Juzzy – A Java based Toolkit for Type-2 Fuzzy Logic

An object-oriented toolkit for the development of type-1, interval type-2 and general type-2 fuzzy systems.

Christian Wagner
Horizon Digital Economy Institute & IMA Group, School of Computer Science
University of Nottingham
Nottingham, United Kingdom
christian.wagner@nottingham.ac.uk

Abstract—In this paper we describe a Java based toolkit for the development of type-1, interval type-2 and (zSlices based) general type-2 Fuzzy Logic Systems (FLSs) which is available as a free download. We describe our motivation for the release of such a toolkit in Java – to improve the accessibility to specifically type-2 FLSs to users both in industry and academia beyond the fuzzy logic research community, as well as to facilitate the application of FLSs in real world applications from web and cloud based deployments to multi-agent based control, for example in smart buildings. To the best of our knowledge it is the first package or toolkit that enables the straightforward design and implementation of type-1, interval and general type-2 FLSs. We review its features and provide sample implementations of the different FLSs, together with explanations and source code. Finally, we conclude with future developments and a call for feedback and contributions to drive the further development of the toolkit.

Keywords—fuzzy systems; type-2; toolkit; fuzzy inference; software; open-source; multi-core; Java

I. INTRODUCTION

Since Zadeh introduced Fuzzy Logic in 1965 [1], it has become a widely applied technique in applications as well as a research domain in its own right. A wide host of variations based on the original concepts have been developed, in particular type-1 and, based on type-2 fuzzy sets introduced in 1975 [2], type-2 fuzzy logic systems (FLSs). Most applications have centered on type-1 FLSs, with an increase in recent years in the number of applications leveraging type-2 FLSs, in particular interval type-2 FLSs [2]. This has been paralleled by a strong research interest in further developing type-2 FLSs, including the exploration and development of general type-2 FLSs (e.g. see [4][5][6]).

Today, research in both interval and general type-2 fuzzy logic and FLSs is rapidly advancing with areas such as Computing With Words [7] and new approaches such as constructing fuzzy sets based on agreement [8] exploring the potential of type-2 fuzzy sets and systems. In comparison to the advances in research, the number of applications of type-2, in particular general type-2 FLSs is still very limited. More importantly, the majority of existing applications that do exist are mostly confined to applications that are research-led by members of the fuzzy logic research community, rather than third party applications entirely conceived and implemented, for example, by practitioners in other research fields and/or industry.

It is a well-known phenomenon in computing that for the adoption of a new method or algorithm, the availability of software implementing said method/algorihm, is essential. Analogously, we believe that both for reproducibility of research outputs as well as the more widespread adoption and application of non-classical fuzzy logic systems (such as interval type-2 and general type-2 fuzzy systems), the provision of free, widely available, open-source software is essential. Previously a software toolbox in the popular R language [9] was introduced which provides the basis for the development of both type-1 and interval type-2 FLSs [10]. R is a free, open source software comparable to the commercial MATLAB® [11] software which is highly popular in engineering and computer science as well as in the life sciences and psychology. In this paper, we are following a different approach by introducing an advanced toolkit supporting type-1, interval type-2 and general type-2 FLS design and implementation based on the object oriented Java language.

The use of Java [12] as a popular software development language enables the use of FLSs in a myriad of applications from cloud/web-based applications to platform independent deployments in intelligent buildings - thus greatly increasing the potential for real world applications of FLSs. Further, the object-oriented nature of Java lends itself to the incremental implementation and design of type-1, interval and general type-2 FLSs, mirroring developments in theory through its implementation and thus facilitating both adoption and extension of/to higher order FLSs (e.g. from type-1 to interval type-2 FLSs).

The paper is intended as both an introduction to the Java based fuzzy logic toolkit as well as an invitation to the fuzzy logic community to join with the development effort, give feedback and make recommendations.

The paper is structured as follows. Section II provides basic background information on the current availability of toolkits and packages for the development of type-1/type-2 applications as well as the Java language in general. Subsequently, Section III describes the Juzzy toolbox and its architecture, followed by a discussion of currently available features in Section IV. Section V demonstrates some of the features through the practical implementation of type-1, interval type-2 and zSlices.
based general type-2 FLSs. Finally, Section V draws some conclusions and discusses the future development of the toolkit.

II. BACKGROUND

Toolboxes and toolkits for the implementation of type-1 Fuzzy Logic Systems (FLSs) are widespread and have led to the application of type-1 FLSs in a myriad of applications. As noted in [10], perhaps the most well-known of these is the Fuzzy Logic Toolbox provided for MATLAB® (The Mathworks, Inc.) which allows users to create type-1 fuzzy inference systems using MATLAB® command-line functions or a graphical user interface (GUI) [11].

In the case of type-2 FLSs, the availability of software and tools is still extremely limited in comparison to type-1 FLSs. While several researchers make source code publicly available (e.g. J. Mendel), the use of source code still requires in-depth familiarity with the material in question and thus does not fulfill the same role as toolkits in terms of making the material easily accessible for people from a variety (in particular non-Computer Science) of backgrounds.

Beyond source code, existing toolkits that support type-2 FLSs include [13] and [10]. In [13], Castro et al. demonstrate a toolbox for MATLAB® that provides tools for the development of interval type-2 fuzzy systems, including a GUI similar to that of the MATLAB® Fuzzy Logic Toolbox [11]. While the toolbox allows the implementation of interval type-2 fuzzy systems using MATLAB®, it, being based on the proprietary MATLAB® software is restricted in terms of its availability (MATLAB® is subject to license fees) as well as its user base (MATLAB® is predominantly used in the physical sciences and engineering communities). The R based toolkit [10] is aimed at enabling the broader usage of type-2 FLS beyond the commercial MATLAB® environment with a particular view to engage research communities where R is highly popular such as Psychology and the Social Sciences.

While MATLAB® and R are both commonly used to interact with and process data both in an academic and industrial setting, they do not lend themselves to building real world applications which are generally written in languages such as C (or derivates such as C++, C#, etc.), Python or Java. The latter is particularly attractive as its platform independence makes it highly suitable for a variety of systems from web-based applications to agent-based control for example as part of intelligent inhabited environments.

Beyond the actual programming language employed, current toolkits which support type-2 FLSs are limited to the development of interval type-2 FLS and thus do not allow the development of novel applications based on general type-2 fuzzy systems which have recently seen strong development in the research community [4][5][6].

This paper introduces and describes Juzzy, a Java based based toolkit which enables the rapid development of type-1, interval type-2 and zSlices based general type-2 FLSs. Juzzy has been designed to provide a straightforward entry to the implementation of different types of FLSs by leveraging the object-oriented nature of Java. It is highly flexible and contains advanced features such as multi-core support for general type-2 fuzzy systems and flexible visualisation options for different types of fuzzy sets.

The toolkit and its features will be further detailed in the following section, followed by implementation examples for type-1, interval type-2 and zSlices based general type-2 FLSs. It is worth clarifying that while this paper contains sample implementations of type-1, interval type-2 and general type-2 FLSs, its purpose is not to compare the performance of the different types of FLSs but to demonstrate their implementation and specifically the steps required to move from a type-1 all the way to a general type-2 FLS implementation (conceptually as well as in actual implementation code).

III. JUZZY – AN OBJECT ORIENTED TOOLKIT

The Java based toolkit – Juzzy, is an object oriented toolkit that supports the development of type-1, interval and zSlices based general type-2 FLSs. It is available online at http://juzzy.wagnerweb.net/ together with documentation and examples. The toolkit is free of charge for non-commercial purposes and we only ask authors/developers to reference this paper when using it.

While the main purpose of Juzzy is to enable the rapid development of different types of FLSs, it is worth to briefly highlight the structure of the toolkit itself. Over the years of development in the theory of type-2 FLSs, the main underlying design requirement has been that when all the uncertainty about the degrees of membership disappears, a type-2 fuzzy set/system should behave like a type-1 fuzzy set/system (e.g. see [14]).

This design requirement is reflected in the actual representations for type-2 fuzzy sets, where interval type-2 fuzzy sets are commonly represented as two type-1 fuzzy sets (an upper and a lower membership function) and zSlices based general type-2 fuzzy sets are represented as a series of weighted interval type-2 fuzzy sets (i.e. zSlices [4]). In Juzzy, this inherent structure is exploited to create a hierarchical structure where an interval type-2 fuzzy set is represented as two type-1 fuzzy sets and a general type-2 fuzzy set can be represented as a series of interval type-2 fuzzy sets.

Figure 1 shows a basic package view of Juzzy while Figure 2 shows a UML diagram of the same class structure. Figure 2 indicates the hierarchical structure of the toolkit, showing generic data types at the most basic level, followed by type-1 sets, interval type-2 and then general type-2 fuzzy sets and systems. Examples are at the top as they rely on all types of sets and FLSs.

While the structure makes the toolkit transparent during development, it also enables a rapid transition between different FLS implementations (e.g. from type-1 to interval type-2) and thus has direct implications on the toolkit’s use. The strong similarities in the implementation of different types of FLSs will be further detailed in Section IV by showing real examples of FLS implementations. Finally, Figure 2 also highlights the built-in support for multi-core architectures which will briefly be discussed in Section IV.
the toolkit and/or to give feedback in terms of which features are most needed.

B. Performance and Multi-Core Support

Juzzy natively supports the use of multiple processors for the processing of zSlices based general type-2 FLSs. The process of computing zSlices based general type-2 FLSs in parallel and the potential increase in processing speed on multi-core architectures has previously been discussed in [4]. On demand (one line of code), the toolkit dynamically allocates individual zLevels of zSlices based general type-2 FLSs to individual processing cores, greatly increasing efficiency and speed in real world applications without additional effort from the developer. Complete examples of single and multi-core implementations of FLSs are included with the toolkit.

C. Visualisation

The Juzzy toolkit functions enable the usage of any visualisation libraries (accessible from Java) and is preconfigured to work with JMathPlot [17] which allows the rapid generation of visualisations for both membership functions and control surfaces. Section V provides a series of examples. Additionally, human-readable output is available from most FLS components created in Juzzy, e.g. a rule base is easily returned in a human-readable format as shown for example in Figure 10.

V. FLS IMPLEMENTATIONS IN JUZZY

We proceed by illustrating some of the features demonstrating the implementation of a type-1, interval type-2 and general type-2 FLSs for the classical waiter-tipping problem which we briefly outline in Section IV.A below. Each further subsection (IV.B-IV.D) provides full detail on the implementation, features and actual code employed to rapidly design, construct and execute the three respective types of FLS.

In particular we will highlight the similarities and differences as we transition from the type-1 to the interval and general type-2 FLSs, providing detail on how the hierarchical structure of Juzzy enables the rapid transition between different types of FLS.

As mentioned above, we do not focus on performance or developing the “best” system using each type of FLS but on illustrating the different FLS types and their implementation. In this context, the fuzzy sets and the rule bases used are only examples and no optimisation techniques have been applied. In a real application it is expected that these properties of FLSs are designed based on one or a variety of techniques such as design based on expert knowledge and consensus or learning techniques (for example using a neural network or evolutionary approach (e.g. [15])).

A. The waiter-tipping problem

In order to demonstrate the implementation of different types of FLS based on the Juzzy toolkit, we will implement the well-known “waiter-tipping” problem in which one would like to determine the amount of tip (as a percentage) one should give to the waiting staff based on two variables: the quality of

IV. TOOLKIT FEATURES

As the scope of the paper does not allow the detailing of all individual features, we will focus on the main features in subsection IV.A, followed by a brief description of the built-in multi-core support and visualisation features in subsections IV.B and IV.C respectively. Section V will demonstrate some of the features of the toolkit through sample implementations of type-1, interval type-2 and zSlices based general type-2 FLSs.

A. Features Overview

The Juzzy toolkit provides all of the features required to construct MISO (Multiple Input Single Output) Mamdani type fuzzy logic inference systems of type-1, interval type-2 or general type-2. Note that MIMO (Multiple Input Multiple Output) systems can be implemented as a series of MISO systems. As Juzzy is provided for free and complete with all source code, it provides a highly extensible platform which will be updated as further components are developed.

For all types of FLS (type-1, interval type-2 and general type-2), the toolkit provides a series of membership function types such as triangular, Gaussian, trapezoidal, etc., some of which are shown in Section V of this paper.

For defuzzification, Juzzy currently supports height and centroid defuzzification, while for the type-reduction of type-2 sets and systems, the KM algorithm [16] is employed. The inclusion of further algorithms for both defuzzification and type-reduction is planned and one of the objectives of this paper is to encourage the research community to contribute to
the food and the level/quality of service provided by the member(s) of waiting staff. In control terms, we thus have a two-inputs (food and service), single output (tip) problem. In our example, both food and service are modelled as variables with domain $[0, 10]$ (i.e. a user would rate these variables with a number between 0 and 10) and the tip is modelled as a variable with domain $[0, 30]$ (i.e. a tip of 0-30%). The following subsections will detail the implementation of a type-1, interval and general type-2 FLS based on this same example in each case.

B. Type-1 FLSs in Juzzy

In order to demonstrate the set-up of a sample type-1 FLS in Juzzy we take a step-by-step approach which is later mirrored in the interval and $z$Slices based general type-2 FLSs.

1) Definition of FLS inputs.

We define both inputs (food quality and service level) as numbers between 0 and 10. Figure 3 shows the definition of input objects in Java using the Juzzy software package.

The input objects play a two-fold role: they define the domain for each input and they allow the association of the actual inputs to the antecedents in the FLS as shown in Figure 8.

```
food = new Input("Food Quality", new Tuple(0,10));
service = new Input("Service Level", new Tuple(0,10));
```

Figure 3. Defining FLS inputs in Java using the Juzzy package. Each input has a defined domain (here $[0, 10]$).

2) Definition of Type-1 Fuzzy Sets for each variable.

Juzzy supports a variety of type-1 membership functions types including Gaussian, triangular and trapezoidal membership functions. In this paper we have chosen to model the variables with a small number of fuzzy sets to maintain a reasonable level of complexity suitable for description – in real applications any number of fuzzy sets is possible.

In terms of the inputs, we model the quality of the food using two simple triangular fuzzy sets as shown in Figure 4. Service is modelled using three Gau-Angle fuzzy sets (another type of membership function available in Juzzy, combining the properties of Gaussian and triangular membership functions) as shown in Figure 5. Finally, the output – the level of tip is modelled using three Gaussian fuzzy sets as shown in Figure 6.

Figure 7 illustrates the setup of the Type-1 fuzzy sets in Java based on the Juzzy toolkit.

```
MF for bad food, MF for great food
```

Figure 4. Triangular type-1 fuzzy sets for bad and great food.

```
MF for unfriendly service, MF for ok service, MF for friendly service
```

Figure 5. Gau-Angle type-1 fuzzy sets for unfriendly, ok and friendly service.

```
MF for low tip, MF for medium tip, MF for high tip
```

Figure 6. Gaussian type-1 fuzzy sets for low, medium and high tip.

```
MF for bad food, MF for great food
```

Figure 7. Source Code snippet showing the setup of the fuzzy sets / membership functions for the type-1 FLC.

3) Definition of Logical Antecedents and Consequents.

In preparation of designing the rule base for the FLS in Juzzy, each previously created membership function is associated with antecedent and consequent objects. Note that each antecedent is labelled (the labels are later used to create a list of rules which can be used during interpretation or visualisation, e.g. Figure 10) and that all antecedents are associated with the respective input defined previously (i.e. food and service). The source code for this step is illustrated in Figure 8.

```
T1_Antecedent lowFood = new T1_Antecedent("Low Food", MFForBadFood, MFForGreatFood, food);
T1_Antecedent greatFood = new T1_Antecedent("Great Food", MFForGreatFood, MFForBadFood, food);
T1_Antecedent unfriendlyService = new T1_Antecedent("Unfriendly Service", MFForUnfriendlyService, MFForFriendlyService, service);
T1_Antecedent okService = new T1_Antecedent("OK Service", MFForOkService, MFForFriendlyService, service);
T1_Antecedent friendlyService = new T1_Antecedent("Friendly Service", MFForFriendlyService, service);
T1_Consequent lowTip = new T1_Consequent("Low Tip", MFForLowTip);
T1_Consequent mediumTip = new T1_Consequent("Medium Tip", MFForMediumTip);
T1_Consequent highTip = new T1_Consequent("High Tip", MFForHighTip);
```

Figure 8 Definition of antecedent and consequent objects in Juzzy
4) Definition of the Rule Base.

Having set up the antecedents and consequents of an FLS, it is straightforward to create a new rule base and to add rules. Figure 9 illustrates the set-up of a new rule base with 6 rules and the actual definition and addition of each rule. Note how the antecedents of the rules are specified as an array, followed by the respective consequent for each rule.

A rule base can be printed using the object’s toString() method. The resulting human-readable output for this example is given in Figure 10.

```java
rulebase = new TI_Rulebase(6);
rulebase.addRule(new TI_Rule(new TI_Antecedent(), [badFood, unfriendlyService], lowTip));
rulebase.addRule(new TI_Rule(new TI_Antecedent(), [badFood, okService], lowTip));
rulebase.addRule(new TI_Rule(new TI_Antecedent(), [badFood, friendlyService], mediumTip));
rulebase.addRule(new TI_Rule(new TI_Antecedent(), [goodFood, unfriendlyService], lowTip));
rulebase.addRule(new TI_Rule(new TI_Antecedent(), [goodFood, okService], mediumTip));
rulebase.addRule(new TI_Rule(new TI_Antecedent(), [goodFood, friendlyService], highTip));
```

Figure 9 Creation of a new type-1 rule base in Juzzy

After specifying the rule base, the FLS is ready for use. During execution, the value of the inputs is updated using each input’s setInput() method (e.g.: food.setInput(8)); and the overall output is generated using the evaluate method of the rulebase (e.g.: output = rulebase.evaluate(1, 100)); The control surface for the FLS when discretising both inputs into 100 steps and employing minimum inference and centroid defuzzification is shown in Figure 11.

2) Definition of Interval Type-2 Fuzzy Sets for each variable.

Interval type-2 fuzzy sets are defined using two individual type-1 fuzzy sets (an upper and a lower membership function). In our example, the food quality is modelled using two triangular interval type-2 fuzzy sets (Figure 12) and the service using three interval type-2 Gaussian and Triangular fuzzy sets (a combination of interval type-2 Gaussian and Triangular membership functions - as in the type-1 case) as shown in Figure 13.

Note that in contrast to the type-1 case, we have chosen to model food with only two rather than three fuzzy sets. Fewer fuzzy sets result in fewer rules needing to be specified, which generally supports the interpretability of the FLS’s rule base. In this case it simplifies the FLS and facilitates the visualisation of all components as part of this paper. Finally, the output – the level of tip – is modelled using three Gaussian interval type-2 fuzzy sets as shown in Figure 14.

Figure 15 illustrates the actual setup of the interval type-2 fuzzy sets in Java based on the Juzzy toolkit. Note how each interval type-2 set is (solely) defined through an upper and a lower type-1 membership function.

C. Interval Type-2 FLSs in Juzzy

Similar to the approach for constructing the Type-1 FLS above, we proceed to set up the IT2 FLS step by step, highlighting in particular the similarities and differences between the FLSs.

1) Definition of FLS inputs.

As the actual inputs to the IT2 FLS are identical to the type-1 FLS case, the set-up of the FLS inputs is identical to the type-1 case (see Figure 3).
discretisation steps for both inputs and employing KM type-reduction [16] is shown in Figure 19.

A comparison between the control surface for the interval type-2 FLS in Figure 19 and the one for the type-1 FLS in Figure 11 may indicate a slightly smoother control surface of the type-2 FLS even though a smaller number of rules was employed. However it is important to note that a comparison of the different FLSs is not the objective of this paper. The FLSs in this article have been designed arbitrarily and there is no ground truth “how much to tip” to serve as a benchmark for the performance of either FLS.

D. General Type-2 FLSs in Juzzy

As described in Section IV, Juzzy natively supports the creation of zSlices based general type-2 fuzzy sets and FLSs [4]. We proceed to design and implement a zSlices based general type-2 FLS (zFLS). In order to maintain tractability of the different zLevels, especially in the figures, four zLevels for the zFLS have been chosen. As such, all zSlices based sets will be modelled using four zSlices. The sequence of steps for the implementation of a zFLS is identical to that of type-1 and interval type-2 FLSs and we briefly review each step below.

1) Definition of FLS inputs.

As the actual inputs to the FLSs are identical to those in the type-1 and interval type-2 FLS case, their setup is also identical and follows Figure 3.

2) Definition of zSlices based General Type-2 Fuzzy Sets for each variable.

In Juzzy, zSlices based general type-2 fuzzy sets are specified as a series of zSlices where each zSlice itself is specified using an interval type-2 fuzzy set and an associated third dimension (secondary membership) (see [4]).

In this example, the individual zSlices based general type-2 fuzzy sets for each variable are based on the interval type-2 fuzzy sets from the previous section. Specifically, the third dimension of each zSlices based general type-2 fuzzy set is formed by creating four zSlices evenly distributed within the Footprint of Uncertainty [16] of the original IT2 fuzzy sets. We have opted for this straightforward approach to generating general type-2 fuzzy sets but depending on the application, the toolkit allows for the precise specification of each zSlice independently.

The food quality and service level are modelled using two triangular zSlices based general type-2 fuzzy sets as shown in

Figure 15. Definition of interval type-2 fuzzy sets in Juzzy. Each IT2 set is based on an upper and a lower type-1 membership function.

3) Definition of Logical Antecedents and Consequents.

As in the type-1 case, to create an FLS, all fuzzy sets are associated with antecedents and consequents in preparation for the construction of fuzzy logic rules. Figure 16 shows the construction of the respective antecedents and consequents in Juzzy for the sample FLS.

Figure 16. Antecedent and consequent definition for the interval type-2 FLC in Juzzy.

4) Definition of the Rule Base.

As there are fewer fuzzy sets than in the type-1 case, there are fewer antecedents, resulting in fewer rules (see Figure 17). A human-readable printout of the rule base is shown in Figure 18.

Figure 17. Creation of a new interval type-2 rule base in Juzzy.

Interval Type-2 Fuzzy Logic System with 4 rules:
IF BedQuality AND UnfriendlyService THEN LowTip
IF BedQuality AND FriendlyService THEN MediumTip
IF GreatFood AND UnfriendlyService THEN LowTip
IF GreatFood AND FriendlyService THEN HighTip

Figure 18. Printout of interval type-2 rule base.

After the specification of the rule base, the FLS is ready for execution in an identical fashion to the type-1 case, i.e. first the inputs are updated and then the rule base is evaluated for the given inputs. For illustration, the control surface for the rule base in Figure 17, employing minimum inference, 100...
Figure 21 and Figure 22. The level of tip meanwhile is modelled using three Gaussian zSlices based general type-2 fuzzy sets as shown in Figure 23. Each of the figures includes both a rear/side view as well as a front-view of the respective sets to facilitate their visualisation. The process of creating the zSlices based general type-2 fuzzy sets in Juzzy is illustrated by the source code in Figure 20.

Note that, as stated in [4], it is a requirement that all zSlices based general type-2 fuzzy sets in a given FLS have the same number of zLevels as this enables the independent computation of each zLevel followed by a simple weighted average calculation to generate the final output.

3) Definition of Logical Antecedents and Consequents.

As in the type-1 and interval type-2 cases, after all fuzzy sets have been defined, all of them are associated with antecedents and consequents in preparation for the construction of fuzzy logic rules.

Figure 24 shows the construction of the respective antecedents and consequents in Juzzy for the FLS.

4) Definition of the Rule Base.

As the antecedents/consequents are the same as in the interval type-2 case, the rule base of the zFLS is identical to that of the IT2 FLS (only the underlying fuzzy sets differ). The construction (in Juzzy) and the printout of the rule base for the zFLS is provided in Figure 25 and Figure 26 respectively.
After the specification of the rule base, the zFLS is ready for execution. During execution, each zLevel can be computed independently on different processors (subject to an available multi-core processing hardware architecture) and the output of all individual zLevels is recombined using a weighted average to produce the overall output (see [4]). The control surface for the zFLS is shown in Figure 27.

As noted, this paper does not aim to compare the performance of type-1, interval type-2 and general type-2 FLSs but to demonstrate the features of the freely available toolkit. In this light, a visual inspection of the control surface for the zFLS in Figure 27 confirms that it is highly similar, potentially somewhat smoother than the interval type-2 case – as is expected as the interval type-2 and zSlices based general type-2 FLSs share similar sets and the same rule base.

Figure 27. Control surface for zSlices based general type-2 FLC.

VI. CONCLUSIONS

As part of this paper we have provided an overview of Juzzy - a Java based Fuzzy Logic Toolkit which is available for download from http://juzzy.wagnerweb.net/ together with documentation and examples. The toolkit is free of charge for non-commercial purposes and we only ask authors/developers to cite this paper when using the toolkit.

We have highlighted our motivation for producing and releasing a Java based toolkit – mainly to provide access to recent developments in fuzzy logic and fuzzy logic systems to both industry and academia beyond the fuzzy logic community, as well as to provide developers a tool to implement fuzzy logic systems in real world applications such as web-based systems or multi-agent based systems (e.g. in smart buildings), where current approaches based on MATLAB® or R are not easily applicable.

We have reviewed the main features of the current toolkit, including the support for type-1, interval and general type-2 fuzzy logic systems, multi-core execution of general type-2 fuzzy systems and visualisation features.

We would like to reiterate that it is not the purpose of this paper to compare the different types of FLS but only to demonstrate and detail their implementation using the Juzzy toolkit. The purpose of the paper is to disseminate information about the availability of the toolkit and to encourage the research community to contribute to the development of the toolkit through suggestions, comments or their own source code contributions.

Java is a well-developed, widely used programming language that enables the platform independent development of applications for a myriad of uses. We hope that this Java based toolkit can support the more widespread adoption of type-1, interval type-2 and general type-2 FLSs.

ACKNOWLEDGMENT

All visualisations produced in this paper have been generated using the Juzzy toolkit which in turn relies on the open source java package JMathPlot [17] to generate graphics from its data.

REFERENCES