Answer to reviewers
jFuzzyLogic: a Java library to design fuzzy logic controllers according to the standard for fuzzy control programming

Pablo Cingolani and Jesús Alcalá-Fdez

Thank you very much for the detailed and careful review of our paper "jFuzzyLogic: a Java library to design fuzzy logic controllers according to the standard for fuzzy control programming". We would like to thank the referees and the Guest Editor for their comments and suggestions, which have substantially improved the paper. We have tried to address all the questions and issues in this new version of the paper, and a detailed description of our modifications is provided below. We hope that our corrections will answer all your concerns, and we look forward to hearing from you.

1 REVIEW NO. 1

Comments to the authors: The authors introduce a Java library for fuzzy controllers. Since it follows a standard IEC 61131-7, it seems to be easily applicable to a lot of problems. The authors also compare own library with existing ones in detail. Thus, readers can access related resources as well. This paper would be acceptable for publication in IJCIS, but some minor modifications are necessary:

Response: We incorporated all suggested changes, which improved readability. Thank you.

2 REVIEW NO. 2

Comments to the authors:

Comments to the authors: The paper presents jFuzzyLogic, an open source library for designing fuzzy logic controllers. jFuzzyLogic is written in Java and respects the FCL (fuzzy control language) IEC-61131-7 specifications. FCL is an instruction list oriented programming language. The aim is to provide a standardized language and parser, which could be interesting for designing industrial fuzzy controllers.

The paper is clear and easy to read. The objectives are well stated and an effort has been made to illustrate the functionalities.

However, some important issues are missing. The following points need to be addressed and discussed.

1- The fuzzy partition concept is not specified in the API. Consequently, there is no way to automatically design regular or irregular grids for a given input or output, for instance a k-MF regular fuzzy partition. More generally, that missing concept prevents the user from imposing constraints on the MFs, for exemple interpretability constraints such as the ones respected by standard fuzzy partitions (Ruspini partitions). Similarly the optimization module API does not include constraints on fuzzy partitions during optimization. This can be seen in Figure 11, where MF optimization results in non overlapping MFs for the v input, which may be risky even if it improves the accuracy on the training data set.

Answer: We do agree that IEC-61131-7 norm is lacking in this basic concept and we would like the IEC committee to address this issue in future releases of the norm.

In order to clarify, we added the following text to the manuscript (section 4.2):

> *It should be noted that, as mentioned in section 2.2, IEC standard does not define or address the concept of fuzzy partition of a variable. For this reason, fuzzy partition are not directly implemented in jFuzzyLogic. In order to produce a complete partition, we should indicate the membership function for each linguistic term.*

We also think that to be able to indicate constrains in the FIS, would be a good extension for jFuzzyLogic. We added the following words to out "Conclusions" section:

> *The jFuzzyLogic software package is continuously being updated and improved. Future work includes: i) using defuzzifier analytic solutions wherever possible (e.g. FIS consisting exclusively of trapezoidal and triangular sets), as it would considerably speed-up processing; ii) adding FIS constraints and developing ways to include them as part of the language extensions; and iii) at the moment, we are developing an implementation of an FCL to C/C++ compiler, allowing easy implementation of embedded control systems.*

Comments to the authors: 2- Connection with data files is not specified through an API. Examples are provided for entering values one by one, or as a command line argument to java, but no example is given to run the FLC using a current data file. The only exception is the optimization example in Section 5.2, which uses a hard set value (training.txt). A data API would be useful to specify the data sources (data base, on-line acquisition..), types (real, fuzzy. ..)and use (learning, inference, validation).

Answer: It is true that data files as source of information are not specified in the API, mostly because they are not part of the IEC norm either. This is due to the fact that the norm is supposed to be "implementable" even in computer lacking of a file a system (e.g. a stand alone microcontroller).

Needless to say, the concept is not only useful, but also trivial to implement given the current API. Following the reviewer's suggestion, we incorporated a simple extension to the command line version of the program. Now it is possible to use a TXT table as input simply by using the new "-txt" command line option.

We added the following lines to the manuscript in section 5.1:

> *When evaluation of multiple input values is required, values can be provided as a tab-separated input file using the command line option* `-txt file.txt`*. This approach can be easily extended for other data sources, such as databases, online acquisition, etc.*

Comments to the authors: 3- Apparently, membership functions (MFs) are systematically discretized. It seems a pity for analytical forms very easy to compute such as trapezoidal or triangular MFs.

Answer: We fully agree with the reviewer's remark. We considered this as part of our future work, so we explicitly added the following lines to the manuscript (same as previous answer):

> *The jFuzzyLogic software package is continuously being updated and improved. Future work includes: i) using defuzzifier analytic solutions wherever possible (e.g. FIS consisting exclusively of trapezoidal and triangular sets), as it would considerably speed-up processing; ii) adding FIS constraints and developing ways to include them as part of the language extensions; and iii) at the moment, we are developing an implementation of an FCL to C/C++ compiler, allowing easy implementation of embedded control systems.*

Comments to the authors: Other minor remarks:

Introduction: The very beginning of the introduction is a bit strange, regarding aforementioned drawbacks, that are actually not mentioned in this paper. Section 2.2 provides -¿ Section 3 provides

Section 2.1: FRB S: not defined. The term Data Base is confusing in Fig.1, might be better to use Fuzzy Partition Base?

Section 2.2: IL is its European language: this sentence is unclear.

Section 3: It seems to me that, as Java, C++ is platform independent (except for graphical interface, topic not addressed here).

Section 4.4: That subsection is not well organized, shouldn'tt the sentence "the last step, etc." be at the beginning? and the discretization issue discussed separately?

Section 4.6: What do you mean by "Obviously, such evaluation is usually not available in real world applications, otherwise we would probably not need to implement a fuzzy system."

Presentation: Fig. 6, 7 and 10 are much too small, please enlarge them.

English spelling and grammar: could be improved. for instance: (abstract)case of study -¿ case study (introduction) allow developpers -¿allows developpers (section 2) membership to the set -¿ membership in the set an FLC -¿ a FLC an FIS -¿ a FIS the KB is comprised by -¿ the KB includes as we shown -¿ as

shown shown bellow -¿ shown below (section 4) the union of all points by to tune functions: not understandable Excecution order -¿ Execution order (section 5) incorrect hyphenation of maximum Memnbership -¿ membership (Fig. 9 title)

Answer: We thank the reviewer for these comments. We updated the manuscript adding all the reviewer's suggestions. Thank you.