fuzzylite
a fuzzy logic control library in C++

Juan Rada-Vilela
jcrada@fuzzylite.com

Abstract—**Fuzzy Logic Controllers (FLCs) are software components found nowadays within well-known home appliances such as washing machines, rice cookers and refrigerators. Their objective is to control certain operations of these devices utilizing rules expressed with the uncertainty of human terms such as *cold*, *heavy* and *fast*. Such a flexibility has made of FLCs a widely used approach to incorporate artificial intelligence not only into home appliances, but also into energy-efficient motors, industrial control systems, human decision making, robots, and much more. State-of-the-art libraries to model FLCs have strong limitations in terms of licensing, cost, design and implementation, all of which have been recently addressed in a free open-source fuzzy logic control library named `fuzzylite`. This paper presents an introduction to fuzzy logic control and the underlying design and operation of `fuzzylite`.**

I. Introduction

A Fuzzy Logic Controller (FLC) is a software component that controls the output variables of a system according to its inputs and a set of rules expressed with the uncertainty of human terms. For example, an FLC can automatically dim the brightness of a lamp according to the ambient light by controlling its output power utilizing a set of rules stated as "if Ambient is *dark* then Power is *high*". Such a simplicity and interpretability makes of FLCs an attractive choice to address problems that involve uncertainty or are just too complex to address with conventional techniques [1].

FLCs are based on fuzzy logic, a field started by L. Zadeh in 1965 [2]. However, state-of-the-art libraries to model FLCs still have strong limitations such as costly and/or restrictive licensing [3], [4], overly complex implementations [5], and some unfortunate design choices [6], hence leaving developers with just a few options and little encouragement to utilize FLCs in their applications. To overcome these limitations, `fuzzylite` was designed and developed as a fuzzy logic control library that is free, open source, commercial friendly, object oriented, simple and easy to use.

The overall goal of this paper is to present an introduction to fuzzy logic control in order to provide the necessary background to start using `fuzzylite`. Specifically, the topics that will be covered are the design and operation of an FLC, and a description of the main features of `fuzzylite`.

The remainder of this paper is structured as follows. Section II describes the most relevant libraries available to model FLCs. Sections III and IV present the design and operation of an FLC. Section V describes the main features of `fuzzylite`. Finally, Section VI ends this paper with conclusions and suggestions for future work.

II. Related Work

The Matlab Fuzzy Logic Toolbox [3] is perhaps the most widely known library to design FLCs. It is built on top of the Matlab computing environment and bundles Mamdani and Takagi-Sugeno controllers, over 11 linguistic terms, four fuzzy logic operators, seven defuzzifiers, four types of hedges, and FLCs can be imported and exported utilizing the Fuzzy Inference System (FIS) format. Matlab and its toolbox are sold separately under restrictive and costly proprietary licenses. The toolbox has not been updated since 2005.

The Octave Fuzzy Logic Toolkit [4] is a library built on top of the Octave computing environment, which shares similarities to Matlab. It provides the same functionality as the Matlab Toolbox, and provides eight additional fuzzy logic operators. Octave and its toolkit are free and open source released under the GNU GPL license. The toolkit was last updated in 2012.

The jFuzzyLogic [6] library is built using the Java programming language. It supports Mamdani controllers and bundles 14 linguistic terms, 13 fuzzy logic operators, five defuzzifiers, and FLCs can be imported and exported utilizing the Fuzzy Control Language (FCL) format. It does not support hedges, it relies on third-party libraries, and the source code mixes graphical elements with the logic of the controller. It is free and open source released under the Apache or GNU LGPL license. The library was last updated in 2012.

III. Design of a Fuzzy Logic Controller

The design of an FLC consists of modeling the system inputs and outputs as linguistic variables, and creating the necessary inference rules that will control the system.

A. Linguistic Variables

A linguistic variable [7] is a set of terms expressed in natural language that can represent the possible values that a system variable can take. For example, the system variable Ambient, is a linguistic variable whose values are in the range $[0, 255]$ and are covered with overlapping terms of equal range defined as *dark* $[0, 127]$, *medium* $[64, 191]$, and *bright* $[128, 255]$.

The set of terms of a linguistic variable is a *crisp set* when each value of the system variable belongs with certainty to one or more of its linguistic terms. For example, a photosensor reading $x = 95$ is dark and medium because it is included within the range of both terms in Ambient, but it is not bright.

The set of terms of a linguistic variable is a *fuzzy set* when each value of the system variable belongs with degrees of certainty to one or more of its linguistic terms according to their respective *membership functions*. For example, modeling the system variable Ambient as shown in Figure 1, where $\mu(x)$ is the degree of certainty of value x, then $x = 84$ gives the fuzzy set $\tilde{x} = 0.68/\text{dark} + 0.32/\text{medium} + 0.0/\text{bright}$. Thus, a crisp set is a special case of a fuzzy set where $\mu(x) \in \{0, 1\}$.

B. Inference Rules

The inference rules are conditional statements that control the system. Each rule consists of an antecedent and a consequent, each of which comprises propositions in the form "variable is term". The propositions in the antecedent can be connected

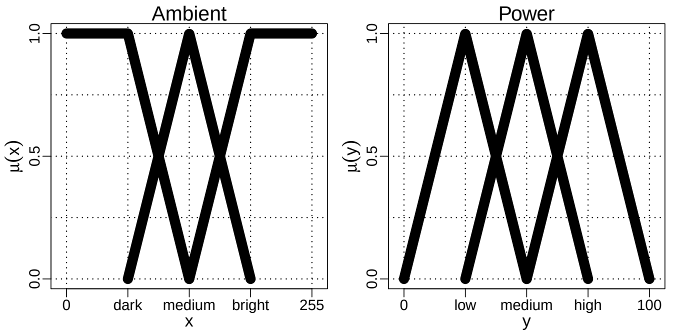

Figure 1. Example of linguistic variables.

by the conjunctive *and* or the disjunctive *or*, both of which are fuzzy logic operators. Differently, the propositions in the consequent are independent from each other and can be separated with a comma or just a symbolic *and*. The term in any proposition can be preceded by a *hedge* that modifies its membership function to model cases such as *very*, *somewhat*, *seldom* and *not*. Additionally, the importance of a rule can be determined by a weight $w \in [0.0, 1.0]$, which is equal to 1.0 if omitted.

The structure of the antecedent of a rule is "**if** variable **is** *[hedge]** term *[***and**/**or** *variable* **is** *[hedge]* term]**". The structure of the consequent is "**then** variable **is** *[hedge]** term *[***and** *variable* **is** *[hedge]* term]* [***with** $w]^?$". In these structures, elements in bold are keywords, within brackets are optional, *-marked may appear zero or more times, and ?-marked may appear once or not at all.

For example, utilizing the linguistic variables defined in Figure 1, the inference rules to control a light dimmer are the following:

- **if** Ambient is dark **then** Power is high
- **if** Ambient is medium **then** Power is medium
- **if** Ambient is bright **then** Power is low

C. Fuzzy Logic Operators

A fuzzy logic operator defines an operation between two values. It can be a *T-Norm* when it models conjunction, or an *S-Norm* when it models disjunction. Examples of T-Norms are the *minimum* and the *algebraic product* between two values, whereas examples of S-Norms are the *maximum* and the *algebraic sum*.

D. Configuration

The FLC described thus far is known as a *Mamdani* controller [8]. Its configuration consists of four fuzzy logic operators and a defuzzifier as follows. The keywords *and* and *or* that connect the propositions in the antecedent of the rules are defined by a T-Norm and an S-Norm, respectively. The consequents of the rules are modified by the *activation operator*, which is a T-Norm. The fuzzy outputs of the rules are accumulated by an *accumulation operator*, which is an S-Norm. Finally, the *defuzzifier* is a method to convert each of the accumulated fuzzy outputs into crisp values. The typical defuzzifier is the *centroid*, which computes a crisp value from the center of mass of the fuzzy set.

Takagi-Sugeno [9] is another well-known FLC. It models the output variables with a set of functions instead of Mamdani's fuzzy sets. Thus, a term is no longer defined by a membership function, but instead by a polynomial function in the form $f(x, y) = ax + by + c$, where x and y are the system inputs, a and b their respective coefficients, and c a constant. The *order* of the controller is given by the maximum degree of the polynomials. For example, a system variable Power, defined as *low* $= 25$, *medium* $= 50$ and *high* $= 75$, has coefficients $a = b = 0$, and hence is a *zero-order* Takagi-Sugeno FLC.

The configuration of a Takagi-Sugeno FLC consist of three fuzzy logic operators and a different kind of defuzzifier. The operators are the same as the first three in Mamdani, that is, a T-Norm and an S-Norm to define the keywords *and* and *or*, and another T-Norm to define the *activation operator*. The *defuzzifier* is different from Mamdani's because the outputs are no longer fuzzy, but are instead values and weights given by their respective functions and activation degrees. The typical defuzzifier is the *weighted average*. For convenience, hereinafter we refer to polynomial functions as membership functions, and to these outputs as fuzzy outputs.

Tsukamoto [10] is a less known FLC based on Mamdani, but configured as a Takagi-Sugeno FLC.

IV. Operation of a Fuzzy Logic Controller

The operation of an FLC at time t consists of three sequential steps known as *fuzzification*, *inference* and *defuzzification* (see Figure 2). The fuzzification step converts the crisp values of the system inputs into fuzzy values. The inference step utilizes the fuzzy input values to activate the relevant fuzzy rules and generate the corresponding fuzzy output values. Lastly, the defuzzification step converts the fuzzy output values into crisp values.

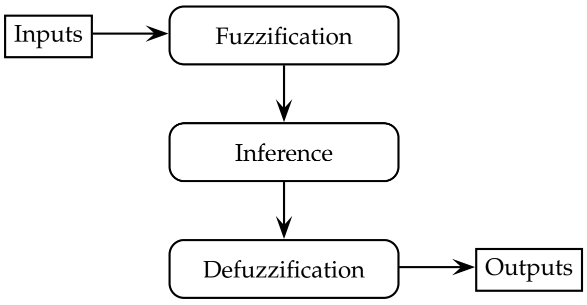

Figure 2. Stages of a Fuzzy Logic Controller.

A. Fuzzification

The fuzzification stage consists of computing the fuzzy values of the the linguistic variables given their respective system inputs. The fuzzy set $\tilde{x}_i$ for input variable i is defined as the membership function values $\mu_{ij}(x_i)$ for each term j on system input x_i. For example, given the photosensor reading $x = 168$, the fuzzy set for Ambient will be $\tilde{x} = 0.0/\text{dark} + 0.36/\text{medium} + 0.64/\text{bright}$.

B. Inference

The inference stage utilizes the fuzzy input values to trigger the inference rules and generate the fuzzy output values. For each rule, the propositions in the antecedent evaluate to the membership functions of their respective terms, all of which have been computed at the fuzzification stage. If the antecedent has more than one proposition, the T-Norms and S-Norms of the respective connectors are utilized to compute a single value from the different membership functions. The single value is then multiplied by the weight of the rule and it is referred to as the *activation degree*. The activation degree is utilized with the activation operator to modify the membership functions of the terms within the propositions of the consequent. In the case of a Mamdani FLC, the fuzzy accumulation operator is utilized to accumulate the modified membership functions from the terms of each output variable.

C. *Defuzzification*

The defuzzification stage consists of converting the fuzzy outputs from each variable into crisp values, which are computed with a *defuzzifier*. Many defuzzifiers have been suggested in the literature [11], but the most common ones are the *centroid* and *maxima* defuzzifiers for Mamdani controllers, and the *weighted average* and *weighted sum* for Takagi-Sugeno or Tsukamoto controllers. The centroid computes the x value of the centre of mass of the fuzzy set. A maximum defuzzifier returns the smallest, mean or largest x value for the maximum membership function. The weighted average and weighted sum are computed on the modified functions utilizing their activation degrees as weights. In the case of Tsukamoto, the defuzzifiers utilize the activation degrees as weights, and the membership functions of the activation degrees as values.

D. *Mamdani vs. Takagi-Sugeno*

The operation of the examples described thus far is described in Figure 3 as a relation between their inputs and outputs. While the relations in both cases seem similar, there are small differences between dark and bright as Takagi-Sugeno shows a straight line, whereas Mamdani shows subtle curves. However, the most important differences are in terms of interpretability, performance and information.

In terms of interpretability, the simpler model of the linguistic terms in Mamdani controllers makes them more easy to interpret and maintain over time than Takagi-Sugeno controllers. However, in terms of performance, Mamdani controllers are more computationally expensive because defuzzifiers generally need to integrate over the resulting fuzzy sets, whereas defuzzifiers in Takagi-Sugeno would consist only of a few arithmetic operations. Lastly, in terms of information, the design of a Takagi-Sugeno controller perhaps requires more information about the system in order to properly define the functions of the output variables.

V. A Fuzzy Logic Control Library

`fuzzylite` is a cross-platform, free open-source fuzzy logic control library programmed in C++ and released under the Apache License. Its goal is to provide the design and operation of FLCs with an

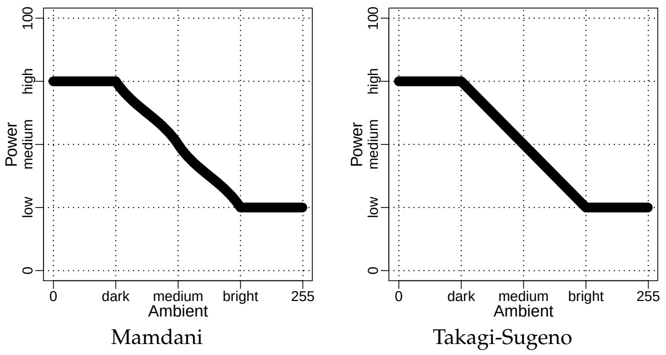

Figure 3. Inputs *vs.* outputs of examples.

object-oriented approach such that controllers can be incorporated into any application in just a few steps without requiring any third-party libraries. Additionally, it comes with an application named `qtfuzzylite` to visually design FLCs and interact with their operation in real time.

As of version 3.1, the following are the main features of `fuzzylite`.

- Controllers: Mamdani, Takagi-Sugeno and Tsukamoto
- Linguistic terms: those in Figure 4 and others
- T-Norms: minimum, algebraic product, bounded difference, drastic product, einstein product, hamacher product
- S-Norms: maximum, algebraic sum, bounded sum, normalized sum, drastic sum, einstein sum, hamacher sum
- Defuzzifiers: centroid, bisector, smallest of maximum, largest of maximum, mean of maximum, weighted average, weighted sum
- Hedges: any, not, extremely, seldom, somewhat, very
- Import and export controllers utilizing the FCL and FIS formats
- Extend and incorporate new linguistic terms, fuzzy logic operators, defuzzifiers and more.

The core classes of `fuzzylite` are shown in Figure 5, where the Engine encapsulates the design and operation of the FLC as described in the previous sections. For more details and examples of `fuzzylite`, please visit `http://www.fuzzylite.com`.

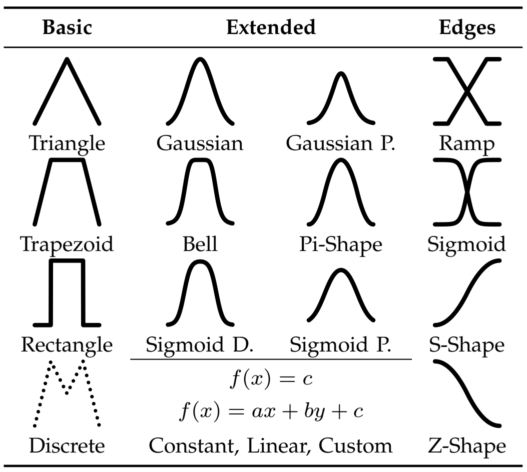

Figure 4. Linguistic terms available in `fuzzylite`.

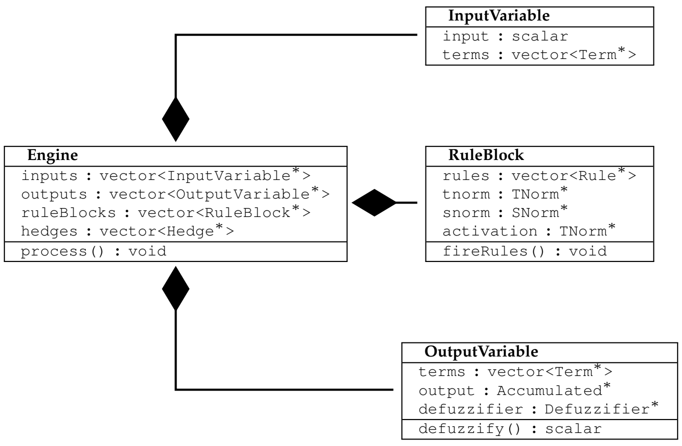

Figure 5. Abstract diagram of `fuzzylite`.

VI. Conclusions and Future Work

Fuzzy logic control provides a powerful alternative to traditional control algorithms. While the examples presented here may not leverage the whole power of FLCs because simple linear functions (such as that of the ramp term) could have achieved the same results, these examples were designed to be very simple in order to better explain the design and operation of an FLC.

Besides the examples, some of the concepts may have been oversimplified in order to provide an introduction to fuzzy logic control from the perspective of a developer. Nonetheless, it still provides an accurate description of the underlying design and operation of an FLC. Furthermore, the exact details can also be found in the source code of `fuzzylite`. For a more advanced introduction to FLCs, the reader is encouraged to refer to [1].

Future development in `fuzzylite` will focus on type-2 FLCs [12] to address uncertainty in the membership functions, the Adaptive Neuro-Fuzzy Inference System (ANFIS) [13] to automatically create and tune an FLC from a given dataset of inputs and outputs, and fuzzy C-Means clustering [14] to label the data as fuzzy clusters.

References

[1] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller — Parts I and II," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 2, pp. 404–435, 1990.

[2] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[3] Matlab, "Fuzzy logic toolbox," http://www.mathworks.com.au/products/fuzzy-logic/index.html, accessed on July, 2013.

[4] L. Markowsky and B. Segee, "The octave fuzzy logic toolkit," in *Proceedings of the International Workshop on Open-Source Software for Scientific Computation*, 2011, pp. 118–125.

[5] S. Rabin, *AI Game Programming Wisdom*. Rockland, USA: Charles River Media, Inc., 2002.

[6] P. Cingolani and J. Alcala-Fdez, "jfuzzylogic: a robust and flexible fuzzy-logic inference system language implementation," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, 2012, pp. 1–8.

[7] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning - Part I," *Information Sciences*, vol. 8, no. 3, pp. 199–249, 1975.

[8] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[9] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-15, no. 1, pp. 116–132, 1985.

[10] R. Shoureshi and Z. Hu, "Tsukamoto-type neural fuzzy inference network," in *Proceedings of the American Control Conference*, vol. 4, 2000, pp. 2463–2467.

[11] W. V. Leekwijck and E. E. Kerre, "Defuzzification: criteria and classification," *Fuzzy Sets and Systems*, vol. 108, no. 2, pp. 159–178, 1999.

[12] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 6, pp. 643–658, 1999.

[13] J.-S. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[14] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, USA: Kluwer Academic Publishers, 1981.