i c_i}{\left(\sum_{i=1}^{j} d_i + \sum_{i=j+1}^{n} c_i \right)} = f''_L(j). \tag{A-11}$$

Combining (A-9) and (A-11), conclude that

$$f''_L(j) < a_j, \qquad \text{when } a_j > f^*_L. \tag{A-12}$$

This completes the proof of the second row of (A-4).

Property 3 (Monotonicity of $f''_L(k)$): It is true that

$$\left. \begin{aligned} f''_L(k-1) \ge f''_L(k), & \quad \text{when } a_k < f^*_L \\ f''_L(k+1) \ge f''_L(k), & \quad \text{when } a_k > f^*_L \end{aligned} \right\}. \tag{A-13}$$

■

This property shows that $f''_L(k)$ is a monotonic function (but not a strictly monotonic function) on both sides of the minimum of the FWA. For example, in Fig. 1 $f''_L(k)$ monotonically decreases to the left of $f^*_L \equiv f''_L(4)$, whereas it monotonically increases to the right of $f^*_L \equiv f''_L(4)$.

Proof: To begin, observe that when $a_k < f_L^*$ Property 2 lets us conclude that $f_L''(k) > a_k$; hence, using (18) in this inequality, it is true that

$$\sum_{i=1}^{k} a_i d_i + \sum_{i=k+1}^{n} a_i c_i > a_k \left(\sum_{i=1}^{k} d_i + \sum_{i=k+1}^{n} c_i \right). \tag{A-14}$$

(A-14) is used below. Next, a formula for $f_L''(k) - f_L''(k-1)$ is obtained as shown in (A-15) at the bottom of the page. Note that going from the second to the third lines of (A-15) has involved a moderate amount of straightforward algebra during which many terms cancel each other. Because $d_k - c_k \geq 0, \sum_{i=1}^{k} d_i + \sum_{i=k+1}^{n} c_i > 0$, and (A-14) is true, the numerator of (A-15) ≤ 0; hence

$$f_L''(k) - f_L''(k-1) \leq 0 \tag{A-16}$$

which completes the proof of the first line of (A-13).

Next, consider the case when $a_k > f_L^*$, for which Property 2 lets us conclude that $f_L''(k) < a_k$; hence, using (18) in this inequality, it is true that

$$\sum_{i=1}^{k} a_i d_i + \sum_{i=k+1}^{n} a_i c_i < a_k \left(\sum_{i=1}^{k} d_i + \sum_{i=k+1}^{n} c_i \right). \tag{A-17}$$

Because the rest of the proof of the second line of (A-13) is so similar to that just given for the first line, its details are left to the reader.

Appendix B
Properties of $f_U''(k)$

Because properties of $f_U''(k)$ and their proofs are so similar to those for $f_L''(k)$, the properties for $f_U''(k)$ are summarized in this Appendix and their proofs are left to the readers. Recall, the maximum of $f_U''(k)$ occurs at $k = k_U$; hence, $f_U^* = f_U''(k_U)$. k_U is the *switch point* for the maximum.

Property 4 (Location of the Maximum): When $k = k_U$, then

$$b_{k_U} \leq f_U''(k_U) = f_U^* < b_{k_U+1} \tag{B-1}$$

where

$$f_U''(k_U) = \frac{\sum_{i=1}^{k_U} b_i c_i + \sum_{i=k_U+1}^{n} b_i d_i}{\sum_{i=1}^{k_U} c_i + \sum_{i=k_U+1}^{n} d_i} \tag{B-2}$$

and [see (6)]

$$\begin{aligned} f_U^* &= \max_{\forall w_i \in [c_i, d_i]} f_U(w_1, \ldots, w_n \mid \alpha_j) \\ &= \max_{k=1,\ldots,n} (f_U''(k)) = f_U''(k_U). \end{aligned} \tag{B-3}$$

■

Property 5 (Location of $f_U''(k)$ in Relation to the Line $y = b_k$): $f_U''(k)$ lies above the line $y = b_k$ when b_k is to the left of f_U^* and lies below the line $y = b_k$ when b_k is to the right of f_U^*, i.e.,

$$f_U''(k) \begin{cases} > b_k, \text{when } b_k < f_U^* \\ < b_k, \text{when } b_k > f_U^* \end{cases}. \tag{B-4}$$

■

Property 6 (Monotonicity of $f_U''(k)$): It is true that

$$\left. \begin{array}{ll} f_U''(k-1) \leq f_U''(k), & \text{when } b_k < f_U^* \\ f_U''(k+1) \leq f_U''(k), & \text{when } b_k > f_U^* \end{array} \right\}. \tag{B-5}$$

■

Appendix C
Disproof of the Karnik–Mendel Upper Bound

As stated just after (15), different initializations can be used for a KM algorithm. When the same initialization as used for the EFWA is chosen, the initialization for finding k_L and f_L^* is

$$w_i = \begin{cases} d_i, & i \leq \lfloor (n+1)/2 \rfloor \\ c_i, & \text{otherwise} \end{cases}. \tag{C-1}$$

Making use of (A-1), if $a_{\lfloor (n+1)/2 \rfloor} \leq f_L''(\lfloor (n+1)/2 \rfloor) < a_{\lfloor (n+1)/2 \rfloor + 1}$, then $f_L''(\lfloor (n+1)/2 \rfloor)$ is the minimum, in which case only one iteration of the KM algorithm is needed to find the minimum. If, however, $f_L''(\lfloor (n+1)/2 \rfloor)$ is not the minimum value, then $f_L''(\lfloor (n+1)/2 \rfloor)$ must obviously satisfy one of the following inequalities:

$$f_L''(\lfloor (n+1)/2 \rfloor) < a_{\lfloor (n+1)/2 \rfloor} \tag{C-2}$$

$$f_L''(\lfloor (n+1)/2 \rfloor) \geq a_{\lfloor (n+1)/2 \rfloor + 1} \geq a_{\lfloor (n+1)/2 \rfloor} \tag{C-3}$$

where the second part of (C-3) follows from the ascending order of the a_i. From (A-4) in Property 2, when $f_L''(k)$ is not the minimum, it follows that

$$f_L^* \begin{cases} > a_k, & \text{when } f_L''(k) > a_k \\ < a_k, & \text{when } f_L''(k) < a_k \end{cases}. \tag{C-4}$$

$$\begin{aligned} f_L''(k) - f_L''(k-1) &= \frac{\sum_{i=1}^{k} a_i d_i + \sum_{i=k+1}^{n} a_i c_i}{\sum_{i=1}^{k} d_i + \sum_{i=k+1}^{n} c_i} - \frac{\sum_{i=1}^{k-1} a_i d_i + \sum_{i=k}^{n} a_i c_i}{\sum_{i=1}^{k-1} d_i + \sum_{i=k}^{n} c_i} \\ &= \frac{\sum_{i=1}^{k} a_i d_i + \sum_{i=k+1}^{n} a_i c_i}{\sum_{i=1}^{k} d_i + \sum_{i=k+1}^{n} c_i} - \frac{\sum_{i=1}^{k} a_i d_i + \sum_{i=k+1}^{n} a_i c_i - a_k(d_k - c_k)}{\sum_{i=1}^{k} d_i + \sum_{i=k+1}^{n} c_i - (d_k - c_k)} \\ &= \frac{(d_k - c_k)\left[a_k\left(\sum_{i=1}^{k} d_i + \sum_{i=k+1}^{n} c_i\right) - \left(\sum_{i=1}^{k} a_i d_i + \sum_{i=k+1}^{n} a_i c_i\right)\right]}{\left(\sum_{i=1}^{k} d_i + \sum_{i=k+1}^{n} c_i\right)\left[\sum_{i=1}^{k} d_i + \sum_{i=k+1}^{n} c_i - (d_k - c_k)\right]} \end{aligned} \tag{A-15}$$

When (C-2) is satisfied, observe from the second line of (C-4) that $f_L^* < a_{\lfloor (n+1)/2 \rfloor}$, and so the minimum must be located to the left of $a_{\lfloor (n+1)/2 \rfloor}$. According to step 3) of the KM algorithm, and the monotonicity of $f_L''(k)$, the value of k in (18) can only be to the left of $\lfloor (n+1)/2 \rfloor$. Similarly, when (C-3) is satisfied, from the first line of (C-4), observe that $f_L^* > a_{\lfloor (n+1)/2 \rfloor}$ and so the minimum must be located to the right of $a_{\lfloor (n+1)/2 \rfloor}$. According to step 3) of the KM algorithm and the monotonicity of $f_L''(k)$, the value of k in (18) can only be to the right of $\lfloor (n+1)/2 \rfloor$.

Consequently, the upper bound on the number of iterations to find k_L and f_L^* should be no larger than $\lfloor (n+1)/2 \rfloor$; hence, the Karnik–Mendel upper bound of n on the number of iterations to find k_L and f_L^* is too conservative.

Finally, when Karnik and Mendel [6] developed their upper bound, properties of the GC comparable to those in Appendix A were not known for the GC, and, they did not connect the GC to the FWA.

Acknowledgment

The authors wish to thank all of the reviewers, especially the one who suggested "big-Oh" values for the KM algorithms be provided.

References

[1] S. Auephanwiriyakul and J. M. Keller, "Analysis and efficient implementation of a linguistic fuzzy c-means," *IEEE Trans. Fuzzy Syst.*, vol. 10, pp. 563–582, Oct. 2002.

[2] W. M. Dong and F. S. Wong, "Fuzzy weighted averages and implementation of the extension principle," *Fuzzy Sets Syst.*, vol. 21, pp. 183–199, 1987.

[3] D. Dubois, H. Fargier, and J. Fortin, "A generalized vertex method for computing with fuzzy intervals," in *Proc. FUZZ IEEE 2004*, Budapest, Hungary, 2004, pp. 541–546.

[4] Y.-Y. Guh, C.-C. Hon, and E. S. Lee, "Fuzzy weighted average: The linear programming approach via Charnes and Cooper's rule," *Fuzzy Sets Syst.*, vol. 117, pp. 157–160, 2001.

[5] Y.-Y. Guh, C.-C. Hon, K.-M. Wang, and E. S. Lee, "Fuzzy weighted average: A max-min paired elimination method," *Comput. Math. Applicat.*, vol. 32, pp. 115–123, 1996.

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

[7] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Upper Saddle River, NJ: Prentice-Hall, 1995.

[8] D. H. Lee and D. Park, "An efficient algorithm for fuzzy weighted average," *Fuzzy Sets Syst.*, vol. 87, pp. 39–45, 1997.

[9] T.-S. Liou and M.-J. J. Wang, "Fuzzy weighted average: An improved algorithm," *Fuzzy Sets Syst.*, vol. 49, pp. 307–315, 1992.

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

[11] 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. Fuzzy Syst.*, vol. 15, no. , pp. 309–322, Apr. 2007.

[12] J. M. Mendel, "Fuzzy sets for words: A new beginning," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, St. Louis, MO, May 2003, pp. 37–42.

[13] D. Wu and J. M. Mendel, "The Linguistic Weighted Average," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Vancouver, BC, Canada, Jul. 2006.

Feilong Liu (S'06) received the B.S degree in automation theory from Chinese Northeastern University, China, in 1995 and the M.S degree in engineering from South China University of Technology, China, in 2000. He is currently pursuing the Ph.D. degree in electrical engineering at the University of Southern California, Los Angeles.

His current research interests include type-2 fuzzy logic theory, artificial intelligence, signal processing, pattern recognition, and applying these technologies to smart oil field problems such as water flooding.

Jerry M. Mendel (S'59–M'61–SM'72–F'78–LF'04) received the Ph.D. degree in electrical engineering from the Polytechnic Institute of Brooklyn, New York.

Currently, he is a Professor of electrical engineering at the University of Southern California, Los Angeles, where he has been since 1974. He has published more than 470 technical papers and is author and/or editor of eight books, including *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions* (Englewood Cliffs, NJ: Prentice-Hall, 2001). His present research interests include type-2 fuzzy logic systems and their applications to a wide range of problems, including target classification, smart oil field technology, and computing with words.

Dr. Mendel is a Distinguished Member of the IEEE Control Systems Society. He was President of the IEEE Control Systems Society in 1986 and is presently Chairman of the Fuzzy Systems Technical Committee and a member of the Administrative Committee of the IEEE Computational Intelligence Society. Among the awards he has received are the 1983 Best Transactions Paper Award from the IEEE Geoscience and Remote Sensing Society, the 1992 Signal Processing Society Paper Award, the 2002 IEEE Transactions on Fuzzy Systems Outstanding Paper Award, a 1984 IEEE Centennial Medal, and an IEEE Third Millenium Medal.