SPRINGER BRIEFS IN APPLIED SCIENCES AND TECHNOLOGY · COMPUTATIONAL INTELLIGENCE

Mauricio A. Sanchez
Oscar Castillo
Juan R. Castro

Type-2 Fuzzy Granular Models

Springer

SpringerBriefs in Applied Sciences and Technology

Computational Intelligence

Series editor

Janusz Kacprzyk, Warsaw, Poland

About this Series

The series "Studies in Computational Intelligence" (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution, which enable both wide and rapid dissemination of research output.

More information about this series at http://www.springer.com/series/10618

Mauricio A. Sanchez · Oscar Castillo
Juan R. Castro

Type-2 Fuzzy Granular Models

Springer

Mauricio A. Sanchez
Faculty of Chemical Sciences
and Engineering
Autonomous University of Baja California
Tijuana
Mexico

Juan R. Castro
Faculty of Chemical Sciences
and Engineering
Autonomous University of Baja California
Tijuana
Mexico

Oscar Castillo
Division of Graduate Studies
Tijuana Institute of Technology
Tijuana, Baja California
Mexico

ISSN 2191-530X ISSN 2191-5318 (electronic)
SpringerBriefs in Applied Sciences and Technology
ISBN 978-3-319-41287-0 ISBN 978-3-319-41288-7 (eBook)
DOI 10.1007/978-3-319-41288-7

Library of Congress Control Number: 2016944415

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG Switzerland

Preface

In this book, several contributions to the area of granular computing are presented: a nature-inspired granulating algorithm, various techniques for forming higher-type information granules, and an application comparison of various types of information granules. All research was done with the intention of analyzing the general properties of some core concepts of granular computing, such as the information granulation, the principle of justifiable granularity, and higher-type information granule formation. All information granules were represented via fuzzy sets, either type-1, interval type-2, or general type-2 fuzzy sets. And all proposed approaches are derivative of hybrid intelligent algorithms, such that they automate the modeling from raw data to final fuzzy granular model.

This book is intended as a reference for engineers who wish to dwell into applications of more complex algorithms inspired by granular computing, for aspiring graduate students who desire to better understand how information granules can be formed, or for scientists who want to keep up to date on current research trends on granular computing focused on higher-type information granule formation, or to assess new areas of opportunity where research has yet to be carried out.

In Chap. 1, a short introduction to the book is given, where a broad set of granular computing core concepts are mentioned, as well as a brief description of contributions, applications, and general explanation of the proposed methods is also shown.

In Chap. 2, recommended background and theory is given, where current definitions of many used computational intelligence techniques are shown, as well as a deeper description of granular computing concepts is used throughout the book.

In Chap. 3, the embodiment of all carried out research is placed here from a nature-inspired information granulating technique, to multiple approaches for forming higher-type information granules.

In Chap. 4, a detailed description of carried out experimentation is shown, where each proposed technique results are summarized, giving acute understanding of the performance capabilities of the proposed algorithms.

In Chap. 5, concluding remarks regarding research done is shown with the focus of granular computing concepts and how they help improve the model performance, meaningfulness, and interpretability.

We would like to express our gratitude to out supporting agency CONACYT for the opportunity given to us by providing us grants to perform our research. We would also like to thank the institution Autonomous University of Baja California for supporting us, our families for keeping us motivated, and all people who have contributed to the success of our research, either directly or indirectly.

Tijuana, Mexico
March 2016

Mauricio A. Sanchez
Oscar Castillo
Juan R. Castro

Contents

Chapter 1
Introduction

In Computational Intelligence, an important area of application is that of model generation, especially when based on sampled data. Such modeling requires the use of specialized algorithms that focus on different aspects based on the needs of the final model or how the data is structured, this entails having special requirements per case. Although the performance of most data modeling techniques is very acceptable, most still lack a certain detail which could enrich them, that is, to ensure the entire process from start to finish focuses solely on the meaningfulness of the model as well as the interpretation.

In this book, the main focus of how information models are determined by carefully analyzing the structure of the data itself and form meaningful models is examined in how model generation gains better insight by using the concept of Granular Computing, and how the benefits of using such concepts help improve the core functionality of modeling from information as well as providing better information models which not only provide a good representation but also has improved interpretation capabilities.

Shown in Fig. 1.1 is a description of how information on some known information in some knowledge domain, as taken from samples, measurements, or readings can be transformed via some information granulation process into a collection of information granules (G1, G2, G3 and G4) which are formed on the basis that each information granule is built upon specific properties of the domain knowledge from which is was formed from. Not only is it sufficient to have information granules which perform well, but also that they are the correct size, that is, that the domain knowledge that each information granule represents is rightfully situated.

There are many research contributions in this book, all focused on applying the concepts of Granular Computing. One such contribution is the proposal of a nature inspired granulating algorithm, which takes a sample of some information and via the concept of Newtonian gravitational forces, finds similarities between all

M.A. Sanchez et al., *Type-2 Fuzzy Granular Models*,
SpringerBriefs in Computational Intelligence, DOI 10.1007/978-3-319-41288-7_1

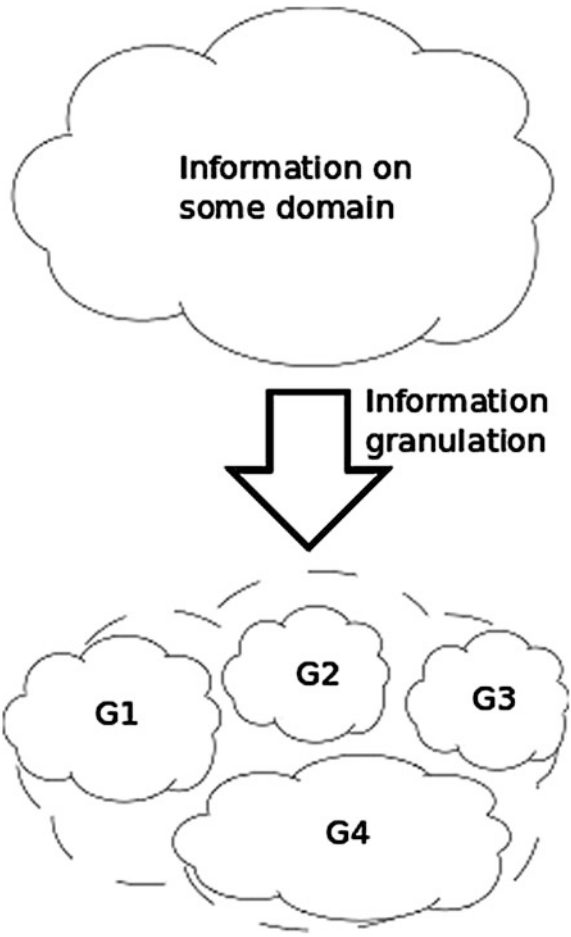

Fig. 1.1 Basic concept of information granulation. From transforming information from some domain into multiple information granules which capture the hidden relationships which is conformed from properties of the domain knowledge

information, thus creating different sized information granules which dynamically adapt to best present the meaningfulness of each individual core concept each information granule embodies; and through a learning technique the framework for a fuzzy inference system is adjusted based on available numerical evidence to form a complete granular model which can be inferred upon. A series of other contributions are also given which define multiple approaches to capturing uncertainty from data and implanting it into higher-type information granules, such as measuring differences in overlapping information granules on different viewpoints of the same domain knowledge; as well as via quantization of dispersion in the numerical evidence which formed each individual information granule; also, through an application of the principle of justifiable granularity as basis for finding dissimilarities in information granules; or by using the concept of numerical

evidence to create non-homogeneous higher-type fuzzy information granules. And lastly, an application example of how Granular Computing is used to solve a mobile robot's motion is also depicted and the performance comparison between different types of fuzzy information granule representations, such that normal fuzzy information granules are compared to higher-type fuzzy information granules.

As information granules require a medium for representing concepts of information, in this book, all representation mediums are in the form of fuzzy sets, such that fuzzy information granules are used throughout. These fuzzy information granules are a good medium because they can be adapted to represent many types of information granules, such as single points in the case of overly specific information granules; intervals, where a anything between such interval represents something of importance; fuzzy sets, such as Gaussian functions which define varying degrees of importance of information depending on the numerical location of such data; and type-2 fuzzy sets, much like Gaussian functions yet they also define a specified degree of uncertainty which lead to the representation of higher-type information granules.

This book is organized as follows: Chap. 2 presents the background and theory used by the rest of the book; Chap. 3 gives a description of all proposed methods; Chap. 4 shows a series of experiments based on the proposed methods, and an application of the usage of fuzzy granular models in a fuzzy control experiment; finally, Chap. 5 presents the conclusions of all carried out research in this book.

Chapter 2
Background and Theory

The focus of this book emphasizes the field of Granular Computing, where by nature is a vast area still under development. This section summarizes this matter as well as derived topics which aid Granular Computing, such as Fuzzy Sets and clustering algorithms.

2.1 Granular Computing

Granular Computing (GrC) is an area which was conceived by L.A. Zadeh in 1979 [1] which tries to adequate information granules to the information data that it tries to model. That is, an information granule is an atomic expression derived from an extracted model via a sample of data, where such granule can be small as to obtain a specific description of a part of a model, or coarser, denoting less precision and more generality. This being controlled by the requirements of the final granular model.

GrC is inspired by how the human brain processes information, in how it takes sets of factual data as to get a representation where a decision is required, then groups such data into different levels of resolution where each level can be ambiguous or very specific, suited to what can more effectively give an answer and finally make a correct decision based on collected data. With this in mind, GrC tries to reproduce this very behavior and take it into a computational algorithm which takes data, forms information granules and obtains a model more in affinity with reality.

It must be noted that GrC does nothing by itself; instead it works in conjunction with other algorithms where it applies its core theories and methodologies in order to obtain better information granules, and as such obtain better granular models.

M.A. Sanchez et al., *Type-2 Fuzzy Granular Models*,
SpringerBriefs in Computational Intelligence, DOI 10.1007/978-3-319-41288-7_2

2.2 Information Granule Representations

A key topic in GrC is information granule representation, where the simplest form of representation is by means of intervals, by which a delimitation of granule size exists thus having clear knowledge of what data is covered by such information granule. A clear example of this could be an age range, where the Universe of Discourse covers from 0 to 100 years old, but since a more specific age range is desired, such as 40–57 years old, this clearly eliminates all data outside this range, i.e. the ranges 0–39 years old and 58–100 years old is ignored. Although an interval is a simple representation, interval arithmetic [2] is not, and still requires more research to fully implement a viable solution to fuse GrC with interval arithmetic.

Another information granule representation schema which is also simple is Set Theory, represented by collection of items of the same type or categorical type. Similar to representation by intervals, an item either belongs or does not to a group, or set. Yet the main difference with intervals is that its mathematical operations are very mature. Although, as its representation schema is simple, more real world information granules cannot be properly represented.

The list of existing information granule representations is quite extensive, e.g. quotient space [3], fuzzy sets [4], rough sets [5], neural networks [6], shadowed sets [7], neutrosophic sets [8]; giving detailed attentions to each one would require much unnecessary attention which will not be given. Therefore, the chosen and used representation for information granules in this book is Fuzzy Sets (FS) since they are a representation which have much similarities with the core concept of GrC, which is to represent human cognition where imprecision is used throughout linguistic variables to represent granular models. With FSs an imprecise belongingness value can be represented where such imprecision is between the interval [0, 1]; zero, being that it does not belong at all to the set, one, being that it completely belongs to the set, and where any in-between value represents a certain degree of belonging to the set. With this in mind, FSs are an excellent representation for information granules, where in comparison with intervals or normal sets, where a datum either belongs or not to a set, with FSs a datum can simultaneously belong to multiple sets, although at different degrees, which is similar to human cognition.

2.3 Principle of Justifiable Granularity

This principle is a technique which purpose is to specify the adequate size of an information granule in such a way that it is not too small and has sufficient coverage of experimental data while at the same time it does not have too much coverage as to over generalize the granule. Figure 2.1 visually shows both these differences.

A double optimization must exist which takes into account both objectives:

1. The information granule must be as specific as possible.
2. The information granule must have sufficient numerical evidence.

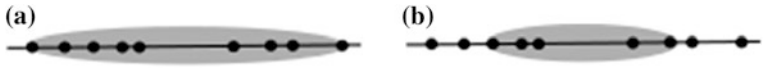

Fig. 2.1 Visual representation of two different objectives in data coverage, where **a** full experimental data coverage is achieved by the information granule, and **b** a more specific information granule covering only a portion of the experimental data

This double optimization is performed twice, as the length of the information granule has two sides which must be optimized, the left side interval from the Median of the data sample and the right side interval from the median of the data sample, as shown in Fig. 2.2. Both intervals, a and b, start from the Med(D) of available data D which formed said information granule.

Each information granule's set, shown in Eq. (2.1), is important, since it is the basis for finding the required lengths used to obtain a meaningful information granule. Where x_k are the individual data which belong to the information granule Ω.

$$\text{set}(x_k \in \Omega) \tag{2.1}$$

The set of is separated into two sections, as shown in Eqs. (2.2) and (2.3), as to conform to each interval, a and b respectively.

$$\text{set}(x_k \in \Omega, \ > Med(\text{D})) \tag{2.2}$$

$$\text{set}(x_k \in \Omega, \ < Med(\text{D})) \tag{2.3}$$

As for the required double optimization, it is obtained by maximizing Eqs. (2.4) or (2.5), for a and b respectively. Which uses a user criterion $\alpha \in [0, \alpha_{max}]$, where α_{max} obtains the smallest possible length achieved by the principle of justifiable granularity, and can be obtained via Eqs. (2.6) and (2.7), for a and b respectively. Described as the natural logarithm of the cardinality of the chosen side (a or b) divided by the length of the closest datum x_1 to *Med(D)*.

$$V(a^*) = max_{a<Med(D)}[V(a)] \tag{2.4}$$

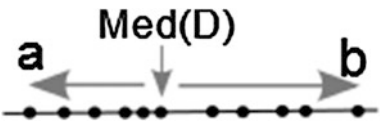

Fig. 2.2 Intervals a and b are separately optimized based on available numerical evidence from the formation of said information granule. Where both lengths start at the median of the information granule

$$V(b^*) = max_{b > Med(D)}[V(b)] \tag{2.5}$$

$$\alpha^a_{max} = \frac{card(x_k \in \Omega, < Med(\mathrm{D}))}{|Med(D) - x_1|} \tag{2.6}$$

$$\alpha^b_{max} = \frac{card(x_k \in \Omega, > Med(\mathrm{D}))}{|Med(D) - x_1|} \tag{2.7}$$

Regarding the double optimization itself, shown in Eqs. (2.8) and (2.9), for a and b respectively. Which is an integration of the probability density function from *Med(D)* to all prototypes of a, or b, multiplied by the user criterion for specificity α. In Appendix A, the demonstration of how Eqs. (2.8) and (2.9) are transformed into computational models is shown. And, in Appendix C.1, the code which computes the values for a and b is shown.

$$V(a) = \mathrm{e}^{(-\alpha|Med(D)-a|)} \int_{a}^{Med(D)} p(x)dx \tag{2.8}$$

$$V(b) = \mathrm{e}^{(-\alpha|b-Med(D)|)} \int_{Med(D)}^{b} p(x)dx \tag{2.9}$$

As an example of the behavior of *V(b)* in respect to the optimal length in respect to the chosen value of α, Fig. 2.3 shows how the peak of each curve locates the optimum length.

Finally, another example is shown, in Fig. 2.4, that demonstrates how the curve optimization affects both intervals, a and b, of the information granule. Here, it can clearly be seen where the optimal length is located in respect the peak of the optimization curve from the principle of justifiable granularity.

A more detailed study of the behavior of the principle of justifiable granularity can be found in [9].

2.4 Data Granulation Algorithms

To obtain granular models, help is needed from algorithms which can create these models from experimental data. In GrC it is much more common to find machine learning techniques rather than traditional statistical methods. With these intelligent algorithms, the most common sub-group of algorithms are clustering algorithms, due to their focus on finding similarities between the data itself and for differentiating between these found groups. This action generates a model from the data.

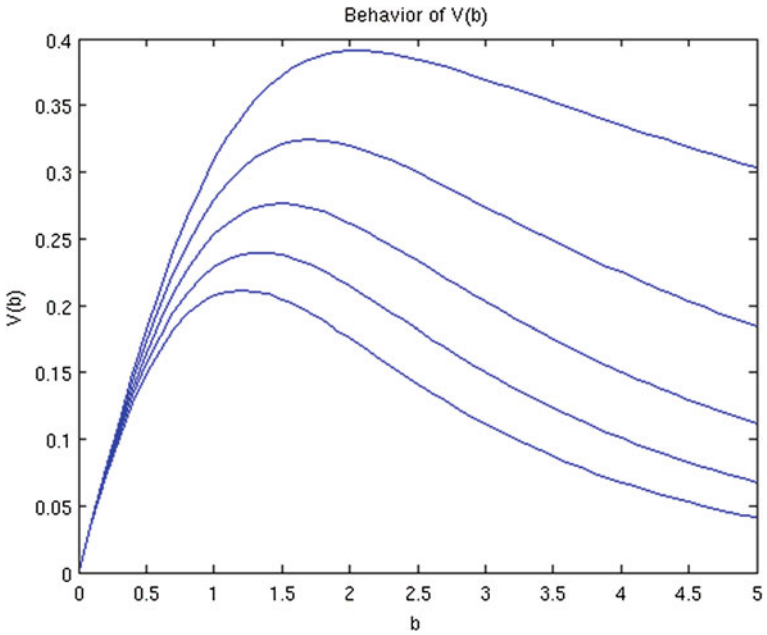

Fig. 2.3 Behavior of *V(b)* in respect to b, with changing values of α

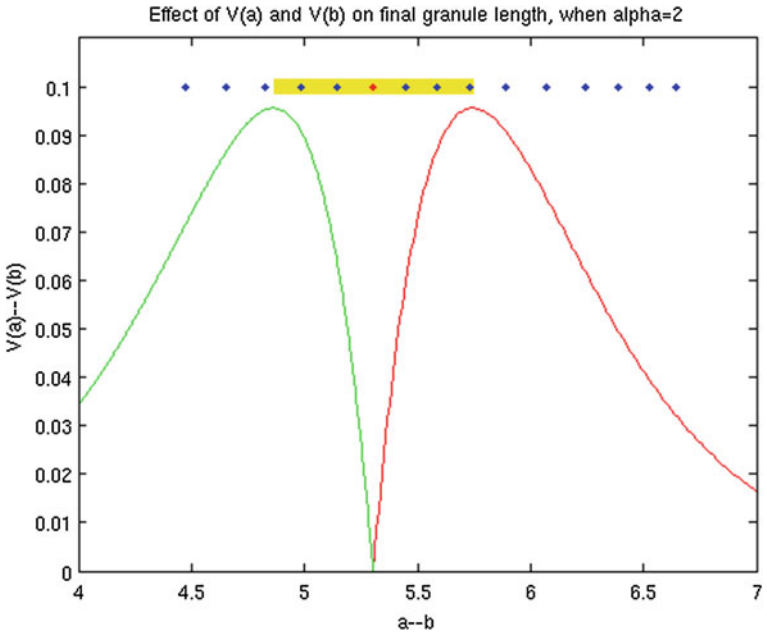

Fig. 2.4 Effect of *V(a)* and *V(b)* on the final size of the information granule, when $\alpha = 2$

Found groups of similar data are now called information granules which represent abstract models of a portion of relevant information as obtained from a given phenomenon. And the process of obtaining such information granules is called *data granulation*.

There is a great quantity of existing clustering algorithms, and among the most applied in GrC are K-Means [10] and Fuzzy C-Means (FCM) [11]. Where the K-Means algorithm contains a partition matrix that specifies which datum exactly belongs to which group, whereas the FCM algorithm specifies how much each datum belongs to each group. In K-means, each datum can only belong to one found group; while in FCM, each datum can belong to multiple groups, although in different degrees.

Many more clustering algorithms exist which perform the task of granulating data, such as the subtractive algorithm [12], or the fuzzy granular gravitational clustering algorithm [13] which will be seen in more detail further down in this document.

2.5 Fuzzy Logic

Fuzzy Logic (FL) can be seen as an advancement from bivariate logic. Where only two values can be expressed $\{0, 1\}$, whereas in FL values can be anything within the interval [0, 1]. This interval can express different degrees of perception, such as *not too much*, *very little*, or *more or less*. When used in a FS, aptly named a Type-1 Fuzzy Set (T1 FS), it can give better perceptual representations since ambiguities can now the modeled. Although many perceptual situations can be modeled using T1 FS, uncertainty is not one. For this, Interval Type-2 Fuzzy Sets (IT2 FSs) can be used, where its model directly handles, apart from imprecision, uncertainty. As for the complete model for Type-2 Fuzzy Sets, where T1 FSs and IT2 FSs are simplifications of Type-2 Fuzzy Sets. General Type-2 Fuzzy Sets (GT2 FSs) exist which in essence have a better handling of uncertainty than IT2 FS, this is due to how uncertainty is represented, instead of a 2D area, it is represented by a 3D volume.

2.5.1 Type-1 Fuzzy Sets

A T1 FS A, expressed by $\mu_A(x)$ where $x \in X$, described as $A = \{(x, \mu_A(x))|x \in X\}$. A visual example is shown in Fig. 2.5, where a generic Gaussian membership function represents a T1 FS.

A Type-1 Fuzzy Logic System (T1 FLS) can be easily described by a block diagram, shown in Fig. 2.6. Where the Fuzzifier takes crisp inputs and maps them into FS; the Inference, based on Rules, maps Fuzzy Sets from the antecedents to FSs from the consequents; finally, the Output Processor defuzzifies and outputs a crisp value.

The rule set for T1 FLSs are in the format as shown in Eq. (2.10) where the relation between the input and output space is mapped. Where R^l is a specific rule, x_p is input p,

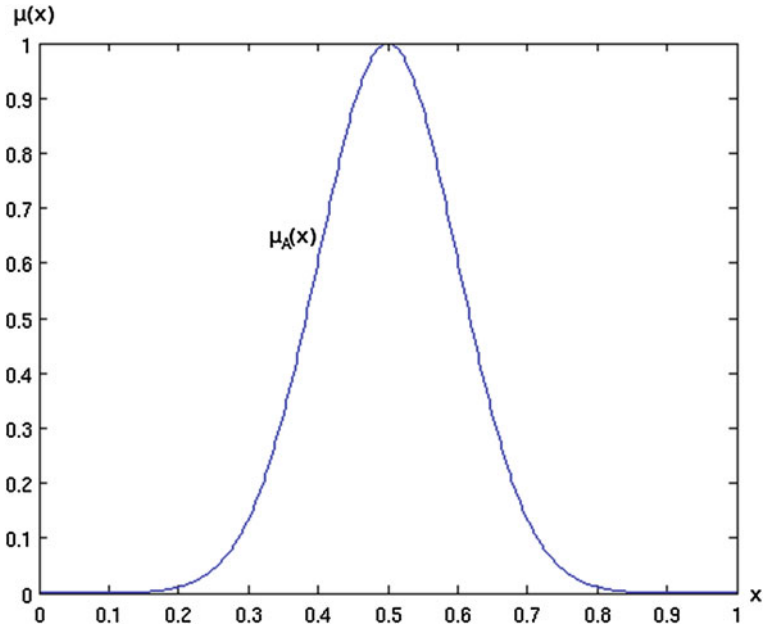

Fig. 2.5 Example of a generic T1 FS in the form of a Gaussian membership function

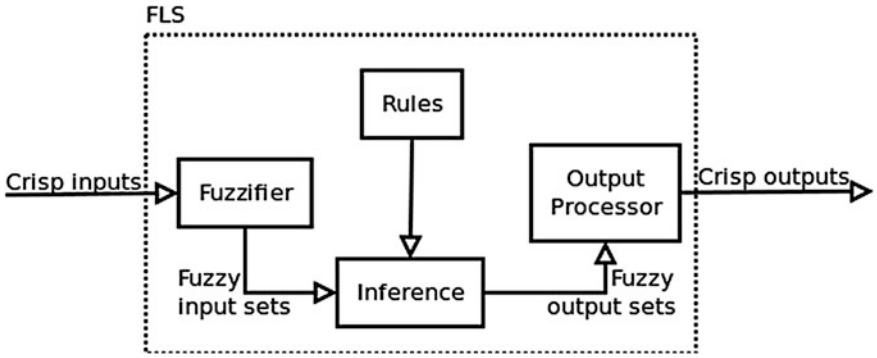

Fig. 2.6 Block diagram describing the main components of a T1 FLS

F_p^l is a membership function on rule l and input p, y is the output on membership function G^l. Both F and G are in the form of $\mu_F(x)$ and $\mu_G(y)$ respectively.

$$R^l : \text{IF } x_1 \text{ is } F_1^l \text{ and } \ldots \text{ and } x_p \text{ is } F_p^l, \quad \text{THEN } y \text{ is } G^l, \quad \text{where } l = 1, \ldots, M \tag{2.10}$$

As for the inference which calculates the compatibility between the antecedents and the consequents, using t-norms ($\tilde{*}$), Eq. (2.11) shows the basic methodology

required to process such t-norms. Where μ_{B^l} is the consequent membership function after being processed by the antecedents and Y is the domain space for the consequents.

$$\mu_{B^l}(y) = \mu_{G^l}(y) \tilde{*} \left\{ \left[\sup_{x_1 \in X_1} \mu_{x_1}(x_1) \tilde{*} \mu_{F_1^l}(x_1) \right] \tilde{*} \cdots \tilde{*} \left[\sup_{x_p \in X_p} \mu_{x_p}(x_p) \tilde{*} \mu_{F_p^l}(x_p) \right] \right\}, \quad y \in Y \tag{2.11}$$

The defuzzification process can be achieved in many ways, each obtaining similar results. Examples of common defuzzifiers are centroid, shown in Eq. (2.12); center-of-sums, shown in Eq. (2.13); or heights, shown in Eq. (2.14). Where y_i is a discrete position from Y, $y_i \in Y$, $\mu_B(y)$ is a FS which has been mapped from the inputs, c_{B^l} denotes the centroid on the lth output, a_{B^l} is the area of the set, and $\bar{y}^l$ is the point which has the maximum membership value in the lth output set.

$$y_c(x) = \frac{\sum_{i=1}^{N} y_i \mu_B(y_i)}{\sum_{i=1}^{N} \mu_B(y_i)} \tag{2.12}$$

$$y_a(x) = \frac{\sum_{l=1}^{M} c_{B^l} a_{B^l}}{\sum_{l=1}^{M} a_{B^l}} \tag{2.13}$$

$$y_h(x) = \frac{\sum_{l=1}^{M} \bar{y}^l \mu_{B^l}(\bar{y}^l)}{\sum_{l=1}^{M} \mu_{B^l}(\bar{y}^l)} \tag{2.14}$$

2.5.2 Type-2 Fuzzy Sets

Type-2 Fuzzy Sets can be used in two forms, by its simplified form, i.e. IT2 FSs; or its complete form, i.e. GT2 FSs. Both will be briefly described in this section.

An IT2 FS $\tilde{A}$ is represented by $\underline{\mu}_{\tilde{A}}(x)$ and $\overline{\mu}_{\tilde{A}}(x)$ which are the lower and upper membership functions respectively of $\mu_{\tilde{A}}(x)$, described as $\tilde{A} = \left\{ \left((x,u), \mu_{\tilde{A}}(x,u) = 1 \right) \middle| \forall x \in X, \forall u \in [0,1] \right\}$, where x is a subset of the Universe Of Discourse X, and u is a mapping of X into [0, 1]. A visual example is shown in Fig. 2.7, where a generic Gaussian membership function with uncertain standard deviation represents an IT2 FS.

An Interval Type-2 Fuzzy Logic System (IT2 FLS) can be easily described by a block diagram, shown in Fig. 2.8. Where the Fuzzifier takes crisp inputs and maps them into FS; the Inference, based on the Rules, maps Fuzzy Sets from the antecedents to FS from the consequents; then, the output can be chosen between

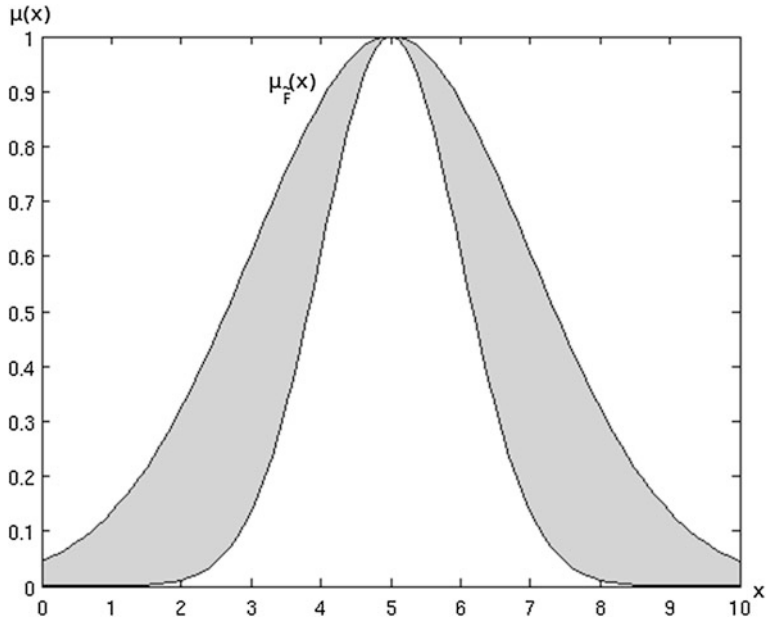

Fig. 2.7 Example of a generic IT2 FS in the form of a Gaussian membership function with uncertain standard deviation

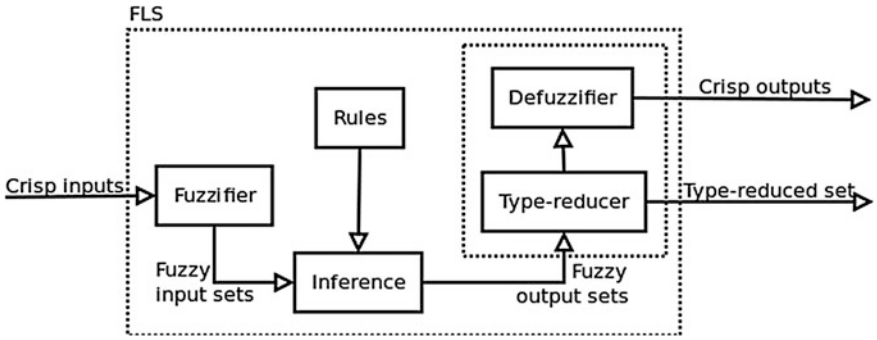

Fig. 2.8 Block diagram describing the main components of an IT2 FLS

obtaining a Type-reduced set (T1 FS) or Output Processor which defuzzifies and outputs a crisp value.

The rule set of an IT2 FLS maintains the same format as for the T1 FLS, but with a small notation difference, as shown in Eq. (2.15).

$$R^l : \text{IF } x_1 \text{ is } \tilde{F}_1^l \text{ and } \ldots \text{ and } x_p \text{ is } \tilde{F}_p^l, \quad \text{THEN } y \text{ is } \tilde{G}^l, \quad \text{where } l = 1, \ldots, M \tag{2.15}$$

The IT2 FLS inference can be summarized by Eq. (2.16). Where $\underline{f}^l$ and $\overline{f}^l$ represent the firing sets defined by Eqs. (2.16) and (2.17), and b is a interval defined by Eqs. (2.18) and (2.19).

$$\mu_{\tilde{B}}(y) = \int_{b \in \left[\left[\underline{f}^1 \tilde{*} \underline{\mu}_{\tilde{G}^1}(y)\right] v \ldots v \left[\left[\underline{f}^N \tilde{*} \underline{\mu}_{\tilde{G}^N}(y)\right]\right], \left[\left[\overline{f}^1 \tilde{*} \overline{\mu}_{\tilde{G}^1}(y)\right] v \ldots v \left[\left[\overline{f}^N \tilde{*} \overline{\mu}_{\tilde{G}^N}(y)\right]\right]\right]} 1/b, \quad y \in Y \tag{2.15}$$

$$\underline{f}^l(x') = \underline{\mu}_{\tilde{F}_1^l}\left(x_1'\right) \tilde{*} \cdots \tilde{*} \underline{\mu}_{\tilde{F}_{p1}^l}\left(x_p'\right) \tag{2.16}$$

$$\overline{f}^l(x') = \overline{\mu}_{\tilde{F}_1^l}\left(x_1'\right) \tilde{*} \cdots \tilde{*} \overline{\mu}_{\tilde{F}_{p1}^l}\left(x_p'\right) \tag{2.17}$$

$$\underline{b}^l(y) = \underline{f}^l \tilde{*} \underline{\mu}_{\tilde{G}^l}(y) \tag{2.18}$$

$$\overline{b}^l(y) = \overline{f}^l \tilde{*} \overline{\mu}_{\tilde{G}^l}(y) \tag{2.19}$$

A common technique for type-reducing an IT2 FLS is center-of-sets (cos), shown in Eq. (2.20). Where $Y_{\cos}$ is an interval set defined by two points $\left[y_l^i, y_r^i\right]$, which are obtained from Eqs. (2.21) and (2.22). Although other type-reducing techniques do exist, such as [14, 15].

$$Y_{\cos}(x) = \int_{y^1 \in \left[y_l^1, y_r^1\right]} \cdots \int_{y^M \in \left[y_l^M, y_r^M\right]} \int_{f^1 \in \left[\underline{f}^1, \overline{f}^1\right]} \cdots \int_{f^M \in \left[\underline{f}^M, \overline{f}^M\right]} 1 \Big/ \frac{\sum_{i=1}^M f^i y^i}{\sum_{i=1}^M f^i} \tag{2.20}$$

$$y_l = \frac{\sum_{i=1}^M f_l^i y_l^i}{\sum_{i=1}^M f_l^i} \tag{2.21}$$

$$y_r = \frac{\sum_{i=1}^M f_r^i y_r^i}{\sum_{i=1}^M f_r^i} \tag{2.22}$$

If a crisp output value is desired, a defuzzification of the type-reduced set y_l and y_r can be done, as shown in Eq. (2.23).

$$y(x) = \frac{y_l + y_r}{2} \tag{2.23}$$

A GT2 FS $\tilde{\tilde{A}}$ described by $\tilde{\tilde{A}} = \left\{ \left((x,u), \mu_{\tilde{\tilde{A}}}(x,u) \right) \middle| \quad \forall \ x \in X, \quad \forall \ u \in [0,1] \right\}$. Where $\mu_{\tilde{\tilde{A}}}(x,u)$ is the set of secondary membership functions which form the uncertainty on the GT2 FS, x is the domain of the primary membership function,

and u is the domain of the secondary membership functions. Figure 2.9 shows a sample GT2 FS as seen from an orthographic top view for a generic Gaussian primary membership function with uncertain mean and a Gaussian secondary membership function. Figure 2.10 also shows the same GT2 FS but with an isometric view.

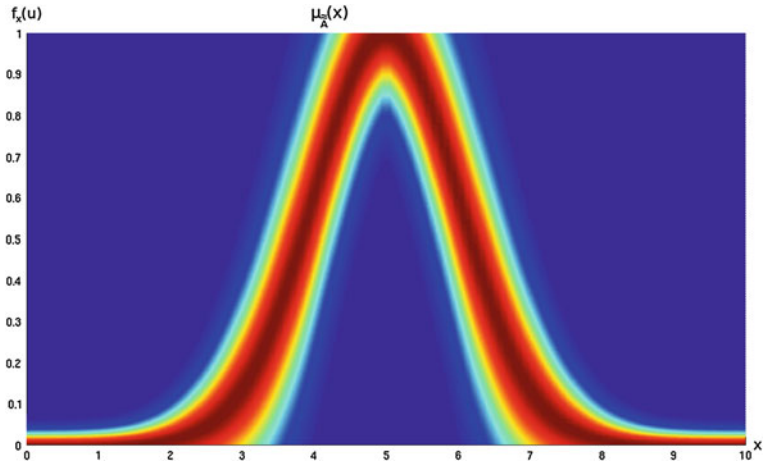

Fig. 2.9 Example of a GT2 FS in the form of a Gaussian primary membership function with uncertain mean and Gaussian secondary membership function, as seen from a orthographic top view

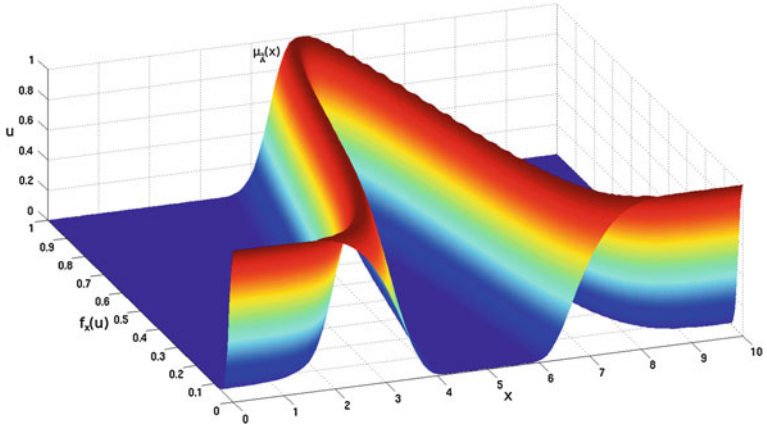

Fig. 2.10 Example of a GT2 FS in the form of a Gaussian primary membership function with uncertain mean and Gaussian secondary membership function, as seen from an isometric view

The rule set for a GT2 FLS maintains the format for T1 FLSs and IT2 FLSs, but with a slight notation difference, shown in Eq. (2.24).

$$R^l : \text{IF } x_1 \text{ is } \tilde{\tilde{F}}_1^l \text{ and} \ldots \text{and } x_p \text{ is } \tilde{\tilde{F}}_p^l, \quad \text{THEN } y \text{ is } \tilde{\tilde{G}}^l, \quad \text{where } l = 1, \ldots, M \tag{2.24}$$

The inference of a GT2 FLS is quite different from the previous shown FLS (T1 and IT2), as the complete and original inference is very computational complex, hence the FLS simplifications of T1 and IT2. Yet it can be summed by two main operations, *meet* and *join*, shown in Eqs. (2.25) and (2.26) respectively. Analogous to the inference operations of either T1 FLS or IT2 FLS, they are used together in order to find the compatibility of the antecedents with the inputs, and accordingly their mapping into the consequents space.

$$\mu_{\tilde{\tilde{A}}}(x) \sqcup \mu_{\tilde{\tilde{B}}}(x) = \left\{ \left[\int_{u \in J_x^u} \int_{u \in J_x^w} f_x(u) \tilde{*} g_x(w) / (u \vee w) \right] \right\} \tag{2.25}$$

$$\mu_{\tilde{\tilde{A}}}(x) \sqcap \mu_{\tilde{\tilde{B}}}(x) = \left\{ \left[\int_{u \in J_x^u} \int_{u \in J_x^w} f_x(u) \tilde{*} g_x(w) / (u \wedge w) \right] \right\} \tag{2.26}$$

The defuzzification section of a GT2 FLS uses a centroid technique, shown in Eq. (2.27). Where $C_{\tilde{\tilde{A}}}$ defines the centroid of a GT2 FS, and θ_i is a combination associated to the secondary degree $f_{x_1}(\theta_1) \tilde{*} \cdots \tilde{*} f_{x_N}(\theta_N)$. It must be noted that this defuzification is of O(n^n), which basically makes it uncomputational as the discretization increases. Although newer techniques exist which can highly reduce this into a more manageable and viable algorithm, via approximations.

$$C_{\tilde{\tilde{A}}} = \int_{\theta_1 \in J_{x_1}} \cdots \int_{\theta_N \in J_{x_N}} [f_{x_1}(\theta_1) \tilde{*} \ldots \tilde{*} f_{x_N}(\theta_N)] \Big/ \frac{\sum_{i=1}^{N} x_i \theta_i}{\sum_{i=1}^{N} \theta_i} \tag{2.27}$$

GT2 FLS approximations reduce the computational complexity of its original set operations, especially the defuzzification. These approximation techniques involve reducing the three-dimensional GT2 FS into multiple, and manageable, IT2 FSs.

One such approximation technique is done by obtaining an α-Plane [16] via the union of the α-cuts on all vertical slices on α_i. An α-Plane can be defined by Eq. (2.28) where $\tilde{A}_\alpha$ is an IT2 FS.

$$\tilde{A}_\alpha = \int_{\forall x \in X} \int_{\forall u \in [0,1]} \{(x, u) | f_x(u) \geq \alpha\} \tag{2.28}$$

Another approximation technique is a zSlice [17], which obtains a plane by calculating the intervals for all vertical cuts on a given z. This zSlice is defined by Eq. (2.29). Here, the difference between an α-Plane and a zSlice is that an α-Plane performs it cuts on $\alpha/\tilde{A}_\alpha$, while a zSlice cuts on $\tilde{Z}_\alpha$ projected unto $X \times Y$.

$$\tilde{A}_z = \int_{\forall x \in X} \int_{\forall y \in [0,1]} z/(x,y) \tag{2.29}$$

When an α-Plane or a zSlice is calculated, the inference of an IT2 FLS is used. The union of all α-Planes or zSlices is defined by Eqs. (2.30) and (2.31) for α-Planes and zSlices respectively. Where $R_{\tilde{A}_\alpha}$ is one horizontal slice at level α.

$$\tilde{\tilde{A}} = \bigcup_{\forall \alpha \in [0,1]} R_{\tilde{A}_\alpha} \tag{2.30}$$

$$\tilde{\tilde{A}} = \bigcup_{\forall z \in [0,1]} \tilde{A}_z \tag{2.31}$$

2.6 Fuzzy Granular Computing

Having seen a description of what both GrC and Fuzzy Logic are, the term Fuzzy Granular Computing can be inferred from a combination of both. It is the direct application of the abstract concepts from GrC specifically when used with Fuzzy Logic. The concept of a FS is analogous to the concept of an Information Granule, where each represents an abstract model for a collection of data. And a collection of FSs, or Fuzzy Information Granules, becomes a Fuzzy Granular model.

References

1. Zadeh, L.A.: Fuzzy sets and information granularity. Adv. Fuzzy Set Theor. Appl. 3–18 (1979)
2. Hickey, T., Ju, Q., Van Emden, M.H.: Interval arithmetic: from principles to implementation. J. ACM **48**(5), 1038–1068 (2001)
3. Zhang, L.Z.L., Zhang, B.Z.B.: Quotient space based multi-granular computing. In: 2005 IEEE International Conference on Granular Computing, vol. 1, p. 98 (2005)
4. Pedrycz, W., Song, M.: Granular fuzzy models: a study in knowledge management in fuzzy modeling. Int. J. Approx. Reason. **53**(7), 1061–1079 (2012)
5. Sai, Y.S.Y., Nie, P.N.P., Chu, D.C.D.: A model of granular computing based on rough set theory. In: 2005 IEEE International Conference on Granular Computing, vol. 1, pp. 233–236 (2005)

6. Pedrycz, W., Vukovich, G.: Granular neural networks. Neurocomputing **36**(1–4), 205–224 (2001)
7. Pedrycz, W., Vukovich, G.: Granular computing with shadowed sets. Int. J. Intell. Syst. **17**(2), 173–197 (2002)
8. Solis, A.R., Panoutsos, G.: Granular computing neural-fuzzy modelling: a neutrosophic approach. Appl. Soft Comput. **13**(9), 4010–4021 (2013)
9. Sanchez, M.A., Castillo, O., Castro, J.R.: An analysis on the intrinsic implementation of the principle of justifiable granularity in clustering algorithms. In: O. Castillo, P. Melin, J. Kacprzyk (eds.) Recent Advances on Hybrid Intelligent Systems, vol. 451. Springer, Berlin, Heidelberg, pp. 121–134 (2013)
10. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
11. Bezdek, J.C., Ehrlich, R., Full, W.: FCM: the fuzzy c-means clustering algorithm. Comput. Geosci. **10**(2), 191–203 (1984)
12. Chen, M.-Y.: A hybrid ANFIS model for business failure prediction utilizing particle swarm optimization and subtractive clustering. Inf. Sci. (Ny) **220**, 180–195 (2013)
13. Sanchez, M.A., Castillo, O., Castro, J.R., Rodríguez-Díaz, A.: Fuzzy granular gravitational clustering algorithm. North Am. Fuzzy Inf. Process. Soc. **2012**, 1–6 (2012)
14. Yeh, C.-Y., Jeng, W.-H.R., Lee, S.-J.: An enhanced type-reduction algorithm for type-2 fuzzy sets. IEEE Trans. Fuzzy Syst. **19**(2), 227–240 (2011)
15. Mendel, J.M.: Enhanced Karnik-Mendel algorithms. IEEE Trans. Fuzzy Syst. **17**(4), 923–934 (2009)
16. Mendel, J., Liu, F., Zhai, D.: α-plane representation for type-2 fuzzy sets: theory and applications. IEEE Trans. Fuzzy Syst. **17**(5), 1189–1207 (2009)
17. Wagner, C., Hagras, H.: Toward general type-2 fuzzy logic systems based on zSlices. IEEE Trans. Fuzzy Syst. **18**(4), 637–660 (2010)

Chapter 3
Advances in Granular Computing

This book is a research compendium in the area of Fuzzy Granular Computing. Two main branches are handled, a proposed fuzzy granulating algorithm, and higher-type information granule formation, although more work was performed on the latter.

3.1 Fuzzy Granular Gravitational Clustering Algorithm

A Fuzzy granulation algorithm was proposed in [1, 2] which uses the concept of gravitational forces to form information granules, and was aptly named Fuzzy Granular Gravitational Clustering Algorithm (FGGCA).

It is based on Newton's Law of Universal Gravitation, as shown in Eq. (3.1) which calculates the gravitational force between two bodies. Where F_g is the gravitational force, G is a gravitational constant which equals 6.674×10^{-11}, m_1 and m_2 are the masses of both bodies, and $\|x_1, x_2\|$ is the distance between the centers of mass between both bodies.

$$F_g = G\frac{m_1 m_2}{\|x_1, x_2\|^2} \tag{3.1}$$

The premise of the algorithm is that gravitational forces are simulated inside a confined and normalized space $[-1, 1]$, where each datum represents a point body with a mass of 100 kg. Here, each attribute, or feature, of a dataset is considered a dimension in Euclidean n-space.

The algorithm starts by first calculating all gravitational forces between each data point, as shown in Eq. (3.2). Where F_{ij} is the gravitational force between the i-th and j-th data points, and $i \neq j$.

M.A. Sanchez et al., *Type-2 Fuzzy Granular Models*,
SpringerBriefs in Computational Intelligence, DOI 10.1007/978-3-319-41288-7_3

$$F_{ij} = G\frac{m_i m_j}{\left\|x_i, x_j\right\|^2} \tag{3.2}$$

The sum of all gravitational forces per each data point must then be calculated, as shown in Eq. (3.3), this calculates an individual gravitational force density for each data point relative to the rest of the data points.

$$F_i^{density} = \sum_{j=1}^{n} F_{ij} \tag{3.3}$$

A premise is given where data points with more gravitational force density are more likely to be actual cluster centers than those with the lowest densities. For this, all densities must be sorted into descending order, from highest density to lowest density. This also rearranges all data points, from $x \in X$ to $x^{sorted} \in X$.

Having sorted all data points, the next step is to start joining nearby, and gravitationally dense, points with each other in a pairwise manner. This process is done iteratively for all data points while the following condition is true: IF $\min\left(\left\|x_i, x_j\right\|\right) < radius$ THEN *start joining data points*. Where $i \neq j$, and *radius* is a user criterion used to decide the maximum size of each information granule. Here, the behavior of different values of *radius* affect in such a way that for a small size, more information granules will be found, whereas if the value is high, there will be less amount of information granules. When the condition is met, it means that x_i is more gravitationally dense than x_j therefore x_i absorbs x_j, as shown in Eqs. (3.4) and (3.5).

$$x_i \cup x_j \tag{3.4}$$

$$m_i \cup m_j \tag{3.5}$$

The joining of x_j unto x_i updates the position of x_i, via Eqs. (3.6)–(3.8). Where $\rho_{barycenter}$ is the gravitational center of mass between both data points, λ is a scaling factor between x_i and x_j.

$$\rho_{barycenter} = \left(\frac{m_j}{m_i + m_j}\right)\left\|x_i, x_j\right\| \tag{3.6}$$

$$\lambda = \frac{\rho_{barycenter}}{\left\|x_i, x_j\right\|} \tag{3.7}$$

$$x_i = x_i + \lambda\left(x_j - x_i\right) \tag{3.8}$$

The described algorithm iterates this process for all $x \in X$ until all data points have been joined together, or the distance in the condition previously shown, with the user criterion of *radius*, is not met. Each iteration of this process usually reduces

the cardinality of X to about half its original size, all while maintaining the original sum of mass in the system.

After this iterative process ends the final step before iterating once again from Eq. (3.2), is to adjust the user criterion value of *radius* in order to control the amount and size of each information granule, this is done via Eq. (3.9). Where Δ is another user set criterion for how fast *radius* will change.

$$radius = radius \times \Delta \tag{3.9}$$

The algorithm will iterate until all remaining interactions are beyond *radius*. Once finished, the centers for each information granule are obtained; the following description will now show how the information granule's size is calculated. These information granule sizes are also found by the help of gravitational forces.

Since FGGCA is fuzzy in nature, membership functions form the rule set of the model. Therefore Gaussian membership functions were chosen to represent each individual fuzzy information granule. As a result, two parameters are required, a location point x (as calculated by the previously described algorithm) and a standard deviation σ to delimit the size of the fuzzy information granule.

To calculate the required σ values, let us define the found cluster center as x^c. Where the premise is to find which cluster center x^c exerts more gravitational force upon x when $x^c \neq x_i$, when found $x_i \in x^c$ is established. This process iterates through all $x \in X$.

The algorithm to find the size of the fuzzy information granules is dictated by first obtaining the sets of data points upon which each x^c have more influence over, as shown in Eq. (3.10). Where F_j is the gravitational force exerted between x_j^c and x_i, with conditional $x_j^c \neq x_i$. Afterwards after x_i iterates through all x_j^c, the F_j which exerts more gravitational influence adds x_i into its set $x_{setOfCluster_j}$.

$$F_j = G\frac{m_j^c m_i}{\left\| x_j^c, x_i \right\|^2} \tag{3.10}$$

When all $x \in X$ are transferred into $x_{setOfCluster_j}$ the σ_k^c values can now be calculated for each x^c, as shown in Eq. (3.11). Where σ_k^c is the standard deviation for each cluster center x^c on the k-th input.

$$\sigma_k^c = std\left(x_{setOfCluster_j}\right) \tag{3.11}$$

The described algorithm so far has given a technique for calculating fuzzy information granules which are represented by the antecedents in a FLS, yet the antecedents have not been reviewed. The proposed granular FLS has Takagi-Sugeno-Kang (TSK) [3–5] consequents. To adjust these linear first Order polynomials, a known method based on Least Square Estimator method (LSE) to adjust all coefficient parameters [6] is used.

The consequent parameter adjustment considers c cluster centers $\{x_1, x_2, \ldots, x_c\}$. The input space variables are represented by y_i, and the output space variables are represented by z_i. Where z_i is in the form of a first Order linear function, as shown in Eq. (3.12), G_i are the input constant parameters, and h_i the constants for each z_i.

$$z_i = G_i y + h_i \tag{3.12}$$

As each x_i defines a fuzzy rule, and training data x exists, the firing strengths ω_i per each rule are calculated via Eq. (3.13), with the consideration that all membership functions are Gaussian. Where each x_i is used alongside the obtained σ^c and x^c.

$$\omega_i = \mathrm{e}^{-\frac{1}{2}\left\|\frac{x_i - x^c}{\sigma^c}\right\|^2} \tag{3.13}$$

A parameter τ_i is defined in order to use LSE, as shown in Eq. (3.14). Which can be rewritten as Eq. (3.15). Now, z_i can be defined to the form shown in Eq. (3.16), which describes a matrix of parameters to be optimized by the LSE method, taking the form of $AX = B$.

$$\tau_i = \frac{\omega_i}{\sum \omega_i} \tag{3.14}$$

$$z = \sum_{i-1}^{c} \tau_i z_i = \sum_{i=1}^{c} \tau_i (G_i x + h_i) \tag{3.15}$$

$$\begin{bmatrix} z_1^T \\ \vdots \\ z_n^T \end{bmatrix} = \begin{bmatrix} \tau_{1,1} x_1^T & \tau_{1,1} & \cdots & \tau_{c,1} x_1^T & \tau_{c,1} \\ \vdots & & & & \\ \tau_{1,n} x_n^T & \tau_{1,n} & \cdots & \tau_{c,n} x_n^T & \tau_{c,n} \end{bmatrix} \begin{bmatrix} G_1^T \\ h_1^T \\ \vdots \\ G_c^T \\ h_c^T \end{bmatrix} \tag{3.16}$$

where,

$$B = \begin{bmatrix} z_1^T \\ \vdots \\ z_n^T \end{bmatrix} \tag{3.17}$$

$$A = \begin{bmatrix} \tau_{1,1} x_1^T & \tau_{1,1} & \cdots & \tau_{c,1} x_1^T & \tau_{c,1} \\ \vdots & & & & \\ \tau_{1,n} x_n^T & \tau_{1,n} & \cdots & \tau_{c,n} x_n^T & \tau_{c,n} \end{bmatrix} \tag{3.18}$$

$$X = \begin{bmatrix} G_1^T \\ h_1^T \\ \vdots \\ G_c^T \\ h_c^T \end{bmatrix} \tag{3.19}$$

Knowing that LSE has the form $AX = B$. Where B, in Eq. (3.17), is a matrix of the output values; A, in Eq. (3.18), is a matrix of constants; and X, in Eq. (3.19), is a matrix of estimated parameters. The final solution is given by Eq. (3.20).

$$X = \left(A^T A\right)^{-1} A^T B \tag{3.20}$$

As already shown, the FGGCA is separated into two sections, apart from the TSK consequent learning algorithm, which are finding the clusters, and calculating the information granule sizes. The two following procedures summarize both this processes. In Appendix C.2, used code is shown.

procedure findClusters(x, *radius*, Δ)

1. LOOP while INTERACTIONS_EXIST==TRUE
2. INTERACTIONS_EXIST=FALSE; initialize exit condition for clustering
3. $F_{ij} = G\frac{m_i m_j}{x_i, x_j{}^2}$; calculate all interacting gravitational forces in the system, where $i \neq j$
4. $F_i^{density} = \sum_{j=1}^{n} F_{ij}$; calculate the gravitational density of F_i
5. $x = sort(x, F_i^{density})$; sort x in reference to $F_i^{density}$. Where the x_i with the highest gravitational density is set as the first value in x, and each x_i is subsequently set as the gravitational density is reduced
6. LOOP foreach i in x_i
7. IF $\min(x_i, x_j) < radius$ THEN; verify if there are points which can interact and join together, where $i \neq j$
8. INTERACTIONS_EXIST=TRUE; indicate that another iteration can be done
9. $m_i \cup m_j$; join the mass of x_j unto x_i
10. $\rho_{barycenter} = \left(\frac{m_j}{m_i + m_j}\right)\left\|x_i, x_j\right\|$; calculate the barycenter distance between x_i and x_j
11. $\lambda = \frac{\rho_{barycenter}}{x_i, x_j}$; calculate a scaling factor λ
12. $x_i = x_i + \lambda\left(x_j - x_i\right)$; update de position of x_i to its new location, based on λ
13. $radius = radius \times \Delta$; update the value of *radius* in relation to Δ
14. RETURN x

procedure findSizeOfGranules(x, x^c)

1. $F_j = G\frac{m_j^c m_i}{x_j^c, x_i^2}$; calculate all interacting gravitational forces in the system, where $x_j^c \neq x_i$
2. $F_j^{index} = \max\left(F_j\right)$; find the index of F_j which exerts a higher gravitational force.
3. $x_{setOfCluster_j} = x_{F_j^{index}}$; assign to a set of $x_{setOfCluster_j}$ the value of x_i which index is F_j^{index}
4. LOOP foreach k in *NumberOfInputs*(x)
5. $\sigma_k^c = std\left(x_{setOfCluster_j}\right)$; calculate the standard deviation for each input variable of each cluster
6. RETURN σ^c

In Appendix B.1, some figures are shown which visually describe the behavior of the FGGCA when dealing with 2D data.

3.2 Higher-Type Information Granule Formation

Most research in this book focuses on the formation of Fuzzy Higher-Type information granules which leads to multiple approaches being presented.

3.2.1 A Hybrid Method for IT2 TSK Formation Based on the Principle of Justifiable Granularity and PSO for Spread Optimization

An initial method for the formation of IT2 TSK FIS was proposed in [7] which introduces a method for using the principle of justifiable granularity as a means to heuristically obtaining an interval which reflects the IT2 FS uncertainty, afterwards its IT2 TSK consequent spreads are optimized via a Particle Swarm Optimization (PSO) algorithm. This method described in more detail in the following paragraphs.

Considering that Gaussian membership functions with uncertain means will be used, shown in Fig. 3.1, the required inputs of the proposed method are cluster centers from any clustering algorithm as to define the initial rule set for the IT2 FIS and subsets of data which are best represented by each cluster center.

The process which obtains the representative subsets for each cluster center is executed via Eq. (3.21). Where $\left\|x_i, x_j\right\|$ is the Euclidean distance in n-space between a cluster x_c and a datum x_i.

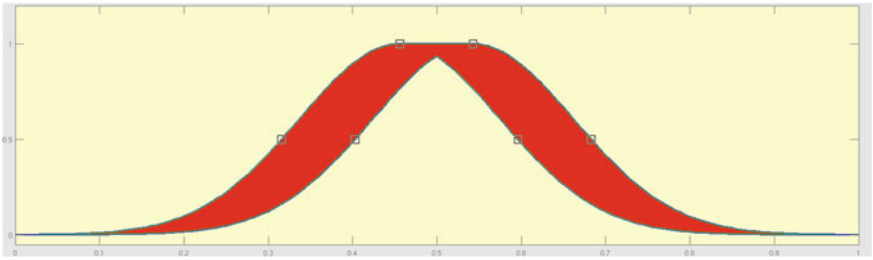

Fig. 3.1 Sample IT2 Gaussian membership function with uncertain mean

$$\|x_c, x_i\| = \sqrt{\sum_{i=1}^{n} (x_c - x_i)^2} \tag{3.21}$$

When all subsets are found the information granule's coverage must be calculated, i.e. the standard deviation σ. This is done through Eq. (3.22). Where $\sigma_{j,k}$ is the standard deviation of the j-th rule and k-th input, x_i is each datum from the subset obtained from Eq. (3.21), $x_c^{j,k}$ is the cluster center for the j-th rule and k-th input, and n is the cardinality of the subset.

$$\sigma_{j,k} = \frac{\sum_{i=1}^{n} (x_i - x_c^{j,k})^2}{n-1} \tag{3.22}$$

Up to this point a Type-1 Gaussian membership function can be created, but the end product is an IT2 Gaussian membership function with uncertain mean. The following process obtains the remaining required parameter to form the IT2 FS.

To obtain the uncertainty on the IT2 FS the principle of justifiable granularity is used as a means to heuristically measure it. Done by forcing each information granule to its largest coverage by using the user criterion value of α_{max} on each side of the interval, as described by Eqs. (2.6) and (2.7). When both intervals a and b are obtained, their difference is used to heuristically measure the uncertainty for the IT2 FS, as shown in Eq. (3.23). Where τ is the measure of uncertainty for a specific information granule.

$$\tau = |a - b| \tag{3.23}$$

The obtained value of τ is used with Eqs. (3.24) and (3.25). Where the center parameter of the IT2 Gaussian membership function with uncertain mean holds the uncertainty of the IT2 FS, i.e. τ offsets the mean of the Gaussian membership function thus adding the uncertainty. This concludes the section for forming Higher-type fuzzy information granules in the antecedents of an IT2 FLS.

$$m_{j,k}^{l} = x_c^{j,k} - \tau_{j,k} \tag{3.24}$$

$$m_{j,k}^{l} = x_c^{j,k} + \tau_{j,k} \tag{3.25}$$

The IT2 linear TSK consequents are calculated in a two step process. First, an LSE algorithm [8, 9] is used twice, as the Gaussian membership function with uncertain mean is parameterized by a left and right T1 Gaussian membership function, the LSE is applied as if two T1 FLSs existed. When all TSK coefficients are obtained, the average of both sets of parameters is used, as shown in Eq. (3.26). Where φ_l and φ_r are the coefficient sets for the left and right side T1 FLSs, and φ is the set used for the proposed IT2 FLS. The code for this process is included in Appendix C.3.

$$\varphi = \frac{\varphi_l + \varphi_r}{2} \tag{3.26}$$

The second part of the process for calculating the coefficients for the IT2 TSK consequents is to obtain the spreads of each coefficient, with format as shown in Eq. (3.27). Where c are the coefficients, x the dependent input variables, and s the spreads.

$$y^i = \sum_{k=1}^{p} c_k^i x_k + c_0^i - \sum_{k=1}^{p} |x_k| s_k^i - s_0^i \tag{3.27}$$

To adjust the spreads, a PSO algorithm [10] was chosen. Where the initial population is randomly generated with values within [0, 0.09], i.e. very small spreads. As the PSO uses multiple parameters which can speed up the convergence to a good result, the ones used here are shown in Table 3.1. Where three values, marked in **bold**, specify how the optimization is desired. The individual acceleration factors are fixed, this causes the PSO to search more slowly inside the confined space, otherwise the spread significantly increases and that is not a desirable behavior. The rest of the parameters are set to typical recommended parameter values.

The objective function for the PSO is another very important topic that must be addressed in order to obtain the best possible solution. Shown in Eq. (3.28) is the manually adjusted objective function. Where θ is the RMSE of the output coverage, and ϑ is the RMSE of the size of the Footprint Of Uncertainty.

$$objVal = 0.8 * \theta + 0.2 * \vartheta \tag{3.28}$$

Table 3.1 PSO parameters used for optimizing the spread of the IT2 TSK linear polynomials

PSO parameter	Value
Number of particles	30
Number of iterations	**50**
Initial value of the individual-best acceleration factor	**0.5**
Final value of the individual-best acceleration factor	0.5
Initial value of the global-best acceleration factor	0.5
Final value of the global-best acceleration factor	**0.5**
Initial value of the inertia factor	0.9
Final value of the inertia factor	0.4

Once the PSO concludes its 50 iterations, the spread values are expected to be quasi-optimal, with some minor error on the FOU coverage, or optimal, with zero error on the FOU coverage.

3.2.2 *Information Granule Formation via the Concept of Uncertainty-Based Information with IT2 FS Representation with TSK Consequents Optimized with Cuckoo Search*

This approach, published in [11] uses the concept of uncertainty-based information as a means to measure uncertainty and use it for the formation of IT2 FS antecedents. Its consequents are IT2 TSK linear polynomials which are optimized via a Cuckoo Search [12] algorithm.

The concepts of uncertainty and information are closely related in such a way that their fundamental characteristic is that involved uncertainty from any situation is a result from information deficiency. Therefore, an assumption can be made where uncertainty is reduced by obtaining new relevant information, .e.g. obtaining new experimental results, and by measuring the difference between both sets of information (a priori and a posteriori), then, can some uncertainty can be measured.

In Fig. 3.2, the shown diagram represents the general idea behind the behavior of uncertainty-based information. A reduction of uncertainty can be obtained via the difference of two uncertain models from the same information. That is, an a priori uncertainty model is obtained with a first sample of information, where as an a posteriori uncertainty model is obtained with a second sample of information related to the same situation.

With an inspiration on uncertainty-based information, higher-type information granules can be formed via the capture and measurement of uncertainty.

Through a first sample of information $D1$, an uncertain model can be created. And through a second sample of information $D2$ another similar uncertain model can be also created. These two models of uncertainty are analogous to the models in

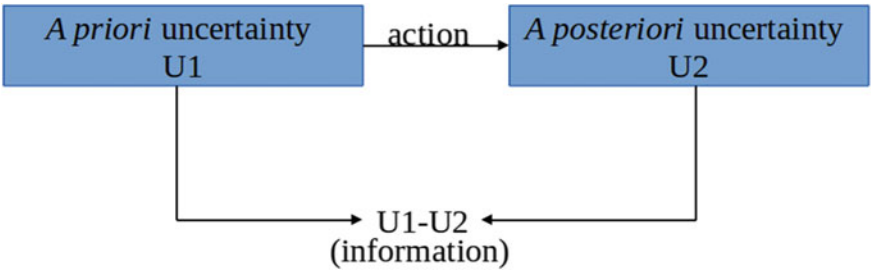

Fig. 3.2 Diagram of the behavior of the uncertainty-based information where uncertainty is reduced by the difference between two uncertain models of the same information

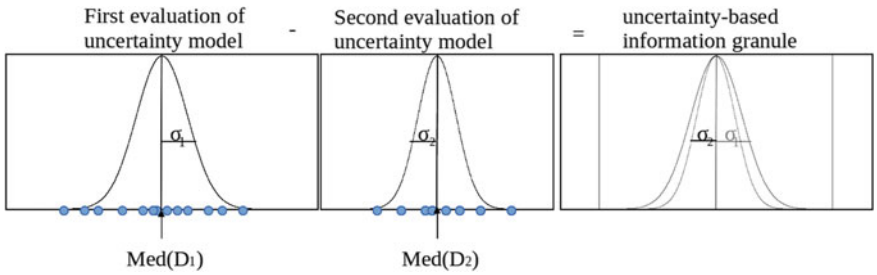

Fig. 3.3 Explanatory diagram of how the proposed approach measures and defines the uncertainty, and forms an IT2 FS with such uncertainty

the theory of uncertainty-based information, a priori and a posteriori uncertainty models. As uncertainty-based information tries to reduce uncertainty by measuring it, for the purpose of this proposed technique, it uses such measurement of uncertainty and integrates it into an IT2 FS.

The proposed approach works by taking two sets of samples from an information source, $D1$ and $D2$, and obtaining two Gaussian Type-1 membership functions. As there will probably be a difference between the two taken samples, this difference is what ultimately reflects the direct measurement of uncertainty which is then imposed unto an IT2 Gaussian membership function with an uncertain standard deviation. This behavior is shown in Fig. 3.3.

An implementation to this approach uses the Subtractive clustering algorithm [6] to obtain rule sets. The following algorithm summarizes how the fuzzy Higher-Type information granules are formed:

1. Obtain first rule set from *D1*, comprised of centers m and standard deviations σ_1. Where σ_1 is obtained by finding sets from data closest to each m^i.
2. Obtain σ_2 via *D2*. Where σ_2 is obtained by finding sets from data closest to each m^i.
3. Form antecedent IT2 membership functions by using found parameters m, σ_1^i and σ_2^i.
4. The consequents are obtained via an optimization of IT2 TSK linear polynomials with a Cuckoo Search optimization algorithm [12].

Another implementation uses the Fuzzy C-Means clustering algorithm [13] to obtain rule sets. This implementation is a better reflection of the concept of a priori and a posteriori uncertainty models. Appendix C.4 shows the code which executes this algorithm.

1. Obtain first rule set from *D1*, comprised of centers m_1 and standard deviations σ_1. Where σ_1 is obtained via the U_1 matrix partition, where the highest value represents to which m_1^i each datum belongs to. This obtains the a priori uncertainty model.
2. Obtain first rule set from $D2$, comprised of centers m_2 and standard deviations σ_2. Since the FCM randomly chooses an initial U partition matrix, to conform to rule set similarity between both executions. The second FCM's U partition

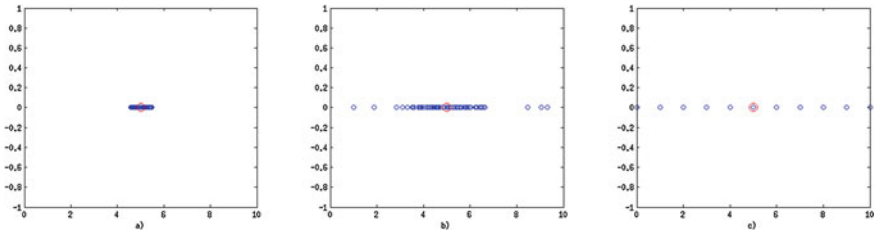

Fig. 3.4 Visual depiction of varying degrees of data dispersion. **a** Low dispersion, **b** medium dispersion, and **c** high dispersion

matrix is initialized with the values of U_1. Where σ_1 is obtained via the U_2 matrix partition, where the highest value represents to which m_1^i each datum belongs to. This obtains the a posteriori uncertainty model.

3. Form antecedents IT2 membership functions by obtaining the mean of m_1^i and m_2^i, and using both σ_1^i and σ_2^i.
4. The consequents are obtained via an optimization of IT2 TSK linear polynomials with a Cuckoo Search optimization algorithm.

3.2.3 *Method for Measurement of Uncertainty Applied to the Formation of IT2 FS*

In this approach, by Sanchez et al. [14] a heuristic methodology is proposed which based on the Coefficient of Variation can measure a degree of uncertainty from a dataset and construct Higher-type information granules in the form of IT2 FSs. The consequents of the IT2 FLS use TSK linear polynomials which are optimized via a Cuckoo Search optimization algorithm.

The premise of the approach is that uncertainty can be interpreted as a case for data dispersion in a sample of data. As shown in Fig. 3.4, where (a) shows a low dispersion scenario where most data samples are very close together, therefore having low dispersion, or in the case of the premise of this approach, low uncertainty; or the case of (b), where although there is a concentration of data near the center there are still data samples existing far from it, this can be interpreted as medium uncertainty; or (c), where all data is equally spaced though the Universe of Discourse, therefore an interpretation is given such that uncertainty is so high that any place could potentially obtain another sample.

The conversion is done via the use of the Coefficient of Variation cv, shown in Eq. (3.29). Where σ is the standard deviation, and m is the mean of the set.

$$cv = \frac{\sigma}{m} \tag{3.29}$$

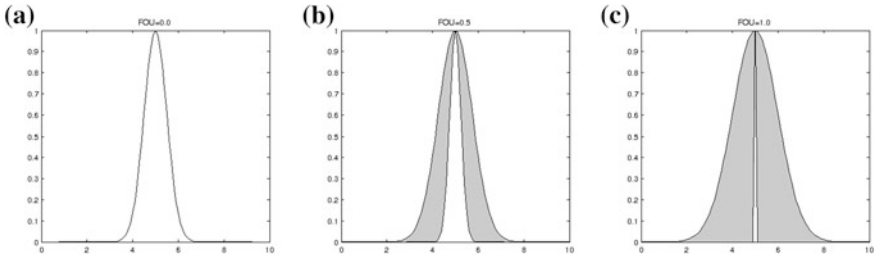

Fig. 3.5 Varying degrees of FOU in an IT2 FS. Where **a** FOU = 0.0 (low dispersion), **b** FOU = 0.5 (medium dispersion), and **c** FOU = 1.0 (high dispersion)

A limitation exists on *cv* which adds a restriction that only non-negative values can be computed.

The dispersion-uncertainty relation, when taking into account the use of IT2 FSs when the FOU is used as a measure of uncertainty, a direct proportion relation is used, as shown in Eq. (3.30). Such relation states that with low dispersion, a small FOU exists, with a medium amount of dispersion, a medium amount of FOU exists, and with a high amount of dispersion, a high amount of FOU exists.

$$c_v \propto FOU \tag{3.30}$$

The dispersion-uncertainty relation can be visually perceived in Fig. 3.5, where different degrees of dispersion are reflected by a proportionally sized FOU.

To form Higher-type information granules for the antecedents of the FLS, a FIS prototype must first be acquired, conformed of a rule set composed of centers for Gaussian membership functions and the accompanying subsets of data which created each prototype. The proposed approach works on each FS independently from each other, each FS is conformed of a center value m, and a standard deviation σ obtained from the subset of sample data δ. And for each FS a cv is calculated using Eq. (3.29). This value is now used to search for a near optimal FOU area in an IT2 FS. The search starts by first considering the highest possible obtainable area $FOU^{\max} = \int \tilde{A}(x)dx = 1$ with the previously obtained σ, as shown in Fig. 3.5.c, achieved when $\sigma_1 = 0$ and $\sigma_2 = 2\sigma$. Here, possible values of FOU are in the interval [0, 1]. The search is then performed with discrete small steps λ until the FOU value which equals c_v is found. The smallest value σ^0 is defined as $\sigma_1 = \sigma_2 = \sigma^0 = \sigma$, as shown in Fig. 3.5a. Each step λ affects σ_1 and σ_2 by a size increment/decrement, as shown in Eq. (3.31). This is done iteratively while $\|\sigma_1, \sigma_2\| \leq 2\sigma$, where $\|\sigma_1, \sigma_2\|$ is the Euclidean distance between σ_1 and σ_2, or until $cv = FOU_i$, and FOU_i is the current iteration of the IT2 FS modified by $\sigma^i_{1,2}$.

$$\sigma^i_{1,2} = \sigma^{i-1}_{1,2} \pm \lambda; \quad i = 0, 1, \ldots, n \tag{3.31}$$

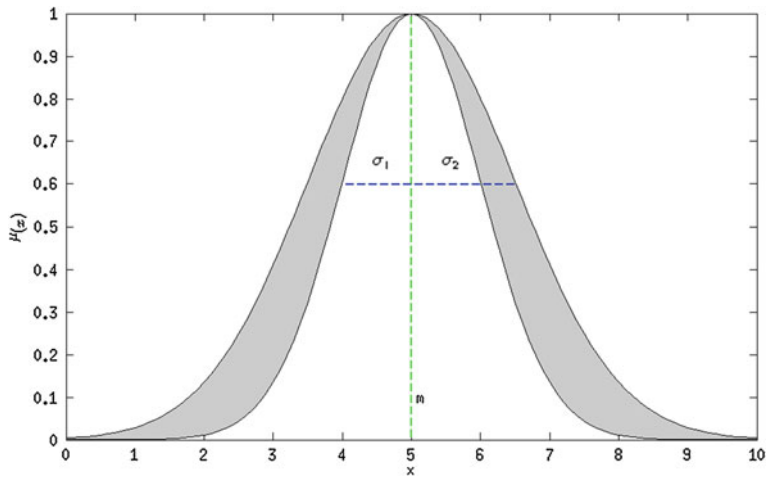

Fig. 3.6 IT2 FS represented by a Gaussian membership function with uncertainty in the standard deviation. Formed with three variables: σ_1,σ_2 and m

When the search has found the values of σ_1 and σ_2 which together form the desired FOU, the IT2 FS can be formed, i.e. a Higher Type Information Granule has been formed, as shown in Fig. 3.6.

As for obtaining the consequents of the FLS, a Cuckoo Search optimization algorithm is used for this purpose. Where IT2 TSK linear polynomials embody the consequents of the FLS.

3.2.4 Formation of GT2 Gaussian Membership Functions Based on the Information Granule Numerical Evidence

This work was done by Sanchez et al. [15], where a technique for the formation of GT2 FS is proposed by means of inspiration on the principle of justifiable granularity, whereby the concept of numerical evidence is used in the formation of IT2 FS.

The proposed technique can be defined by the following steps:

1. Use any clustering algorithm to obtain an initial set of cluster centers (C) and sets of data (D) which formed each information granule.
2. Calculate the required parameters to form a Gaussian primary membership function. Extract an individual c_i, from $c \in C$, and using the subset d_i, from $d \in D$, obtain the standard deviation σ_i.

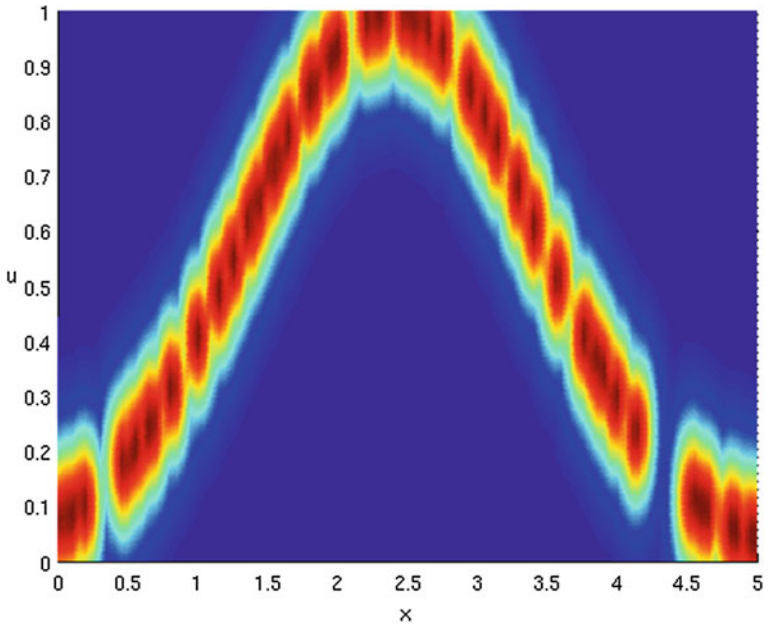

Fig. 3.7 Proposed GT2 Gaussian membership function with constant $\boldsymbol{\sigma}$ for all secondary membership functions. Top view

3. Initialize σ for Gaussian secondary membership functions. Where the initial value of $\sigma = 0.1$.
4. For each data point belonging to the subset d_i, create a secondary membership function with revolution on $fx(u)$ axis, following u of the primary membership function.

Two approaches exist to how the value of σ for the secondary membership functions changes and how it affects the GT2 FS. The first approach maintains a fixed value of σ for all secondary membership functions. Appendix C.5 shows code which creates this specific type of GT2 membership functions. Figures 3.7 and 3.8, show a top view and orthogonal view of a sample GT2 FS with a constant σ for the secondary membership functions.

The other proposed approach to forming GT2 Gaussian membership functions is to change σ for the secondary membership functions based on u. Where the closest points to the center will increase in size, since more data is expected which could potentially fall there, whereas the sides have less possibilities of obtaining new data points, therefore reducing the size. Appendix C.6 shows code which creates this specific type of GT2 membership functions. Figures 3.9 and 3.10 depicts a sample Gaussian primary membership function with σ for the secondary membership function dependent on u.

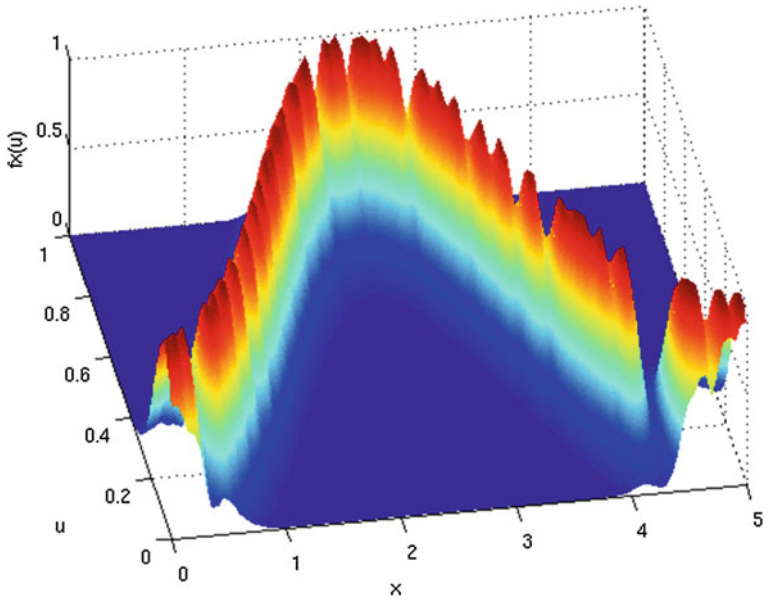

Fig. 3.8 Proposed GT2 Gaussian membership function with constant σ for all secondary membership functions. Orthogonal view

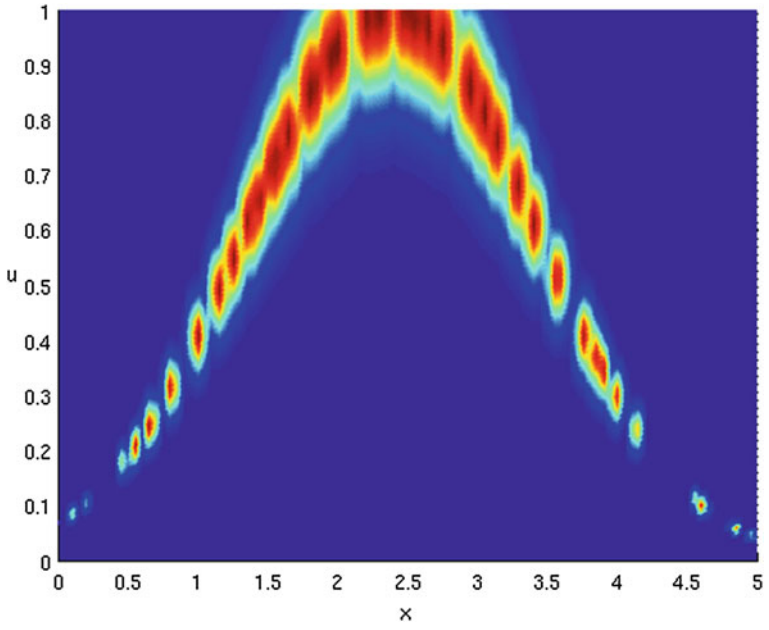

Fig. 3.9 Proposed GT2 Gaussian membership function with σ dependent on u for all secondary membership functions. Orthogonal view

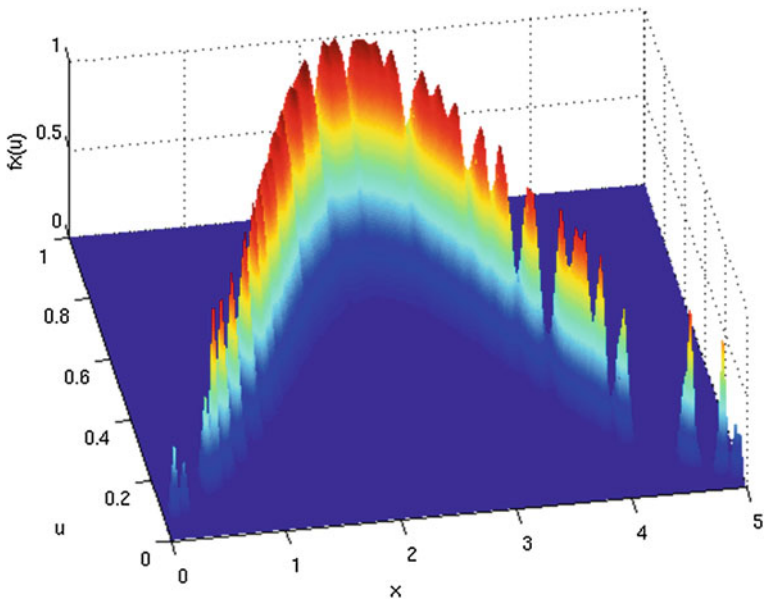

Fig. 3.10 Proposed GT2 Gaussian membership function with $\boldsymbol{\sigma}$ dependent on $\boldsymbol{u}$ for all secondary membership functions. Orthogonal view

In Appendix B.2, additional figures are shown for an application in solving the Iris dataset. For both fixed, and σ dependent on u.

References

1. Sanchez, M.A., Castillo, O., Castro, J.R., Rodríguez-Díaz, A.: Fuzzy granular gravitational clustering algorithm. North Am. Fuzzy Inf. Process. Soc. **2012**, 1–6 (2012)
2. Sanchez, M.A., Castillo, O., Castro, J.R., Melin, P.: Fuzzy granular gravitational clustering algorithm for multivariate data. Inf. Sci. (Ny) **279**, 498–511 (2014)
3. Buckley, J.J.: Sugeno type controllers are universal controllers. Fuzzy Sets Syst. **53**(3), 299–303 (1993)
4. Takagi, T., Sugeno, M., Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man. Cybern. **SMC-15**(1), 116–132 (1985)
5. Sugeno, M., Kang, G.: Structure identification of fuzzy model. Fuzzy Sets Syst. **28**(1), 15–33 (1988)
6. Chiu, S.L.: Fuzzy model identification based on cluster estimation. J. Intell. Fuzzy Syst. **2**, 267–278 (1994)
7. Sanchez, M.A., Castro, J.R., Perez-Ornelas, F., Castillo, O.: A hybrid method for IT2 TSK formation based on the principle of justifiable granularity and PSO for spread optimization. In: 2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), pp. 1268–1273 (2013)
8. Jang, J.S.R.: ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans. Syst. man Cybern. **23**(3), 665–685 (1993)

9. Jang, J.-S.R.: Fuzzy modeling using generalized neural networks and Kalman filter algorithm. In: Proceedings of the ninth National conference on Artificial intelligence, pp. 762–767 (1991)
10. Kennedy, J., Eberhart, R.: Particle swarm optimization. In Proceedings of ICNN'95—International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
11. Sanchez, M.A., Castillo, O., Castro, J.R.: Information granule formation via the concept of uncertainty-based information with Interval Type-2 Fuzzy Sets representation and Takagi–Sugeno–Kang consequents optimized with Cuckoo search. Appl. Soft Comput. **27**, 602–609 (2015)
12. Yang, X.-S.: Cuckoo search via Lévy flights. In: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), pp. 210–214 (2009)
13. Bezdek, J.C., Ehrlich, R., Full, W.: FCM: the fuzzy c-means clustering algorithm. Comput. Geosci. **10**(2), 191–203 (1984)
14. Sanchez, M.A., Castillo, O., Castro, J.R.: Method for measurement of uncertainty applied to the formation of interval type-2 fuzzy sets. In: Melin, P., Castillo, O., Kacprzyk, J. (eds.) Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization, vol. 601, pp. 13–25. Springer International Publishing, Cham (2015)
15. Sanchez, M.A., Castro, J.R., Castillo, O.: Formation of general type-2 Gaussian membership functions based on the information granule numerical evidence. In: 2013 IEEE Workshop on Hybrid Intelligent Models and Applications (HIMA), pp. 1–6 (2013)

Chapter 4
Experimentation and Results Discussion

All experimentation is focused on the previously shown approaches to fuzzy information granulation. Most experiments were done with benchmark datasets which will be described in the following paragraph. Two benchmark dataset types were used: classification, and identification.

Classification type benchmarks datasets are described in Table 4.1, showing name, number of features, number of classes, and sample size. Where these datasets are iris, wine, glass identification, seeds, image segmentation, Haberman's survival, and mammographic mass [1]. The iris dataset, has 4 input features (petal length, petal width, sepal length, and sepal width), and 3 outputs (iris setosa, iris virginica, and iris versicolor) with 50 samples of each flower type, with a total of 150 elements in the dataset. The wine dataset, with 13 input features of different constituents (Alcohol, malic acid, ash, alkalinity of ash, magnesium, total phenols, flavanoids, nonflavanoid phenols, proanthocyanidins, color intensity, hue, OD280/OD315 of diluted wines, and proline) identifying 3 distinct italian locations where the wine came from. With 59, 71, and 48 elements respectively in each class, for a total of 178 elements in the whole dataset. The glass identification dataset, has 9 input variables (refractive index, sodium, magnesium, aluminum, silicon, potassium, calcium, barium, and iron), and 7 classes (building windows float processed, building windows non float processed, vehicle windows float processed, containers, tableware, and headlamps). With 70, 76, 17, 13, 9, and 29 elements respectively in each class, for a total of 214 elements in the whole dataset. The seeds dataset, with 7 input features (area, perimeter, compactness, length of kernel, width of kernel, asymmetry coefficient, and length of kernel groove) and 3 output classes (Kama, Rosa, and Canadian) with 70 samples of each class, for a total of 210 elements in the dataset. The image segmentation dataset, with 19 input features (column of the center pixel region, row of the center pixel region, number of pixels in a region, result of line extraction algorithm, lines of high contrast, mean contrast of horizontally adjacent pixels in the region, standard deviation contrast of horizontally adjacent pixels in the region, mean contrast of vertically adjacent pixels in the region, standard deviation contrast of vertically adjacent pixels in the region,

M.A. Sanchez et al., *Type-2 Fuzzy Granular Models*,
SpringerBriefs in Computational Intelligence, DOI 10.1007/978-3-319-41288-7_4

Table 4.1 Description of classification benchmark datasets used for experimentation

Dataset	No. of features	No. of classes	Sample size
Iris	4	3	150
Wine	13	3	178
Glass	9	6	214
Seed	7	3	210
Image segmentation	19	7	2310
Haberman's survival	3	2	306
Mammographic mass	5	2	830

Table 4.2 Description of identification benchmark datasets used for experimentation

Dataset	No. of inputs	No. of outputs	Sample size
Complex curve	1	1	94
Engine behavior	2	2	1199
Thermex behavior	1	1	236
Bodyfat	13	1	252
Housing	13	1	506

average of region, average over region of R, average over region of B, average over region of G, measure of excess red, measure of excess blue, measure of excess green, transformation of RGB, saturation mean, and hue mean), and 7 classes (brickface, sky, foliage, cement, window, path, and grass) with 330 samples of each class, for a total of 2310 samples in the whole dataset. The Haberman's survival dataset, has 3 input features (age of patient at time of operation, year of operation, and number of positive axillary nodes detected), and 2 output classes (patient survived 5 years or longer, and patient died within 5 years) with 225 and 81 samples respectively for each class, for a total of 306 samples in the dataset. And, the mammographic mass dataset, with 5 input features (BI-RADS assessment, age, shape, margin, and density) and 2 output classes (benign, and malignant) with 427 benign samples and 403 malignant samples, totaling 830 samples in the dataset.

Identification type benchmarks datasets are described in Table 4.2, showing name, number of features, and sample size. Where these datasets are complex curve [2], engine behavior [1], thermex behavior [2], bodyfat [3], and housing [1]. Where these datasets are a complex curve, with 1 input (x) and 1 output (y), and 94 total samples. Engine behavior dataset, with 2 inputs (fuel rate, speed) and 2 outputs (torque, nitrous oxide emissions), and 1199 total samples. Thermex behavior dataset, with 1 input (temperature) and 1 output (thermex), with 236 total samples. The bodyfat dataset, with 13 input (age, weight, height, neck circumference, chest circumference, abdomen 2 circumference, hip circumference, thigh circumference, knee circumference, ankle circumference, biceps (extended) circumference, forearm circumference, and wrist circumference) features and 1 output (bodyfat percentage), with 252 elements in total. The housing dataset with 13 inputs (per capita crime rate per town, proportion of residential land zoned lots over 25,000 sq. ft., proportion of

non-retail business acres per town, 1 if tract bounds Charles river, 0 otherwise, nitric oxides concentration, average number of rooms per dwelling, Proportion of owner-occupied units built prior to 1940, weighted distances to five Boston employment centers, index of accessibility to radial highways, Full-value property-tax rate per $10,000, pupil-teacher ratio by town, $1000(Bk - 0.63)^2$, where Bk is the proportion of blacks by town, and percent lower status of the population) and 1 output (median value of owner occupied homes in each neighborhood), with 506 samples in total.

4.1 Granulation Algorithms

The first set of shown experiments will be for the work done in Sect. 3.1, as this is a clustering algorithm the classification accuracy is measured via a confusion matrix [4]. In Table 4.3, the accuracy of the classification performance is measured, where letters in **bold** show the best achieved performance, and "–" signifies no published results exists. Here, result comparison was made with SC^3SR [5], ASC_1 [6], LODE [7], ASC_2 [8], CE-RCTO [9], BH [10], MPC-KMeans [11], SCC [12],

Table 4.3 Classification accuracy percentage of various algorithms with multiple datasets

Algorithm	Dataset						
	Iris	Wine	Glass	Seeds	Image segmentation	Haberman's survival	Mammographic mass
FGGCA	**97.22**	96.66	**93.645**	**95.572**	90.586	**76.195**	82.72
SC^3SR	97.1	–	66.4	–	–	–	–
ASC_1	91.8	70	–	–	74.2	–	–
LODE	96	–	–	–	–	77.3	–
ASC_2	95.7	–	–	–	–	–	–
CE-RCTO	89.5	–	73.15	–	**93.63**	–	–
BH	89.98	–	–	–	–	–	–
MPC-KMeans	–	82.2	–	–	–	–	–
SCC	–	**97.1**	64.9	–	–	–	–
Spectral CAT	97	97	70	–	65	–	–
LDA-KM	–	83	51	–	–	–	–
Boost-NN	–	–	–	75.6	–	–	–
DGP	–	–	–	–	–	–	**86**
AMA	–	96	87	–	–	65	–
Bac0+Bac1	–	–	–	–	–	70.59	–
CGCA	–	–	–	92	–	–	–
FLD	–	–	64.79	–	93.43	–	–
IP-DCA-VNS	96.7	93.2	80.4	–	–	–	–
TPMSVM +GA	–	–	–	–	–	72.89	–
DJS	91.3	71.9	43.9	–	57.6	–	–

Spectral CAT [13], LDA-KM [14], Boost-NN [15], DGP [16], AMA [17], Bac0 +Bac1 [18], CGCA [19], FLD [20], IP-DCA-VNS [21], TPMSVM+GA [22], TPMSVM+GA [22], and DJS [23].

These results show that the performance of the proposed FGGCA is very good when contrasted against the other shown clustering algorithms. Although the FGGCA obtained better results in 4 cases out of 7. Yet in the other 3 cases, a similar result was achieved. And generally speaking, where the other clustering algorithms sometimes achieve high performance, they sometimes also achieve a very low performance, whereas FGGCA is much more stable in achieving a high performance throughout all tests.

4.2 Higher-Type Information Granule Algorithms

These sets of experiments are for the proposed approach shown in Sect. 3.2.1. For this case, performance is measured by the output coverage. As Higher-type information granules are more robust against information uncertainty, three types of noise was inserted into a complex curve dataset: no noise, 10 and 0 dBi. Where dBi is a measure of power of noise. In Figs. 4.1, 4.2 and 4.3, three instances of the FOU coverage is shown: the first, when no noise is inserted into the dataset, a thin FOU is present, as there is no variation, yet the general behavior of the complex curve is easily followed by the IT2 FIS which was created by the proposed method; the second, with 10 dBi noise inserted into the dataset, it is clearly visible that although there is noise the FOU has a very good coverage and still follows the center of the noise behavior fairly well; the third, although there is significant noise at 0 dBi, yet the FOU still manages to mostly cover all output points, and still achieving to

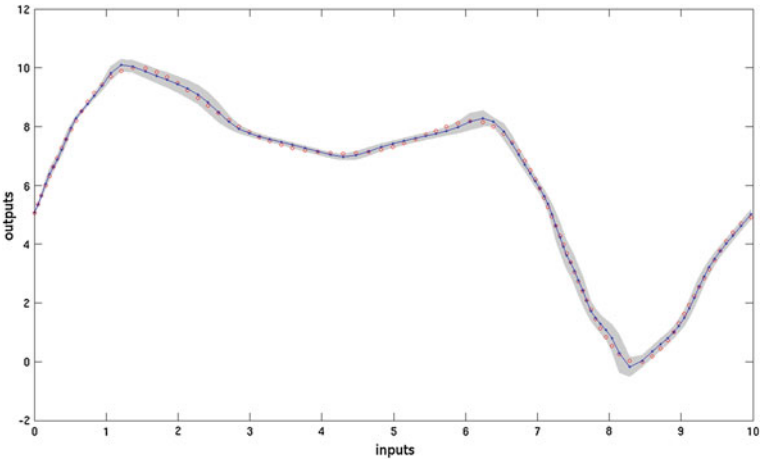

Fig. 4.1 The final result of the formed IT2 FIS alongside the FOU, when there is no added noise

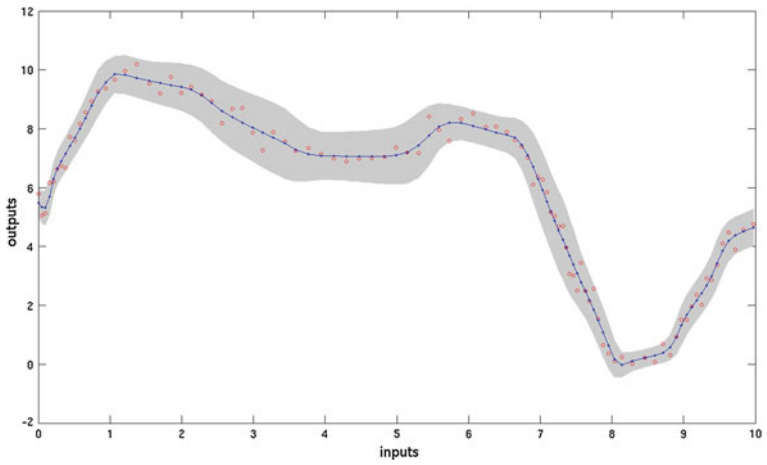

Fig. 4.2 The final result of the formed IT2 FIS alongside the FOU, when there is 10 dBi noise

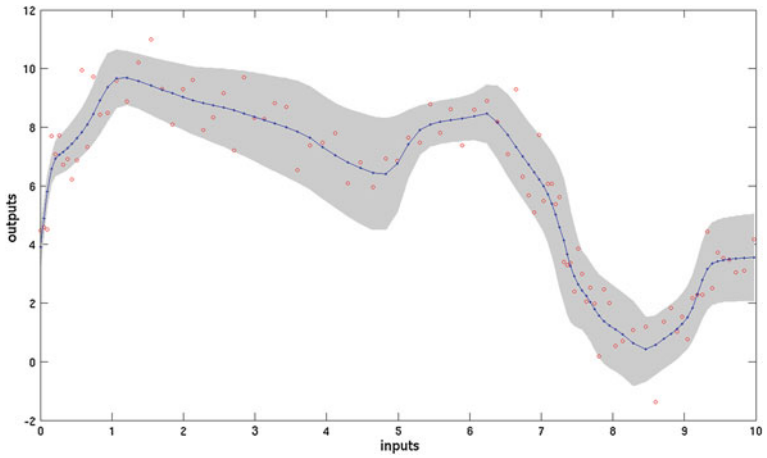

Fig. 4.3 The final result of the formed IT2 FIS alongside the FOU, when there is 0 dBi noise

follow the general behavior of the curve. Additional results are shown in Appendix B.3.

More experiments were executed upon this approach to Higher-type information granulation, from Sect. 3.2.2, now using classification datasets, such as Iris, Wine, and Glass. Results are shown Table 4.4, where the classification accuracy for Iris, Wine and Glass datasets depicting the minimum, mean, maximum, and standard deviation after 30 execution runs. As well as the RMSE for the previously shown complex curve dataset. These results show a general good performance throughout tested benchmark datasets.

Table 4.4 Obtained results for various benchmark datasets

	Classification accuracy %			RMSE
	Iris	Wine	Glass	Complex curve
Min	86.66	88.88	44.18	0.36
Mean	93.99	93.05	68.43	0.69
Max	100	97.22	86.04	1
Std	5.164	5.982	15.592	0.181

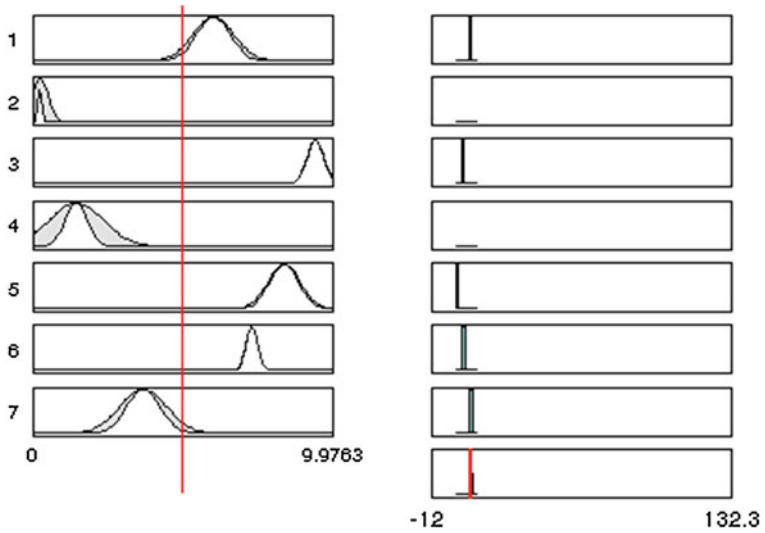

Fig. 4.4 IT2 FIS for solving the complex curve dataset

The next set of experiments pertains to the approach shown in Sect. 3.2.3, where the output coverage is used as a performance measure. Two experiments will be shown, for a complex curve and thermex behavior. In this case, Hold-out type test was done, where 40 % of the dataset was used for training and the other 60 % for testing, all chosen randomly. Figures 4.4 and 4.5 show the formed IT2 FISs for both cases, complex curve and thermex behavior. Here, the main point of interest is the fact that the antecedent FSs have different sized FOUs, meaning that each IT2 FS's FOU is adjusted to best fit the data it is representing.

The coverage for these two experiments can be perceived by Figs. 4.6 and 4.7, for the complex curve, and thermex behavior, respectively. Where the use of linear IT2 polynomials cause the observed behavior where the output FOU has difficulty in adapting to gradient changes. Yet the FOU still manages to cover 100 % of the behavior of both curves. In Appendix B.4, an additional experiment is shown.

These two experiments are more focused on showing the performance of the formed antecedents, which are IT2 FSs, rather than the adjusted IT2 linear TSK

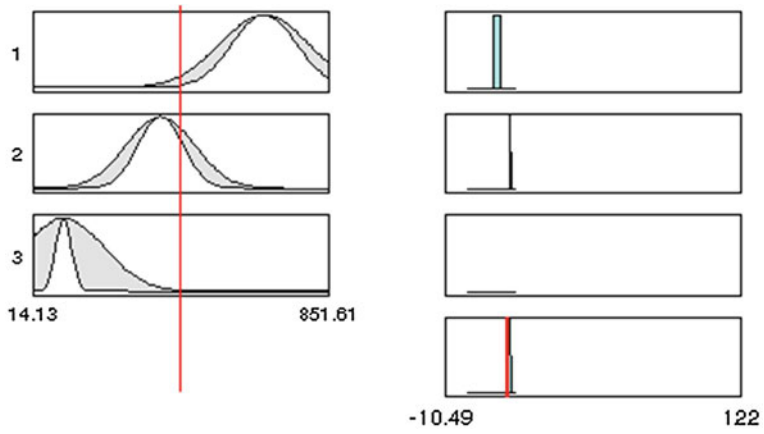

Fig. 4.5 IT2 FIS for solving the thermex behavior dataset

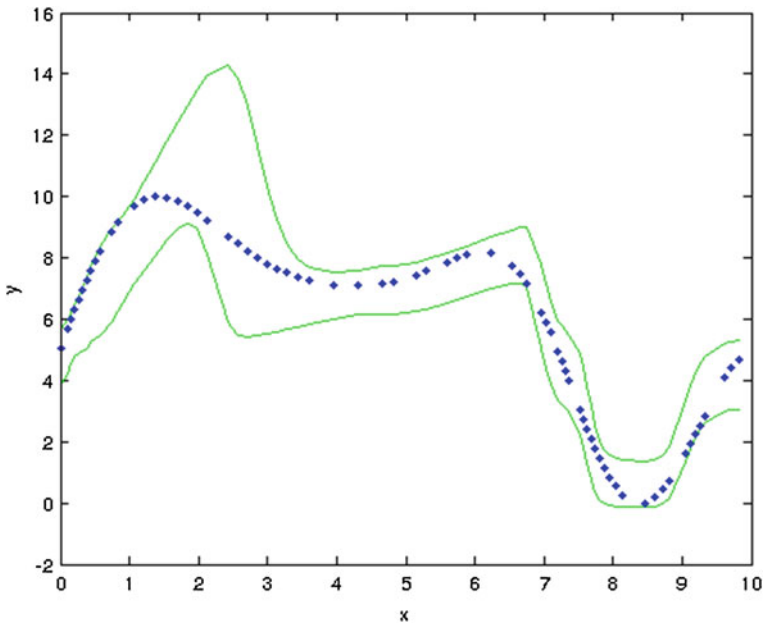

Fig. 4.6 Output coverage for the complex curve dataset

consequents. Where an important point is the heterogeneously sized FOUs of each antecedents FS, each adapted to the available data which formed them. And considering such adaptations in FOU size, the end result is acceptable, such that 100 % coverage was achieved by the output FOU.

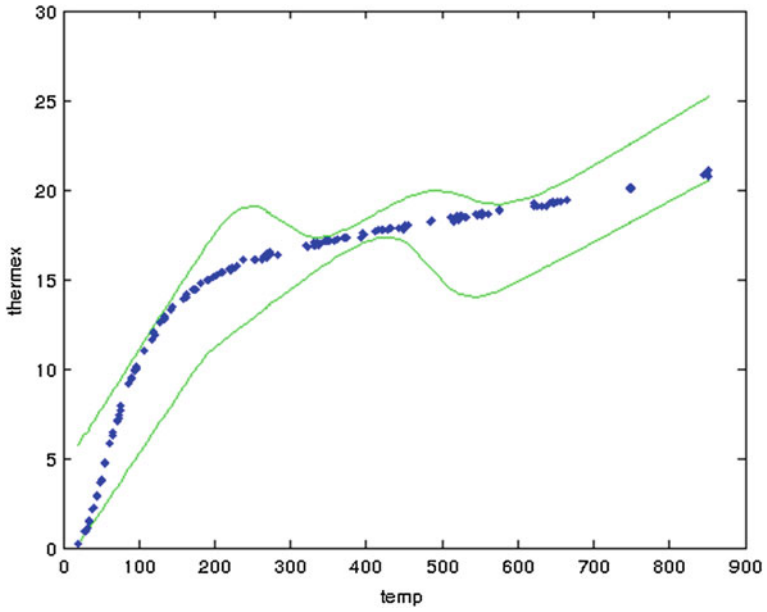

Fig. 4.7 Output coverage for the thermex behavior dataset

4.3 Application. General Type-2 Fuzzy Controller

An application of information granulation implementing FSs was made to demonstrate the differences between noise resilience of T1 FSs, IT2 FSs and GT2 FSs in a mobile robot controller. This experiment was presented by [24]. As Higher-type information granules (IT2 FSs and GT2 FSs) intrinsically handle uncertainty within their model, it is expected that both would perform better when external perturbations are added, and that T1 FSs would not be able to properly handle such external perturbations as well as its counterparts.

The models used for this experiment is that of a unicycle mobile robot [25] which has two driving wheels located on the same rear axis, and a front free moving wheel. Figure 4.8 shows a graphical description of the robot model used.

The dynamics of the mobile robot model assumes that the free moving wheel can be ignored, such that Eqs. (4.1) and (4.2) model the plant used in the simulation. Where, $q = (x, y, \theta)^T$ is the vector of the configuration coordinates, $\upsilon = (v, w)^T$ is the vector of velocities, $\tau = (\tau_1, \tau_2)$ is the vector of torques applied to the wheels of the robot where τ_1 and τ_2 denote the torques of the right and left wheel, (x, y) is the position in the X–Y (world) reference frame, θ is the angle between the heading direction and the x-axis, v and w are the linear and angular velocities.

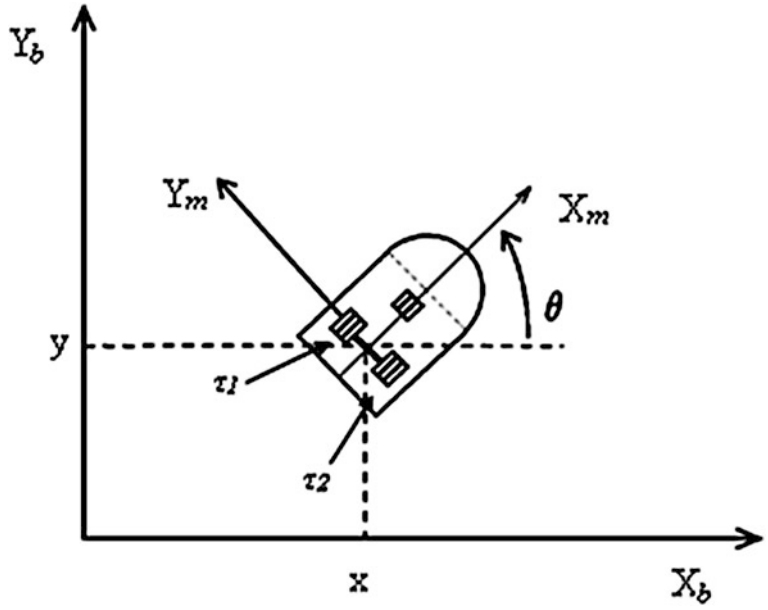

Fig. 4.8 Mobile robot model

$$M(q)\dot{v} + C(q,\dot{q})v + Dv = \tau + P(t) \tag{4.1}$$

$$\dot{q} = \underbrace{\begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix}}_{J(q)} \underbrace{\begin{bmatrix} v \\ w \end{bmatrix}}_{\upsilon} \tag{4.2}$$

A non-holonomic constraint exists in this system, corresponding to a no-slip wheel condition preventing the robot from moving sideways, as shown in Eq. (4.3).

$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0 \tag{4.3}$$

The Fuzzy Controller (FC) was modeled after [26]. Where linear (ϑ_d) and angular (w_d) velocity errors were used for input variables, and right (τ_1) and left (τ_2) torques for outputs. Chosen membership functions used for this FC are Trapezoidal MFs for Negative (N) and Positive (N) linguistic terms, and Triangular MFs for Zero (Z) linguistic term. The rule base can be seen in Table 4.5.

Table 4.5 Rule base used by the FC

ev (Input 1)	*ew* (Input 2)	Output 1	Output 2
N	N	N	N
N	Z	N	Z
N	P	N	P
Z	N	Z	N
Z	Z	Z	Z
Z	P	Z	P
P	N	P	N
P	Z	P	Z
P	P	P	P

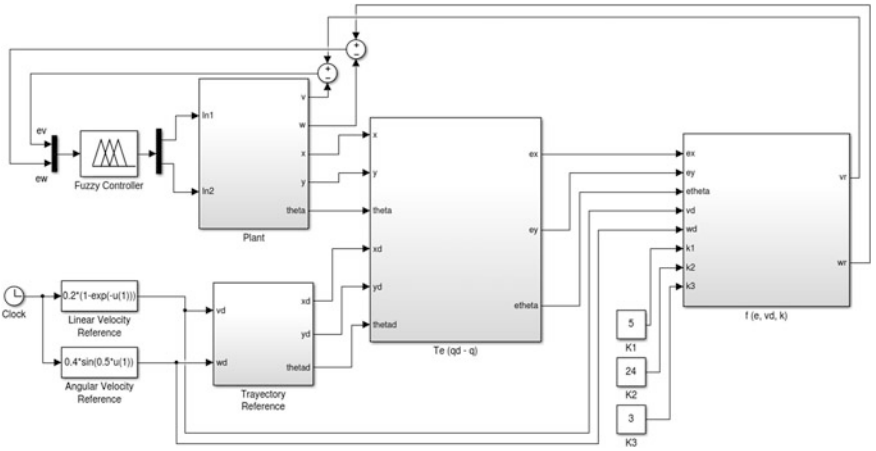

Fig. 4.9 Complete FC of the mobile robot

Finally, Fig. 4.9 shows the complete system used for simulation the robot controller.

Results for the simulation were performed using various external perturbations: band-limited white noise, pulse generated noise, and uniform random number noise. Configuration for the different types of external perturbation are as follows: for band-limited white noise, the height of the Power Spectral Density was equal to 0.1; for pulse generated noise, the amplitude was 1, the period was set to 1, the pulse width (%) was set to 1, and the phase delay was set to 0; and for the uniform random number noise, the limits where set to [−1, 1].

The test criterions chosen to compare the performance of T1, IT2 and GT2 FSs are ISE, IAE, ITSE, and ITAE, shown in Eqs. (4.4)–(4.7) respectively.

$$ISE = \int_0^\infty e^2(t)dt \tag{4.4}$$

$$IAE = \int_0^\infty |e(t)|dt \tag{4.5}$$

$$ITSE = \int_0^\infty e^2(t)tdt \tag{4.6}$$

$$ITAE = \int_0^\infty |e(t)|tdt \tag{4.7}$$

All simulation results are concentrated in Tables 4.6, 4.7 and 4.8, where the three different types of external perturbations are inserted into the system. Additional behavioral results are shown in Appendix B.5.

Table 4.6 Performance index results when band-limited white noise is inserted into the system as an external perturbation

Performance index	v			w		
	T1FC	IT2FC	GT2FC	T1FC	IT2FC	GT2FC
ITAE	135.20	120.30	116.80	137.10	119.50	114.10
ITSE	143.70	110.10	103.70	153.20	110.90	101.40
IAE	13.71	12.18	11.80	13.96	12.20	11.70
ISE	14.97	11.33	10.61	15.98	11.60	10.63

Table 4.7 Performance index results when pulse generated noise is inserted into the system as an external perturbation

Performance index	v			w		
	T1FC	IT2FC	GT2FC	T1FC	IT2FC	GT2FC
ITAE	27.37	26.37	24.22	18.05	15.67	13.20
ITSE	9.44	7.07	5.27	7.14	3.70	2.26
IAE	2.90	2.95	2.74	2.17	2.15	1.90
ISE	1.12	1.11	0.91	0.99	0.94	0.79

Table 4.8 Performance index results when uniform random number noise is inserted into the system as an external perturbation

Performance index	v			w		
	T1FC	IT2FC	GT2FC	T1FC	IT2FC	GT2FC
ITAE	80.09	64.65	59.39	81.90	65.16	59.85
ITSE	51.69	33.00	28.44	51.40	31.42	26.74
IAE	8.52	7.10	6.56	8.67	7.17	6.67
ISE	5.80	4.22	3.69	5.83	4.10	3.63

In all accounts, the logical expected results is reached, where the GT2 FC has the best performance, followed by the IT2 FC, and finally the lowest performing FC is the T1 FC. These results are expected due to how uncertainty is not integrated into the model of the T1 FC, whereas the IT2 FC has some degree of uncertainty representation, and the GT2 FC has even more handle on uncertainty.

References

1. Frank, A., Asuncion, A.: {UCI} machine learning repository. In: University of California Irvine School of Information, vol. 2008, no. 14/8. University of California, School of Information and Computer Sciences, Irvine (2010)
2. U.S. The MathWorks, Inc., Natick, Massachusetts, "MATLAB Release 2013b." (2013)
3. Meyer, M., Vlachos, P.: {StatLib} data archive (1989)
4. Stehman, S.V.: Selecting and interpreting measures of thematic classification accuracy. Remote Sens. Environ. **62**(1), 77–89 (1997)
5. Qian, Q., Chen, S., Cai, W.: Simultaneous clustering and classification over cluster structure representation. Pattern Recognit. **45**(6), 2227–2236 (2012)
6. Taşdemir, K.: Vector quantization based approximate spectral clustering of large datasets. Pattern Recognit. **45**(8), 3034–3044 (2012)
7. Meo, R., Bachar, D., Ienco, D.: LODE: a distance-based classifier built on ensembles of positive and negative observations. Pattern Recognit. **45**(4), 1409–1425 (2012)
8. Vu, V.-V., Labroche, N., Bouchon-Meunier, B.: Improving constrained clustering with active query selection. Pattern Recognit. **45**(4), 1749–1758 (2012)
9. Yu, Z., Wong, H.-S., You, J., Yu, G., Han, G.: Hybrid cluster ensemble framework based on the random combination of data transformation operators. Pattern Recognit. **45**(5), 1826–1837 (2012)
10. Hatamlou, A.: Black hole: a new heuristic optimization approach for data clustering. Inf. Sci. (Ny) **222**, 175–184 (2013)
11. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: Twenty-First International Conference on Machine learning—ICML '04, p. 11 (2004)
12. Cai, W., Chen, S., Zhang, D.: A simultaneous learning framework for clustering and classification. Pattern Recognit. **42**(7), 1248–1259 (2009)
13. David, G., Averbuch, A.: SpectralCAT: categorical spectral clustering of numerical and nominal data. Pattern Recognit. **45**(1), 416–433 (2012)
14. Ding, C., Li, T.: Adaptive dimension reduction using discriminant analysis and k-means clustering. In: International Conference on Machine Learning, pp. 521–528 (2007)
15. Athitsos, V., Sclaroff, S.: Boosting nearest neighbor classifiers for multiclass recognition. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition—Work (2005)
16. Ludwig, S.A.: Prediction of breast cancer biopsy outcomes using a distributed genetic programming approach. In: Proceedings of the 1st ACM International Health Informatics Symposium, pp. 694–699 (2010)
17. Dash, D., Cooper, G.F.: Model averaging for prediction with discrete Bayesian networks. J. Mach. Learn. Res. **5**, 1177–1203 (2004)
18. Zhang, Y.Z.Y., Street, W.N.: Bagging with adaptive costs. IEEE Trans. Knowl. Data Eng. **20** (5) (2008)
19. Charytanowicz, M., Niewczas, J.: Complete gradient clustering algorithm for features analysis of X-ray images. Inf. Technol. Biomed. **69**, 15–24 (2010)

20. Gao, D., Jun, D., Changming, Z.: Integrated Fisher linear discriminants: an empirical study. Pattern Recognit. **47**(2), 789–805 (2013)
21. Thi, L., An, H., Hoai, L., Dinh, P.: New and efficient DCA based algorithms for minimum sum-of-squares clustering. Pattern Recognit. **47**, 388–401 (2014)
22. Wang, Z., Shao, Y.-H., Wu, T.-R.: A GA-based model selection for smooth twin parametric-margin support vector machine. Pattern Recognit. **46**(8), 2267–2277 (2013)
23. Daneshgar, A., Javadi, R., Razavi, B.S.: Clustering using isoperimetric number of trees. Pattern Recognit. **46**(12), 3371–3382 (2012)
24. Sanchez, M.A., Castillo, O., Castro, J.R.: Generalized Type-2 Fuzzy Systems for controlling a mobile robot and a performance comparison with Interval Type-2 and Type-1 Fuzzy Systems. Expert Syst. Appl. **42**(14), 5904–5914 (2015)
25. Fukao, T., Nakagawa, H., Adachi, N.: Adaptive tracking control of a nonholonomic mobile robot. IEEE Trans. Robot. Autom. **16**(5), 609–615 (2000)
26. Martínez-Soto, R., Castillo, O., Castro, J.R.: Genetic algorithm optimization for type-2 non-singleton fuzzy logic controllers. Recent Adv. Hybrid Approaches Des. Intell. Syst. **547**, 3–18 (2014)

Chapter 5
Conclusions

Multiple approaches were suggested for the formation of Higher-type information granules with Type-2 Fuzzy Set representation. Among these approaches, one implementation made direct use of the principle of justifiable granularity in order calculate the individual coverage for all information granules, obtaining very good performance, and another approach suggested a method for using the concept of the principle of justifiable granularity in order to create GT2 FSs. Other approaches were also proposed which measured and captured uncertainty for the formation of Higher-type information granules, one such approach used information theory to overlap two sample information granules in order to directly obtain the uncertainty between both granules, the other approach used calculated the amount of data dispersion and used it as a means to provide a FOU to IT2 FSs.

In general, many approaches were tried in this book which form Higher-type information granules, all varying in their performance, but having similar results in general terms. Showing that there are many possible approaches for creating Higher-type information granules.

A clustering algorithm with granular computing concepts was created which used gravitational forces as the premise for finding relations between all data. With the premise that similar information must belong to the same information granules, gravitational forces were used to group together closely related data and find such relevant information granules. When experimentation was performed, the results were very promising as obtained performance was very comparable with recent state of the art clustering algorithm, showing that the concept of granular computing, when applied, does obtain great performance.

An algorithm was created which used the principle of justifiable granularity in order to calculate the best possible information granule size based on the evidence which initially formed such granules. When experiments were performed, results showed good performance, once again showing that the integration of granular computing concepts can give benefits to existing algorithms.

Multiple approaches were tested where Higher-type information granules were created. Due to the nature of uncertainty, these approaches used concepts from

M.A. Sanchez et al., *Type-2 Fuzzy Granular Models*,
SpringerBriefs in Computational Intelligence, DOI 10.1007/978-3-319-41288-7_5

information theory and statistics in order to measure uncertainty, either directly or indirectly, and use these uncertainty measures in the formation of Higher-type information granules, where they were represented by Type-2 FSs.

Considering FSs were used as representation for information granules, FLSs had to be used in order to process the granular models. Existing two types of FLS types, Mamdani and TSK, ultimately, TSK was chosen as they typically perform better when modeling from data. Yet they had to be optimized from the data itself in order perform well. Cuckoo Search optimization algorithm was chosen for all these optimizations as it is simple to use and yet it obtains very acceptable results, although any other optimization algorithm could be used, such that there is no marriage between the proposed approaches and the Cuckoo Search optimization algorithm. When dealing with the optimization of IT2 FLSs, the output of these systems is an interval which should provide enough coverage for all output reference data, such that the objective function had to be multi objective, where he coverage was to be assured and at the same time the coverage was not to overextend as to over-generalize the output.

Appendix A

$$\rho(x) \sim N(0,\sigma) \equiv \frac{1}{\sqrt{2\pi}} \mathrm{e}^{\left(-\frac{1}{2}x^2\right)}$$

$$V(b) = \mathrm{e}^{(-\alpha|b-u|)} \int_u^b \rho(x)\,\mathrm{d}x; \quad \alpha \in [0, \alpha_{max}]$$

$$V(b^*) = max_{b\,>\,med(D)}[V(b)]$$

$$V(a^*) = max_{a<med(D)}[V(a)]$$

$$\rho(x) \sim N(u,\sigma) \equiv \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{\left(-\frac{1}{2}\left(\frac{x-u}{\sigma}\right)^2\right)}$$

$$erf \equiv \frac{2}{\sqrt{\pi}} \int_0^x \mathrm{e}^{-t^2}\,\mathrm{d}t$$

$$\rho(x) \sim N(0,\sigma) \equiv \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{\left(-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2\right)}$$

$$z_b = \frac{\sqrt{2}}{2}\left(\frac{b-u}{\sigma}\right)$$

$$\int_\mu^b \rho(x)\,\mathrm{d}x = \frac{1}{2} erf(z_b)$$

$$z_a = \frac{\sqrt{2}}{2}\left(\frac{a-u}{\sigma}\right)$$

M.A. Sanchez et al., *Type-2 Fuzzy Granular Models*,
SpringerBriefs in Computational Intelligence, DOI 10.1007/978-3-319-41288-7

$$\int_{a}^{\mu} \rho(x)\,\mathrm{d}x = -\frac{1}{2} erf(z_a)$$

$$v(b) = \frac{1}{2}\mathrm{e}^{(-\alpha b)} erf(z_b)$$

$$v(a) = -\frac{1}{2}\mathrm{e}^{(-\alpha a)} erf(z_a)$$

$$erf(x) = \frac{2}{\sqrt{\pi}} \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)k!}$$

$$\frac{d}{dx}[erf(x)] = \frac{2}{\sqrt{\pi}}\mathrm{e}^{-x^2}$$

$$z_k = \frac{x_k - \bar{x}}{\sigma}$$

$$\rho(z) - N(\bar{x}, \sigma) \equiv \frac{1}{\sigma\sqrt{2\pi}}\mathrm{e}^{\left(-\frac{1}{2}z_k^2\right)}$$

Appendix B

Appendix B.1

See Figs. B.1.1, B.1.2, B.1.3 and B.1.4.

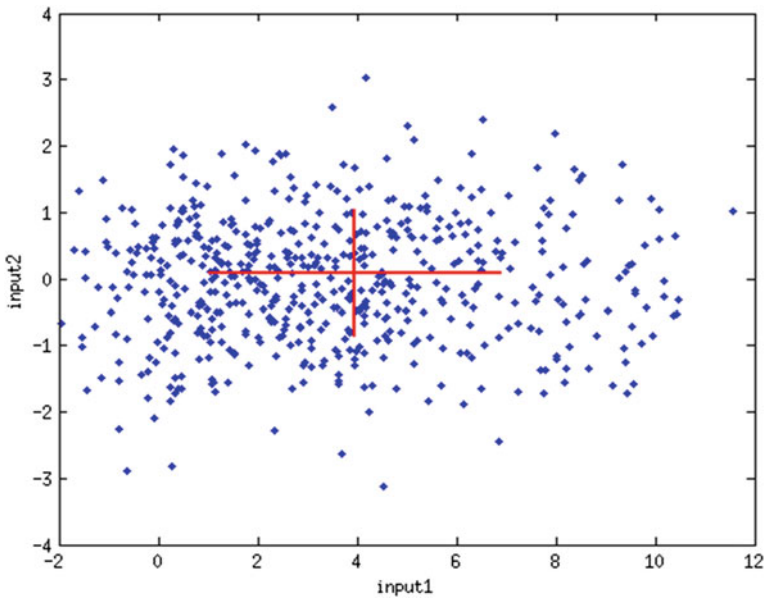

Figure B.1.1 Synthetic cloud of data with point concentration bias towards the left side, used to demonstrate the found representative center as well as the length of the σ for the Gaussian membership functions

M.A. Sanchez et al., *Type-2 Fuzzy Granular Models*,
SpringerBriefs in Computational Intelligence, DOI 10.1007/978-3-319-41288-7

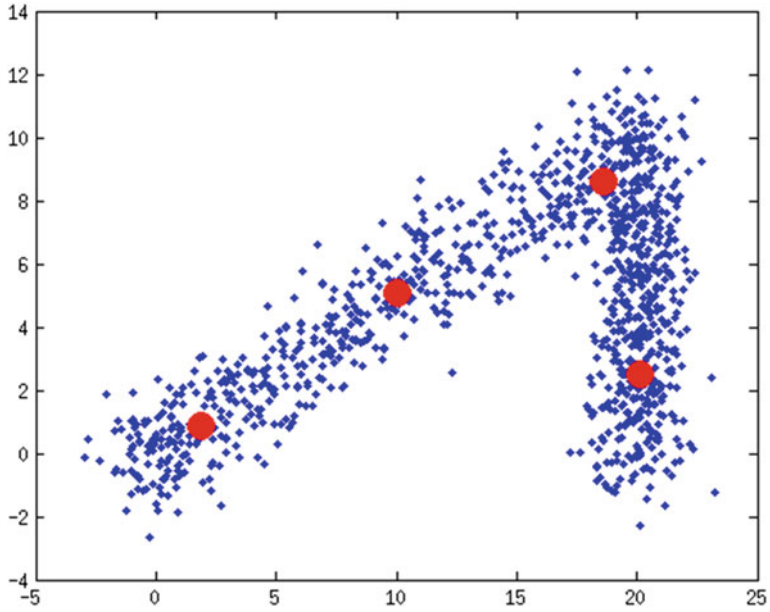

Figure B.1.2 Synthetic patterned dataset which shows the behavior of the clustering section of the FGGCA of how it distributes the representative clusters between the whole dataset

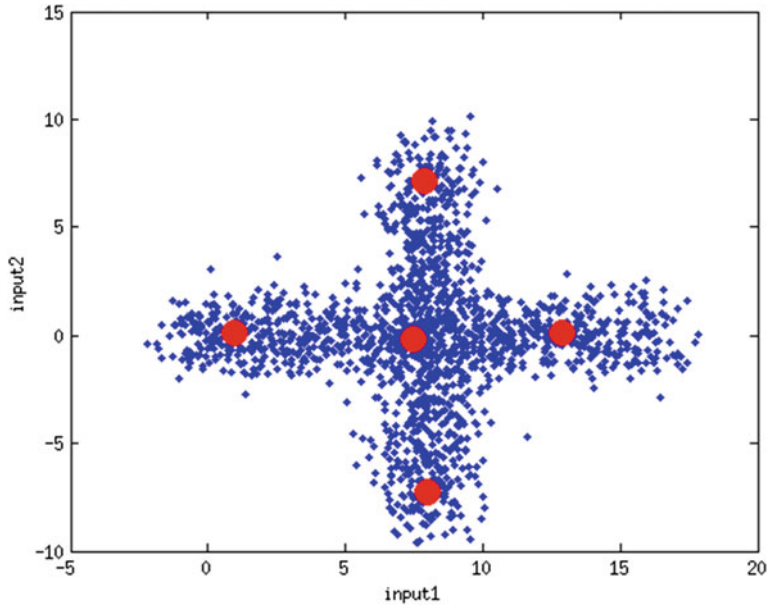

Figure B.1.3 Synthetic dataset with a cross pattern, where the solid discs show the location of 5 cluster centers

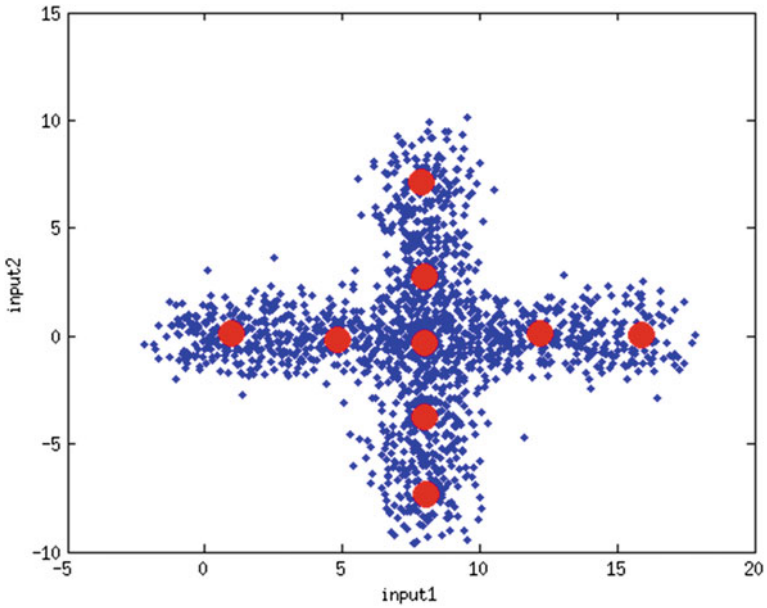

Figure B.1.4 Synthetic dataset with a cross pattern, where the solid discs show the location of 9 cluster centers

Appendix B.2

The following figures (Figs. B.2.1, B.2.2, B.2.3 and B.2.4) are from the iris dataset as seen from the primary function, where the functions can be seen as incomplete, yet they were built from existing numerical evidence, where there is no data, there was no evidence in the set for that specific membership function.

The following figures (Figs. B.2.5, B.2.6, B.2.7 and B.2.8) are the same membership function from the iris dataset, but as seen from an orthogonal point of view to better appreciate the formed General Type-2 Gaussian membership functions.

The respective membership functions for the iris dataset formed with a dependency on u will be shown in the following figures (Figs. B.2.9, B.2.10, B.2.11 and B.2.12) from the primary membership function's point of view.

The following figures (Figs. B.2.13, B.2.14, B.2.15 and B.2.16) show the same membership functions from the iris dataset, dependent on u, but from an orthogonal point of view.

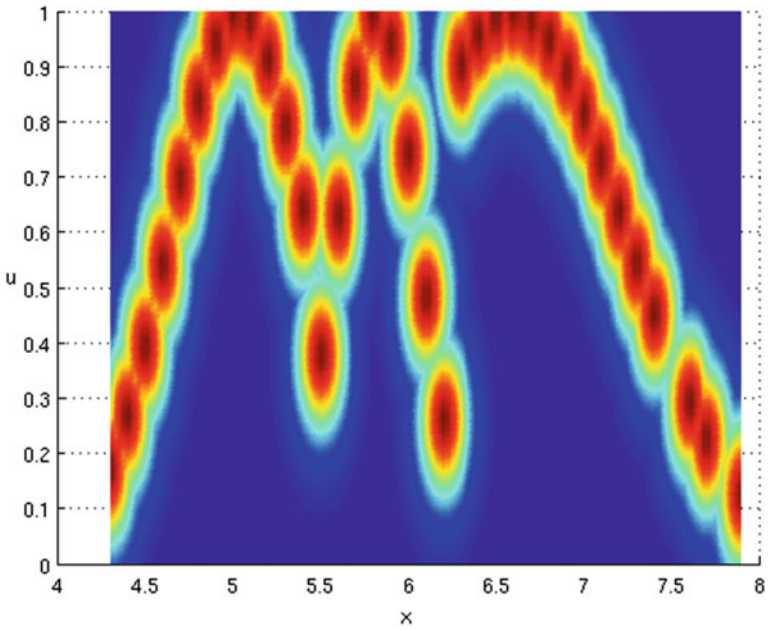

Figure B.2.1 General Type-2 Gaussian membership function built from the iris dataset as seen from the primary function's point of view. Shown here is the sepal length

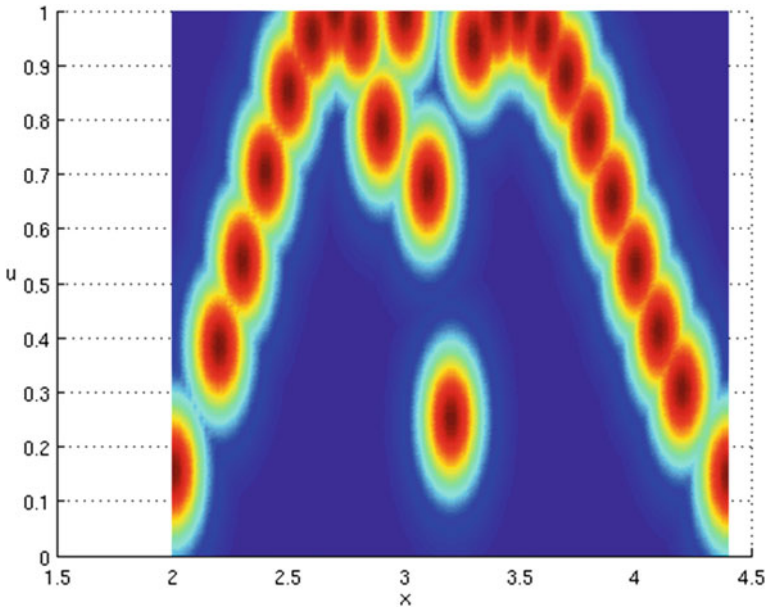

Figure B.2.2 General Type-2 Gaussian membership function built from the iris dataset as seen from the primary function's point of view. Shown here is the sepal width

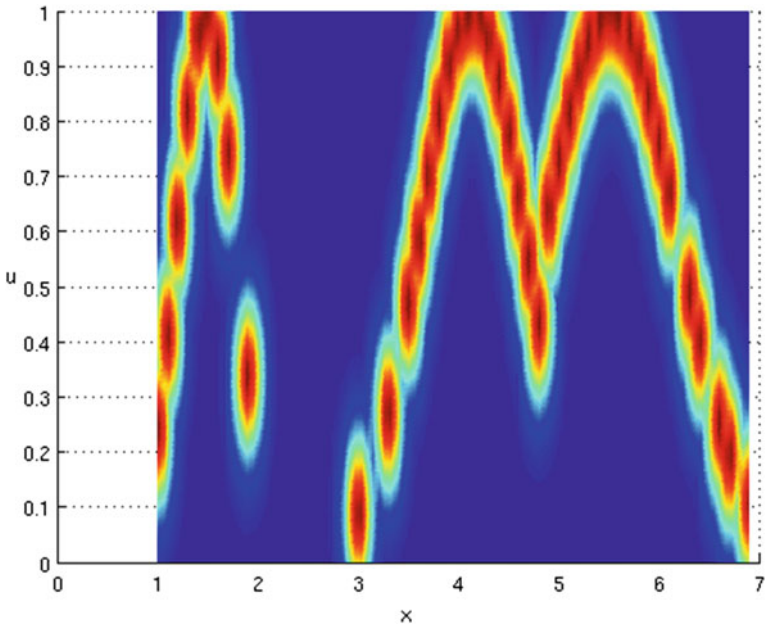

Figure B.2.3 General Type-2 Gaussian membership function built from the iris dataset as seen from the primary function's point of view. Shown here is the petal length

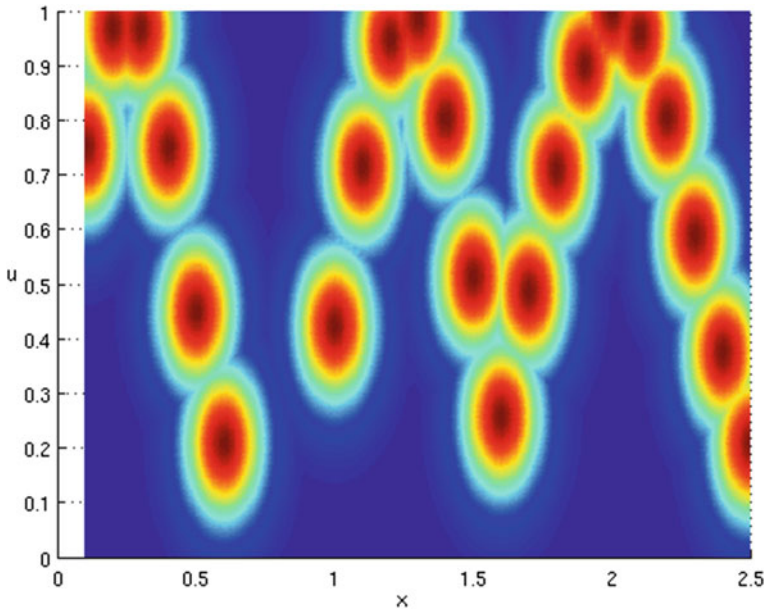

Figure B.2.4 General Type-2 Gaussian membership function built from the iris dataset as seen from the primary function's point of view. Shown here is the petal width

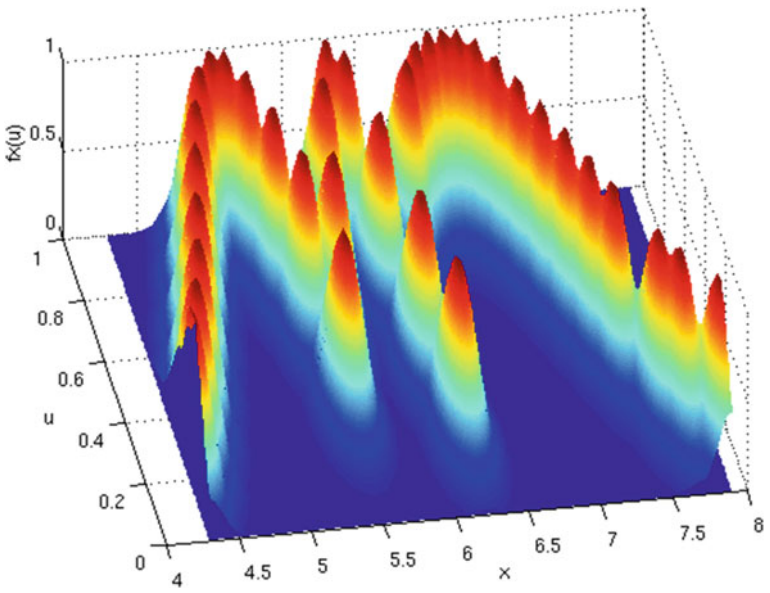

Figure B.2.5 General Type-2 Gaussian membership function built from the iris dataset as seen from an orthogonal point of view. Shown here is the sepal length

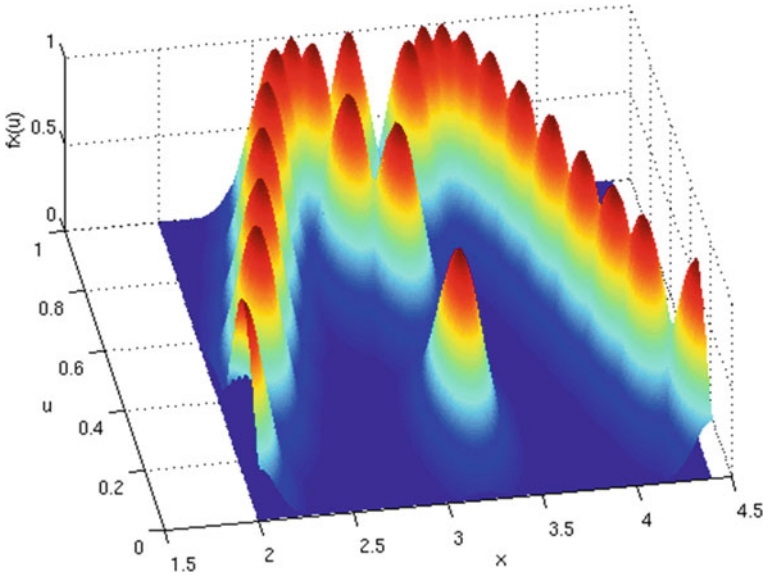

Figure B.2.6 General Type-2 Gaussian membership function built from the iris dataset as seen from an orthogonal point of view. Shown here is the sepal width

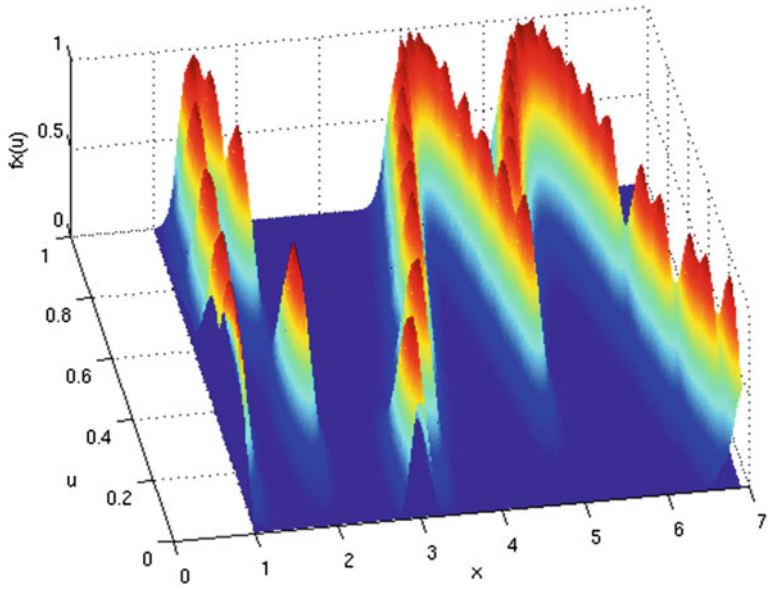

Figure B.2.7 General Type-2 Gaussian membership function built from the iris dataset as seen from an orthogonal point of view. Shown here is the petal length

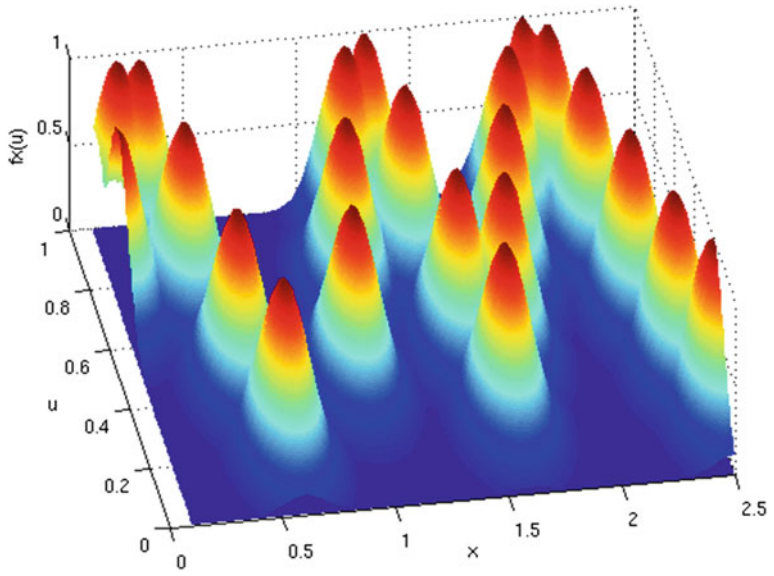

Figure B.2.8 General Type-2 Gaussian membership function built from the iris dataset as seen from an orthogonal point of view. Shown here is the petal width

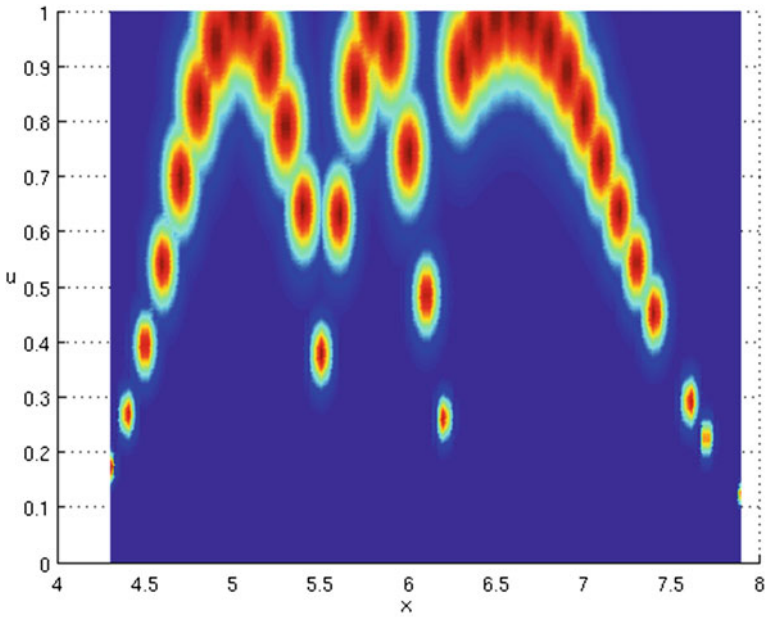

Figure B.2.9 General Type-2 Gaussian membership function built from the iris dataset as seen from the primary function's point of view. Shown here is the sepal length

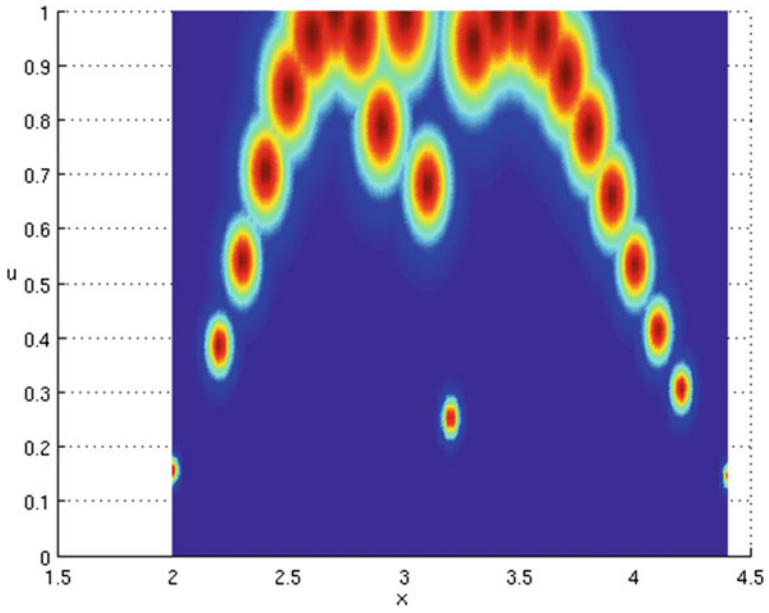

Figure B.2.10 General Type-2 Gaussian membership function built from the iris dataset as seen from the primary function's point of view. Shown here is the sepal width

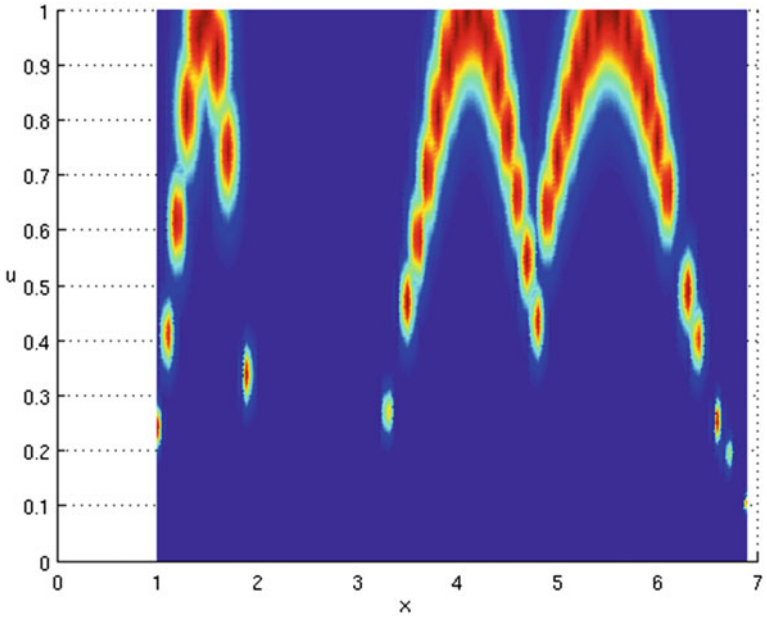

Figure B.2.11 General Type-2 Gaussian membership function built from the iris dataset as seen from the primary function's point of view. Shown here is the petal length

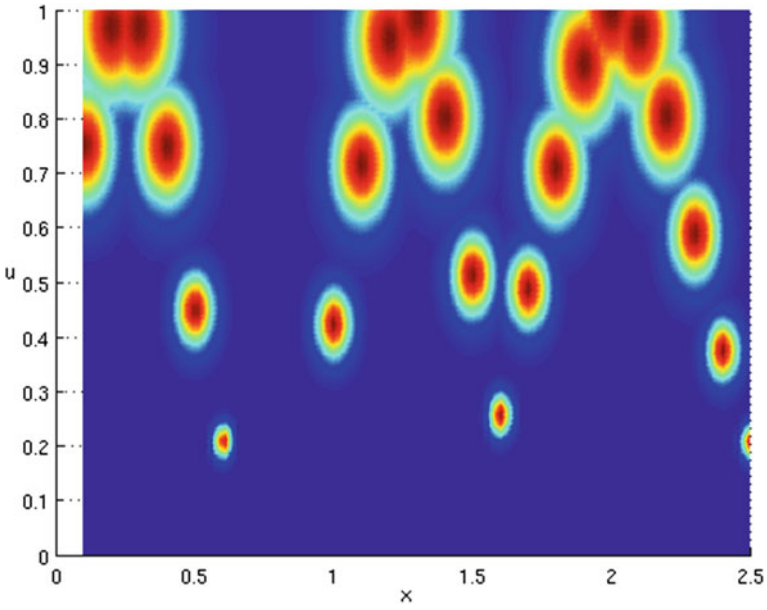

Figure B.2.12 General Type-2 Gaussian membership function built from the iris dataset as seen from the primary function's point of view. Shown here is the petal width

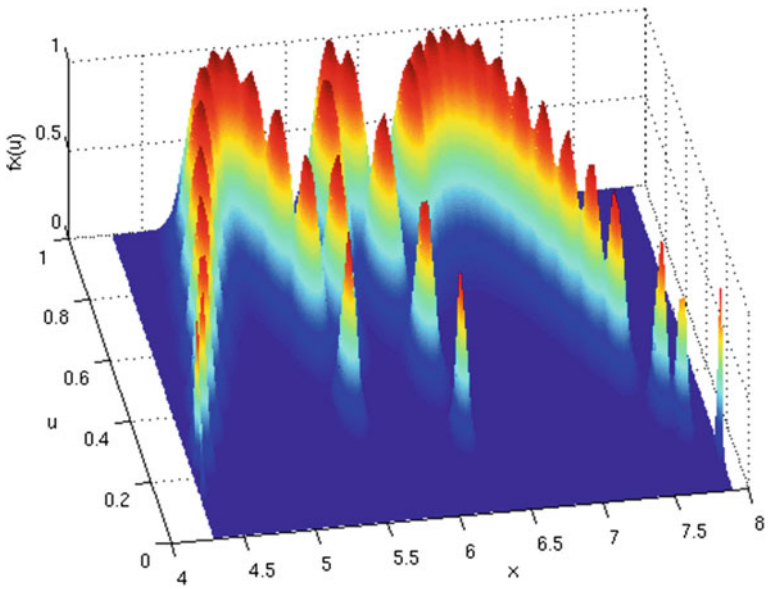

Figure B.2.13 General Type-2 Gaussian membership function built from the iris dataset as seen from the primary function's point of view. Shown here is the sepal length

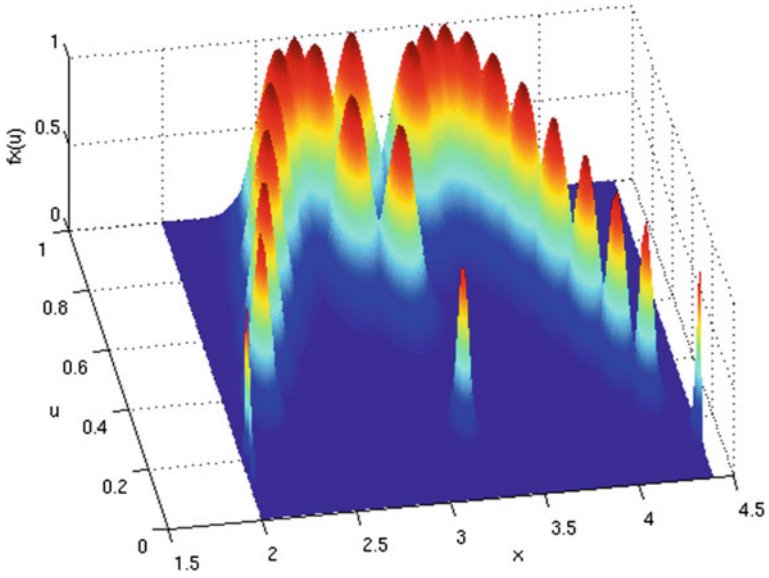

Figure B.2.14 General Type-2 Gaussian membership function built from the iris dataset as seen from the primary function's point of view. Shown here is the sepal width

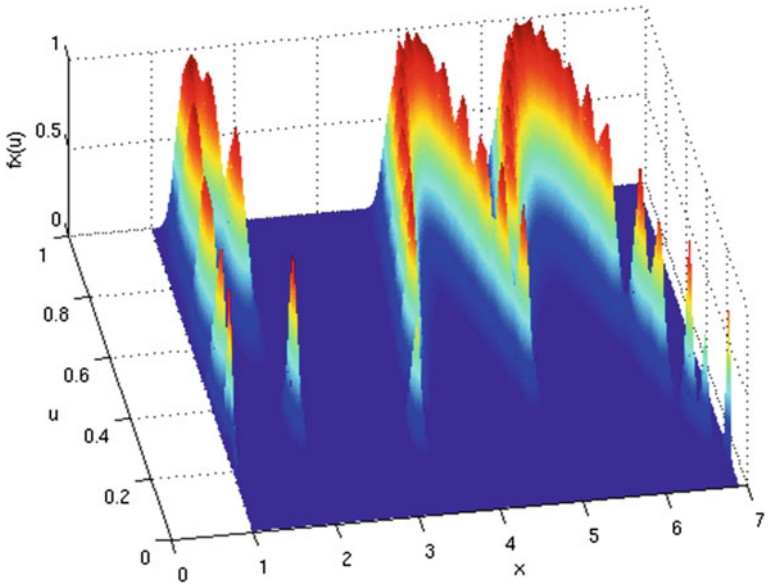

Figure B.2.15 General Type-2 Gaussian membership function built from the iris dataset as seen from the primary function's point of view. Shown here is the petal length

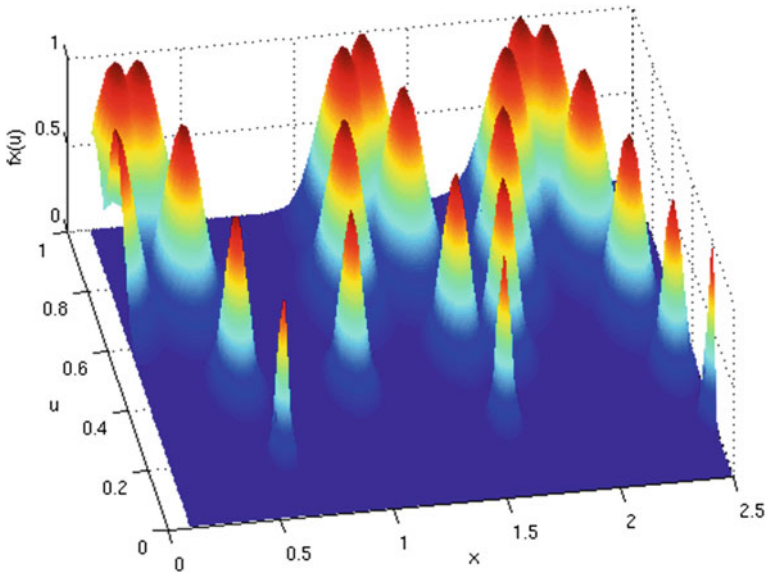

Figure B.2.16 General Type-2 Gaussian membership function built from the iris dataset as seen from the primary function's point of view. Shown here is the petal width

Appendix B.3

The following figures (Figs. B.3.1, B.3.2 and B.3.3) show the behavior of the proposed approach when dealing with different degrees of noise, in this case, the benchmark Iris dataset was used.

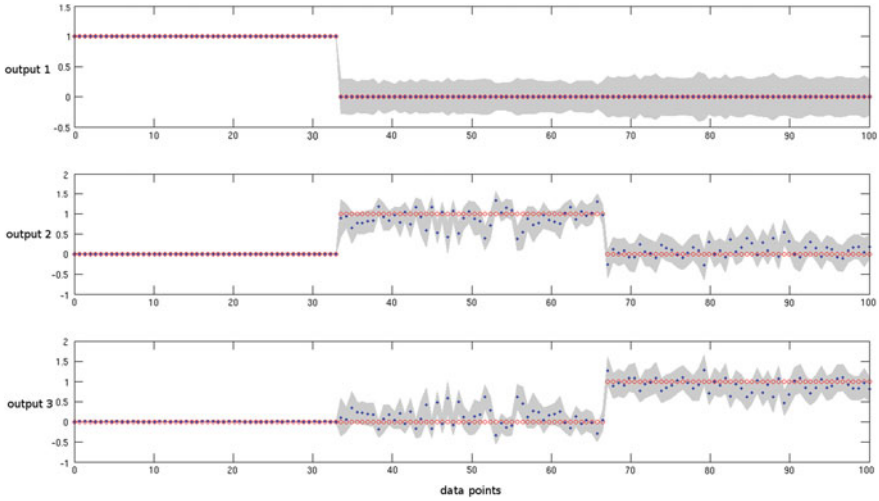

Figure B.3.1 The final result of the formed IT2 FIS alongside the FOU, when there is no noise

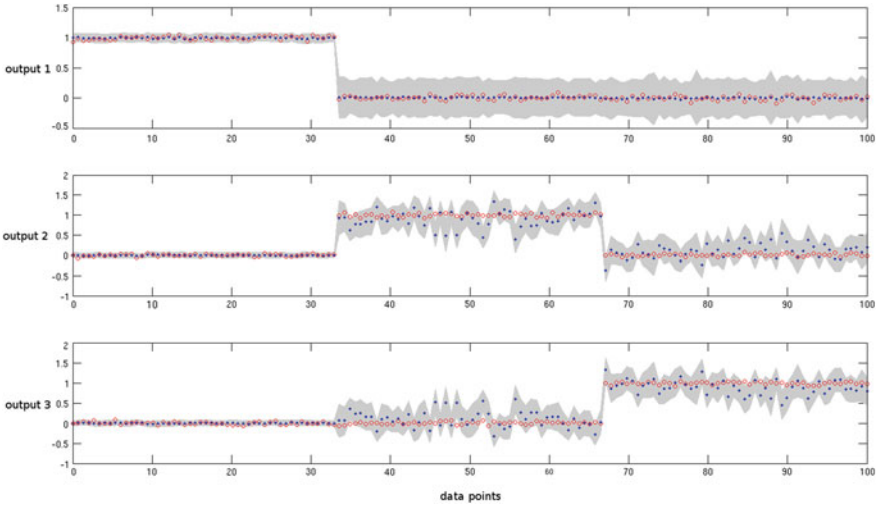

Figure B.3.2 The final result of the formed IT2 FIS alongside the FOU, when there is 30 dBi noise

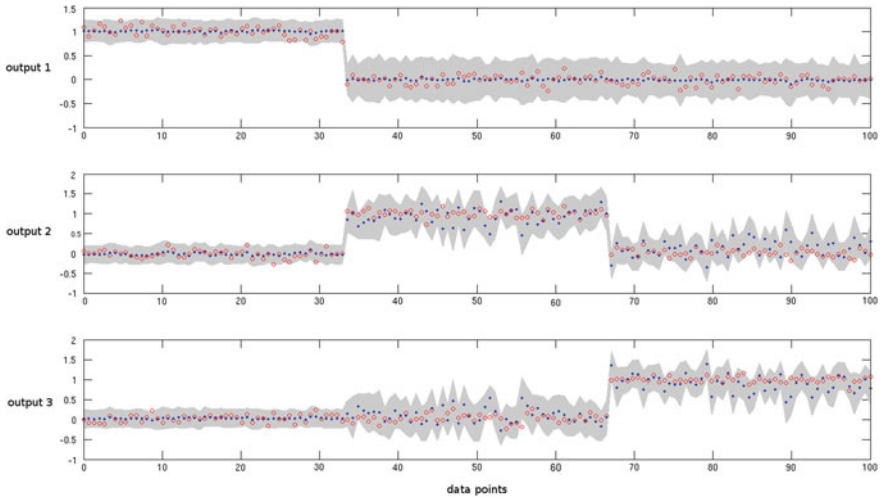

Figure B.3.3 The final result of the formed IT2 FIS alongside the FOU, when there is 20 dBi noise

Appendix B.4

Figure B.4.1 shows the rule set for an IT2 FLS as formed for the benchmark Engine dataset, and the following figures (Figs. B.4.2 and B.4.3) are the output behavior of the output coverage.

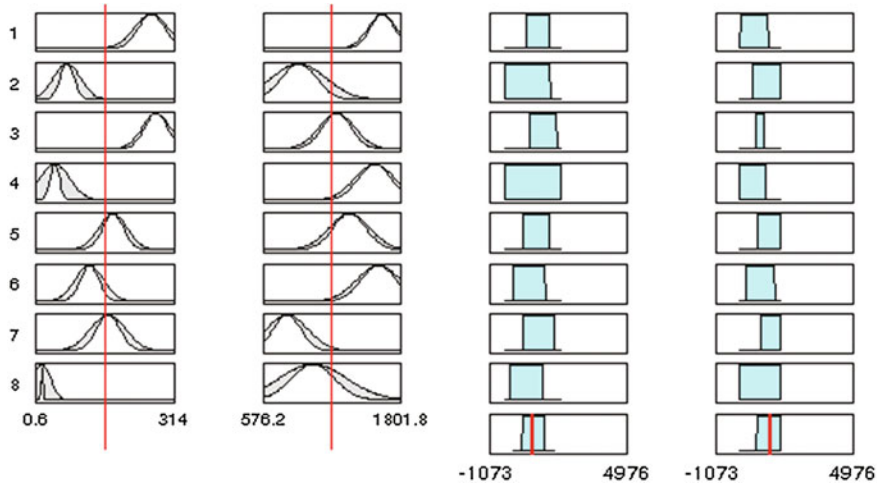

Figure B.4.1 IT2 FIS for solving the engine behavior dataset

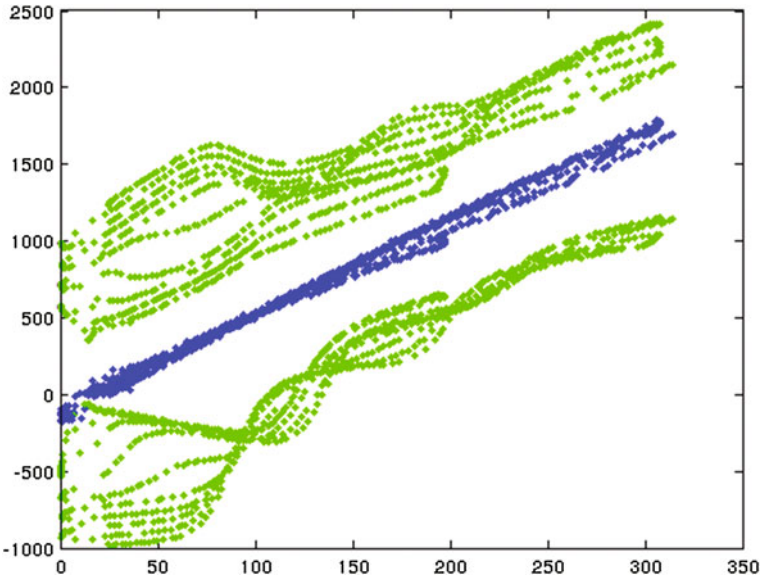

Figure B.4.2 Output coverage for the engine dataset. For the first output of the FLS

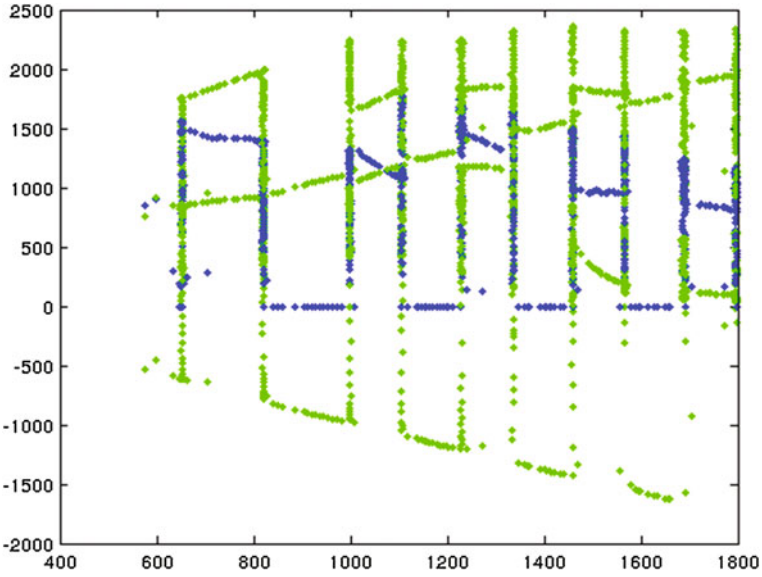

Figure B.4.3 Output coverage for the engine dataset. For the second output of the FLS

Appendix B.5

As an example of the relation between noise and FC performance, the following figure (Fig. B.5.1) shows the obtained Performance Index values for each FC (T1, IT2, and GT2) when inserting band-limited white noise into the system. Here, the performance of the GT2FC is better with respect to both IT2FC and T1FC.

To illustrate the behavior of the FC with respect to the reference, in a noisy environment, the outputs *v* and *w* are shown in the following figures (Figs. B.5.2,

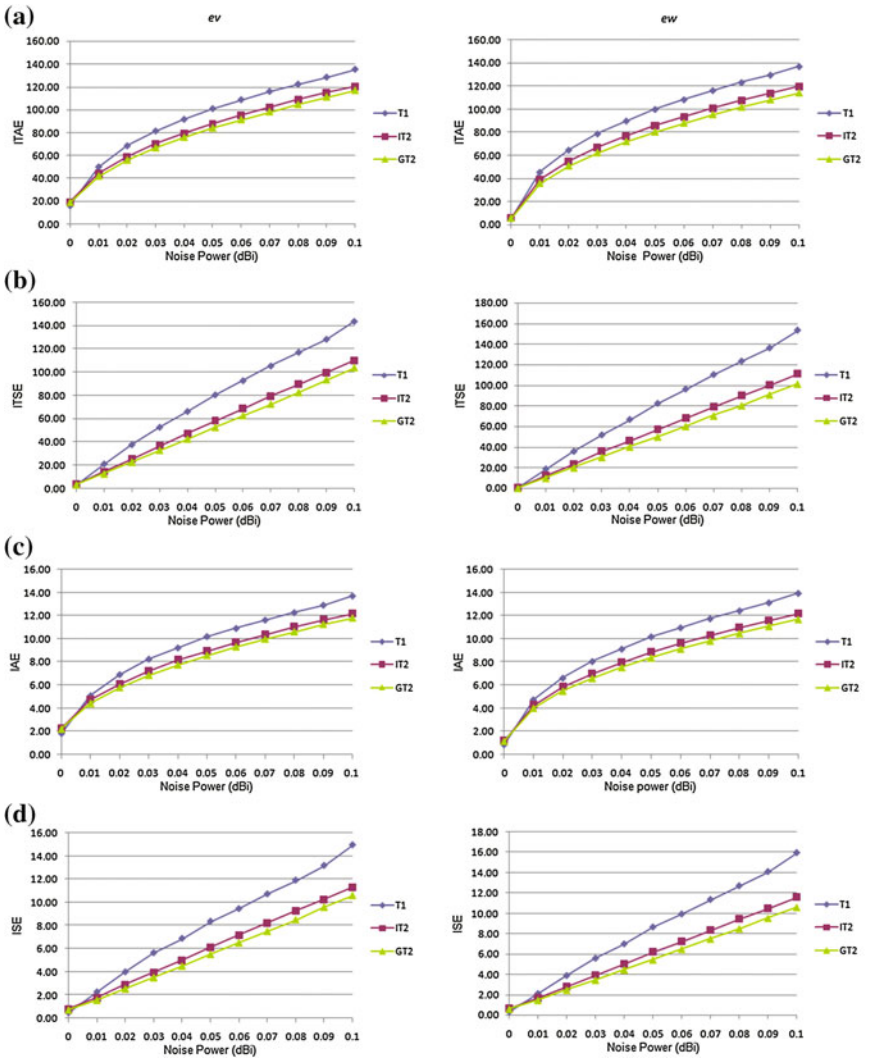

Figure B.5.1 Behavior of various performance indices in relation to the amount of noise perturbations present in the system, when T1FC, IT2FC, and GT2FC are used. **a** ITAE, **b** ITSE, **c** IAE, and **d** ISE. Band-limited white noise is used as external perturbation

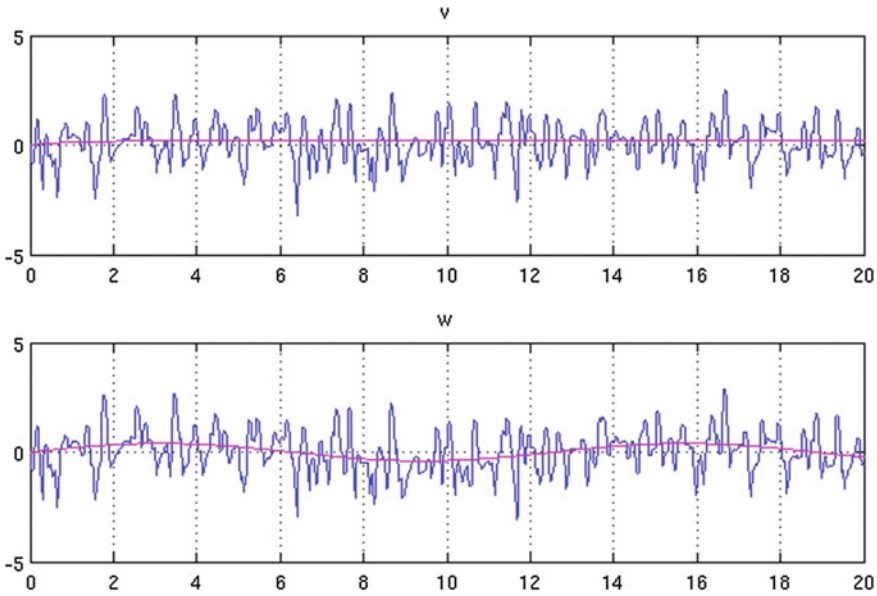

Figure B.5.2 Reference/FC results, for v and w, using a T1FC where the perturbations are added by band-limited white noise

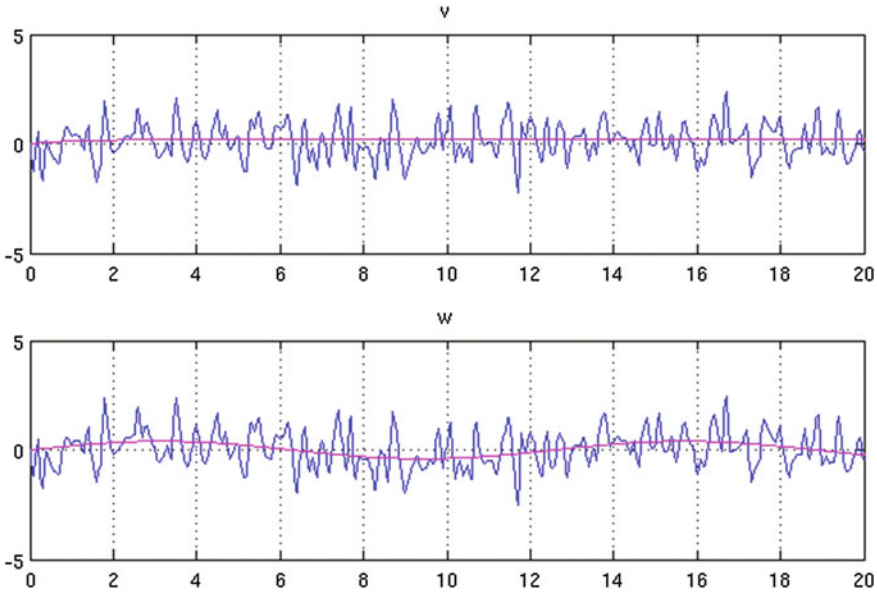

Figure B.5.3 Reference/FC results, for v and w, using an IT2FC where the perturbations are added by band-limited white noise

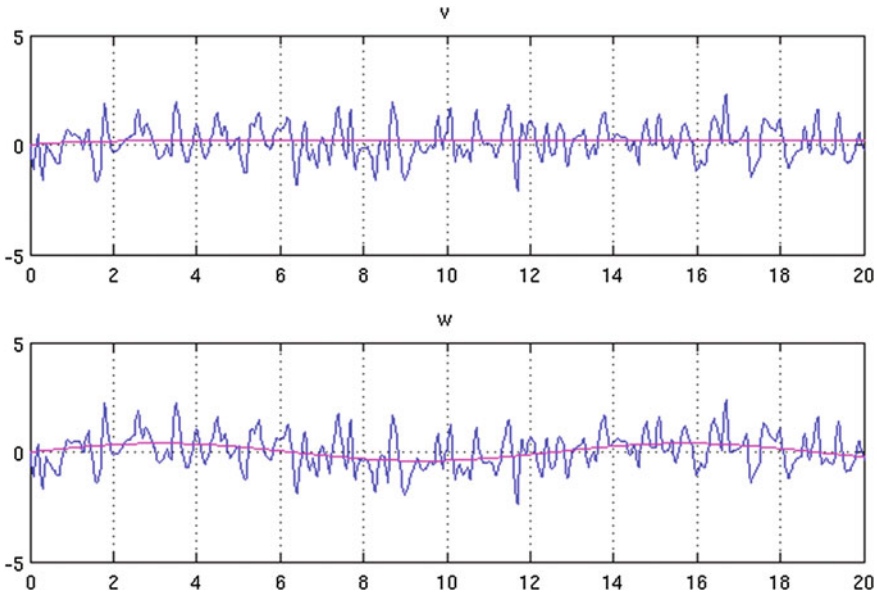

Figure B.5.4 Reference/FC results, for *v* and *w*, using a GT2FC where the perturbations are added by band-limited white noise

B.5.3 and B.5.4), for T1FC, IT2FC, and GT2FC, respectively. Where the selected perturbation is band-limited white noise, and a 20 second space was used to obtain the samples.

Appendix C

Appendix C.1

```
function [aopt,bopt,x] = intervalinfogranule(x,alphaA,alphaB,mu)
   sigma = std(x);
   [x,~] = sort(x);
   xmin = min(x); xmax = max(x);
   % upper bound of the information granule
   b = linspace(mu,xmax,1000);
   f  = exp(-alphaB*abs(b-mu));
   zb = (sqrt(2)/2)*(b-mu)/sigma;
   Ipxb = erf(zb)/2;
   Vb = f.*Ipxb;
   [~,idx] = max(Vb);
   bopt = abs(b(idx)-mu);
   % lower bound of the information granule
   a = linspace(xmin,mu,1000);
   f  = exp(-alphaA*abs(a-mu));
   za = (sqrt(2)/2)*(a-mu)/sigma;
   Ipxa = -erf(za)/2;
   Va = f.*Ipxa;
   [~,idx] = max(Va);
   aopt = abs(a(idx)-mu);
```


M.A. Sanchez et al., *Type-2 Fuzzy Granular Models*,
SpringerBriefs in Computational Intelligence, DOI 10.1007/978-3-319-41288-7

Appendix C.2

```
function [centers,sigmas] = FGGA(inputs,outputs,radius,modifier)
%% Normalize data
x = inputs;
x = [x,outputs];
maxX = max(max(x));
x = x./maxX;
%% Initial values
SIZE = size(x(:,1),1);
numVars = size(x,2);
mass = 100;
G = 6.67e-11;
m = ones(1,SIZE).*mass;
y = x;
x = [x, zeros(size(x(:,1),1),2)];
INTERACTIONS_EXIST = 1;
iters = 0;
%% Main loop
while INTERACTIONS_EXIST == 1
  INTERACTIONS_EXIST = 0;
  iters = iters + 1;
  % Calculate all interacting forces in the system
  F = zeros(SIZE,SIZE);
  dist = zeros(SIZE,SIZE);
  for i=1:1:SIZE
    for j=1:1:SIZE
      if i ~= j
        dSum = 0;
        for k=1:numVars
          dSum = dSum + (x(i,k)-x(j,k))^2;
        end
```

```
            dist(i,j) = sqrt(dSum);
            F(i,j) = (G*m(i)*m(j)) / dSum;
        else
            dist(i,j) = 3;
        end
    end
end
% Reorder particule data
[~,idx] = sort(sum(F),'descend');
% Unite particles
for i=1:1:SIZE
    idxi = idx(i);
    [minD,md]=min(dist(idxi,:));
    op1 = idxi;
    op2 = md;
    if m(op1)~=-1 && m(op2)~=-1 && op1~=op2 && minD<radius
        INTERACTIONS_EXIST = 1;
        if m(op1) >= m(op2)
            a = op1;
            b = op2;
        else
            a = op2;
            b = op1;
        end
        m(a) = m(a) + m(b);
        % % Move paticles according to masses % %
        for k=1:numVars
```

```
            dSum = dSum + (x(a,k)-x(b,k))^2;
        end
        d = sqrt(dSum);
        distp = (m(b)/(m(a)+m(b)))*d;
        t = (1/d)*distp;
        x(a,:) = x(a,:) + t*(x(b,:)-x(a,:));
        % % % % % % % % % % % % % % % % % % % % %
        m(b) = -1;
        x(b,:) = -2;
        SIZE = SIZE - 1;
      end
    end
    % Clean up elements from 'm' and 'x'
    m(m==-1) = [];
    condition=x(:,1)==-2;x(condition,:)=[];
    radius = radius * modifier;
end
%% Calculate radius of influence per cluster point found
lvalue = zeros(size(y,1),SIZE);
for i=1:size(y)
    F    = zeros(SIZE,1);
    dist = zeros(SIZE,1);
    for j=1:SIZE
        if ~isequal(x(j,1:numVars),y(i,1:numVars))
            dSum = 0;
            for n=1:numVars
                dSum = dSum + (x(j,n)-y(i,n))^2;
```

```
            end
            dist(j) = sqrt(dSum);
            F(j) = (G*m(j)*100)/dSum;
        end
    end
    [~,idx] = max(F);
    lvalue(i,idx) = dist(idx);
end
for i=1:SIZE
    temp = lvalue(:,i);
    temp(temp==0) = [];
    x(i,numVars+1) = mean(temp);
end
%% Format output data and Denormalize
rowToDel = [];
centers = x(:,1:size(inputs,2)).*maxX;
sigmas  = x(:,numVars+1).*(maxX);
for i=1:size(sigmas,1)
    if sigmas(i) == 0 || isnan(sigmas(i))
        rowToDel = [rowToDel;i];
    end
end
sigmas(rowToDel,:)  = [];
centers(rowToDel,:) = [];
sigmas = repmat(sigmas,1,size(inputs,2));
```

Appendix C.3

```
function fismat = it2fuzzyfy(Xin,Xout,centers)
y = Xin;
y = [y,Xout];
SIZE = size(centers,1);
alpha = 0; % generalized information granules
lSigmas = zeros(SIZE,size(centers,2));
rSigmas = zeros(SIZE,size(centers,2));
sigmas  = zeros(SIZE,size(centers,2));
% Loop variables
for k=1:size(centers,2)
   lvalue = zeros(size(y,1),SIZE);
   % Loop data points
   for i=1:size(y,1)
      dist = zeros(SIZE,1);
      % Loop cluster centers
      for j=1:SIZE
         if ~isequal(centers(j,:),y(i,:))
            d = (centers(j,k)-y(i,k))^2;
            dist(j) = sqrt(d);
         end
      end
      [~,idx] = min(dist);
      lvalue(i,idx) = y(i,k);
   end
   % Calculate sigmas for inputs and outputs
   for i=1:SIZE
      temp = lvalue(:,i);
      temp(temp==0) = [];
```

```
        % If no elements in set, insert a minimal value
        if isempty(temp)
            tempVar = abs(max(y)-min(y));
            lSigmas(i,k) = (tempVar(k)/100);
            rSigmas(i,k) = (tempVar(k)/100);
            sigmas(i,k)  = (tempVar(k)/100);
        % If elements in set exists, calculate lengths A and B
        else
            temp = sort(temp,'ascend');
            sigmas(i,k) = sqrt((1/(size(temp,1)))*sum((temp-centers(i,k)).^2));
            if ~isnan(temp)
                [lTemp,rTemp] = intervalinfogranule(temp,alpha,alpha,centers(i,k));
                if lTemp==0 || rTemp==0
                    tempVar = max(y)-min(y);
                    lSigmas(i,k) = (tempVar(k)/100);
                    rSigmas(i,k) = (tempVar(k)/100);
                elseif lTemp <= rTemp
                    lSigmas(i,k) = (lTemp/3);
                    rSigmas(i,k) = (rTemp/3);
                else
                    lSigmas(i,k) = (rTemp/3);
                    rSigmas(i,k) = (lTemp/3);
                end
            end
        end
    end
end
```

```
sigmas(~sigmas) = 0.01;
 centersAnt = centers(:,1:size(Xin,2));
lSigmasAnt = lSigmas(:,1:size(Xin,2));
rSigmasAnt = rSigmas(:,1:size(Xin,2));
mSigma = abs(rSigmasAnt - lSigmasAnt);
xBounds = [];
[numData,numInp]   = size(Xin);
[~,numOutp] = size(Xout);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
% Distance multipliers
distMultp = (1.0 / sqrt(2.0)) ./ sigmas;
[numRule,~] = size(centersAnt);
sumMu = zeros(numData,1);
muVals = zeros(numData,1);
dxMatrix = zeros(numData,numInp);
muMatrix = zeros(numData,numRule * (numInp + 1));
for i=1:numRule
    for j=1:numInp
        dxMatrix(:,j) = (Xin(:,j) - (centersAnt(i,j)-mSigma(i,j))) * distMultp(i,j);
    end
    dxMatrix = dxMatrix .* dxMatrix;
    if numInp > 1
        muVals(:) = exp(-1 * sum(dxMatrix'));
    else
        muVals(:) = exp(-1 * dxMatrix');
    end
```

```
        sumMu = sumMu + muVals;
        colNum = (i - 1)*(numInp + 1);
        for j=1:numInp
            muMatrix(:,colNum + j) = Xin(:,j) .* muVals;
        end
        muMatrix(:,colNum + numInp + 1) = muVals;
    end % endfor i=1:numRule
    sumMuInv = 1.0 ./ sumMu;
    for j=1:(numRule * (numInp + 1))
        muMatrix(:,j) = muMatrix(:,j) .* sumMuInv;
    end
    % Compute the TSK equation parameters
    lOutEqns = muMatrix \ Xout;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
    % Distance multipliers
    distMultp = (1.0 / sqrt(2.0)) ./ sigmas;
    sumMu = zeros(numData,1);
    muVals = zeros(numData,1);
    dxMatrix = zeros(numData,numInp);
    muMatrix = zeros(numData,numRule * (numInp + 1));
    for i=1:numRule
        for j=1:numInp
            dxMatrix(:,j) = (Xin(:,j) - (centersAnt(i,j)+mSigma(i,j))) * distMultp(i,j);
        end
        dxMatrix = dxMatrix .* dxMatrix;
        if numInp > 1
```

```
        muVals(:) = exp(-1 * sum(dxMatrix'));
      else
        muVals(:) = exp(-1 * dxMatrix');
      end
      sumMu = sumMu + muVals;
      colNum = (i - 1)*(numInp + 1);
      for j=1:numInp
        muMatrix(:,colNum + j) = Xin(:,j) .* muVals;
      end
      muMatrix(:,colNum + numInp + 1) = muVals;
    end % endfor i=1:numRule
    sumMuInv = 1.0 ./ sumMu;
    for j=1:(numRule * (numInp + 1))
      muMatrix(:,j) = muMatrix(:,j) .* sumMuInv;
    end
    % Compute the TSK equation parameters
    rOutEqns = muMatrix \ Xout;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
    % the clusters input dimensions
    [numRule,~] = size(centersAnt);
    % Set the FIS name as 'sug[numInp][numOutp]'
    theStr = sprintf('sug%g%g',numInp,numOutp);
    fismat.name=theStr;
    % FIS type
    fismat.type = 'sugeno';
    fismat.andMethod    = 'prod';
```

```
fismat.orMethod     = 'probor';
fismat.impMethod    = 'prod';
fismat.aggMethod    = 'max';
fismat.defuzzMethod = 'cos';
% Set the input variable labels
for i=1:numInp
   theStr = sprintf('in%g',i);
   fismat.input(i).name = theStr;
end
% Set the output variable labels
for i=1:numOutp
   theStr = sprintf('out%g',i);
   fismat.output(i).name = theStr;
end
% Set the input variable ranges
if isempty(xBounds)
   % No data scaling range values were specified, use the actual minimum and
   % maximum values of the data.
   minX = min(Xin);
   maxX = max(Xin);
else
   minX = xBounds(1,1:numInp);
   maxX = xBounds(2,1:numInp);
end
ranges = [minX ; maxX]';
for i=1:numInp
  fismat.input(i).range = ranges(i,:);
```

```
end
% Set the output variable ranges
if isempty(xBounds)
    % No data scaling range values were specified, use the actual minimum and
    % maximum values of the data.
    minX = min(Xout);
    maxX = max(Xout);
else
    minX = xBounds(1,numInp+1:numInp+numOutp);
    maxX = xBounds(2,numInp+1:numInp+numOutp);
end
ranges = [minX ; maxX]';
for i=1:numOutp
  fismat.output(i).range = ranges(i,:);
end
% Set the input membership function labels
for i=1:numInp
    for j=1:numRule
        theStr = sprintf('in%gcluster%g',i,j);
        fismat.input(i).mf(j).name = theStr;
    end
end
% Set the output membership function labels
for i=1:numOutp
    for j=1:numRule
        theStr = sprintf('out%gcluster%g',i,j);
        fismat.output(i).mf(j).name = theStr;
```

```
    end
  end
  % Set the input membership function types
  for i=1:numInp
    for j=1:numRule
      fismat.input(i).mf(j).type = 'igaussmtype2';
    end
  end
  % Set the output membership function types
  for i=1:numOutp
    for j=1:numRule
      fismat.output(i).mf(j).type = 'linear';
    end
  end
  % Set the input membership function parameters
  for i=1:numInp
    for j=1:numRule
      fismat.input(i).mf(j).params = ...
        [sigmas(j,i) centersAnt(j,i)-mSigma(j,i) centersAnt(j,i)+mSigma(j,i)];
    end
  end
  % Set the output membership function parameters
  for i=1:numOutp
    for j=1:numRule
      lOutParams = reshape(lOutEqns(:,i),numInp + 1,numRule);
      rOutParams = reshape(rOutEqns(:,i),numInp + 1,numRule);
      params = (lOutParams(:,j)'+rOutParams(:,j)')/2;
```

```
        outputSpread = ones(size(params)) .* 0.01;
        fismat.output(i).mf(j).params = [params outputSpread];
      end
    end
    % Set the membership function pointers in the rule list
    colOfEnum = (1:numRule)';
    for j=1:numRule
      for i=1:numInp
        fismat.rule(j).antecedent(i) = colOfEnum(j);
      end
      for i=1:numOutp
        fismat.rule(j).consequent(i) = colOfEnum(j);
      end
     % Set the antecedent operators and rule weights in the rule
      fismat.rule(j).weight=1;
      fismat.rule(j).connection=1;
    end
end
```

Appendix C.4

```
function [fis] = UBIGF_igaussstype2(trD1,trD2,numInputs,numOutputs,K)
fis = newfistype2('IT2FIS','sugeno','prod','probor','prod','sum','cos');
%% Data normalization: [-1,1]
maxX = max(max(trD1));
trD1 = trD1./maxX;
%% Cluster analysis
[centers,U] = fcm(trD1,K);
%% Data de-normalization
centers = centers .* maxX;
trD1    = trD1.*maxX;
rangeA1 = min(trD1(:,1:numInputs));
rangeB1 = max(trD1(:,1:numInputs));
[~,idx] = max(U);
%% Build antecedents. First Pass
for i=1:numInputs
   rA = rangeA1(i);
   rB = rangeB1(i);
   fis = addvartype2(fis,'input',strcat('x',num2str(i)),[rA rB]);
   for j=1:K
      set = trD1(idx==j,:);
      subset = set(:,i);
      m = centers(j,i);
      if size(set,1)==0
         tempVar = max(trD1)-min(trD1);
         s1 = tempVar(i)/100;
      else
         s1 = std(subset);
      end
      if s1==0
         tempVar = max(trD1)-min(trD1);
         s1 = tempVar(i)/100;
      end
      Flm = strcat(strcat('F',num2str(j)),num2str(i));
      fis = addmftype2(fis,'input',i,Flm,'igaussstype2',[s1 s1 m]);
   end
```

```
end
%% Data normalization: [-1,1]
maxX = max(max(trD2));
trD2 = trD2./maxX;
%% Cluster analysis
[centers,U] = fcm2(trD2,K,U);
%% Data de-normalization
centers = centers .* maxX;
trD2    = trD2.*maxX;
rangeA2 = min(trD2(:,1:numInputs));
rangeB2 = max(trD2(:,1:numInputs));
[~,idx] = max(U);
%% Build antecedents. Second Pass
for i=1:numInputs
   fis.input(i).range(1) = min(fis.input(i).range(1),rangeA2(i));
   fis.input(i).range(2) = max(fis.input(i).range(2),rangeB2(i));
   for j=1:K
      set    = trD2(idx==j,:);
      subset = set(:,i);
      m = centers(j,i);
      tempVar = max(trD2)-min(trD2);
      if size(set,1)==0
         s2 = tempVar(i)/100;
      else
         s2 = std(subset);
      end
      if s2==0
```

```
            tempVar = max(trD1)-min(trD1);
            s2 = tempVar(i)/100;
        end
        fis.input(i).mf(j).params(3) = (fis.input(i).mf(j).params(3) + m) / 2;
        fis.input(i).mf(j).params(2) = s2;
    end
end
%% Build and initialize interval TSK consequents
initParams = zeros(1,(numInputs+1)*2);
for i=1:numOutputs
    rA = min(rangeA1(i),rangeA2(i));
    rB = max(rangeB1(i),rangeB2(i));
    fis = addvartype2(fis,'output',strcat('y',num2str(i)),[rA rB]);
end
for i=1:K
    for j=1:numOutputs
        Glm = strcat(strcat('G',num2str(j)),num2str(i));
        fis = addmftype2(fis,'output',j,Glm,'linear',initParams);
    end
end
%% Build rule list
ruleList = ones(K, numInputs+numOutputs+2);
for i = 2:1:K
    ruleList(i,1:numInputs+numOutputs) = i;
end
fis = addruletype2(fis,ruleList);
%%  Optimize interval TSK consequents
D = [trD1;trD2];
inD  = D(:,1:numInputs);
outD = D(:,(numInputs+1):end);
nVars = size(fis.output(1).mf(1).params,2) * fis.rule(end).antecedent(1) *
size(fis.output,2);
fis   = cuckoo_search(25,nVars,fis,inD,outD);
```

Appendix C.5

```
function set = gaussgaussGMF_A(x,card,primCenter,primSigma)
    discretization = size(x,2);
    secSigma = 0.1;
    fu = gaussmf(card,[primSigma primCenter]);  % fu
    u = linspace(0,1,discretization);           % u
    fxu = zeros(discretization,discretization); % fxu
    [X,Y] = meshgrid(x,u);
    for i=1:size(card,1)
        zTemp = exp( -((X-card(i)).^2 + (Y-fu(i)).^2) / (2*secSigma^2) );
        fxu = max(fxu,zTemp);
    end
    set.X{1} = x';
    for i=1:discretization
        set.fXU{i} = [u' fxu(:,i)];
    end
```

Appendix C.6

```
function set = gaussgaussGMF_B(x,card,primCenter,primSigma)
    discretization = size(x,2);
    secSigma = 0.1;
    fu = gaussmf(card,[primSigma primCenter]);  % fu
    u = linspace(0,1,discretization);           % u
    fxu = zeros(discretization,discretization); % fxu
    [X,Y] = meshgrid(x,u);
for i=1:size(card,1)
    sigmaTemp = secSigma * fu(i);
    zTemp = exp( -((X-card(i)).^2 + (Y-fu(i)).^2) / (2*sigmaTemp^2) );
    fxu = max(fxu,zTemp);
end
set.X{1} = x';
for i=1:discretization
    set.fXU{i} = [u' fxu(:,i)];
end
```

Index

M.A. Sanchez et al., *Type-2 Fuzzy Granular Models*,
SpringerBriefs in Computational Intelligence, DOI 10.1007/978-3-319-41288-7