A Direct Approach for Determining the Switch Points in the Karnik-Mendel Algorithm

Chao Chen, *Student Member, IEEE,* Robert John, *Senior Member, IEEE,* Jamie Twycross and Jonathan M. Garibaldi, *Member, IEEE*

***Abstract*—The Karnik-Mendel algorithm is used to compute the centroid of interval type-2 fuzzy sets, determining the switch points needed for the lower and upper bounds of the centroid, through an iterative process. It is commonly acknowledged that there is no closed-form solution for determining such switch points. Many enhanced algorithms have been proposed to improve the computational efficiency of the Karnik-Mendel algorithm. However, all of these algorithms are still based on iterative procedures. In this paper, a direct approach based on derivatives for determining the switch points without multiple iterations has been proposed, together with mathematical proof that these switch points are correctly determining the lower and upper bounds of the centroid. Experimental simulations show that the direct approach obtains the same switch points, but is more computationally efficient than any of the existing (iterative) algorithms. Thus, we propose that this algorithm should be used in any application of interval type-2 fuzzy sets in which the centroid is required.**

***Index Terms*—Karnik-Mendel algorithm, centroid, interval type-2, fuzzy sets, iterative, closed-form, direct approach.**

I. Introduction

THE Karnik-Mendel (KM) algorithm was the first algorithm proposed to determine the switch points when computing the centroids of interval type-2 (IT2) fuzzy sets [1]. With its fast convergence, it is still the most widely used algorithm for computing the switch points [2, 3, 4, 5, 6]. It was originally shown that the maximum number of iterations of the KM algorithm is N, which is the number of discrete points in the universe of discourse for an IT2 fuzzy set. This was believed to be extremely conservative and it was subsequently proved by Liu [7] that the maximum number of iterations of the KM algorithm is $(N+1)/2$. A much smaller number $(N+2)/4$ was given in [8] as the maximum number of iterations, without proof. Though these numbers are much smaller than N, they are still believed to be conservative [3]. Many studies by simulations show that the KM algorithm converges in from two to six iterations, regardless of N [9].

Many attempts have been made to improve the efficiency of the KM algorithm. The enhanced KM (EKM) algorithm introduces a better initialisation, which "on average ... can save about two iterations" [9]. Also, some algorithmic improvements to simplify computations were introduced to reduce the computational cost for each iteration. An improved iterative algorithm with stopping condition (IASC) was first proposed in [10] and further refined in [11]. Further improvements to the IASC have been made in the enhanced IASC (EIASC) in [12]. Both the IASC and EIASC algorithms have been reported to be superior to the KM and EKM algorithms when N is small (e.g. $N \leqslant 100$). However, their computational costs increase rapidly as N increases since many possible switch points have to be evaluated before finding the correct ones [3].

Rather than calculating the exact centroids, closed-form solutions, which are much more efficient than iterative algorithms, have been provided for approximations. For example, the Nie-Tan (NT) method as given in [13]. It has been demonstrated that the NT method can give a very good approximation to the KM algorithm. As an extension of the NT method, a better approximation with Taylor-series is provided in [14]. Closed-form formulae for calculating the centroids of a general type-2 fuzzy set are proposed in [15], where linear connections between the centroid endpoints of any α plane and that of $\alpha = 0$ and $\alpha = 1$ planes have been introduced. However, the calculations of the centroid end points of the $\alpha = 0$ and $\alpha = 1$ planes are still based on the iterative KM algorithm. Other algorithms to improve efficiency have also been proposed recently, such as [16] and [17], although these too are iterative.

To the best of our knowledge, all existing algorithms, such as the EKM and the EIASC, for determining the switch points are iterative-based methods. In this paper, a direct approach which can compute the switch points based on derivatives, without multiple iterations, is proposed. We use the term 'iteration' here to mean the repeated calculation of the position of the switch points. Within all approaches there are common calculations that require looping (e.g. cumulative sum), which we do not consider as iterations of the main algorithm.

The rest of the paper is organised as follows. The mathematical formulation of the KM algorithm is introduced in Section II. Section III briefly reviews some of the well-known iterative algorithms for the switch points in the KM algorithm. The new direct approach is introduced in Section IV, followed by a proof in Section V to show that the switch points calculated by the direct approach are the same as obtained via the KM algorithm. Experimental results for comparisons are shown and discussed in Sections VI and VII. Finally, a conclusion is provided in Section VIII. Note that all variables and constants defined in this paper are real numbers.

The authors are with the Laboratory for Uncertainty in Data and Decision Making (LUCID), the Intelligent Modelling and Analysis (IMA) and the Automated Scheduling Optimisation and Planning (ASAP) Research Groups, School of Computer Science, University of Nottingham, Nottingham, Jubilee Campus, NG8 1BB UK e-mail: {chao.chen, robert.john, jamie.twycross, jon.garibaldi}@nottingham.ac.uk.

Manuscript received *** **, 2016; revised *** **, 2017; accepted *** **, 2017. Date of publication *** **, 2017; date of current version *** **, 2017.
Digital Object Identifier ***

II. The KM algorithm

Let an IT2 fuzzy set $\tilde{A}$ be based on

$$\begin{aligned} x_i &\in X, \qquad i = 1, 2, ..., N \\ J_i &\equiv [\underline{u}_i, \bar{u}_i], \qquad 0 \leqslant \underline{u}_i \leqslant \bar{u}_i \leqslant 1 \end{aligned}$$

where x_i is the primary variable in the discrete universe of discourse X (note that x_i is in ascending order for i from 1 to N), J_i represents the membership grade interval for the primary variable x_i, and N is the number of discrete points in the universe of discourse.

For any given embedded type-1 fuzzy set, with membership grades $u_i \in J_i$ for all i, of such an IT2 fuzzy set $\tilde{A}$, the centroid is defined as:

$$c = \frac{\sum_{i=1}^{N} x_i u_i}{\sum_{i=1}^{N} u_i} .$$

If the above centroid c is computed for all embedded type-1 fuzzy sets, a centroid interval $C = [c_l, c_r]$ can be obtained where

$$\begin{aligned} c_l &= \inf_{\forall u_i \in J_i} \frac{\sum_{i=1}^{N} x_i u_i}{\sum_{i=1}^{N} u_i} \\ c_r &= \sup_{\forall u_i \in J_i} \frac{\sum_{i=1}^{N} x_i u_i}{\sum_{i=1}^{N} u_i} \end{aligned}$$

In fact, there is no need to compute the centroids for all embedded type-1 fuzzy sets to get the centroid interval. It is well known that the endpoints c_l and c_r of the centroid interval can also be expressed as (the derivation can be found in [9]):

$$c_l = \frac{\sum_{i=1}^{L} x_i \bar{u}_i + \sum_{i=L+1}^{N} x_i \underline{u}_i}{\sum_{i=1}^{L} \bar{u}_i + \sum_{i=L+1}^{N} \underline{u}_i} \quad (1)$$

$$c_r = \frac{\sum_{i=1}^{R} x_i \underline{u}_i + \sum_{i=R+1}^{N} x_i \bar{u}_i}{\sum_{i=1}^{R} \underline{u}_i + \sum_{i=R+1}^{N} \bar{u}_i} \quad (2)$$

where L and R, which are integer indices in the range of $[1, N-1]$, are the switch points for selecting $\underline{u}_i$ or $\bar{u}_i$.

As stated by Mendel [3], there is no closed-form solution for such switch points L and R, and hence, for c_l and c_r. By utilising the properties of the switch points such that

$$\begin{aligned} x_L \leqslant c_l \leqslant x_{L+1} \\ x_R \leqslant c_r \leqslant x_{R+1} \end{aligned}$$

the KM algorithm can be used to find them iteratively [1].

III. The iterative algorithms

In this section, two commonly used iterative algorithms are briefly introduced, to establish terminology and notation. Rather than introduce the original KM and IASC algorithms, their enhancements (the EKM and EIASC algorithms) are reviewed as they are more efficient than the original algorithms.

A. The EKM algorithm

The EKM algorithm is summarised in Table I. Compared to the original KM algorithm, the EKM algorithm introduces a better initialisation (Step 2) for the starting position and a change of the termination condition (Step 4) to remove an unnecessary iteration [9]. The EKM algorithm can save, on average, about two iterations. Simplified calculations (Step 5) are also introduced to save computational costs for each iteration.

B. The EIASC algorithm

The EIASC algorithm is summarised in Table II. Compared to the original IASC algorithm, the EIASC algorithm introduced a new stopping criterion (Step 4) [12]. Also, the starting point for computing c_r has been changed to N (Step 2) since the switch point R for c_r has been shown to be generally greater than $N/2$ [9].

IV. The Direct Approach (DA) algorithm

An arbitrary c in the centroid interval C, represented as Equation 3, can be expanded to Equation 4. Equation 4 can then be transformed to Equation 5 by substituting $x_1 + \sum_{j=2}^{i} \delta x_j$ for all corresponding $x_i (i > 1)$, where

$$\delta x_j = x_j - x_{j-1} .$$

For example, x_3 in Equation 4 is represented as $(x_1 + \delta x_2 + \delta x_3)$ in Equation 5. Note that δx_j is always positive since x_i (defined in Section II), and hence x_j, is in ascending order. This transformation is made to allow the sign of the partial derivatives to be easily identified, as shown in Equations 7 to 11. After the transformation, by rearranging and aggregating all the items with x_1 and δx_j for $j = 2, 3, ..., N$ in Equation 5, c can finally be represented as Equation 6. We can then compute the partial derivative of c with respect to u_j in the pattern of Equations 7 to 8. Then, this partial derivative $\frac{\partial c}{\partial u_j}$ can be represented as Equation 9 for specific $j \in [1, N]$.

It is noted by Karnik and Mendel [1] and Wu and Mendel [9] that equating the partial derivative of c with respect to u_j to zero does not give us any information about the value of u_j that maximises or minimises c. However, it can be observed that the sign of the partial derivative of c with respect to u_j does not depend on the value of u_j. Hence, it is clear that, when the partial derivative is negative, it is necessary to take the largest possible value of u_j in order to minimise c. When the partial derivative is positive, it is necessary to take the smallest possible value of u_j in order to minimise c. That is, to obtain c_l (the minimum centroid), one must set u_j to $\bar{u}_j$ when the partial derivative is negative, and to $\underline{u}_j$ when the partial derivative is positive. We observe that for any given value of u_j, this partial derivative is monotonically increasing with j (from 1 to N). Hence, for c_l, there must be a switch point $k \in [1, N]$ for which $\frac{\partial c}{\partial u_k} \leq 0$ and $\frac{\partial c}{\partial u_{k+1}} \geq 0$. Based on the above, it is possible to find the switch point directly by locating the value of k where the sign of the partial derivative changes. The same principles can be used to find the switch point k for the maximum centroid c_r, swapping $\bar{u}_j$ and $\underline{u}_j$ in u_j. The DA algorithm is such an algorithm, deploying the derivatives of

Step	The EKM Algorithm for computing c_l	The EKM Algorithm for computing c_r
1	Sort x_i $(i = 1, 2, ..., N)$ in ascending order and match $\underline{u}_i$ and $\bar{u}_i$ accordingly with their respective x_i.	
2	Set $k = [N/2.4]$, which is the nearest integer to $N/2.4$, and compute $a = \sum_{i=1}^{k} x_i \bar{u}_i + \sum_{i=k+1}^{N} x_i \underline{u}_i$ $b = \sum_{i=1}^{k} \bar{u}_i + \sum_{i=k+1}^{N} \underline{u}_i$	Set $k = [N/1.7]$, which is the nearest integer to $N/1.7$, and compute $a = \sum_{i=1}^{k} x_i \underline{u}_i + \sum_{i=k+1}^{N} x_i \bar{u}_i$ $b = \sum_{i=1}^{k} \underline{u}_i + \sum_{i=k+1}^{N} \bar{u}_i$
	$c = a/b$	
3	Find $k' \in [1, N-1]$ such that $x_{k'} < c \leqslant x_{k'+1}$	
4	If $k' = k$, set $c_l = c$ and stop;	If $k' = k$, set $c_r = c$ and stop;
	Otherwise, go to Step 5;	
5	Compute $s = sign(k' - k)$, and $a' = a + s \sum_{i=min(k,k')+1}^{max(k,k')} x_i(\bar{u}_i - \underline{u}_i)$ $b' = b + s \sum_{i=min(k,k')+1}^{max(k,k')} (\bar{u}_i - \underline{u}_i)$	Compute $a' = a + s \sum_{i=min(k,k')+1}^{max(k,k')} x_i(\underline{u}_i - \bar{u}_i)$ $b' = b + s \sum_{i=min(k,k')+1}^{max(k,k')} (\underline{u}_i - \bar{u}_i)$
	$c' = a'/b'$	
6	Set $c = c'$, $a = a'$, $b = b'$ and $k = k'$. Go to Step 3.	

TABLE I: The EKM algorithm for computing the centroid end points (c_l and c_r) of an IT2 Fuzzy Set. Note that for the case described in Section II, Step 1 is not necessary since x_i has already been defined in ascending order. Table I is adapted from [3, 9].

Step	The EIASC Algorithm for computing c_l	The EIASC Algorithm for computing c_r
1	Sort x_i $(i = 1, 2, ..., N)$ in ascending order and match $\underline{u}_i$ and $\bar{u}_i$ accordingly with their respective x_i.	
2	Initialise $k = 0$ and	Initialise $k = N$ and
	$a = \sum_{i=1}^{N} x_i \underline{u}_i$	
	$b = \sum_{i=1}^{N} \underline{u}_i$	
3	Compute $k = k + 1$	Compute
	$a = a + x_k(\bar{u}_k - \underline{u}_k)$	
	$b = b + \bar{u}_k - \underline{u}_k$	
	$c = a/b$	
		$k = k - 1$
4	If $c \leqslant x_{k+1}$, set $c_l = c$ and stop;	If $c \geqslant x_k$, set $c_r = c$ and stop;
	Otherwise, go to Step 3;	

TABLE II: The EIASC algorithm for computing the centroid end points (c_l and c_r) of an IT2 Fuzzy Set. Note that for the case described in Section II, Step 1 is not necessary since x_i has already been defined in ascending order. Table II is adapted from [3, 12].

c with respect to each u_j, and hence providing a method for directly obtaining the switch points. Figure 1 illustrates the monotonically increasing trend of the partial derivative $\frac{\partial c}{\partial u_j}$ and the location of the switch point.

Since δx_j is always positive, it can be observed in Equation 9 that the partial derivative consists of two parts, which are the positive part pos_j and the negative part neg_j:

$$\frac{\partial c}{\partial u_j} = pos_j + neg_j$$

where pos_j and neg_j are presented in Equations 10 and 11, which can be summarised as Equations 12 and 13. Having all the partial derivatives, the switch points L for c_l and R for c_r can be obtained as follows.

For L and hence c_l:

1) Calculate pos_j by setting all the u_i to be $\bar{u}_i$ and calculate neg_j by setting all the u_i to be $\underline{u}_i$, for all $j \in [1, N]$.
2) Calculate all the partial derivatives $\frac{\partial c}{\partial u_j}$ as pos_j + neg_j.
3) Find the smallest $k \in [1, N-1]$ such that $\frac{\partial c}{\partial u_{k+1}} \geqslant 0$.
4) If k exists, set $L = k$; otherwise, set $L = N - 1$.
5) Compute c_l by Equation 1.

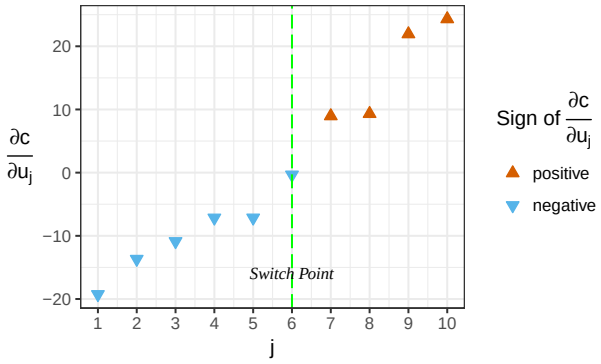

Fig. 1: An illustration of the DA algorithm showing the position of the switch point for an arbitrary example where N is 10.

For R and hence c_r:

1) Calculate pos_j by setting all the u_i to be $\underline{u}_i$ and calculate neg_j by setting all the u_i to be $\bar{u}_i$, for all $j \in [1, N]$.
2) Calculate all the partial derivatives $\frac{\partial c}{\partial u_j}$ as $pos_j + neg_j$.
3) Find the largest $k \in [1, N-1]$ such that $\frac{\partial c}{\partial u_k} \leqslant 0$.
4) If k exists, set $R = k$; otherwise, set $R = 1$.
5) Compute c_r by Equation 2.

It should be noted that the denominator $(\sum_{i=1}^{N} u_i)^2$ in Equations 12 and 13, which must be positive, can be neglected in calculations to save computational costs. In other words, without the risk of changing the sign of $\frac{\partial c}{\partial u_j}$, pos_j and neg_j can be simplified as presented in Equations 14 and 15. Also, in practice, there is no need to calculate the partial derivatives one by one, as cumulative sums and vectorised operations can be used. The pseudo-code for obtaining L and c_l is shown in Algorithm 1. R and c_r are calculated in a similar manner, making the substitutions as described above (pseudo-code is omitted due to space constraints).

V. Proof of the DA algorithm

Proposition 1. *It is clear that when $\frac{\partial c}{\partial u_j} < 0$, u_j should be its maximum value, $\bar{u}_j$, to minimise c and it should be its minimum value, $\underline{u}_j$, to maximise c. Conversely, when $\frac{\partial c}{\partial u_j} > 0$, u_j should be $\underline{u}_j$ to minimise c and it should be $\bar{u}_j$ to maximise c. When $\frac{\partial c}{\partial u_j} = 0$, c is not dependent on the value of u_j.*

Theorem 1. *The index L obtained by the DA algorithm, as described in Section IV, is the correct switch point for calculating the minimum centroid c_l.*

Proof. Given that δx_i is always positive, it is clear (from Equations 14 and 15) that for any $j \in [1, N]$, pos_j attains its maximum value when u_i is $\bar{u}_i$ and neg_j attains its maximum value when u_i is $\underline{u}_i$; hence $\frac{\partial c}{\partial u_j}$ is also maximal. It can be observed from Equation 8 that $\frac{\partial c}{\partial u_j}$ is monotonically increasing in j. This is because the only difference of $\frac{\partial c}{\partial u_j}$ for $j = n-1$ and $j = n$ is the n^{th} term ($n \in [2, N]$), which is negative for $j = n-1$ and positive for $j = n$.

If there exists a smallest $k \in [1, N-1]$ such that $\frac{\partial c}{\partial u_{k+1}} \geqslant 0$, then k is the correct switch point L (for calculating c_l) on the basis that:

1) L cannot be less than k, for the following reason. For every $j \in [1, k]$, it must be the case that $\frac{\partial c}{\partial u_j} < 0$. Therefore, by proposition 1, u_j should be its maximum value $\bar{u}_j$ to minimise c. Now suppose L is less than k. Then, as L is the switch point, every $u_{j\in[L+1,k]}$ should be its minimum value $\underline{u}_j$, which by contradiction is not possible.
2) L cannot be greater than k, for the following reason. Assume L is greater than k. Then, as L is the switch point every $u_{j\in[k+1,L]}$ should be its maximum value $\bar{u}_j$. By proposition 1, it must be the case that $\frac{\partial c}{\partial u_{k+1}} < 0$, in which case u_{k+2} must be changed from $\underline{u}_{k+2}$ to $\bar{u}_{k+2}$. This means $\frac{\partial c}{\partial u_{k+2}}$, if it exists, must also be negative. By induction, it can be deduced that $\frac{\partial c}{\partial u_N}$ must be negative. This is in contradiction to the fact that $\frac{\partial c}{\partial u_N} \geqslant 0$. Thus, the assumption is incorrect. Hence, L cannot be greater than k.

If there does not exist a $k \in [1, N-1]$ such that $\frac{\partial c}{\partial u_{k+1}} > 0$, then for every $j \in [1, N-1]$, again it must be the case that $\frac{\partial c}{\partial u_j} < 0$. Also, $\frac{\partial c}{\partial u_N}$ must be 0. Thus, there is no contradiction for the switch point L to be $N-1$.

□

Theorem 2. *The index R obtained by the DA algorithm, as described in Section IV, is the correct switch point for calculating the maximum centroid c_r.*

Proof. The proof is similar to above. □

VI. Experimental comparison

To further investigate the performance of the new DA, comparisons in terms of time efficiency between the new approach and two of the most widely used algorithms (the EKM and the EIASC algorithms) were conducted. The platform was a laptop with Intel Core i7-3720QM CPU @ 2.60GHz and 8GB memory, running Windows 7 Professional 64bit Service Pack 1. The programming language and software environment is R x64 version 3.2.3. Computational costs were measured by the user time returned by the built-in function *proc.time()* in the R environment.

A. Examples of IT2 fuzzy sets

In this section, three example IT2 fuzzy sets are used to verify the correctness of the DA algorithm by comparing the switch points with the EKM algorithm and the EIASC algorithm. Specifically, the vector X, containing x_i, has 101 discrete values from 0 to 10 by a step size of 0.1. $\bar{u}_i$ and $\underline{u}_i$ are obtained by the following membership functions for each type of example IT2 fuzzy sets.

1) Symmetric Gaussian membership functions with uncertain deviation:

$$\bar{u}_i = \exp\left(-0.5\left(\frac{x_i - 5}{1.75}\right)^2\right)$$

$$\underline{u}_i = \exp\left(-0.5\left(\frac{x_i - 5}{0.25}\right)^2\right)$$

$$
\begin{aligned}
c &= \frac{\sum_{i=1}^{N} x_i u_i}{\sum_{i=1}^{N} u_i} && (3)\\
&= \frac{x_1u_1 + x_2u_2 + x_3u_3 \cdots + x_Nu_N}{\sum_{i=1}^{N} u_i} && (4)\\
&= \frac{x_1u_1 + (x_1+\delta x_2)u_2 + (x_1+\delta x_2+\delta x_3)u_3 \cdots + (x_1+\delta x_2 \cdots + \delta x_N)u_N}{\sum_{i=1}^{N} u_i} && (5)\\
&= x_1 + \frac{(\sum_{i=2}^{N} u_i)\delta x_2}{\sum_{i=1}^{N} u_i} + \frac{(\sum_{i=3}^{N} u_i)\delta x_3}{\sum_{i=1}^{N} u_i} \cdots + \frac{(\sum_{i=N}^{N} u_i)\delta x_N}{\sum_{i=1}^{N} u_i} && (6)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial c}{\partial u_j} &= \frac{\partial}{\partial u_j}\left(x_1 + \frac{(\sum_{i=2}^{N} u_i)\delta x_2}{\sum_{i=1}^{N} u_i} + \frac{(\sum_{i=3}^{N} u_i)\delta x_3}{\sum_{i=1}^{N} u_i} \cdots + \frac{(\sum_{i=N}^{N} u_i)\delta x_N}{\sum_{i=1}^{N} u_i}\right) && (7)\\
&= \frac{\partial}{\partial u_j}(x_1) + \frac{\partial}{\partial u_j}\left(\frac{(\sum_{i=2}^{N} u_i)\delta x_2}{\sum_{i=1}^{N} u_i}\right) + \frac{\partial}{\partial u_j}\left(\frac{(\sum_{i=3}^{N} u_i)\delta x_3}{\sum_{i=1}^{N} u_i}\right) \cdots + \frac{\partial}{\partial u_j}\left(\frac{(\sum_{i=N}^{N} u_i)\delta x_N}{\sum_{i=1}^{N} u_i}\right) && (8)\\
&= \begin{cases}
\left(0 - \frac{(\sum_{i=2}^{N} u_i)\delta x_2}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=3}^{N} u_i)\delta x_3}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=4}^{N} u_i)\delta x_4}{(\sum_{i=1}^{N} u_i)^2} \cdots - \frac{(\sum_{i=N-2}^{N} u_i)\delta x_{N-2}}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=N-1}^{N} u_i)\delta x_{N-1}}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=N}^{N} u_i)\delta x_N}{(\sum_{i=1}^{N} u_i)^2}\right) & |j=1\\
\left(0 + \frac{(\sum_{i=1}^{1} u_i)\delta x_2}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=3}^{N} u_i)\delta x_3}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=4}^{N} u_i)\delta x_4}{(\sum_{i=1}^{N} u_i)^2} \cdots - \frac{(\sum_{i=N-2}^{N} u_i)\delta x_{N-2}}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=N-1}^{N} u_i)\delta x_{N-1}}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=N}^{N} u_i)\delta x_N}{(\sum_{i=1}^{N} u_i)^2}\right) & |j=2\\
\left(0 + \frac{(\sum_{i=1}^{1} u_i)\delta x_2}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{2} u_i)\delta x_3}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=4}^{N} u_i)\delta x_4}{(\sum_{i=1}^{N} u_i)^2} \cdots - \frac{(\sum_{i=N-2}^{N} u_i)\delta x_{N-2}}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=N-1}^{N} u_i)\delta x_{N-1}}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=N}^{N} u_i)\delta x_N}{(\sum_{i=1}^{N} u_i)^2}\right) & |j=3\\
\vdots & \\
\left(0 + \frac{(\sum_{i=1}^{1} u_i)\delta x_2}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{2} u_i)\delta x_3}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{3} u_i)\delta x_4}{(\sum_{i=1}^{N} u_i)^2} \cdots + \frac{(\sum_{i=1}^{N-3} u_i)\delta x_{N-2}}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=N-1}^{N} u_i)\delta x_{N-1}}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=N}^{N} u_i)\delta x_N}{(\sum_{i=1}^{N} u_i)^2}\right) & |j=N-2\\
\left(0 + \frac{(\sum_{i=1}^{1} u_i)\delta x_2}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{2} u_i)\delta x_3}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{3} u_i)\delta x_4}{(\sum_{i=1}^{N} u_i)^2} \cdots + \frac{(\sum_{i=1}^{N-3} u_i)\delta x_{N-2}}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{N-2} u_i)\delta x_{N-1}}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=N}^{N} u_i)\delta x_N}{(\sum_{i=1}^{N} u_i)^2}\right) & |j=N-1\\
\left(0 + \frac{(\sum_{i=1}^{1} u_i)\delta x_2}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{2} u_i)\delta x_3}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{3} u_i)\delta x_4}{(\sum_{i=1}^{N} u_i)^2} \cdots + \frac{(\sum_{i=1}^{N-3} u_i)\delta x_{N-2}}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{N-2} u_i)\delta x_{N-1}}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{N-1} u_i)\delta x_N}{(\sum_{i=1}^{N} u_i)^2}\right) & |j=N
\end{cases} && (9)
\end{aligned}
$$

$$
pos_j = \begin{cases}
0 & |j=1\\
0 + \frac{(\sum_{i=1}^{1} u_i)\delta x_2}{(\sum_{i=1}^{N} u_i)^2} & |j=2\\
0 + \frac{(\sum_{i=1}^{1} u_i)\delta x_2}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{2} u_i)\delta x_3}{(\sum_{i=1}^{N} u_i)^2} & |j=3\\
\vdots & \\
0 + \frac{(\sum_{i=1}^{1} u_i)\delta x_2}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{2} u_i)\delta x_3}{(\sum_{i=1}^{N} u_i)^2} \cdots + \frac{(\sum_{i=1}^{N-3} u_i)\delta x_{N-2}}{(\sum_{i=1}^{N} u_i)^2} & |j=N-2\\
0 + \frac{(\sum_{i=1}^{1} u_i)\delta x_2}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{2} u_i)\delta x_3}{(\sum_{i=1}^{N} u_i)^2} \cdots + \frac{(\sum_{i=1}^{N-2} u_i)\delta x_{N-1}}{(\sum_{i=1}^{N} u_i)^2} & |j=N-1\\
0 + \frac{(\sum_{i=1}^{1} u_i)\delta x_2}{(\sum_{i=1}^{N} u_i)^2} + \frac{(\sum_{i=1}^{2} u_i)\delta x_3}{(\sum_{i=1}^{N} u_i)^2} \cdots + \frac{(\sum_{i=1}^{N-1} u_i)\delta x_N}{(\sum_{i=1}^{N} u_i)^2} & |j=N
\end{cases} \quad (10)
$$

$$
neg_j = \begin{cases}
-\frac{(\sum_{i=2}^{N} u_i)\delta x_2}{(\sum_{i=1}^{N} u_i)^2} \cdots - \frac{(\sum_{i=N-1}^{N} u_i)\delta x_{N-1}}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=N}^{N} u_i)\delta x_N}{(\sum_{i=1}^{N} u_i)^2} & |j=1\\
-\frac{(\sum_{i=3}^{N} u_i)\delta x_3}{(\sum_{i=1}^{N} u_i)^2} \cdots - \frac{(\sum_{i=N-1}^{N} u_i)\delta x_{N-1}}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=N}^{N} u_i)\delta x_N}{(\sum_{i=1}^{N} u_i)^2} & |j=2\\
-\frac{(\sum_{i=4}^{N} u_i)\delta x_4}{(\sum_{i=1}^{N} u_i)^2} \cdots - \frac{(\sum_{i=N-1}^{N} u_i)\delta x_{N-1}}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=N}^{N} u_i)\delta x_N}{(\sum_{i=1}^{N} u_i)^2} & |j=3\\
\vdots & \\
-\frac{(\sum_{i=N-1}^{N} u_i)\delta x_{N-1}}{(\sum_{i=1}^{N} u_i)^2} - \frac{(\sum_{i=N}^{N} u_i)\delta x_N}{(\sum_{i=1}^{N} u_i)^2} & |j=N-2\\
-\frac{(\sum_{i=N}^{N} u_i)\delta x_N}{(\sum_{i=1}^{N} u_i)^2} & |j=N-1\\
0 & |j=N
\end{cases} \quad (11)
$$

$$pos_j = \begin{cases} 0 & \big| j = 1 \\ \frac{(\sum_{i=1}^{1} u_i)\delta x_2}{(\sum_{i=1}^{N} u_i)^2} \left[+ \frac{(\sum_{i=1}^{2} u_i)\delta x_3}{(\sum_{i=1}^{N} u_i)^2} \cdots + \frac{(\sum_{i=1}^{j-1} u_i)\delta x_j}{(\sum_{i=1}^{N} u_i)^2} \right] & \big| j \in [2, N] \end{cases} \quad (12)$$

$$neg_j = \begin{cases} \left[- \frac{(\sum_{i=j+1}^{N} u_i)\delta x_{j+1}}{(\sum_{i=1}^{N} u_i)^2} \cdots - \frac{(\sum_{i=N-1}^{N} u_i)\delta x_{N-1}}{(\sum_{i=1}^{N} u_i)^2} \right] - \frac{(\sum_{i=N}^{N} u_i)\delta x_N}{(\sum_{i=1}^{N} u_i)^2} & \big| j \in [1, N-1] \\ 0 & \big| j = N \end{cases} \quad (13)$$

$$pos_j = \begin{cases} 0 & \big| j = 1 \\ \left(\sum_{i=1}^{1} u_i\right) \delta x_2 \left[+ \left(\sum_{i=1}^{2} u_i\right) \delta x_3 \cdots + \left(\sum_{i=1}^{j-1} u_i\right) \delta x_j \right] & \big| j \in [2, N] \end{cases} \quad (14)$$

$$neg_j = \begin{cases} \left[- \left(\sum_{i=j+1}^{N} u_i\right) \delta x_{j+1} \cdots - \left(\sum_{i=N-1}^{N} u_i\right) \delta x_{N-1} \right] - \left(\sum_{i=N}^{N} u_i\right) \delta x_N & \big| j \in [1, N-1] \\ 0 & \big| j = N \end{cases} \quad (15)$$

input : $X, \bar{U}, \underline{U}$, vectors of the primary variable, the upper membership grades, and the lower membership grades, respectively; x_i, $\bar{u}_i$, and $\underline{u}_i$ denote elements of the respective vectors;
output: L, the switch point; c_l, the lower bound of the centroid;

$X' \leftarrow \{x_i - x_{i-1} \mid i = 2, 3, ..., N\}$, a vector of consecutive differences of elements of X ;
$S^1 \leftarrow \{\sum_{i=1}^{j} \bar{u}_i \mid j = 1, 2, ..., N-1\}$, a vector of the cumulative sum of the first $N-1$ elements of $\bar{U}$;
$S^2 \leftarrow \{\sum_{i=N}^{j} \underline{u}_i \mid j = N, ..., 3, 2\}$ a vector of the cumulative sum of the last $N-1$ elements of $\underline{U}$ in the reverse order ;
$T^P \leftarrow \{x'_i \cdot s_i^1 \mid i = 1, 2, ..., N-1\}$, where x'_i and s_i^1 are the i^{th} element of vectors X' and S^1 respectively ;
$T^N \leftarrow \{x'_{N-i} \cdot s_i^2 \mid i = 1, 2, ..., N-1\}$, where s_i^2 is the i^{th} element of the vector S^2 ;
$D^P \leftarrow \{0, \sum_{i=1}^{j} t_i^P \mid j = 1, 2, ..., N-1\}$, where t_i^P is the i^{th} element of the vector T^P ;
$D^N \leftarrow \{\sum_{i=1}^{j} -t_i^N, 0 \mid j = N-1, ..., 2, 1\}$, where t_i^N is the i^{th} element of the vector T^N ;
$D \leftarrow \{d_i^P + d_i^N \mid i = 1, 2, ..., N\}$, where d_i^P and d_i^N are the i^{th} element of vectors D^P and D^N respectively ;
Find the smallest $k \in 1, 2, ..., N-1$ such that $d_{k+1} \geqslant 0$, where d_{k+1}, which represents $\frac{\partial c_l}{\partial u_{k+1}}$, is the $(k+1)^{th}$ element of the vector D ;
if k *exists* **then** $L \leftarrow k$ **else** $L \leftarrow N-1$;
Compute c_l by Equation 1;

Algorithm 1: Pseudo code for obtaining L, the switch point; and c_l, the lower bound of the centroid.

2) Gaussian upper membership function and triangular lower membership function:

$$\bar{u}_i = \begin{cases} \exp\left(-0.5\left(\frac{x_i-2}{5}\right)^2\right) & \big| 0 \leqslant x_i \leqslant 7.185 \\ \exp\left(-0.5\left(\frac{x_i-9}{1.75}\right)^2\right) & \big| 7.185 < x_i \leqslant 10 \end{cases}$$

$$\underline{u}_i = \begin{cases} \frac{0.6(x_i+5)}{19} & \big| 0 \leqslant x_i \leqslant 2.6 \\ \frac{0.4(14-x_i)}{19} & \big| 2.6 < x_i \leqslant 10 \end{cases}$$

3) Piecewise Gaussian membership functions:

$$\bar{u}_i = \max \begin{cases} \exp\left(-\left(\frac{x_i-3}{\sqrt{8}}\right)^2\right) \\ 0.8\exp\left(-\left(\frac{x_i-6}{\sqrt{8}}\right)^2\right) \end{cases}$$

$$\underline{u}_i = \max \begin{cases} 0.5\exp\left(-\left(\frac{x_i-3}{\sqrt{2}}\right)^2\right) \\ 0.4\exp\left(-\left(\frac{x_i-6}{\sqrt{2}}\right)^2\right) \end{cases}$$

As shown in Table III, the switch points L and R obtained with the examples by three algorithms are all the same.

	Example 1		Example 2		Example 3	
	L	R	L	R	L	R
DA	36	65	27	69	32	57
EKM	36	65	27	69	32	57
EIASC	36	65	27	69	32	57

TABLE III: The switch points L and R obtained by three algorithms based on the example fuzzy sets.

B. Generalised IT2 fuzzy sets

1) Generalised bell-shaped IT2 fuzzy sets: It was assumed that the vector X, containing x_i, is uniformly distributed from 0 to 10. $\bar{u}_i$ and $\underline{u}_i$ are defined by generalised bell-shaped function:

$$\bar{u}_i = \frac{1}{1 + \left(\frac{x_i - c}{\bar{a}}\right)^{2b}}$$

$$\underline{u}_i = \frac{1}{1 + \left(\frac{x_i - c}{\underline{a}}\right)^{2b}}$$

where $\underline{a}$ and b are randomly selected between 1 and 2; $\bar{a}$ is the multiplication of $\underline{a}$ with a random number between 1 and 2; c is a random number between 0 and 10.

2) Generalised randomly-shaped IT2 fuzzy sets: This experimental comparison was designed to be similar to the first comparison in [12]. It was assumed that vectors X and $\bar{U}$, containing x_i and $\bar{u}_i$ respectively, are uniformly distributed from 0 to 1. $\underline{u}_i$ is the multiplication of $\bar{u}_i$ with a random number between 0 and 1.

Comparisons are made for these two types of IT2 fuzzy sets separately. In each comparison, N, which is the length of X, is set to be between 10 and 2000 with a step size of 10 (giving 200 different values of N). For each value of N, 5000 Monte Carlo simulations were made and the computational time costs were aggregated to be compared for each algorithm.

In all the 2×10^6 simulations, the DA algorithm gave the same switch points as those given by the KM and the EIASC algorithms. Computational time comparisons are shown in Figures 2 and 3. It can be observed that the three algorithms compared are similar when N is very small. It should be noted that the EIASC algorithm is shown to be better than the EKM algorithm when N is smaller than 1000 [12]. However, in our experiments, the EIASC algorithm is only more efficient than other algorithms when N is around 10. In contrast, DA clearly outperforms other algorithms when N is larger than 100 regardless of the shape of fuzzy sets.

VII. Discussion

As can be observed in Figures 2 and 3, the computational time of the EIASC algorithm increases most rapidly among the three algorithms. As has already been discussed, this is because the number of switch points to be evaluated for the EIASC algorithm increases along with the increase of N. In other words, the number of iterations for the EIASC algorithm increases rapidly. In contrast, the EKM algorithm can achieve its final result in from two to six iterations, regardless of N. Thus, the computational time for the EKM algorithm increases less significantly than the EIASC algorithm. However, the computational time does increase linearly because the size of the vectors involved in the computing process increases along with the increase of N.

Regardless of the shape of fuzzy sets, our newly proposed DA algorithm performs the best among the three algorithms, although it can be considered as a brute-force approach since all the partial derivatives have to be computed before locating the switch points. However, vectorised operations and the use of cumulative sum make the computing of partial derivatives quite simple. Thus, the DA algorithm is still competitive for very small values of N and clearly outperforms the EKM, the EIASC for $N \gtrsim 100$.

VIII. Conclusion

In this paper, a direct approach based on derivatives for determining the switch points for calculating the centroid of an interval type-2 fuzzy set has been introduced. A derivation of the algorithm, psuedo-code for calculating the switch points, and a mathematical proof of correctness of the switch points

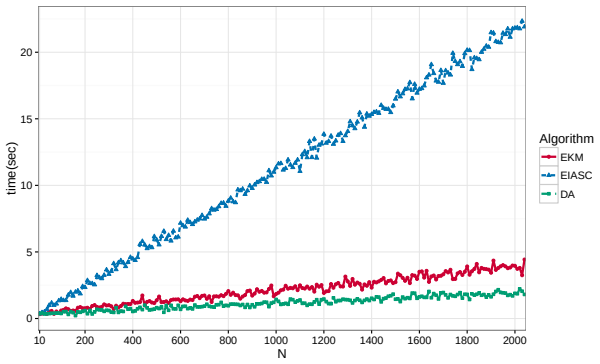

Fig. 2: A comparison of computational time costs for different algorithms based on generalised bell-shaped IT2 fuzzy sets described in Section VI.

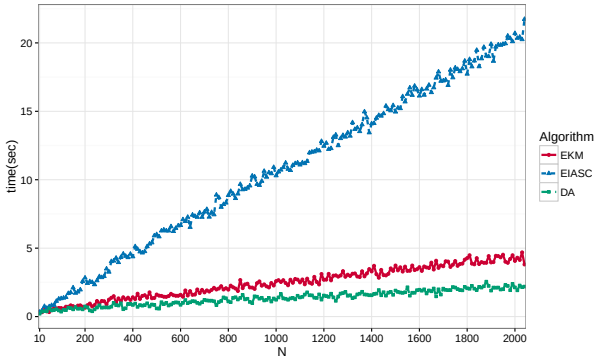

Fig. 3: A comparison of computational time costs for different algorithms based on generalised randomly-shaped IT2 fuzzy sets described in Section VI.

have been given for the proposed approach. By empirical simulations, it has been shown that DA is superior to all other iterative algorithms (including the EKM and the EIASC) in time efficiency. It should be noted that the DA algorithm is in fact a brute force method which requires the calculations of all partial derivatives. While it can be noted that the partial derivatives are in ascending order and the switch points are located where the sign of partial derivatives changes, it would be interesting for an approach to find the switch points without calculating all the partial derivatives.

In conclusion, we have contributed a new algorithm for determining the switch points for calculating the lower and upper bounds of the centroids of an interval type-2 fuzzy set. Given that our algorithm clearly outperform the EKM and the EIASC algorithms, we suggest that this new DA algorithm should always be used when N, the number of discretizations of the universe of discourse, $\gtrsim 100$.

References

[1] N. Karnik and J. Mendel, "Centroid of a type-2 fuzzy set," *Information Sciences*, vol. 132, no. 14, pp. 195–220, 2001.

[2] J. M. Mendel, "On KM Algorithms for Solving Type-2 Fuzzy Set Problems," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 3, pp. 426–446, 2013.

[3] J. Mendel, H. Hagras, W.-W. Tan, W. W. Melek, and H. Ying, *Introduction To Type-2 Fuzzy Logic Control: Theory and Applications*, 1st ed. Wiley-IEEE Press, 2014.
[4] J. M. Mendel and M. R. Rajati, "On Computing Normalized Interval Type-2 Fuzzy Sets," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 5, pp. 1335–1340, 2014.
[5] M. Nie and W. W. Tan, "Ensuring the Centroid of an Interval Type-2 Fuzzy Set," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 4, pp. 950–963, 2015.
[6] T. Kumbasar, "Revisiting KM algorithms: A Linear Programming approach," in *Proceedings IEEE International Conference on Fuzzy Systems*, 2015, pp. 1–6.
[7] F. Liu and J. M. Mendel, "Aggregation using the fuzzy weighted average as computed by the Karnik-Mendel algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 1, pp. 1–12, 2008.
[8] H. Wu and J. M. Mendel, "Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 5, pp. 622–639, 2002.
[9] D. Wu and J. M. Mendel, "Enhanced Karnik–Mendel algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 4, pp. 923–934, 2009.
[10] M. Melgarejo, "A fast recursive method to compute the generalized centroid of an interval type-2 fuzzy set," in *Proceedings North American Fuzzy Information processing Society*, 2007, pp. 190–194.
[11] K. Duran, H. Bernal, and M. Melgarejo, "Improved iterative algorithm for computing the generalized centroid of an interval type-2 fuzzy set," in *Proceedings North American Fuzzy Information Processing Society*, 2008, pp. 1–5.
[12] D. Wu and M. Nie, "Comparison and practical implementation of type-reduction algorithms for type-2 fuzzy sets and systems," in *Proceedings IEEE International Conference on Fuzzy Systems*, 2011, pp. 2131–2138.
[13] M. Nie and W. W. Tan, "Towards an efficient type-reduction method for interval type-2 fuzzy logic systems," in *Proceedings IEEE International Conference on Fuzzy Systems*, 2008, pp. 1425–1432.
[14] J. M. Mendel and X. Liu, "New closed-form solutions for Karnik-Mendel algorithm+defuzzification of an interval type-2 fuzzy set," in *Proceedings IEEE International Conference on Fuzzy Systems*, 2012, pp. 1–8.
[15] M. Nie and W. W. Tan, "Closed form formulas for computing the centroid of a general type-2 fuzzy set," in *Proceedings IEEE International Conference on Fuzzy Systems*, 2015, pp. 1–8.
[16] S. Chakraborty, A. Konar, A. Ralescu, and N. R. Pal, "A Fast Algorithm to Compute Precise Type-2 Centroids for Real-Time Control Applications," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 340–353, 2015.
[17] S. M. Salaken, A. Khosravi, S. Nahavandi, and D. Wu, "Effect of different initializations on EKM algorithm," in *Proceedings IEEE International Conference on Fuzzy Systems*, 2015, pp. 1–6.

Chao Chen received the B.Eng. degree in Electronic and Information Engineering from Tianjin University of Technology, Tianjin, China, in 2003, and the M.Sc. degree (Distinction) in Management of Information Technology from University of Nottingham, Nottingham, UK, in 2012. He is currently a Ph.D. student, with the Vice-Chancellor's Scholarship for Research Excellence, in School of Computer Science, University of Nottingham. He is a member of the Laboratory for Uncertainty in Data and Decision Making (LUCID) and the Intelligent Modelling and Analysis (IMA) Research Group. His current research interests include the modelling of time series forecasting with fuzzy logic systems. Particularly, he has an interest in the optimisation of fuzzy models with the architecture of the adaptive-network-based fuzzy inference system (ANFIS).

Robert John received the B.Sc. (Hons.) degree in mathematics from Leicester Polytechnic, Leicester, U.K., the M.Sc. degree in statistics from UMIST, Manchester, U.K., and the Ph.D. degree in Fuzzy Logic from De Montfort University, Leicester, U.K., in 1979, 1981, and 2000, respectively. He worked in industry for 10 years as a mathematician and knowledge engineer developing knowledge based systems for British Gas and the financial services industry. Bob spent 24 years at De Montfort University. He has over 150 research publications of which about 50 are in international journals with over 6000 citations. Bob joined the University of Nottingham in 2013 where he heads up the research group ASAP in the School of Computer Science. The Automated Scheduling, Optimisation and Planning (ASAP) research group carries out multi-disciplinary research into mathematical models and algorithms for a variety of real world optimisation problems. He is also a member of LUCID.

Jamie Twycross is an Assistant Professor in Computer Science at the University of Nottingham. He has a B.Sc. (Hons) in Mathematical Physics from Imperial College, London, an M.Sc. in Evolutionary and Adaptive Systems from the University of Sussex, and a Ph.D. in Computer Science from the University of Nottingham. His main research interest is in Computational Biology, where he works at the interface of computer science and biology to develop and apply computational and mathematical approaches to address biological and digital problems. He has expertise in computational and mathematical modelling, data analytics, machine learning, and software engineering. He is a member of the Intelligent Modelling and Analysis Group, and leads the Modelling Group in the Synthetic Biology Research Centre at the University of Nottingham.

Jonathan M. Garibaldi received the B.Sc. (Hons.) degree in physics from Bristol University, Bristol, U.K., in 1984, and M.Sc. degree and Ph.D. degree from the University of Plymouth, Plymouth, U.K., in 1990 and 1997, respectively. Prof. Garibaldi is currently Head of School of Computer Science, University of Nottingham, Head of the Intelligent Modelling and Analysis (IMA) Research Group, and a member of the Lab for Uncertainty in Data and Decision Making (LUCID). His main research interests include modelling uncertainty and variation in human reasoning, and in modelling and interpreting complex data to enable better decision making, particularly in medical domains. Prof. Garibaldi is the current Editor-in-Chief of IEEE Transactions on Fuzzy Systems.