
Article:
Rubio-Solis, A., Melin, P., Martinez-Hernandez, U. et al. (2018) General type-2 radial basis function neural network: a data-driven fuzzy model. IEEE Transactions on Fuzzy Systems. ISSN: 1063-6706

https://doi.org/10.1109/TFUZZ.2018.2858740

General Type-2 Radial Basis Function Neural Network: A Data-Driven Fuzzy Model

Adrian Rubio-Solis[1], Patricia Melin[2], Uriel Martinez-Hernandez[3] and George Panoutsos[1]

***Abstract*—This paper proposes a new General Type-2 Radial Basis Function Neural Network (GT2-RBFNN) that is functionally equivalent to a GT2 Fuzzy Logic System (FLS) of either Takagi-Sugeno-Kang (TSK) or Mamdani type. The neural structure of the GT2-RBFNN is based on the α-planes representation, in which the antecedent and consequent part of each fuzzy rule uses GT2 Fuzzy Sets (FSs). To reduce the iterative nature of the Karnik-Mendel algorithm, the Enhaned-Karnik-Mendel (EKM) type-reduction and three popular direct-defuzzification methods, namely the 1) Nie-Tan approach (NT), the 2) Wu-Mendel uncertain bounds method (WU) and the 3) Biglarbegian-Melek-Mendel algorithm (BMM) are employed. For that reason, this paper provides four different neural structures of the GT2-RBFNN and their structural and parametric optimisation. Such optimisation is a two-stage methodology that first implements an Iterative Information Granulation approach to estimate the antecedent parameters of each fuzzy rule. Secondly, each consequent part and the fuzzy rule base of the GT2-RBFNN is trained and optimised using an Adaptive Gradient Descent method (AGD) respectively. Several benchmark data sets, including a problem of identification of a nonlinear system and a chaotic time series are considered. The reported comparative analysis of experimental results is used to evaluate the performance of the suggested GT2 RBFNN with respect to other popular methodologies.**

***Index Terms*—General Type-2 FLSs, Radial Basis Function Neural Networks, α-plane representation, fuzzy modelling.**

I. Introduction

GENERAL Type-2 Fuzzy Logic is now well established and is gaining more and more in popularity [1]–[6]. This is mainly credited to the capability of General Type-2 Fuzzy Sets (GT2 FSs) to better handle and minimise the effect of high levels of uncertainty with respect to other high order FSs such as Interval Type-2 Fuzzy Sets (IT2 FSs) [7]–[14]. Compared to Type-1 Fuzzy Sets (T1 FSs) and IT2 FSs, a GT2 FS weights uncertainty nonuniformly and is described by a Memberhip Function (MF) that is characterised by more parameters, so using GT2 FSs allows for more design degrees of freedom [7], [14]. Furthermore, a GT2 FS is characterised by a Footprint of Uncertainty (FOU) and an MF (secondary MF), where uncertainty can be modelled with any degree between 0 and 1, whereas T1 and IT2 FSs associate uncertainty only to crisp values of 0 or 1 respectively [9].

[1] A. Rubio-Solis and G. Panoutsos are with the department of Automatic Control and Systems (ACSE), The University of Sheffield, Sheffield, S1 3JD UK. a.rubiosolis, g.panoutsos@sheffield.ac.uk

[2] Patricia Melin is with the Division of Graduate Studies Tijuana Institute of Technology Tijuana, Mexico. pmelin@tectijuana.mx

[3] Uriel Martinez-Hernandez is with the Department of Electronic and Electrical Engineering, Faculty of Engineering and Design, University of Bath, Bath, BA2 7AY, UK. u.martinez@bath.ac.uk

TABLE I: ABBREVIATIONS AND THEIR DEFINITIONS.

Abbreviation	Definition
A2-C0	Antecedents are type-2 fuzzy sets and Consequents are type-0 fuzzy sets (crisp)
AED	Average of End-points Defuzzification
AGD	Adaptive Gradient Descent
BMM	Biglarbegian Melek Mendel approach
COS	Center Of Sets (type-reduction)
EKM	Enhanced Karnik-Mendel algorithm
FOU	Foot Print of Uncertainty
GT2 FS	General Type-2 Fuzzy Set
IIG	Iterative Information Granulation
IWA	Interval Weighted Average
LMF	Lower Membership Function
MF	Membership Function
NT	Nie-Tan simplification.
RBFNN	Radial Basis Function Neural Network
T1 FLS	Type-1 Fuzzy Logic System
T2 FLS	Type-2 Fuzzy Logic System
TR	Type Reduction.
TSK	Takagi Sugeno Kang
UMF	Upper Membership Function
WM UBs	Wu Mendel Uncertainty Bounds

As indicated in [7], a GT2 FLS can be thought as a high order fuzzy set uncertainty model with more flexibility. Therefore, a GT2 Fuzzy Logic System has the potential to outperform not only the use of FLSs of T1, but also to provide a performance than an FLS with IT2 FSs cannot achieve [7]. Although GT2 FLSs are still in their infancy, the number of aplications of higher order fuzzy systems has experienced an important increase during the past five years [15], in particular in areas such as Pattern Recognition [12], [13], Automatic Control [2], [16], Image Processing [17] and Robotics [1], [3], [18]. In this applied context, the usage of GT2 FSs usually increases the computational complexity with respect to T1 and IT2 FLSs. This is clearly compensated not only by a higher model accuracy but also with a better treatment of uncertainty that can be obtained by using GT2 FSs as well as due to new computing technologies. For example in [19], a Mamdani fuzzy neural network with a hidden layer that employs GT2 FSs was proposed. In [19], a comparison about the prediction of noisy time series between the proposed GT2 neural network (NN), a monolithic network and an IT2 NN revealed the superiority of GT2 models to better manage uncertainty. In [17], the authors developed an edge detection system based on a morphological gradient technique and GT2 FSs.

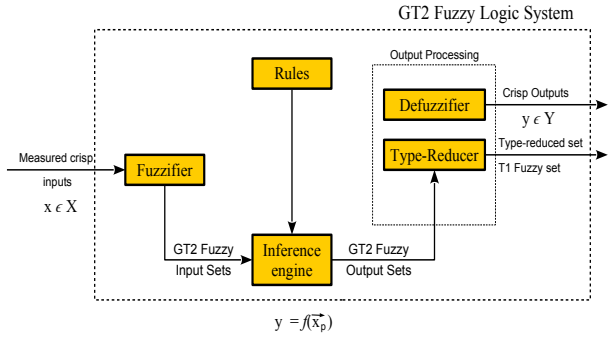

Fig. 1: General Type-2 Fuzzy Logic System (GT2 FLS, Taken from [9]).

According to [17], the proposed GT2 edge detection architecture showed a higher performance than IT2 and T1 FLSs for edge detection when image processing is under high levels of noise. Similar to T1 and IT2 FLSs, a GT2 FLS involves a similar architecture as illustrated in Fig. 1. Specifically, an FLS can be regarded of GT2 if only one of the associated FSs is of GT2 [7]. In this sense, several efforts have been made to represent GT2 FLSs (or T2 FLSs) [20]–[22]. Particularly, horizontal slice-representation allows using everything learned in IT2 FSs theory [9]. According to the $\alpha-$cut decomposition theorem, $\alpha-$cuts decomposition offers a practical way to represent GT2 FLSs (including of IT2 and T1). This is because a GT2 FLS can be represented as the union of all its $\alpha-$planes raised to a level α, where each $\alpha-$plane is the union of its $\alpha-$cuts [9]. Thus, based on the $\alpha-$cuts decomposition theorem, at each input $x' = \vec{x}_p$, a GT2 FLS simultaneously uses $\alpha-$cuts for each vertical slice over the secondary MF domain and the associated $\alpha-$planes [9].

Based on the $\alpha-$plane representation, in this paper a new General Type-2 Radial Basis Function Neural Network (GT2-RBFNN) that is functionally equivalent to a GT2 Mamdani (or TSK) FLS is suggested. To provide a high trade-off between accuracy and model simplicity, two different GT2 RBFNN structures are implemented. On the one hand, to reduce the iterative nature of the Karnik-Mendel method (KM), a GT2 RBFNN with an Enhanced KM algorithm is suggested. On the other hand, three different GT2 RBFNN structures based on direct-defuzzification methods are also presented, i.e. a GT2 RBFNN with a a) Nie-Tan approach, a b) Wu-Mendel Uncertainty bounds method and a c) Biglarbegian-Melek-Mendel procedure. A learning methodology based on an Iterative Information Granulation process (IIG) and an Adaptive Gradient Descent (AGD) approach is implemented to identify the parameters of each antecedent and consequent in the rule base of a GT2 RBFNN. The major contributions of the GT2 RBFNN are twofold. The first contribution is the proposal of a novel RBFNN based on GT2 FSs. Current applications only focus on novel learning methodologies and the implementation of metaheuristics to improve the generalisation properties of the RBFNN. The suggested GT2 RBFNN incorporates GT2 FSs not only to better model and minise the effects of uncertainty, but also to provide a higher level of model accuracy than its counterparts the RBFNN and the IT2 RBFNN. Compared to ensemble of neural networks where uncertainty is viewed as a measure of disagreement among on some inputs, a GT2 RBFNN treats uncertainty as a deficiency that results not only from imprecise boundaries in the FSs of an RBFN and IT2 RBFNN, but also as consequence of information-based imprecision. The second contribution is the proposal of GT2 RBFNN structures based on direct-defuzzification methods and the implementation of an adaptive learning for model simplification and improvement of the convergence of a traditional gradient descent approach.

The rest of this paper is organised as follows: In Sections II and III, a brief review of T2 FSs and the functional equivalence between the RBFNN and GT2 FLSs is provided. Sections IV and V detail the architecture of a GT2-RBFNN with an EKM and three simplified neural structures respectively. In Section V, a parameter identification approach for the GT2 RBFNN models is described. A comparative analysis and a discussion of experiments results are presented in Sections VI and VII correspondingly. Finally, conclusions are drawn in section IX.

II. General Type-2 Fuzzy Logic

This section provides a brief review of General Type-2 Fuzzy Sets (GT2 FSs) and theory of α-plane representation.

A. Definition of a General Type-2 Fuzzy Set

A General Type-2 Fuzzy Set (GT2 FS) denoted by $\tilde{A}$ (also called T2 FS) is characterised by a bivariate MF $\mu_{\tilde{A}}(x,u) \subseteq [0,1]$ on the Cartesian product $\mu_{\tilde{A}} : X \times [0,1]$, where the primary variable is $x \in X$. And the $y-axis$ is called secondary variable or primary MF $u \in J_x \subseteq [0,1]$ as illustrated in Fig. 2. Thus, $\tilde{A}$ is represented by:

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

$\{\mu_{\tilde{A}}(u) | u \in U\}$ is a vertical slice of $\mu_{\tilde{A}}(x,u)$ and it can also be represented by its $\alpha-$cut decomposition.

B. $\alpha-$plane Representation

An $\alpha-$plane for a GT2 FS $\tilde{A}$ is denoted by $\tilde{A}_\alpha$, is the union of the primary MFs of $\tilde{A}$ whose secondary grades are greater than or equal to α $(0 \leq \alpha \leq 1)$

$$\tilde{A}_\alpha = \{(x,u), \mu_{\tilde{A}}(x,u) \geq \alpha | x \in X, u \in [0,1]\} \quad (2)$$

where the lower and upper limits for $\tilde{A}_\alpha$ are defined by

$$\begin{cases} LMF(\tilde{A}_\alpha) = a_{\tilde{\alpha}} \\ UMF(\tilde{A}_\alpha) = b_{\tilde{\alpha}} \end{cases} \quad (3)$$

That means when $\tilde{A}_\alpha$ is raised to level α, it is a plane at that level that can be obtained by connecting all the corresponding $\alpha-cuts$ of the associated vertical slices of the secondary MFs of $x \in X$ [7]. Hence, the horizontal-slice representation of a GT2 FS $\tilde{A}$ is defined by

$$\tilde{A} = \sup_{\alpha \in [0,1]} \alpha / \left[\int_{x \in X} [a_\alpha(x), b_\alpha(x)] / x \right] = \bigcup_{\alpha \in [0,1]} \alpha / \tilde{A}_\alpha \quad (4)$$

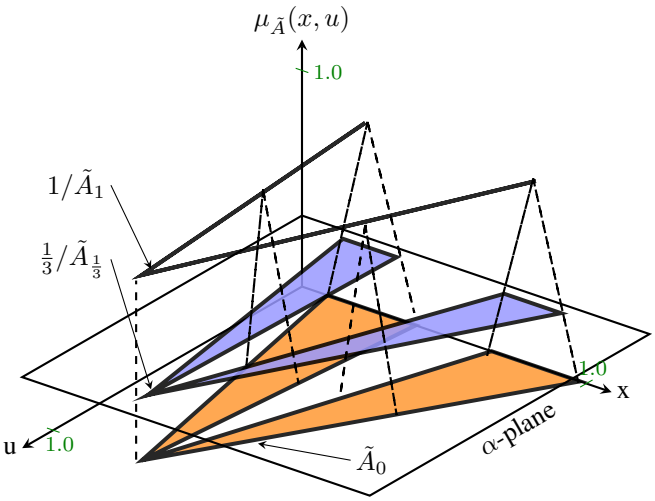

Fig. 2: Some $\alpha-$planes raised to level α for a GT2 FS (Taken from [7]).

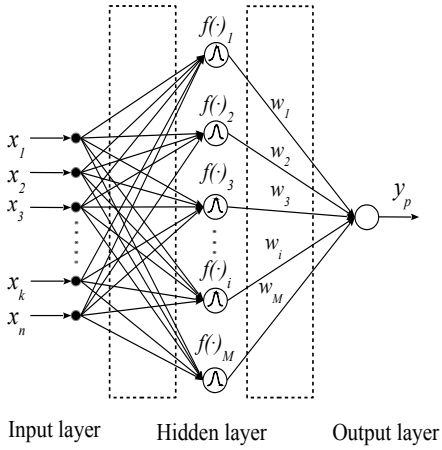

Fig. 3: Radial Basis Function Neural Network (RBFNN, Taken from [23]).

III. Radial Basis Function Neural Network and General Type-2 Fuzzy Logic Systems

It has been proven that under some mild conditions the RBFNN can be viewed as a Type-1 Fuzzy Logic System of either Mamdani or Takagi-Sugeno-Kang type (TSK) [23], [24]. This equivalence has been further extended in [24] in order to design an Interval Type-2 RBFNN (IT2 RBFNN) with a Karnik-Mendel type-reduction in which all the fuzzy sets are of Interval Type-2. An RBFNN can be regarded as an FLSs whose main inference engine is interpreted as an adaptive filter [7], [24]–[26]. It resembles an additive weighted combination of the MFs of the fired-rule output sets in the hidden layer of the RBFNN (See Fig. 3) [7]. Thereby, every hidden receptive unit in the RBFNN is functionally equivalent to a fuzzy rule R^i described by a multi-variable Gaussian MF $\mu_{R^i}(\vec{x}_p, y_p) = \mu_{R^i}[x_1, \ldots, x_n, y]$, where the input vector $\vec{x}_p \in X_1 \times \ldots X_n$ and the implication engine is defined as:

$$\mu_{R^i}(\vec{x}_p, y) = \mu_{A^i \to G^i} = \left[T_{k1}^n \mu_{F_k^i}(x_k) \star \mu_{G^i(y)}\right] \quad (5)$$

Where $\star$ is the minimum $t-norm$ that represents the shortest Euclidean distance to the input vector $\vec{x}_p$. And each receptive unit is the ith fuzzy rule:

$$R^i: \text{IF } x_1 \text{ is } F_1^i \text{ and} \ldots \text{IF } x_k \text{ is } F_k^i \text{ and} \ldots \text{IF } x_n \text{ is } F_n^i \text{ THEN } y \text{ is } G^i;\ i = 1, \ldots, M \quad (6)$$

So that, the firing strength f_i of each receptive unit is

$$\mu_{A^i \to G^i}(\vec{x}_p, y) = \prod_{k=1}^{n} \mu_{F_k^i}(x_k) = f_i\left(\exp\left[-\frac{\sum_{k=1}^{n}(x_k - m_{ki})^2}{\sigma_i^2}\right]\right) \quad (7)$$

where $A_i = F_1^i \times \ldots \times F_n^i$ - m_{ki} and σ_i are the center and width of a multi-variable Gaussian MF respectively. By combining all the rules in the output layer, y_p is [27]

$$y_p = \frac{\sum_{i=1}^{M} \mu_{A^i \to G^i}(\vec{x}_p, y) w_i}{\sum_{i=1}^{M} \mu_{A^i \to G^i}(\vec{x}_p, y)} \quad (8)$$

Strictly speaking, any kind of FLS enhancement might be directly applicable to the RBFNN theory because the structure of its fuzzy rule base in going from T1 FSs to T2 FSs does not change; it is the way the associated antecedents and consequents are modelled [7]. Thus, an RBFNN can be functionally equivalent to a kind of GT2 FLS that is based on the horizontal-slice representation if an RBFNN consists of:

I. An input layer with a singleton fuzzification.
II. The T-norm operator used to compute each rule's firing strength is multiplication (meet).
III. The secondary MF of each GT2 FS is convex.
IV. The $\alpha-$cut of each T1 secondary MF $\tilde{A}$, $\tilde{A}_\alpha$ is given by a set of the lower and upper firing strengths $\left[\underline{f}_i^\alpha, \bar{f}_i^\alpha\right]$ as described in Fig. 4.

The structure of an RBFNN can be viewed as GT2 Takagi-Sugeno-Kang FLS if for each ith fuzzy rule

$$\tilde{R}_\alpha^i: \text{IF } x_1 \text{ is } \tilde{F}_1^i \text{ and} \ldots \text{IF } x_k \text{ is } \tilde{F}_k^i \text{ and} \ldots \text{IF } x_n \text{ is } \tilde{F}_n^i \text{ THEN } y \text{ is } \tilde{g}^i(\vec{x}_p);\ i = 1, \ldots, M \quad (9)$$

whereas for a Mamdani inference fuzzy system, the consequent part is defined as 'y is $\tilde{G}^i$'. For a GT2 RBFNN of Mamdani (TSK) type, when $\vec{x}_p = x_l'$, a vertical slice in the ith receptive unit for the ith antecedent $\tilde{F}_k^i$ is activated, and its $\alpha-$cut decomposition is given by:

$$\tilde{F}_k^i(x_l') \Leftrightarrow \mu_{\tilde{A}^i \to \tilde{G}^i}(\vec{x}_p, y) = \sup_{\alpha \in [0,1]} \alpha / \left[\underline{f}_i^\alpha, \bar{f}_i^\alpha\right] \quad (10)$$

For simplicity, it is used $\underline{f}_i^\alpha(\vec{x}_p) = \underline{f}_i^\alpha$. Hence, the level α firing set in each receptive unit is defined by

$$F_i^\alpha \equiv \left[\underline{f}_i^\alpha, \bar{f}_i^\alpha\right], \alpha \in [0,1] \quad (11)$$

Built upon a horizontal-slice representation, a GT2 RBFNN can be defined as [28]

1. A horizontal-slice Mamdani (TSK) FLS that is analogous to an IT2 FLS where a number of operations described for IT2 FSs theory occur for each horizontal slice [7].
2. A Wagner-Hagras (WH) GT2 RBFNN FLS that results from the union over α of the horizontal-slice Mamdani (TSK) FLSs [14].

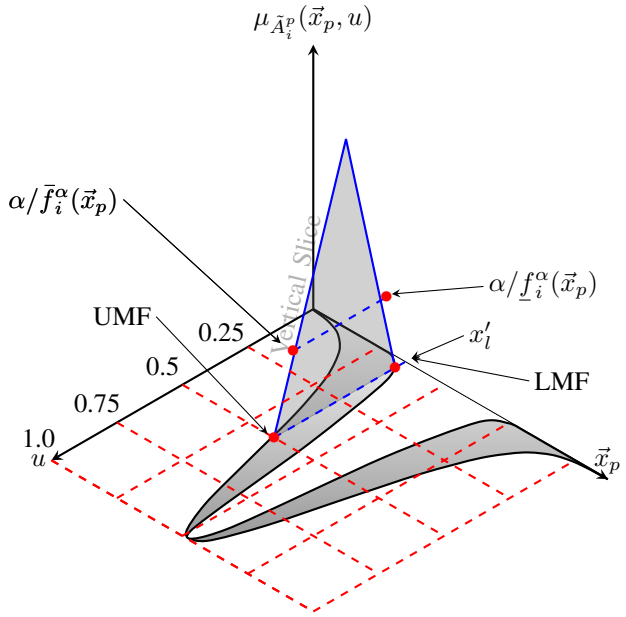

Fig. 4: Singleton fuzzification and triangle secondary MF that is activated when $\vec{x}_p = x_l'$ for the ith receptive unit of the RBFNN

IV. General Type-2 Radial Basis Function Neural Network (GT2 RBFNN)

Based on Fig. 5, this section describes the GT2 RBFNN structure that can be viewed as a Mamdani (TSK) GT2 FLS with an Enhanced Karnik-Mendel (EKM) type-reduction layer, where all the FSs are of GT2. For demonstration purposes, here a GT2 RBFNN with an uncertain width $\sigma_i = [\sigma_i^1, \sigma_i^2]$ and a fixed mean m_k^i is implemented. A horizontal-slice representation is used for the simplest Mamdani (TSK) GT2 RBFNN structure that consists of a singleton fuzzification with a secondary MF that is convex, a Center-Of-Sets (COS) type reduction that uses an EKM and an average of end-points defuzzification (AED) as described in Fig. 6. To avoid additional parameters, and as shown in 4, the secondary MFs are vertical slices, a triangle function is employed where its base is equal to $\bar{f}_i^0 - \underline{f}_i^0$ and its Apex location given by

$$\text{Apex}(\vec{x}_p) = \underline{f}_i^0(\vec{x}_p) + \frac{1}{2} w[\bar{f}_i^0(\vec{x}_p) - \underline{f}_i^0(\vec{x}_p)]; w \in [0,1] \quad (12)$$

A. GT2RBFNN Input Layer

The proposed GT2 RBFNN is a Multi-Input-Single-Output FLS, in which the input data is a multidimensional crisp vector represented by $\vec{x}_p = [x_1, \ldots, x_n] \in R^n$ where only the current state is fed into the layer and then forwarded to next layer.

B. General Type-2 RBF Layer

It is assumed a singleton fuzzification, i.e. for each value x_k only a T1 vertical slice for an antecedent GT2 FS $\tilde{F}_k^i$ is activated. Compared to an IT2 RBFNN, to describe each horizontal slice in the GT2 layer of a GT2 RBFNN, a number of S firing strengths $[\bar{f}_i^{\alpha_s}, \underline{f}_i^{\alpha_s}]$ is required (See Fig. 6) where $\alpha_2 > \alpha_1$. At input $\vec{x}_p$, for each fuzzy rule in the GT2-Mamdani (TSK) RBFNN, only one firing interval $F_i^{\alpha_s}$ is activated for level α_s in the GT2 RBF layer as (See Fig. 6, 7 and 8) [29]:

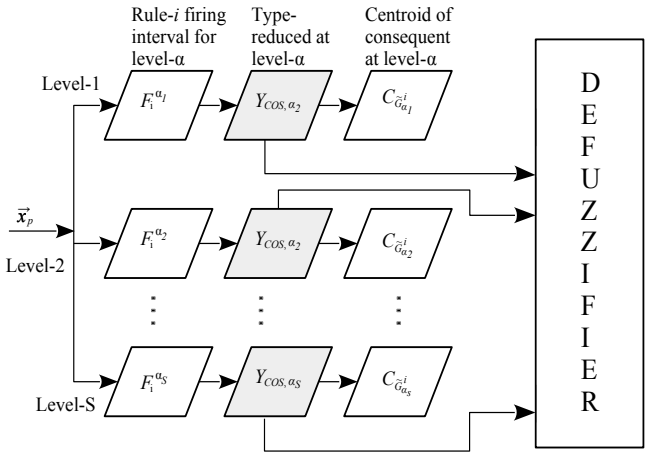

Fig. 5: GT2 Mamdani computations for an RBFNNN (Taken from [7]).

$$F_i^{\alpha_s} := \begin{cases} F_i^{\alpha_s} = [\underline{f}_i^{\alpha_s}(\vec{x}_p), \bar{f}_i^{\alpha_s}(\vec{x}_p)] \\ \underline{f}_i^{\alpha_s}(\vec{x}_p) = \exp\left[-\sum_{k=1}^{n}\left(\frac{x_k - m_k^i}{\sigma_i^2}\right)^2\right]_{\alpha_s} \\ \bar{f}_i^{\alpha_s}(\vec{x}_p) = \exp\left[-\sum_{k=1}^{n}\left(\frac{x_k - m_k^i}{\sigma_i^1}\right)^2\right]_{\alpha_s} \end{cases} \quad (13)$$

Note the term α is not a variable [7]. The subscript $'s'$ is used to denote each α-level in the GT2 RBFNN.

C. Type-reduction Layer

In the type reduction layer, a Center Of Sets Type Reduction (COS TR) is used. This layer performs a mathematical operation that maps a GT2 FS into a T1 FS. Hence, the centroid of each consequent at the α_s-plane is computed as:

$$C_{\tilde{G}_{\alpha_s}^i} = \alpha_s / [w_{l,\alpha_s}^i, w_{r,\alpha_s}^i] \quad (14)$$

According to [7], [29], for a Mamdani GT2 RBFNN, $[w_{l,\alpha_s}^i, w_{r,\alpha_s}^i]$ is an Interval Weighted Average (IWA) that is used along with the firing interval $F_i^{\alpha_s}$ to compute the reduced set $[y_l^{\alpha_s}(\vec{x}_p), y_r^{\alpha_s}(\vec{x}_p)]$ for α_s-level as:

$$y_l^{\alpha_s} = \frac{\sum_{i=1}^{L_{\alpha_s}} w_{l,\alpha_s}^i \bar{f}_i^{\alpha_s} + \sum_{i=L_{\alpha_s}+1}^{M} w_{l,\alpha_s}^i \underline{f}_i^{\alpha_s}}{\sum_{i=1}^{L_{\alpha_s}} \bar{f}_i^{\alpha_s} + \sum_{i=L_{\alpha_s}+1}^{M} \underline{f}_i^{\alpha_s}} \quad (15)$$

$$y_r^{\alpha_s} = \frac{\sum_{i=1}^{R_{\alpha_s}} w_{r,\alpha_s}^i \underline{f}_i^{\alpha_s} + \sum_{i=R_{\alpha_s}+1}^{M} w_{r,\alpha_s}^i \bar{f}_i^{\alpha_s}}{\sum_{i=1}^{R_{\alpha_s}} \underline{f}_i^{\alpha_s} + \sum_{i=R_{\alpha_s}+1}^{M} \bar{f}_i^{\alpha_s}} \quad (16)$$

where $Y_{COS,\alpha_s} = 1/[y_l^{\alpha_s}(\vec{x}_p), y_r^{\alpha_s}(\vec{x}_p)]$. For a TSK GT2 RBFNN, a normalised $A2-C0$ GT2 FLS version is used in which the antecedents are GT2 FSs, and the associated consequent is $g_{i,\alpha_s} = c_0^{i,\alpha_s} x_0 + c_1^{i,\alpha_s} x_1 + \ldots + c_n^{i,\alpha_s} x_n$, where $x_0 = 1$ and $c_m^{i,\alpha_s}, (m = 0, \ldots, n)$ are crips numbers.

$$y_l^{\alpha_s} = \frac{\sum_{i=1}^{L_\alpha} g_{i,\alpha_s} \bar{f}_i^{\alpha} + \sum_{i=L_\alpha+1}^{M} g_{i,\alpha_s} \underline{f}_i^{\alpha}}{\sum_{i=1}^{L} \bar{f}_i^{\alpha} + \sum_{i=L_\alpha+1}^{M} \underline{f}_i^{\alpha}} \quad (17)$$

$$y_r^{\alpha_s} = \frac{\sum_{i=1}^{R_\alpha} g_{i,\alpha_s} \underline{f}_i^{\alpha} + \sum_{i=R_\alpha+1}^{M} g_{i,\alpha_s} \bar{f}_i^{\alpha}}{\sum_{i=1}^{R_\alpha} \underline{f}_i^{\alpha} + \sum_{i=R_\alpha+1}^{M} \bar{f}_i^{\alpha}} \quad (18)$$

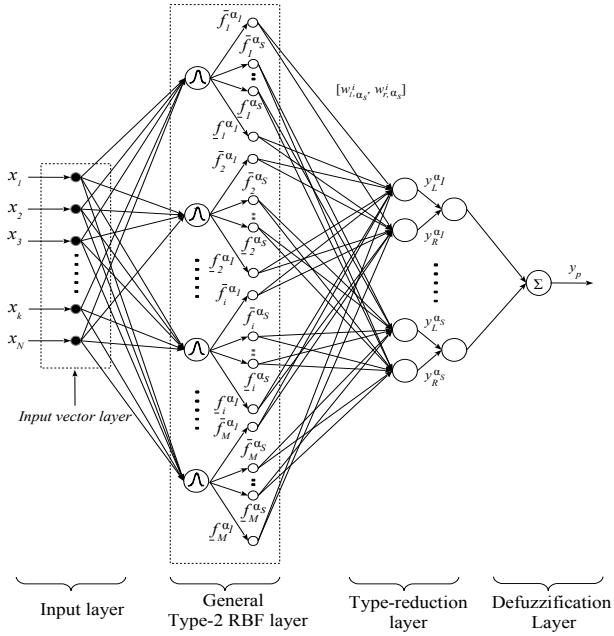

Fig. 6: GT2 RBFNN with an EKM type-reduction layer.

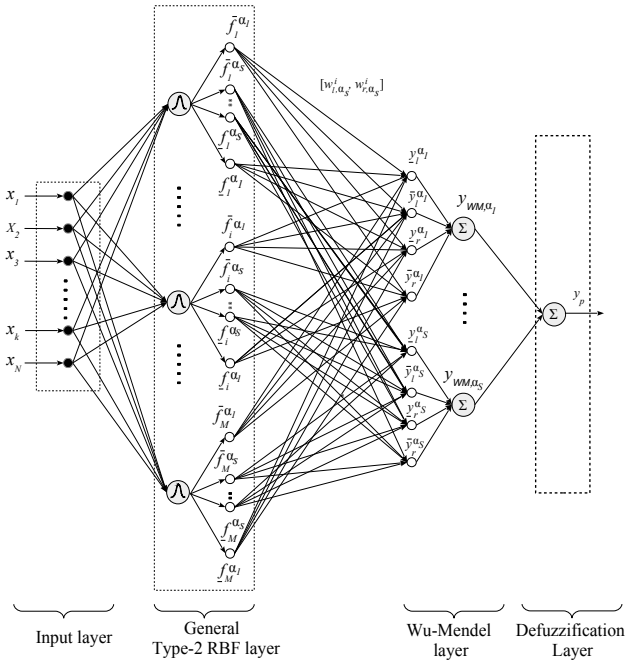

Fig. 7: Wu-Mendel GT2 RBFNN.

D. Defuzzification Layer

This layer performs defuzzification that consists of a process of aggregation of all horizontal slices. Here, the Average of End-Points Defuzzification (AEPD) is used [14]:

$$y_p(\vec{x}_p) = \sum_{s=1}^{S} \alpha_s [(y_l^{\alpha_s}(\vec{x}_p) + y_r^{\alpha_s}(\vec{x}_p))/2] \Big/ \sum_{s=1}^{S} \alpha_s \quad (19)$$

V. Simplified General Type-2 Radial Basis Function Neural Network

In this paper, a GT2 RBFNN that employs a direct-defuzzification algorithm as an output layer is called Simplified General Type-2 Radial Basis Function Neural Network (SGT2 RBFNN). For practical reasons, particularly for real world T2 FLSs, the need to bypass the iterative nature of KM algorithms that results from the number of permutations that are needed to calculate the reduced set has become a priority. Type-reduction is usually used as going from a T2 FS to a T1 FS [30]. In this paper, the term direct-defuzzification and closed-form type reduction are used indistinctly to refer to the mapping that goes from a GT2 FS to a crisp number (type-0). Due to their simplicity and accuracy with respect to KM algorithms, in this paper three popular direct-defuzzification approaches [30] are selected, i.e. a) Nie-Tan closed-form (NT) [31], b) Wu-Mendel Uncertain Bounds approach (WU) [32] and c) the Biglarbegian-Melek-Mendel method (BMM) [33].

A. Simplified Wu-Mendel GT2-RBFNN

The second simplified structure is a Mamdani GT2-RBFNN that employs the Wu-Mendel Uncertain Bounds method and that is called WM GT2-RBFNN for short. For each α-level in the GT2-RBFNN, the WM method replaces the type reduction with an approach that calculates the inner and outer-bound sets for the type reduced of IT2 FLSs [32]. As shown in Fig. 7, for each input vector $\vec{x}_p$, the WM GT2RBFNN output is

$$y_p(\vec{x}_p) = \sum_{s=1}^{S} \alpha_s y_{WM,\alpha_s} \Big/ \sum_{s=1}^{S} \alpha_s \quad (20)$$

For each $\alpha-$level, y_{WM,α_s} is computed as:

$$y_{WM,\alpha_s} = \frac{1}{4}\left(\underline{y}_l^{\alpha_s}(\vec{x}_p) + \overline{y}_l^{\alpha_s}(\vec{x}_p) + \underline{y}_r^{\alpha_s}(\vec{x}_p) + \overline{y}_r^{\alpha_s}(\vec{x}_p)\right) \quad (21)$$

where

$$\underline{y}_l^{\alpha_s} = \overline{y}_l^{\alpha_s} - \left[F_p \times \frac{\sum_{i=1}^{M} \underline{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^{i} - w_{l,\alpha_s}^{1}\right) \sum_{i=1}^{M} \bar{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^{M} - w_{l,\alpha_s}^{i}\right)}{\sum_{i=1}^{M} \underline{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^{i} - w_{l,\alpha_s}^{1}\right) + \sum_{i=1}^{M} \bar{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^{M} - w_{l,\alpha_s}^{i}\right)}\right] \quad (22)$$

$$\overline{y}_r^{\alpha_s} = \underline{y}_r^{\alpha_s} + \left[F_p \times \frac{\sum_{i=1}^{M} \bar{f}_i^{\alpha_s}\left(w_{r,\alpha_s}^{i} - w_{r,\alpha_s}^{1}\right) \sum_{i=1}^{M} \underline{f}_i^{\alpha_s}\left(w_{r,\alpha_s}^{M} - w_{r,\alpha_s}^{i}\right)}{\sum_{i=1}^{M} \bar{f}_i^{\alpha_s}\left(w_{r,\alpha_s}^{i} - w_{r,\alpha_s}^{1}\right) + \sum_{i=1}^{M} \underline{f}_i^{\alpha_s}\left(w_{r,\alpha_s}^{M} - w_{r,\alpha_s}^{i}\right)}\right] \quad (23)$$

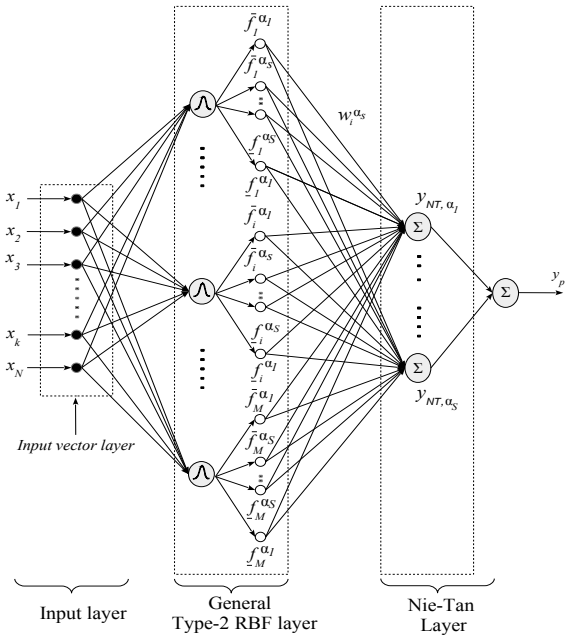

Fig. 8: Nie-Tan GT2 RBFNN.

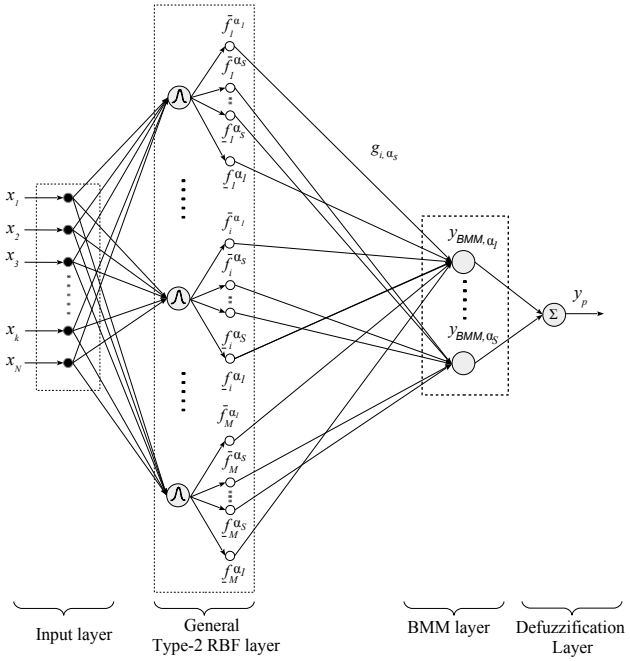

Fig. 9: Biglarbegian-Melek-Mendel GT2 RBFNN.

Where $F_p = \sum_{i=1}^{M} \left(\bar{f}_i^{\alpha_s} - \underline{f}_i^{\alpha_s}\right) / \sum_{i=1}^{M} \bar{f}_i^{\alpha_s} \sum_{i=1}^{M} \underline{f}_i^{\alpha_s}$. in which, consequents w_{i,α_s} are different for each $\alpha-$level. The terms $\overline{y}_l^{\alpha_s}$ and $\underline{y}_r^{\alpha_s}$ are given by

$$\overline{y}_l^{\alpha_s} = \min\left\{\sum_{i=1}^{M} \underline{f}_i^{\alpha_s} w_{l,\alpha_s}^i / \sum_{i=1}^{M} \underline{f}_i^{\alpha_s}, \sum_{i=1}^{M} \bar{f}_i^{\alpha_s} w_{l,\alpha_s}^i / \sum_{i=1}^{M} \bar{f}_i^{\alpha_s}\right\} \tag{24}$$

$$\underline{y}_r^{\alpha_s} = \max\left\{\sum_{i=1}^{M} \bar{f}_i^{\alpha_s} w_{r,\alpha_s}^i / \sum_{i=1}^{M} \bar{f}_i^{\alpha_s}, \sum_{i=1}^{M} \underline{f}_i^{\alpha_s} w_{r,\alpha_s}^i / \sum_{i=1}^{M} \underline{f}_i^{\alpha_s}\right\} \tag{25}$$

B. Simplified Nie-Tan GT2-RBFNN

The second structure is a GT2-RBFNN that uses the Nie-Tan method as a direct-defuzzification layer as illustrated in Fig. 8. The Nie-Tan is a direct-defuzzification method initially developed for IT2 FLSs. Such method uses the vertical representation of the Footprint of Uncertainty (FOU) [31] before the process of dedifuzzification to finally compute the centroid of the IT2 FS. The NT layer can be considered a zero order Taylor series approximation of Karnik-Mendel+dedifuzzification methods. It has been proved the Nie-Tan operator is equivalent to an exhaustive and accurate type-reduction for both discrete and continuous IT2 FSs [31]. Although there has been improvements on the Nie-Tan operator, in this paper, the centroid y_{NT,α_s} at each $\alpha-$level is calculated as:

$$y_{NT,\alpha_s} = \frac{\sum_{i=1}^{M} w_i^{\alpha_s} \left(\underline{f}_i^{\alpha_s} + \bar{f}_i^{\alpha_s}\right)}{\sum_{i=1}^{M} \underline{f}_i^{\alpha_s} + \sum_{i=1}^{M} \bar{f}_i^{\alpha_s}} \tag{26}$$

For each input vector $\vec{x}_p$, the NT GT2RBFNN output $y_p(\vec{x}_p)$ is calculated as:

$$y_p(\vec{x}_p) = \sum_{s=1}^{S} \alpha_s y_{NT,\alpha_s} \Big/ \sum_{s=1}^{S} \alpha_s \tag{27}$$

C. Simplified Biglarbegian-Melek-Mendel GT2-RBFNN

As an alternative to computing the output of a TSK GT2-RBFNN is the Biglarbegian-Melek-Mendel closed form equation [33]. The last simplified neural structure that is called TSK BMM GT2-RBFNN for short. According to Fig. 9 y_p is:

$$y_p(\vec{x}_p) = \sum_{s=1}^{S} \alpha_s y_{BMM,\alpha_s} \Big/ \sum_{s=1}^{S} \alpha_s \tag{28}$$

for each α_s-plane:

$$y_{BMM,\alpha_s} = m_{\alpha_s} y_m^{\alpha_s} + n_{\alpha_s} y_n^{\alpha_s} \tag{29}$$

In which $y_m^{\alpha_s} = \sum_{i=1}^{M} g_{i,\alpha_s} \underline{f}_i^{\alpha_s} / \sum_{i=1}^{M} \underline{f}_i^{\alpha_s}$ and $y_n^{\alpha_s} = \sum_{i=1}^{M} g_{i,\alpha_s} \bar{f}_i^{\alpha_s} / \sum_{i=1}^{M} \bar{f}_i^{\alpha_s}$. The terms $\underline{f}_i^{\alpha_s}$ and $\bar{f}_i^{\alpha_s}$ are calculated using (13), and consequents g_{i,α_s} and adaptation parameters m_{α_s} and n_{α_s} are different for each $\alpha-$level, where $g_{i,\alpha_s} = c_0^{i,\alpha_s} x_0 + \ldots c_m^{i,\alpha_s} x_m$ for $x_0 = 1$.

VI. Learning Methodology of the GT2 RBFNN

To identify the optimal parameters of the GT2 RBFNN, and its neural structure, a two-stage learning methodology based on the concept of Iterative Information Granulation (IIG) and an Adaptive Gradient Descent (AGD) approach is implemented. IIG is a clustering technique whose main essence is to discover a structure in data while producing representatives called granules [34]. Such granules are formed based on a data compatibility measure, and their geometrical properties are used to estimate the initial values of each antecedent in the GT2 RBFNN (See flow diagram, Fig. 10). Similarly to [26], the number of fuzzy rules or hidden units in the GT2 RBFNN is initially approximated by using the gradient of the compatibility curve that is obtained by the IIG.

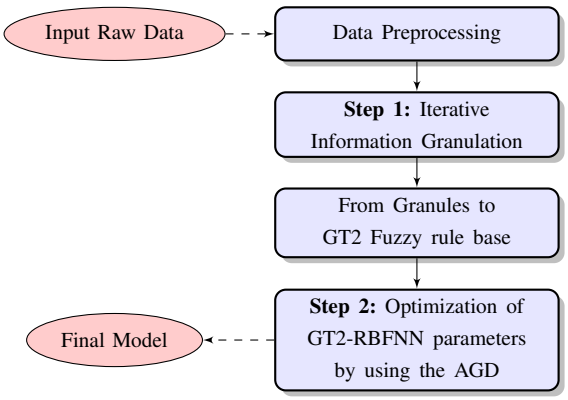

Fig. 10: Parameter identification applied to the GT2 RBFNN.

In a second stage, AGD is applied to optimise the parameters $[\sigma_i^1, \sigma_i^2]$, and m_k^i and to determine the optimal number of fuzzy rules according to cross-validation results.

A. Iterative Information Granulation

In this work, IIG is used not only to granulate/cluster data (Fig. 11), but also used as an approximation to the optimal number of fuzzy rules (hidden nodes) in the GT2 RBFNN as well as the initial values for $[\sigma_i^1, \sigma_i^2]$ and m_k^i of each MF [26]. The process of IIG is based on a compatibility index $C(A, B)$ that defines how good is the merging operation of any two granules A and B. IIG consists of two main steps: [34], [35]:

- Find the two most 'compatible' information granules A and B by using Eq. (30) and merge them together as a new information granule $g_i = (l_{ki}, u_{ki})$ [26]. Where g_i is defined by its lower and upper corners (l_{ik}, u_{ik}) for the dimension $'k'$ and $i = 1, \ldots, M$.
- Repeat the process of finding the two most compatible granules until a satisfactory data abstraction level is achieved. Where the compatibility 'C' is defined as [34]:

$$C(A,B) = D_{MAX} - d_{A,B} \cdot e^{\left(-\alpha_g \frac{card_{A,B}/Cardinality_{MAX}}{L_{A,B}/Length_{MAX}}\right)} \tag{30}$$

Such as D_{MAX}, $Length_{MAX}$ and the term $Cardinality_{MAX}$ is the maximum possible distance and length of a granule and the total number of granules in the data set respectively. $d_{A,B}$ is the weighted multidimensional average distance of the resulting granule with w_k playing the importance weight for the dimension k. In Eq. (30), α_g weights the requirements between distance and cardinality/length and L_{AB} is the multidimensional length of the resulting granule g_i, where:

$$d_{A,B} = \frac{1}{n}\sum_{k=1}^{n} w_k(max(u_{Ak}, u_{Bk}) - min(l_{Ak}, l_{Bk})) \tag{31}$$

g_i is used as a fuzzy constraint to extract the initial parameters of LMF and UMF (m_k and σ_i) which are calculated as:

$$m_k^i = \frac{1}{2}(l_{ik} - u_{ik});\ m_k^i = [m_1^i, ..., m_n^i];\ i = 1, \ldots, M \tag{32}$$

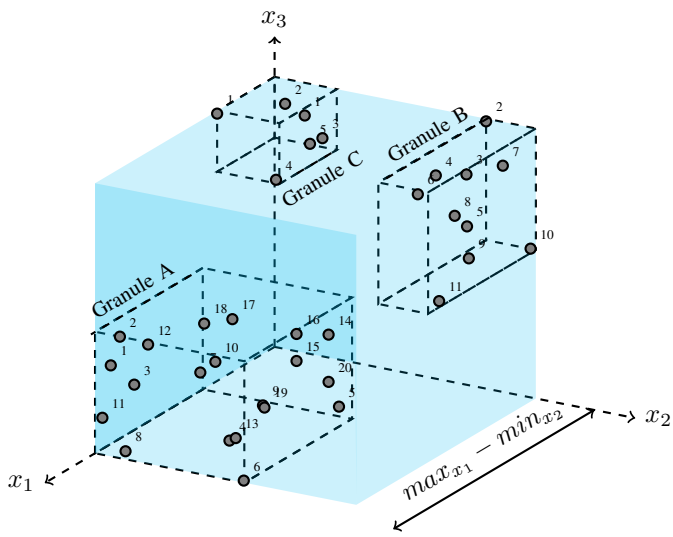

Fig. 11: 3-D Example of final Data space for a set of granules A, B and C.

$$(\sigma_i^1)^2 = \frac{1}{r}\left(\sum_{j=1}^{r} \|m_k^j - m_k^i\|\right)^{1/2} \quad j \neq i;\ \sigma_i^2 = \sigma_i^1 - \Delta\sigma_i \tag{33}$$

in which $j \neq i$, and j is the nearest neighbour to the ith fuzzy rule, and $r \geq 2$ [36].

B. Adaptive Gradient Descent Approach (AGD)

After structure identification, the common parameters m_k^i and $[\sigma_i^1, \sigma_i^2]$ of the antecedent GT2 MFs as well as the weighting factors $[w_{l,\alpha_s}^i, w_{r,\alpha_s}^i]$ at each $\alpha-$level of a Mamdani (TSK) GT2 RBFNN should be optimised. Here, an Adaptive Gradient Descent (AGD) approach that evaluates the Root-Mean-Square Error (RMSE) and uses a performance index $P_i = \frac{1}{P}\sum_{p=1}^{P} e_p^2$ is applied [37]. For each p input-output training data $(\vec{x}_p, d_p)$; $p = 1, \ldots, P$, a cost function $E_p = \frac{1}{2}(e_p^2)$ is also defined, where the error $e_p = (y_p(\vec{x}_p) - d_p)$, and d_p is the desired pattern. To increase the convergence performance of a typical Gradient Descent approach and avoid getting trapped in a local minimum, a momentum term γ is introduced. A self-tuning learning rate β is defined to enhance the learning performance of the GT2-RBFNN. As feedback information, at each current and previous learning iteration 't', the change trend of P_i is evaluated and used to adjust the value of γ and β as follows:

- if $Pi(t+1) \geq P_i(t)$ Then

$$\beta(t+1) = h_d\alpha(t),\ \gamma(t+1) = 0$$

- if $Pi(t+1) < P_i(t)$ and $\left|\frac{\Delta Pi}{Pi(t)}\right| < \delta$ Then

$$\beta(t+1) = h_i\alpha(t),\ \gamma(t+1) = \gamma_0 \tag{34}$$

- if $Pi(t+1) < P_i(t)$ and $\left|\frac{\Delta Pi}{P_i(t)}\right| \geq \delta$ Then

$$\beta(t+1) = \beta(t),\ \gamma(t+1) = \gamma(t)$$

where $h_d, (0 < h_d < 1)$ and $h_i, (1 < h_i)$ are the decreasing and increasing factors respectively and δ is a threshold rate for the RMSE. Thus, by using an EKM type reduction, at each $\alpha-$level of a GT2 RBFNN, the AGD must be able to track the corresponding parameters σ_i and m_k^i in the antecedent active branch in which the value of L_α and R_α may change [24]. As pointed out in section III, a GT2 RBFNN is analogous to an IT2 FLS where all the IT2 FS computations occur for each horizontal slice and their aggregation is carried out by a defuzzification process [7]. Hence, for each $\alpha-$level, the final AGD equations for the consequents $[w_{l,\alpha}, w_{r,\alpha}]$ of a Mamdani GT2-RBFNN are updated as:

$$\Delta w_{l,\alpha_s}^i(p+1) = -\beta \frac{\partial E_p(\vec{x}_p)}{\partial w_{l,\alpha_s}^i} + \gamma \Delta w_{l,\alpha_s}^i(p) \quad (35)$$

$$\Delta w_{r,\alpha_s}^i(p+1) = -\beta \frac{\partial E_p(\vec{x}_p)}{\partial w_{r,\alpha_s}^i} + \gamma \Delta w_{r,\alpha_s}^i(p) \quad (36)$$

For a TSK GT2-RBFNN, the consequent coefficients c_m^{i,α_s} are updated according to

$$\Delta c_m^{i,\alpha_s}(p+1) = -\beta \frac{\partial E_p(\vec{x}_p)}{\partial c_m^{i,\alpha_s}} + \gamma \Delta c_m^{i,\alpha_s}(p) \quad (37)$$

To update the common parameters m_k^i and $\left[\sigma_i^1, \sigma_i^2\right]$

$$\Delta m_k^i(p+1) = -\beta \frac{\partial E_p(\vec{x}_p)}{\partial m_k^i} + \gamma \Delta m_k^i(p) \quad (38)$$

$$\Delta \sigma_i^1(p+1) = -\beta \frac{\partial E_p(\vec{x}_p)}{\partial \sigma_i^1} + \gamma \Delta \sigma_i^1(p) \quad (39)$$

$$\Delta \sigma_i^2(p+1) = -\beta \frac{\partial E_p(\vec{x}_p)}{\partial \sigma_i^2} + \gamma \Delta \sigma_i^2(p) \quad (40)$$

Therefore, the derivatives $\partial E_p(\vec{x}_p)/\partial m_k^i$, $\partial E_p(\vec{x}_p)/\partial \sigma_i^1$ and $\partial E_p(\vec{x}_p)/\partial \sigma_i^2$ should equal the addition of their updates for each $\alpha-$level as follows

$$\frac{\partial E_p}{\partial m_k^i} = \sum_{s=1}^{S} \left.\frac{\partial E_p}{\partial m_k^i}\right|_{\alpha_s} \quad (41)$$

where $\frac{\partial E_p}{\partial m_k^i}|_{\alpha_s}$ is the partial derivative of E_p with respect to the parameter m_k^i at the sth $\alpha-$level.

$$\left.\frac{\partial E_p}{\partial m_k^i}\right|_{\alpha_s} = 2 d_\alpha e_p \left[\left(\frac{\partial y_l^{\alpha_s}}{\partial \bar{f}_i^{\alpha_s}} + \frac{\partial y_r^{\alpha_s}}{\partial \bar{f}_i^{\alpha_s}} \right) \frac{\partial \bar{f}_i^{\alpha_s}}{\partial m_k^i} + \left(\frac{\partial y_l^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} + \frac{\partial y_r^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} \right) \frac{\partial \underline{f}_i^{\alpha_s}}{\partial m_k^i} \right] \quad (42)$$

Where $\partial E_p(\vec{x}_p)/\partial y_p(\vec{x}_p) = \alpha_s 2 \sum_{s=1}^{S} \alpha_s$, and the derivative $\partial y_p(\vec{x}_p)/\partial y_l^{\alpha_s} = \partial y_p(\vec{x}_p)/\partial y_r^{\alpha_s}$. $d_\alpha = \alpha_s / 4 \sum_{s=1}^{S} \alpha_s$ is used to simplify notation. To update σ_i^1 and σ_i^2

$$\frac{\partial E_p}{\partial \sigma_i^1} = \sum_{s=1}^{S} \left.\frac{\partial E_p}{\partial \sigma_i^1}\right|_{\alpha_s}; \quad \frac{\partial E_p}{\partial \sigma_i^2} = \sum_{s=1}^{S} \left.\frac{\partial E_p}{\partial \sigma_i^2}\right|_{\alpha_s} \quad (43)$$

where

$$\left.\frac{\partial E_p}{\partial \sigma_i^1}\right|_{\alpha_s} = 2 d_\alpha e_p \left[\left(\frac{\partial y_l^{\alpha_s}}{\partial \bar{f}_i^{\alpha_s}} + \frac{\partial y_r^{\alpha_s}}{\partial \bar{f}_i^{\alpha_s}} \right) \frac{\partial \bar{f}_i^{\alpha_s}}{\partial \sigma_i^1} \right] \quad (44)$$

$$\left.\frac{\partial E_p}{\partial \sigma_i^2}\right|_{\alpha_s} = 2 d_\alpha e_p \left[\left(\frac{\partial y_l^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} + \frac{\partial y_r^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} \right) \frac{\partial \underline{f}_i^{\alpha_s}}{\partial \sigma_i^2} \right] \quad (45)$$

where the derivatives $\partial \bar{f}_i^{\alpha_s}/\partial \sigma_i^2$ and $\partial \underline{f}_i^{\alpha_s}/\sigma_i^1$ are zero. At each $\alpha-$plane, the AGD approach tracks each permutation that results for the implementation of an EKM method in order to calculate the derivatives with respect to each weight $\partial E_p(\vec{x}_p)/\partial w_{l,\alpha_s}^i$ and $\partial E_p(\vec{x}_p)/\partial w_{r,\alpha_s}^i$ and the coefficients c_m^{i,α_s} of a GT2 RBFNN of Mamdani and TSK type respectively. In [24-26], a detailed description of this calculation for an IT2 fuzzy NN and an IT2 RBFNN with a Mamdani inference and using a KM method is provided.

C. AGD for simplified GT2-RBFNN

Compared to a GT2-RBFNN that utilises an EKM algorithm, direct-defuzzification-based structures do not need a sorting process. Thereby, the implementation of the AGD to identify the parameters of a GT2-RBFNN of Mamdani (TSK) type results much simpler.

1) Wu-Mendel GT2-RBFNN: To update the parameters of a WM GT2RBFNN of Mamdani type, the derivatives with respect to the weighting factors w_{l,α_s}^i and w_{r,α_s}^i, and common parameters σ_i^1, σ_i^2, and m_k^i are:

$$\frac{\partial E_p}{\partial w_{l,\alpha_s}^i} = d_\alpha e_p \frac{\partial y_{WM,\alpha_s}}{\partial w_{l,\alpha_s}^i}; \quad \frac{\partial E_p}{\partial w_{r,\alpha_s}^i} = d_\alpha e_p \frac{\partial y_{WM,\alpha_s}}{\partial w_{r,\alpha_s}^i} \quad (46)$$

$$\frac{\partial E_p}{\partial m_k^i} = \sum_{s=1}^{S} \left.\frac{\partial E_p}{\partial m_k^i}\right|_{\alpha_s} \quad (47)$$

so that:

$$\left.\frac{\partial E_p}{\partial m_k^i}\right|_{\alpha_s} = d_\alpha e_p \left[\left(\frac{\partial \underline{y}_l^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} + \frac{\partial \overline{y}_l^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} + \frac{\partial \underline{y}_r^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} + \frac{\partial \overline{y}_r^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} \right) \frac{\partial \underline{f}_i^{\alpha_s}}{\partial m_k^i} \right] \quad (48)$$

where $d_\alpha = \alpha_s / 4 \sum_{s=1}^{S} \alpha_s$, and to update $[\sigma_i^1, \sigma_i^2]$

$$\frac{\partial E_p}{\partial \sigma_i^1} = \sum_{s=1}^{S} \left.\frac{\partial E_p}{\partial \sigma_i^1}\right|_{\alpha_s}; \quad \frac{\partial E_p}{\partial \sigma_i^2} = \sum_{s=1}^{S} \left.\frac{\partial E_p}{\partial \sigma_i^2}\right|_{\alpha_s} \quad (49)$$

so that

$$\left.\frac{\partial E_p}{\partial \sigma_i^1}\right|_{\alpha_s} = d_\alpha e_p \left[\left(\frac{\partial \underline{y}_l^{\alpha_s}}{\partial \bar{f}_i^{\alpha_s}} + \frac{\partial \overline{y}_l^{\alpha_s}}{\partial \bar{f}_i^{\alpha_s}} + \frac{\partial \underline{y}_r^{\alpha_s}}{\partial \bar{f}_i^{\alpha_s}} + \frac{\partial \overline{y}_r^{\alpha_s}}{\partial \bar{f}_i^{\alpha_s}} \right) \frac{\partial \bar{f}_i^{\alpha_s}}{\partial \sigma_i^1} \right] \quad (50)$$

$$\left.\frac{\partial E_p}{\partial \sigma_i^2}\right|_{\alpha_s} = d_\alpha e_p \left[\left(\frac{\partial \underline{y}_l^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} + \frac{\partial \overline{y}_l^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} + \frac{\partial \underline{y}_r^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} + \frac{\partial \overline{y}_r^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} \right) \frac{\partial \underline{f}_i^{\alpha_s}}{\partial \sigma_i^2} \right] \quad (51)$$

Where the term $\partial \overline{y}_l / \partial w_{l,\alpha_s}^i$ and $\partial \underline{y}_l / \partial w_{l,\alpha_s}^i$ is

$$\frac{\partial \overline{y}_l}{\partial w_{l,\alpha_s}^i} = \begin{cases} \underline{f}_i / \sum_{i=1}^{M} \underline{f}_i, & \min\left\{y_l^{(0)}, y_l^{(M)}\right\} = y_l^{(0)} \\ \underline{f}_i / \sum_{i=1}^{M} \underline{f}_i, & \min\left\{y_l^{(0)}, y_l^{(M)}\right\} = y_l^{(M)} \end{cases} \quad (52)$$

$$\frac{\partial \underline{y}_l^{\alpha_s}}{\partial w_{l,\alpha_s}^i} = \frac{\partial \overline{y}_l^{\alpha_s}}{\partial w_{l,\alpha_s}^i} - (F_p)\left[\frac{\left(\underline{f}_i^{\alpha_s}\right)\sum_{i=1}^{M}\left(\bar{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^M - w_{l,\alpha_s}^i\right)\right) - \left(\bar{f}_i^{\alpha_s}\right)\sum_{i=1}^{M}\left(\underline{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^i - w_{l,\alpha_s}^1\right)\right)}{\left(\sum_{i=1}^{M}\underline{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^i - w_{l,\alpha_s}^1\right) + \sum_{i=1}^{M}\bar{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^M - w_{l,\alpha_s}^i\right)\right)^2}\right] \tag{53}$$

A similar procedure is used to calculate the terms $\partial \overline{y}_r^{\alpha_s}/\partial w_{r,\alpha_s}^i$ and $\partial \underline{y}_r^{\alpha_s}/\partial w_{r,\alpha_s}^i$. To exemplify the computation of the derivatives $\partial \bar{f}_i^{\alpha_s}/\partial m_k^i$, $\partial \underline{f}_i^{\alpha_s}/\partial m_k^i$, $\partial \underline{y}_r^{\alpha_s}/\partial \bar{f}_i^{\alpha_s}$, $\partial \overline{y}_r^{\alpha_s}/\partial \underline{f}_i^{\alpha_s}$, $\partial \overline{y}_r^{\alpha_s}/\partial \bar{f}_i^{\alpha_s}$ and $\partial \underline{y}_r^{\alpha_s}/\partial \underline{f}_i^{\alpha_s}$, the calculation of $\partial \overline{y}_l^{\alpha_s}/\partial \underline{f}_i^{\alpha_s}$ and $\partial \underline{y}_l^{\alpha_s}/\partial \underline{f}_i^{\alpha_s}$ is shown below:

$$\frac{\partial \overline{y}_l^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} = \begin{cases} \dfrac{w_{r,\alpha_s}^i\left(\sum_{i=1}^{M}\bar{f}_i^{\alpha_s}\sum_{i=1}^{M}\underline{f}_i^{\alpha_s}\right) - \left(\sum_{i=1}^{M}\left(\bar{f}_i^{\alpha_s} - \underline{f}_i^{\alpha_s}\right)\sum_{i=1}^{M}\bar{f}_i^{\alpha_s}\right)}{\left(\sum_{i=1}^{M}\bar{f}_i^{\alpha_s}\sum_{i=1}^{M}\underline{f}_i^{\alpha_s}\right)^2}, & \min\left\{y_r^{(0),\alpha_s}, y_r^{(M),\alpha_s}\right\} = y_r^{(0),\alpha_s} \\ 0, & \min\left\{y_r^{(0),\alpha_s}, y_r^{(M),\alpha_s}\right\} = y_r^{(M),\alpha_s} \end{cases} \tag{54}$$

$$\frac{\partial \underline{y}_l^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} = \frac{\overline{y}_l^{\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} + \left[\left(\frac{-v_q\left(\sum_{i=1}^{M}\bar{f}_i^{\alpha_s}\sum_{i=1}^{M}\underline{f}_i^{\alpha_s}\right) - \sum_{i=1}^{M}\left(\bar{f}_i^{\alpha_s} - \underline{f}_i^{\alpha_s}\right)\sum_{i=1}^{M}\bar{f}_i^{\alpha_s}}{\left(\sum_{i=1}^{M}\bar{f}_i^{\alpha_s}\sum_{i=1}^{M}\underline{f}_i^{\alpha_s}\right)^2}\right) + v_r(w_{l,\alpha_s}^i - w_{l,\alpha_s}^1)\left(\frac{\sum_{i=1}^{M}\left(\bar{f}_i^{\alpha_s} - \underline{f}_i^{\alpha_s}\right)}{\sum_{i=1}^{M}\bar{f}_i^{\alpha_s}\sum_{i=1}^{M}\underline{f}_i^{\alpha_s}}\right)\right] \tag{55}$$

in which, the terms v_q and v_r are:

$$v_q = \left(\frac{\sum_{i=1}^{M}\underline{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^i - w_{l,\alpha_s}^1\right)\sum_{i=1}^{M}\bar{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^M - w_{l,\alpha_s}^i\right)}{\sum_{i=1}^{M}\underline{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^i - w_{l,\alpha_s}^1\right) + \sum_{i=1}^{M}\bar{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^M - w_{l,\alpha_s}^i\right)}\right)$$

$$v_r = \left(\frac{v_l\sum_{i=1}^{M}\bar{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^M - w_{l,\alpha_s}^i\right) - \left(\sum_{i=1}^{M}\underline{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^i - w_{l,\alpha_s}^1\right)\sum_{i=1}^{M}\bar{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^M - w_{l,\alpha_s}^i\right)\right)}{\left(\sum_{i=1}^{M}\underline{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^i - w_{l,\alpha_s}^1\right) + \sum_{i=1}^{M}\bar{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^M - w_{l,\alpha_s}^i\right)\right)^2}\right)$$

In which v_l is used to simplify notation as $v_l = \sum_{i=1}^{M}\underline{f}_i^{\alpha_s}\left(w_{l,\alpha_s}^i - w_{l,\alpha_s}^1\right) + \sum_{i=1}^{M}\bar{f}_i\left(w_{l,\alpha_s}^M - w_{l,\alpha_s}^i\right)$.

---

2) Nie-Tan GT2-RBFNN: For a NT GT2-RBFNN of Mamdani type, the AGD equations are defined for the consequent weight of each $\alpha-$level as an spike $w_i^{\alpha_s}$ and updated as:

$$\Delta w_i^{\alpha_s}(p+1) = -\beta\frac{\partial E_p(\vec{x}_p)}{\partial w_i^{\alpha_s}} + \gamma\Delta w_i^{\alpha_s}(p) \tag{56}$$

To implement the AGD for a Mamdani NT GT2-RBFNN, the derivative $\partial E_p(\vec{x}_p)/\partial w_i^{\alpha_s}$ in (56) is updated as

$$\frac{\partial E_p(\vec{x}_p)}{\partial w_i^{\alpha_s}} = \frac{\partial E_p(\vec{x}_p)}{\partial y_{NT,\alpha_s}(\vec{x}_p)}\frac{\partial y_{NT,\alpha_s}(\vec{x}_p)}{\partial w_i^{\alpha_s}} \tag{57}$$

where: $\partial E_p(\vec{x}_p)/\partial y_{NT,\alpha_s}(\vec{x}_p) = \alpha_s/\sum_{s=1}^{S}\alpha_s$ and

$$\frac{\partial y_{NT,\alpha_s}}{\partial w_i^{\alpha_s}} = \frac{\underline{f}_i^{\alpha_s} + \bar{f}_i^{\alpha_s}}{\sum_{i=1}^{M}\underline{f}_i^{\alpha_s} + \sum_{i=1}^{M}\bar{f}_i^{\alpha_s}} \tag{58}$$

Consequently, σ_i^1, σ_i^2 and m_k^i are adjusted as

$$\frac{\partial E_p(\vec{x}_p)}{\partial \sigma_i^1} = \frac{\partial E_p(\vec{x}_p)}{\partial y_{NT,\alpha_s}}\left[\frac{\partial y_{NT,\alpha_s}}{\partial \bar{f}_i^{\alpha_s}}\frac{\partial \bar{f}_i^{\alpha_s}}{\partial \sigma_i^1}\right] \tag{59}$$

$$\frac{\partial E_p(\vec{x}_p)}{\partial \sigma^2} = \frac{\partial E_p(\vec{x}_p)}{\partial y_{NT,\alpha_s}}\left[\frac{\partial y_{NT,\alpha_s}}{\partial \underline{f}_i^{\alpha_s}}\frac{\partial \underline{f}_i^{\alpha_s}}{\partial \sigma_i^2}\right] \tag{60}$$

where

$$\frac{\partial y_{NT,\alpha_s}}{\partial \bar{f}_i^{\alpha_s}} = \frac{\partial y_{NT,\alpha_s}}{\partial \underline{f}_i^{\alpha_s}} = \frac{w_i^{\alpha_s} - y_{NT,\alpha_s}}{\sum_{i=1}^{M}\underline{f}_i^{\alpha_s} + \sum_{i=1}^{M}\bar{f}_i^{\alpha_s}} \tag{61}$$

in which

$$\frac{\partial \bar{f}_i^{\alpha_s}}{\partial \sigma_i^1} = 2\bar{f}_i^{\alpha_s}\frac{(x_k - m_k^i)^2}{(\sigma_i^2)^3};\ \frac{\partial \underline{f}_i^{\alpha_s}}{\partial \sigma_i^2} = 2\underline{f}_i^{\alpha_s}\frac{(x_k - m_k^i)^2}{(\sigma_i^2)^3} \tag{62}$$

$$\frac{\partial \underline{f}_i^{\alpha_s}}{\partial m_k^i} = 2\underline{f}_i^{\alpha_s}\frac{(x_k - m_k^i)}{(\sigma_i^2)^2};\ \frac{\partial \bar{f}_i^{\alpha_s}}{\partial m_k^i} = 2\bar{f}_i^{\alpha_s}\frac{(x_k - m_k^i)}{(\sigma_i^1)^2} \tag{63}$$

For a TSK GT2-RBFNN, the AGD approach follows a similar procedure described in Eq. (59-63). However, the consequent coefficients c_m^{i,α_s} for each $\alpha-$level are updated as:

$$\frac{\partial E_p(\vec{x}_p)}{\partial c_m^{i,\alpha_s}} = \frac{\partial E_p(\vec{x})_p}{\partial y_{NT,\alpha_s}} \left(\frac{x_m(\underline{f}_i^{\alpha_s} + \bar{f}_i^{\alpha_s})}{\sum_{i=1}^{M} \underline{f}_i^{\alpha_s} + \sum_{i=1}^{M} \bar{f}_i^{\alpha_s}} \right) \tag{64}$$

where $m = 0, \ldots, n$, such that $x_0 = 0$ and $c_m^{i,\alpha_s} = 1$.

3) Biglarbegian-Melek-Mendel GT2-RBFNN: Based on the AGD approach, in order to update the parameters of a TSK BMM GT2-RBFNN with an uncertain $[\sigma_i^1, \sigma_i^2]$, fixed mean m_k^i and consequents g_{i,α_s} for each $\alpha-$level

$$\frac{\partial E_p}{\partial c_m^{i,\alpha_s}} = 4 d_\alpha e_p x_m \left(\frac{\underline{f}_i^{\alpha_s}}{\sum_{s=1}^{S} \underline{f}_i^{\alpha_s}} + \frac{\bar{f}_i^{\alpha_s}}{\sum_{s=1}^{S} \bar{f}_i^{\alpha_s}} \right) \tag{65}$$

To update σ_i^1, σ_i^2 and m_k^i

$$\frac{\partial E_p}{\partial \sigma_i^1} = 8 d_\alpha e_p n_{\alpha_s} \left(\frac{g_{i,\alpha_s} - y_n^{\alpha_s}}{\sum_{i=1}^{M} \bar{f}_i^{\alpha_s}} \right) \left(\bar{f}_i^{\alpha_s} \frac{\sum_{k=1}^{N} \left(x_k - m_k^i\right)^2}{(\sigma_i^1)^3} \right) \tag{66}$$

$$\frac{\partial E_p}{\partial \sigma_i^2} = 8 d_\alpha e_p m_{\alpha_s} \left(\frac{g_{i,\alpha_s} - y_m^{\alpha_s}}{\sum_{i=1}^{M} \underline{f}_i^{\alpha_s}} \right) \left(\underline{f}_i^{\alpha_s} \frac{\sum_{k=1}^{N} \left(x_k - m_k^i\right)^2}{(\sigma_i^2)^3} \right) \tag{67}$$

And

$$\frac{\partial E_p}{\partial m_k^i} = 8 d_\alpha e_p m_{\alpha_s} \left(\frac{g_{i,\alpha_s} - y_m^{\alpha_s}}{\sum_{i=1}^{M} \underline{f}_i^{\alpha_s}} + \frac{g_{i,\alpha_s} - y_n^{\alpha_s}}{\sum_{i=1}^{M} \bar{f}_i^{\alpha_s}} \right) \times \left[\bar{f}_i^{\alpha_s} \frac{\sum_{k=1}^{N} \left(x_k - m_k^i\right)}{(\sigma_i^1)^2} + \underline{f}_i^{\alpha_s} \frac{\sum_{k=1}^{N} \left(x_k - m_k^i\right)}{(\sigma_i^2)^2} \right] \tag{68}$$

Please note for each $\alpha-$level, a different value for the coeffcients c_m^{i,α_s} is employed.

VII. Performance Verification

In this section, three different examples are used to compare the performance of the GT2 RBFNN structures with some well known algorithms such as the ANFIS, a Sequential Adaptive Fuzzy Inference System (SAFIS) [38], a network of Functionally Weighted Single-Input-Rule-Modules connected to a Fuzzy Inference System (FWSIRM-FIS) [38], Support Vector Regression (SVR), RBFNN of T1 and IT2, an ensemble of T1 RBFNNs based on a Negative Correlation Learning (E-RBFNN) [39], Support Vector Machine (SVM) [40], Least Square SVM (LS-SVM) [40] and an IT2 Fuzzy Neural Network with support vector regression (IT2-FNN-SVR) [41]. While the first example involves the modelling of 10 real-world benchmark data sets for multiclass classification and regression problems, the last two examples are used for nonlinear plant identification and chaotic time series prediction in the precense of randomness and Gaussian noise respectively. For the ANFIS, RBFNN, IT2-RBFNN and GT2 RBFNN models and E-RBFNN, all the simulations are carried out in MATLAB 2014 environment in an intel Core i7, 2.7 GHZ CPU. Similar to a GT2 RBFNN, an AGD version is implemented to train the RBFNN and the IT2 RBFNN [24], [42].

A. Example 1: Modelling of Benchmarck Data sets for Multiclass Classification and Regression

This example compares the performance of a GT2 RBFNN, RBFNN, IT2 RBFNN, E-RBFNN, ANFIS, SVM and LS-SVM on five real-world benchmark data sets for regression and five data sets for multiclass classification. In Table II and III, the specifications of the data sets are listed. The associated distributions of the data sets are unknown and most of them noisy-free. As indicated in Tables II and III, for cross-validation purposes the number of samples for training (column train) and testing (column test) are randomly selected. By increasing/decreasing by one the number of hidden units initially estimated by the IIG algorithm, the optimal number of hidden units (fuzzy rules) in the GT2 RBFNN are selected based on cross-validation results. In Tables II-III, column *fuzzy rules* is used to indicate the optimal value for the number of hidden units for the RBFNN, IT2 RBFNN and GT2 RBFNN models. It is selected a granulation factor of $\alpha_g = 0.3$, an initial value for $\Delta\sigma_i = 0.1$, $\sigma_i^1 = 1.0$ and for $w_{l,\alpha_s}^i = 1.0$ and $w_{r,\alpha_s}^i = 1.0$. For a TSK GT2 RBFNN with a BMM method, it was determined that the best value for $m_\alpha = 0.9$ and $n_\alpha = 0.1$. For all GT2 RBFNN models and for the E-RBFNN, it was found the best trade-off between accuracy and model simplicity is achieved by using three horizontal slices and 4 units in the hidden layer. In Tables IV and V, the generalisation performance of SVM, and LS-SVM presented in [41] is compared to the average performance results of 20 trials for the ANFIS, RBFNN, IT2 RBFNN, E-RBFNN and GT2 RBFNN. As indicated in [40], SVM and LS-SVM usually achieves a good generalisation performance. This heavily depends on the combination of values for the cost parameter C and kernel parameter γ. Therefore, for each data set a large number of combinations to find the appropiate the C and γ is required. Opposite to this, from Tables II-V, it can be observed that a GT2 RBFNN needs a small number of hidden units to obtain a higher generalisation performance with respect to SVM and LS-SVM. This model simplification compensates the associated learning time that in most cases is similar to the time used to train an IT2 RBFNN, an ANFIS and less to an E-RBFNN, in particular, the simplified GT2 RBFNNs.

TABLE II: SPECIFICATION OF MULTICLASS CLASSIFICATION.

	# Samples		*Number of*		
Datasets	Train	Test	Attributes	Classes	Fuzzy Rules
Iris	100	50	4	3	3
Wine	118	60	13	3	3
Glass	142	72	9	6	6
Segment	1540	770	19	7	7
Shuttle	43500	14500	9	7	9

TABLE III: SPECIFICATION OF REGRESSION DATA SETS.

Datasets	Train	Test	# Attributes	# Fuzzy Rules
Pyrim	49	25	27	3
Housing	337	169	13	4
Space-ga	2071	1036	6	5
Abalone	2784	1393	8	7
HPC	927	103	8	8

TABLE IV: AVERAGE PERFORMANCE OF 20 TRIALS OF GT2-RBFNN, IT2-RBFNN, RBFNN, E-RBFNN, ANFIS, SVM AND LS-SVM.

Dataset			Iris		Wine		Glass		Segment		Shuttle	
Model			Testing (%)	Training Time (s)	Testing (%)	Training Time (s)	Testing (%)	Training Time (s)	Testing (%)	Training Time (s)	Testing (%)	Training Time (s)
GT2-RBFNN	Mamdani	EKM	99.23	1.57	100.0	1.88	70.22	2.01	99.28	98.12	100.0	10911.9
		NT	99.62	1.40	99.80	1.70	70.53	1.80	98.01	59.25	99.98	6329.1
		WM	99.82	1.78	100.0	1.94	70.01	2.41	99.07	84.91	100.0	9873.1
	TSK	EKM	100.0	1.75	100.0	1.99	69.23	3.84	99.41	93.10	99.22	13301.9
		NT	99.11	1.62	100.0	1.89	69.25	4.76	98.09	79.82	100.0	8802.1
		BMM	98.39	1.39	100.0	2.18	70.30	3.78	99.10	88.12	100.0	14119.8
IT2-RBFNN	Mamdani	EKM	98.14	1.55	99.12	1.63	67.02	1.73	96.78	55.72	100.0	7301.4
		NT	97.71	1.31	98.97	1.55	67.59	1.51	98.80	43.11	99.29	5909.1
		WM	98.25	1.58	98.78	1.71	67.25	1.45	97.19	60.08	100.0	8021.1
	TSK	EKM	96.65	1.64	99.45	1.62	67.79	1.65	98.80	63.15	100.0	8722.1
		NT	97.73	1.39	98.56	1.42	67.07	1.58	97.99	57.19	99.93	7503.1
		BMM	96.01	1.09	98.87	1.61	67.12	1.68	98,10	64.16	99.87	8002.1
RBFNN			93.10	1.16	97.71	1.24	65.86	1.44	94.31	38.33	96.90	5204.1
E-RBFNN			95.93	4.28	98.31	2.70	66.69	3.01	95.11	42.68	98.29	32117.2
ANFIS			92.21	1.23	93.44	1.89	65.12	2.39	91.04	28.12	95.21	18366.9
SVM			92.21	0.075	98.37	0.075	67.83	0.2871	96.53	14.30	99.74	2864.0
LS-SVM			96.28	0.0021	97.63	0.0043	67.22	0.0097	96.12	4.302	99.82	24767.0

TABLE V: AVERAGE RMSE OF 20 TRIALS OF GT2-RBFNN, IT2-RBFNN, RBFNN, E-RBFNN, ANFIS, SVM, LS-SVM.

Dataset			Pyrim		Housing		Space-ga		Abalone		HPC (RMSE)	
Model			Testing (RMSE)	Training Time (s)	Testing (RMSE)	Training Time	Testing (RMSE)	Training Time (s)	Testing (RMSE)	Training Time (s)	Testing (RMSE)	Training Time (s)
GT2-RBFNN	Mamdani	EKM	0.0411	17.88	0.0845	2.78	0.0219	67.12	0.0299	60.11	5.170	173.1
		NT	0.0397	15.61	0.0801	1.89	0.0310	54.20	0.0375	45.19	5.281	151.0
		WM	0.0388	18.03	0.0811	2.72	0.0205	68.10	0.0307	63.03	5.310	177.9
	TSK	EKM	0.0478	19.41	0.0816	2.83	0.0198	71.06	0.0255	66.16	5.392	189.0
		NT	0.0401	14.99	0.0802	2.39	0.0109	56.19	0.0301	50.24	5.210	169.2
		BMM	0.0428	18.77	0.0833	2.91	0.0165	75.89	0.0270	68.09	5.280	199.3
IT2-RBFNN	Mamdani	EKM	0.0604	14.08	0.0919	2.30	0.0469	47.19	0.0579	42.10	5.870	158.1
		NT	0.0678	13.02	0.0973	1.88	0.0466	44.06	0.0609	38.04	5.723	140.1
		WM	0.0699	14.58	0.1104	1.93	0.0679	51.02	0.0599	47.26	5.640	158.3
	TSK	EKM	0.0645	14.90	0.1095	2.52	0.0397	56.69	0.0601	50.30	5.560	166.2
		NT	0.0609	14.72	0.1020	1.79	0.0487	47.63	0.0544	42.51	5.504	157.2
		BMM	0.0689	15.26	0.1118	2.33	0.0481	61.19	0.0520	54.47	5.670	175.7
RBFNN			0.0780	11.03	0.1167	1.55	0.0519	43.19	0.0689	35.67	6.120	134.8
E-RBFNN			0.0482	22.14	0.0987	7.55	0.0411	73.22	0.0309	51.05	5.549	229.1
ANFIS			0.0988	12.18	0.0988	2.03	0.0914	38.12	0.1233	38.12	13.490	144.0
SVM			0.1280	0.0315	0.0976	0.0085	0.0648	52.75	0.0764	113.1	10.406	105.3
LS-SVM			0.1272	0.0388	0.0704	0.0343	0.0330	3.510	0.0746	7.674	8.820	6.981

To take full advantage of the equivalence between a GT2 RBFNN and GT2 FLSs, in this example a GT2 RBFNN with an EKM is used to provide some insights about the HPC data. High Performance Concrete (HPC) data is a collection of 1030 multi-dimensional samples where each set of points represents 8 inputs variables (cement, fly ash, water, superplasticiser, coarse aggregate, fine aggregate, age of testing and blast furnace slag, kg/m^3) and one ouptut (Concrete Compressive Strength-MPa, CCS) [43], [44]. To illustrate model performance and physical interpretation, in Fig. 12-14, the data fit for CCS prediction for a Mamdani GT2 RBFNN with an EKM with 8 fuzzy rules and its variable effect surface for the ingredients cement and fly ash and final rule distribution for the input superplasticiser are presented respectively. A variable effect surface is created by keeping $N-2$ input variables constant and ploting the remaining varying input variables. Here, the average of each input variable is used as a constant for the $N-2$ variables. As indicated in [24], by using variable effect surfaces, expert's opinion can confirm the behaviour of specific input variables with respect to a desired output.

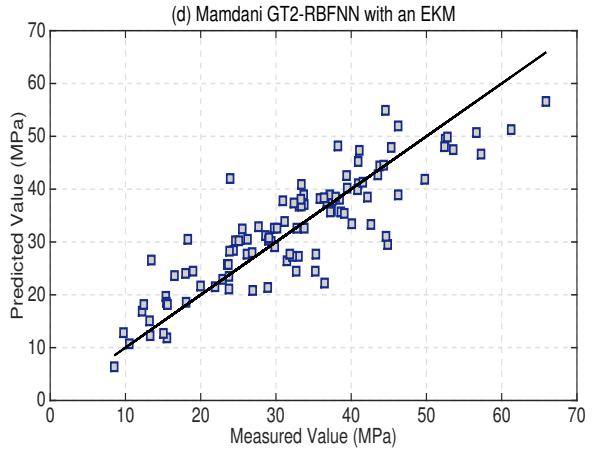

Fig. 12: Testing Data Fit for the HPC compressive strength using a Mamdani GT2-RBFNN with an EKM type-reduction layer.

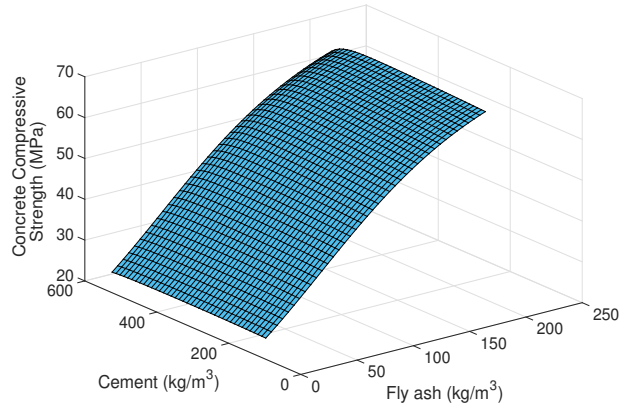

Fig. 13: Variable effect surface for the ingredients: Cement vs Fly ash.

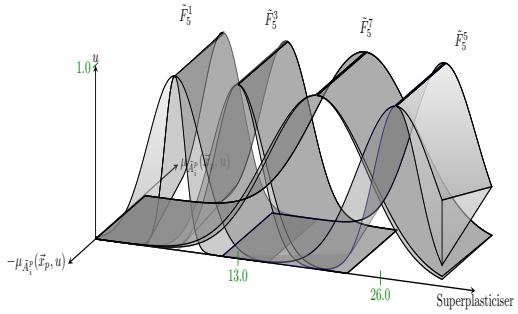

Fig. 14: Final fuzzy rule distribution of a Mamdani GT2-RBFNN with an EKM type-reducer.

B. Example 2: Nonlinear Plant Identification

This example is to identify the nonlinear plant described by the equation below [38]:

$$y(t+1) = f(y(t), y(t-1), u(t)) = \frac{y(t)y(t-1)(y(t)-0.5)}{1+y^2(t)+y^2(t-1)} + 1 \quad (69)$$

The equilibrium state of the unforced system given by Eq. (69) is $(0, 0)$. As [38], the training data consists of 5000×3 input vectors $[y(t)\ \ y(t-1)\ \ u(t)]$ and one output $y(t+1)$. The signal $u(t)$ has been randomly generated by a uniform distribution in the region $[-1.5, 1.5]$. For testing purposes, a data set of 200 observations has been generated where the input $u(t)$ is given by $u(t) = sin(2\pi t/25)$. The experimental setup for the GT2 RBFNN models consists of a number of 3 horizontal slices, a granulation factor of α_g and three fuzzy rules. An initial value for $\Delta\sigma_i = 0.05$, $\sigma_i^1 = 1.0$ and each factor $w_{l,\alpha_s}^i = w_{r,\alpha_s}^i = 1.0$. Based on simulation results, it was found for a TSK GT2 RBFNN with a BMM type reduction, the best value for $m_\alpha = 0.85$ and $n_\alpha = 0.15$. For an E-RBFNN, it was determined the optimal value to provide a high level of generalisation is with 4 hidden units, where each has 3 fuzzy rules. Table VI shows the average generalisation performance of 20 trials, the number of parameters per each model as well as the Average Training Time **ATT** of each GT2 RBFN model with respect to an FWSIRM [38], SANFIS [38], RBFNN, IT2 RBFNN [24], E-RBFNN [39] and the ANFIS system According to Table VI, the highest trade-off between accuracy and model simplicity is obtained by the RBFNN of GT2 using a NT algorithm. From Table VI, it it is clear for most of GT2 RBFNN models the training time is comparable to that of some models such as the BPNN and RBFNN. It is worth noting, the generalisation performance of an E-RBFNN is higher than an IT2 RBFNN and similar to a GT2 RBFNN. Both, GT2 RBFNN and E-RBFNN treat uncertainty as measure for ambiguity. However, a GT2 RBFNN quantifies uncertainty as a deficiency that results not only from imprecise boundaries in the fuzzy sets (vagueness or fuzziness), but also as nonspecificity that refers to information-based imprecision, whereas an E-RBFNN defines ambiguity as a variation of the output of the ensemble members over unlabeled data. That means, uncertainty quantification is useful in an ensemble only if there is a disagreement among on some inputs [45].

TABLE VI: COMPARISON OF THE AVERAGE PERFORMANCE OF 20 TRIALS OF DIFFERENT MODELS IN EXAMPLE 2.

Model			Testing RMSE		Number of Parameters	ATT (s)
			Mean	Best		
GT2-RBFNN	Mamdani	EKM	0.0266	0.0151	33	35.12
		NT	0.0289	**0.0134**	23	28.19
		WM	0.0288	0.0188	33	33.92
	TSK	EKM	0.0267	0.0192	42	38.19
		NT	0.0201	0.0177	42	31.68
		BMM	0.0256	0.0163	42	36.03
IT2-RBFNN	Mamdani	EKM	0.0445	0.0276	21	23.11
		NT	0.0339	0.0194	18	21.71
		WM	0.0439	0.0312	21	27.02
	TSK	EKM	0.0458	0.0374	24	28.11
		NT	0.0479	0.0348	24	21.06
		BMM	0.0481	0.0365	24	27.19
RBFNN			0.0501	0.0408	15	19.20
E-RBFNN			0.0470	0.0161	60	41.12
ANFIS			0.0580	0.0474	106	6.01
FWSIRM-FIS			0.0494	0.0274	45	1.13
SANFIS				0.0221	85	NA
BPNN			0.0939	0.0611	151	94.49

TABLE VII: PERFORMANCE OF THE MAMDANI (TSK) GT2-RBFNN AND OTHER MODELS WITH A TRAINING NOISE σ = 0.2 in example 3.

		Mamdani GT2-RBFNN			TSK GT2-RBFNN			IT2 -FNN		IT2-RBFNN	E-RBFNN
		EKM	**NT**	**WM**	**EKM**	**NT**	**WM**	**SVR-(N)**	**SVR-(F)**	**EKM**	
Number of Parameters		65	45	65	45	45	45	103	103	35	105
Number of Rules		5	5	5	5	5	5	6	6	5	5
Training RMSE ($\sigma = 0.2$)		0.091	0.089	0.088	0.092	0.086	0.087	0.234	0.233	0.125	0.110
Test RMSE	Clean	0.060	0.071	0.061	0.064	0.060	0.062	0.085	0.083	0.085	0.082
	σ = 0.1	0.067	0.068	0.070	0.073	0.075	0.071	0.105	0.103	0.092	0.089
	σ = 0.3	0.097	0.107	0.095	0.108	0.096	0.102	0.186	0.180	0.122	0.117

TABLE VIII: PERFORMANCE OF THE MAMDANI (TSK) GT2-RBFNN AND OTHER MODELS WITH A NOISE $\sigma = 0.3$ in example 3.

Parameters		Mamdani GT2-RBFNN			TSK GT2-RBFNN			IT2 -FNN		IT2-RBFNN	E-RBFNN
		EKM	**NT**	**WM**	**EKM**	**NT**	**WM**	**SVR-(N)**	**SVR-(F)**	**EKM**	
Number of Parameters		65	45	65	45	45	45	103	103	35	105
Number of Rules		5	5	5	5	5	5	6	6	5	5
Training RMSE (σ = 0.3)		0.111	0.108	0.122	0.121	0.117	0.114	0.349	0.347	0.133	0.148
Test RMSE	Clean	0.085	0.088	0.083	0.079	0.069	0.078	0.127	0.121	0.092	0.120
	σ = 0.1	0.109	0.096	0.091	0.081	0.083	0.105	0.138	0.131	0.127	0.132
	σ = 0.3	0.125	0.118	0.131	0.133	0.127	0.129	0.188	0.184	0.144	0.159

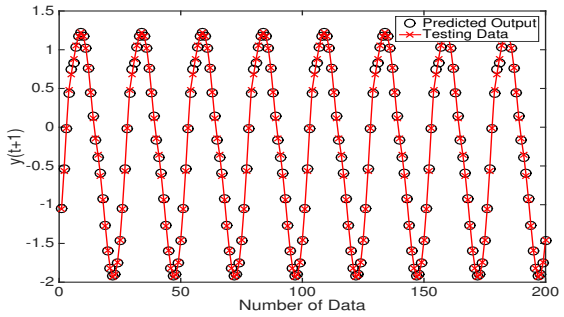

Fig. 15: Testing data, and the output of the GT2-RBFNN with an WM direct defuzzification and 3 fuzzy rules.

In other words, a GT2 RBFNN can be viewed as an ensemble of interval Type-2 FLSs where all the IT2 FSs computations occurr for each α-level and ambiguity is nonuniformly weighted. In this example, a GT2 RBFNN results more practical than an ensemble, especially because it is a more compact model with less parameters and less expensive in terms of computational burden. To exemplify the performance of GT2 RBFNN models, in Fig. 15, the identification result for a GT2 RBFNN with a WM method is shown.

C. Example 3: Noisy Chaotic Time-Series Prediction

As the last experiment, a time-series prediction problem to evaluate the performance of the GT2-RBFNN is employed. The Mackey-Glass chaotic time series is generated from the following differential equation [41]:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+10x(t-\tau)} - 0.1x(t) \tag{70}$$

For comparison reasons with previous results, the parameters $\tau = 30$, $x(0) = 1.2$. Four past values were employed to predict $x(t)$ where the input data format is used as:

$$[x(t-24), x(t-18), x(t-12), x(t-6); x(t)]$$

A number of 1000 patterns were generated from the observation $t = 124$ to $t = 1123$. For cross-validation purposes, the input data was divided into two subsets, i.e. a) 50% for training and b) 50% for testing. For cross-validation purposes, two different types of training data were created by adding Gaussian noise with a standard deviation of $\sigma = 0.2$ $\sigma = 0.3$ and with a mean of 0 to the original data $x(t)$. This type of noise has been selected because it usually occurs in real situations and it is frequently employed to verify model robustness [25-29]. For testing data, three data sets were created from the original data set. The first consists of the original 500 values. The last two testing data sets were created by adding a Gaussian noise with a $\sigma = 0.2$ and $\sigma = 0.3$. To compare the performance of the GT2-RBFNN to other existing interval type-2 fuzzy modelling methodologies, namely: a) an IT2FNN-SVR-(N), b) an IT2-FNN-SVR-(F) and an c) EKM IT2-RBFNN and d) E-RBFNN. The first two models a) and b) were introduced in [41]. The IT2-FNN-SVR is a six-layer interval type-2 fuzzy neural network with support vector machine regression that uses two different types of input nodes. For the first type, the input nodes in an IT2-FNN-SVR simply forwards each numerical data and is called IT2-FNN-SVR-(N) for short. Thus, the output of the IT2-FNN-SVR-(N) is a bounded interval which is described in terms the lower and upper limits of its Footprint Of Uncertainty (FOU). An IT2-FNN-SVR-(F) uses an input node layer that fuzzifies the input numerical data. The third IT2 methodology is an IT2-RBFNN with an EKM approach. And the last methodology is an ensemble of RBFNNs suggested in [39]. According to our experiments, it was determined a number of 3 horizontal slices for an GT2 RBFNN, and 3 hidden units with 3 fuzzy rules each for an E-RBFNN produce the highest balance between model performance and model simplicity. For statistical purposes, each experiment was repeated 10 times, and the RMSE average is used as a comparison performance index. Table VII and VIII show the training and testing results for the prediction of the Mackey-Glass time-series. From Table V, it can be viewed that in general GT2 neural structures outperform the IT2-FNN-SVR and its counterpart the IT2-RBFNN with an EKM. It is also worth noting, the superiority of the GT2-RBFNN is confirmed not only for validation purposes, but also in relation to the number of parameters.

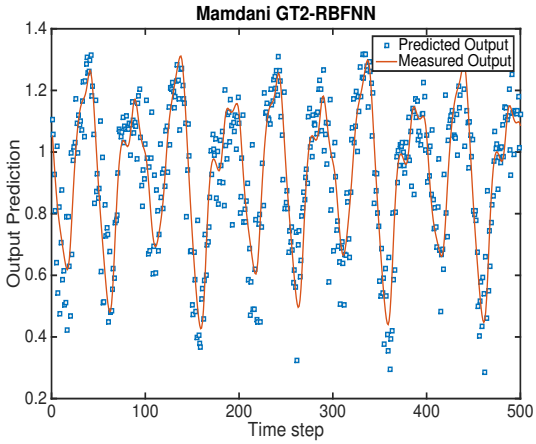

Fig. 16: Testing prediction of a Mamdani GT2-RBFNN with an EKM type-reduction and a noise level of $\sigma = 0.3$ and RMSE = 0.125 that correspond to a training stage with a level of noise of $\sigma = 0.3$.

Hence, the highest accuracy is achieved by a GT2-RBFNN with a WM and NT method respectively. In relation to Table VI, the higher the noise level of the training and testing data, the better the performance of the GT2-RBFNN models with respect to the IT2 fuzzy models. Particularly those GT2 models with an EKM and NT direct defuzzification and of Mamdani type. Finally, in Fig. 16, the testing data-fit of a random experiment using a Mamdani GT2-RBFNN with an EKM and noise level of $\sigma = 0.3$ is illustrated.

VIII. Summary and Discussion

From the comparative analysis presented in previous section, the following summarisation and disscussion is provided:

a) By using GT2 FSs, model accuracy of an RBFNN can be improved importantly. Compared to its counterparts the RBFNN and IT2 RBFNN, a higher tradeoff between accuracy and model simplicity is provided. The term *model simplicity* is used because compared to other existing fuzzy models of T1 and T2, a reduced number of fuzzy rules, and hence of parameters is required to obtain similar or better results.

b) Two problems that involve the treatment of *randomness* for nonlinear plant identification, and for the prediction of *nosisy* chaotic time series was provided. Compared to an RBFNN of T1 or IT2, a GT2 RBFNN weights uncertainty non uniformly. This allows an RBFNN to better model the effecs of uncertainty. That means, an RBFNN with GT2 FSs quantifies uncertainty as a deficiency that results from imprecise boundaries of the associated FSs, so using GT2 FSs accounts to minimise information-based imprecision.

c) From tables IV-VI, column training time is the average time of training epochs spent by each model. As can be noted, the training speed of a GT2 RBFNN with simplified structures is similar to the RBFNN, faster to the ensemble of RBFNNs and similar to the ANFIS model when it comes to modeling large size data sets.

d) As illustrated in example 2, a GT2 RBFNN not only inherits the ability of NNs to approximate complex funtions, but also the ability of fuzzy logic models to provide some insights about the system being modelled.

f) By using GT2 FSs usually increases the computational complexity, however this time can be compensated by an improvement in model performance and a model simplification that can be reached by fuzzy structures based on direct-defuzzification algorithms.

e) Further to point f), in terms of computation, the application of a Gradient Descent approach (GD) to identify the parameters of a GT2 FLS with KM methods ('*or EKM*') usually results more expensive than the parameter identification for an RBFNN of T1 or IT2. This is due to the number of iterations that are needed to calculate not only the associated derivatives but also to track the permutations created during the sorting process of any KM method [46]. A GD is usually not globally convergent. Thus, a number of optimisation methods based on metaheuristics have been proposed [47]. To make this less severe, in this paper an Adaptive version of a GD approach that includes a momentum term to avoid getting trapped in a local minimum and to speed up the GD convergence is suggested.

IX. Conclusions

This paper presents a General Type-2 Radial Basis Function Neural Network (GT2 RBFNN) that is functionally equivalent to a GT2 FLS based on the $\alpha-$plane representation, in which the main inference engine can be viewed as a TSK or Mamdani system. A detailed description of the neural structure and its corresponding parametric optimisation of a GT2-RBFNN with an EKM, and three simplified GT2-RBFNN models that employs three different direct-defuzzification approaches is provided. To offer a comprehensive performance analysis, experimental results about the modelling of ten data sets for multiclass classification and regression problems is provided. Two problems for nonlinear identification and for the prediction of chaotic time series in the precense of randomness and Gaussian noise are considered. Based on experimental results, the suggested model is not only able to outperform its counterparts the RBFNN of type-1 and the Interval Type-2 Radial Basis Function Neural Network (IT2 RBFNN), but also to better treat and minimse the effects of uncertainty. It can be also observed from the simulation results that compared to other methodologies, including an ensemble of RBFNNs, the number of parameters of a GT2 RBFNN is usually smaller. Further developments of the GT2 RBFNN may be related with further advances of Type-2 Fuzzy Logic methodologies, Neural Networks and learning. Particularly to reduce the computational complexity and increase model performance. A future study will be also in terms of the evaluation of the GT2 RBFNN to formulate knoweldge in a transparent way to interpretation and analysis of complex systems.

References

[1] M. A. Sanchez, O. Castillo, and J. R. Castro, "Generalized type-2 fuzzy systems for controlling a mobile robot and a performance comparison with interval type-2 and type-1 fuzzy systems," *Expert Systems with Applications*, vol. 42, no. 14, pp. 5904–5914, 2015.

[2] O. Castillo, L. Amador-Angulo, J. R. Castro, and M. Garcia-Valdez, "A comparative study of type-1 fuzzy logic systems, interval type-2 fuzzy logic systems and generalized type-2 fuzzy logic systems in control problems," *Information Sciences*, vol. 354, pp. 257–274, 2016.

[3] A. Sarabakha, C. Fu, E. Kayacan, and T. Kumbasar, "Type-2 fuzzy logic controllers made even simpler: From design to deployment for uavs," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 6, pp. 5069–5077, 2018.

[4] E. Camci and E. Kayacan, "Game of drones: Uav pursuit-evasion game with type-2 fuzzy logic controllers tuned by reinforcement learning," in *Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on*. IEEE, 2016, pp. 618–625.

[5] S. Greenfield and F. Chiclana, "Slicing strategies for the generalised type-2 mamdani fuzzy inferencing system," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2016, pp. 195–205.

[6] U. Martinez-Hernandez and A. A. Dehghani-Sanij, "Adaptive bayesian inference system for recognition of walking activities and prediction of gait events using wearable sensors," *Neural Networks*, vol. 102, pp. 107–119, 2018.

[7] J. M. Mendel, "General type-2 fuzzy logic systems made simple: a tutorial," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 5, pp. 1162–1182, 2014.

[8] ——, "Advances in type-2 fuzzy sets and systems," *Information sciences*, vol. 177, no. 1, pp. 84–110, 2007.

[9] ——, *Uncertain rule-based fuzzy systems: introduction and new directions*. Springer, 2017.

[10] A. Rubio-Solis, A. Baraka, G. Panoutsos, and S. Thornton, "Data-driven interval type-2 fuzzy modelling for the classification of imbalanced data," in *Practical Issues of Intelligent Innovations*. Springer, 2018, pp. 37–51.

[11] P. Melin, O. Castillo, C. I. Gonzalez, J. R. Castro, and O. Mendoza, "General type-2 fuzzy edge detectors applied to face recognition systems," in *Fuzzy Information Processing Society (NAFIPS), 2016 Annual Conference of the North American*. IEEE, 2016, pp. 1–6.

[12] O. Linda and M. Manic, "General type-2 fuzzy c-means algorithm for uncertain fuzzy clustering," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 5, pp. 883–897, 2012.

[13] R. A. Aliev, W. Pedrycz, B. G. Guirimov, R. R. Aliev, U. Ilhan, M. Babagil, and S. Mammadli, "Type-2 fuzzy neural networks with fuzzy clustering and differential evolution optimization," *Information Sciences*, vol. 181, no. 9, pp. 1591–1608, 2011.

[14] C. Wagner and H. Hagras, "Toward general type-2 fuzzy logic systems based on zslices," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 4, pp. 637–660, 2010.

[15] ——, "zslicestowards bridging the gap between interval and general type-2 fuzzy logic," in *Fuzzy Systems, 2008. FUZZ-IEEE 2008.(IEEE World Congress on Computational Intelligence). IEEE International Conference on*. IEEE, 2008, pp. 489–497.

[16] O. Castillo and L. Amador-Angulo, "A generalized type-2 fuzzy logic approach for dynamic parameter adaptation in bee colony optimization applied to fuzzy controller design," *Information Sciences*, 2017.

[17] P. Melin, C. I. Gonzalez, J. R. Castro, O. Mendoza, and O. Castillo, "Edge-detection method for image processing based on generalized type-2 fuzzy logic," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 6, pp. 1515–1525, 2014.

[18] J. Andreu-Perez, F. Cao, H. Hagras, and G.-Z. Yang, "A self-adaptive online brain machine interface of a humanoid robot through a general type-2 fuzzy inference system," *IEEE Transactions on Fuzzy Systems*, 2016.

[19] F. Gaxiola, P. Melin, F. Valdez, and O. Castillo, "Generalized type-2 fuzzy weight adjustment for backpropagation neural networks in time series prediction," *Information Sciences*, vol. 325, pp. 159–174, 2015.

[20] C. Wagner and H. Hagras, "zslices based general type-2 flc for the control of autonomous mobile robots in real world environments," in *Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on*. IEEE, 2009, pp. 718–725.

[21] S. Coupland and R. John, "Geometric type-1 and type-2 fuzzy logic systems," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 1, pp. 3–15, 2007.

[22] J. M. Mendel and F. Liu, "On new quasi-type-2 fuzzy logic systems," in *Fuzzy Systems, 2008. FUZZ-IEEE 2008.(IEEE World Congress on Computational Intelligence). IEEE International Conference on*. IEEE, 2008, pp. 354–360.

[23] J.-S. Jang and C.-T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE transactions on Neural Networks*, vol. 4, no. 1, pp. 156–159, 1993.

[24] A. Rubio-Solis and G. Panoutsos, "Interval type-2 radial basis function neural network: a modeling framework," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 2, pp. 457–473, 2015.

[25] A. Rubio-Solis, G. Panoutsos, and S. Thornton, "A data-driven fuzzy modelling framework for the classification of imbalanced data," in *Intelligent Systems (IS), 2016 IEEE 8th International Conference on*. IEEE, 2016, pp. 302–307.

[26] A. R. Solis and G. Panoutsos, "Granular computing neural-fuzzy modelling: A neutrosophic approach," *Applied Soft Computing*, vol. 13, no. 9, pp. 4010–4021, 2013.

[27] U. Martinez-Hernandez, A. Rubio-Solis, G. Panoutsos, and A. A. Dehghani-Sanij, "A combined adaptive neuro-fuzzy and bayesian strategy for recognition and prediction of gait events using wearable sensors," in *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[28] F. Liu, "An efficient centroid type-reduction strategy for general type-2 fuzzy logic system," *Information Sciences*, vol. 178, no. 9, pp. 2224–2236, 2008.

[29] J. M. Mendel, "Type-2 fuzzy sets and systems: an overview," *IEEE computational intelligence magazine*, vol. 2, no. 1, pp. 20–29, 2007.

[30] D. Wu, "Approaches for reducing the computational cost of interval type-2 fuzzy logic systems: overview and comparisons," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 1, pp. 80–99, 2013.

[31] M. Nie and W. W. Tan, "Towards an efficient type-reduction method for interval type-2 fuzzy logic systems," in *Fuzzy Systems, 2008. FUZZ-IEEE 2008.(IEEE World Congress on Computational Intelligence). IEEE International Conference on*. IEEE, 2008, pp. 1425–1432.

[32] 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.

[33] M. Biglarbegian, W. W. Melek, and J. M. Mendel, "On the stability of interval type-2 tsk fuzzy logic control systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 3, pp. 798–818, 2010.

[34] W. Pedrycz and A. Bargiela, "Granular clustering: a granular signature of data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 32, no. 2, pp. 212–224, 2002.

[35] A. Rubio-Solis and G. Panoutsos, "Iterative information granulation for novelty detection in complex datasets," in *Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on*. IEEE, 2016, pp. 953–960.

[36] ——, "Fuzzy uncertainty assessment in rbf neural networks using neutrosophic sets for multiclass classification," in *Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1591–1598.

[37] A. Rubio-Solis, U. Martinez-Hernandez, and G. Panoutsos, "Evolutionary extreme learning machine for the interval type-2 radial basis function neural network: A fuzzy modelling approach," in *World Congress on Computational Intelligence*. IEEE, 2018.

[38] C. Li, J. Gao, J. Yi, and G. Zhang, "Analysis and design of functionally weighted single-input-rule-modules connected fuzzy inference systems," *IEEE Transactions on Fuzzy Systems*, 2016.

[39] A. Rubio-Solis and G. Panoutsos, "An ensemble data-driven fuzzy network for laser welding quality prediction," in *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[40] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2012.

[41] C.-F. Juang, R.-B. Huang, and W.-Y. Cheng, "An interval type-2 fuzzy-neural network with support-vector regression for noisy regression problems," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 4, pp. 686–699, 2010.

[42] U. Martinez-Hernandez, T. J. Dodd, M. H. Evans, T. J. Prescott, and N. F. Lepora, "Active sensorimotor control for tactile exploration," *Robotics and Autonomous Systems*, vol. 87, pp. 15–27, 2017.

[43] M.-Y. Cheng, J.-S. Chou, A. F. Roy, and Y.-W. Wu, "High-performance concrete compressive strength prediction using time-weighted evolutionary fuzzy support vector machines inference model," *Automation in Construction*, vol. 28, pp. 106–115, 2012.

[44] M.-Y. Cheng, P. M. Firdausi, and D. Prayogo, "High-performance concrete compressive strength prediction using genetic weighted pyramid operation tree (gwpot)," *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 104–113, 2014.

[45] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Advances in neural information processing systems*, 1995, pp. 231–238.

[46] 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.

[47] J.-q. Li, J.-d. Wang, Q.-k. Pan, P.-y. Duan, H.-y. Sang, K.-z. Gao, and Y. Xue, "A hybrid artificial bee colony for optimizing a reverse logistics network system," *Soft Computing*, vol. 21, no. 20, pp. 6001–6018, 2017.