

DEVELOPMENT OF A SINGLE-VARIABLE FUZZY CONTROLLER IN PYTHON WITHOUT LIBRARIES

Ricardo Francisco Martínez-González

Tecnológico Nacional de Mexico – IT Veracruz, Veracruz, Veracruz, México

ABSTRACT

This paper describes the development of a single-variable fuzzy controller implemented entirely in Python, without relying on external libraries. The coding process was structured into four distinct stages, utilizing an object-oriented programming paradigm for defining membership functions. We developed custom membership functions for both input and output variables, along with methods to evaluate their respective degrees of membership. Inference rules were manually applied to determine the controller's output. Testing involved evaluating these membership functions for both input and output. A key result derived from this process is the cumulative value, which facilitates the calculation of the centroid of the minima of the output variable's functions, ultimately yielding the controller's output. The successful implementation of this fuzzy controller from scratch highlights its potential for future development into more complex controllers and encourages further exploration in this field.

KEYWORDS

Fuzzy Controller, Object-Oriented Programming, Python, Virus, Control System

1. INTRODUCTION

Fuzzy logic can be described as an interpretive system where input and output elements are related to sets with ill-defined boundaries, thus possessing a continuous degree of membership, unlike traditional binary logic [1]. One of the primary applications of fuzzy logic is in the design of control systems, which process one or more inputs to generate outputs that act on specific mechanisms [2]. This application is often very useful in the engineering world, since in this field, almost all theories that characterize the real world do so in an approximate manner, which is very useful for dealing with the uncertainty of real-world problems. These problems must process information with a certain degree of uncertainty, imprecision, ambiguity, or incomplete information [3].

In [4], Zadeh mentioned that classes of objects found in the real world often do not precisely follow the membership criterion; therefore, he established a fuzzy set as a class with a continuous degree of membership. This fact also laid the groundwork for managing operators that can work with fuzzy sets. Zadeh himself, in [5], established the foundations for the use of fuzzy sets in system control, describing what he called state equations for fuzzy systems. The foundational work of Zadeh on fuzzy sets has been widely applied in various fields, including decision-making and pattern recognition [6]. The ability of fuzzy logic to handle imprecise information makes it a powerful tool for complex systems where traditional exact mathematical models are difficult to derive [7].

Now, we will discuss the programming language used to implement the fuzzy controller. Python is a general-purpose, high-level language that has enjoyed significant popularity mainly due to three characteristics [8]:

- a) The code is automatically compiled and executed, making it suitable for use as a scripting language and in web applications.
- b) Python's ability to provide the high speed necessary for computationally intensive tasks, as it integrates C and C++, is a testament to its adaptability and robustness. This feature makes it suitable for implementing complex systems like fuzzy controllers, instilling confidence in the language's capabilities among the audience. Its extensive libraries for numerical computation, such as NumPy, further enhance its suitability for scientific and engineering applications [9].
- c) It features clear and logical writing for large and small applications.

Another great advantage of using Python lies in the large community of developers who create and support new development tools and the massive number of existing libraries for almost every task imaginable. This rich ecosystem significantly accelerates development time and allows for rapid prototyping [10].

2. PYTHON LIBRARIES FOR CREATING FUZZY CONTROLLERS

Among the existing options for creating fuzzy controllers using Python, libraries such as pyFUME, Fuzzylab, and Simpful stand out. In the case of the first library, [11] presents the usage of pyFUME as a Python package that provides a set of classes for defining fuzzy models from the data provided. pyFUME contains functions for estimating the antecedent sets and consequent parameters of a Takagi-Sugeno fuzzy model.

Eduardo Avelar et al. [12] identified the need to create a library that could adapt to the specific needs of a controller project for mobile robot navigation. This highlights the ongoing demand for specialized tools tailored to particular robotic applications [13].

Finally, the Simpful library allows for rapid and efficient configuration of fuzzy sets and inference rules, facilitating the creation of Mamdani- and Sugeno-type models with minimal code [14].

As discussed in this section, libraries are attractive because they facilitate the reconstruction and reuse of codes for subsequent projects. Therefore, it was decided to take the first steps toward generating a custom tool capable of determining the response of a controller with one input variable and one output variable using triangular membership functions.

3. CODE DEVELOPMENT

The coding process was structured into four distinct stages, each serving a specific purpose and providing the necessary level of abstraction to make each stage simple, both in writing and in verifying. These stages are: Stage 1: Membership Function Definition, Stage 2: Inference Rule Application, Stage 3: Testing and Evaluation, and Stage 4: Output Determination. Figure 1 shows a diagram of the four stages that make up the controller code.

For the fuzzy controller's coding and to ensure the code had the potential to be reusable, the "object" paradigm was chosen for each of the membership functions. The "membership" objects will have two methods: one constructor, which establishes the shape of the membership function and its parameters. The second method consists of an evaluation, where a numeric value is entered and returns the degree of membership in that function. This method will be widely used to simplify the evaluation and centroid determination stages.

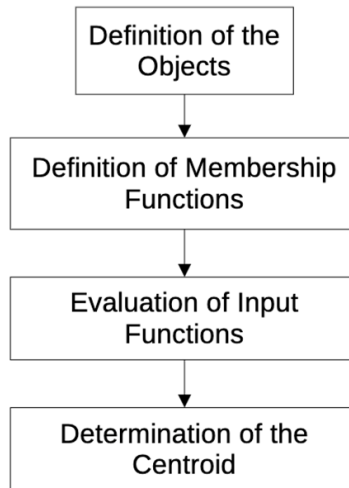


Figure 1. Stages of the code for the developed fuzzy controller.

Only the triangular membership function is available for code simplification purposes, although more shapes can easily be included, since one of the object's attributes is its "shape." Below is Code 1, which defines the "mf" class and the methods described above.

Code 1. Definition of the mf class.

```

class mf:
    def __init__(self, tipo, vertices):
        self.tipo = tipo
        self.axis = vertices

    def eval(self, entrada):
        if self.tipo == 'triangular':
            if entrada >= 0 and entrada < self.axis[0]:
                self.salida = 0
            elif entrada >= self.axis[0] and entrada < self.axis[1]:
                self.salida = (entrada - self.axis[0]) / (self.axis[1] - self.axis[0])
            elif entrada >= self.axis[1] and entrada < self.axis[2]:
                self.salida = 1 - ((entrada - self.axis[1]) / (self.axis[2] - self.axis[1]))
            elif entrada >= self.axis[2] and entrada <= 1:
                self.salida = 0
        
```

3.1. Coding the Core of the Developed Fuzzy Controller

First, the membership functions are created for the input variable. For this application, three triangular functions are used for the input and another three for the output variable; however, since these are objects, more functions are possible.

As explained in the central part of this section, the membership function is initialized with a type and values defined for the desired application. Another point worth noting is that both the functions for the input and output variables are treated as the same class; therefore, the only way to differentiate them is by their name. Code 2 shows how the objects for the membership functions for the controller's input and output variables were defined; the objects are f11, f12, and f13 for the input, and f21, f22, and f23 for the output.

Code 2. Creating the Objects for the Membership Functions.

```
# Funciones de entrada
f11=mf('triangular',[0, 0, 0.3])
f12=mf('triangular',[0.2, 0.5, 0.8])
f13=mf('triangular',[0.7, 1, 1])

# Funciones de salida
f21=mf('triangular',[0, 0, 0.3])
f22=mf('triangular',[0.3, 0.5, 0.7])
f23=mf('triangular',[0.7, 1, 1])
```

After creating the objects, it is necessary to evaluate the input variable to obtain its membership level in each membership function. The "eval" method was made to do this. This method stores the membership level, which has a numerical value in the membership function, in the "output" attribute. In the case of Code 3, a sweep is performed on the input value because the control function is being determined, and it is necessary to decide on the controller's response to various input values.

Code 3. Evaluating the Functions for Input and Output Variables.

```
step=1000; entrada=[1/step]; res=[];

for b in range(step):
    #Evaluación de la fm de entrada
    f11.eval(entrada[b])
    f12.eval(entrada[b])
    f13.eval(entrada[b])

    iter=100; x=[1/iter]; aux1=[]; aux2=0;
    for a in range(iter):
        x.append(x[a]+1/iter)

        f21.eval(x[a])
        f22.eval(x[a])
        f23.eval(x[a])
```

This is also a sweep in the case of output functions; it aims to find all the values belonging to those functions, which will then be evaluated following the inference rules. However, since the goal is to find the entire control function, performing a sweep on all output functions for each step in the sweep of the input functions is necessary. To achieve this, "for" loops were used, covering each sweep's full range.

The next part of the code applies the inference rules. At this point, the rules are applied manually, but for this controller, they are as follows:

- if f13 then f21;
- if f12 then f22;
- if f11 then f23;

Finally, Code 4 performs the set operations based on the rules mentioned earlier. The result is then accumulated to determine the centroid. Half of the accumulated value is calculated for the centroid. With a final cycle, the accumulation is performed again, and the iteration where it exceeds half of the previous accumulator is found. This is the value that will be delivered as the centroid and output of the controller. These operations are performed for each step in the sweep of the input variable.

Code 4. Application of Fuzzy Controller Operation Rules and Centroid Determination

```
# Depende de las reglas
aux1.append(max(min(f21.salida,f13.salida),min(f22.salida,f12.salida),
                min(f23.salida,f11.salida)))
aux2=aux2+aux1[a] #Acumulador para centroide

aux3=aux2/2
aux2=0
# Determinación del centroide
for a in range(iter):
    aux2=aux2+aux1[a]
    if aux2 >= aux3:
        res.append(x[a])
        break
```

4. SIMULATION TESTS

The first test performed was for the membership functions for the input variable. Since these were conducted using objects, after evaluation, it is necessary to "collect" the result of each function in the "output" attribute of its respective object. Thus, tests were performed using an input of 0.7 and with functions with the parameters described in Table 1.

Table 1. Presentation of the objects corresponding to the membership functions for the input variable.

Function Type	Parameters
Triangular	[0, 0, 0.3]
Trapezoidal	[0.1, 0.4, 0.6, 0.9]
Triangular	[0.5, 1, 1]

Given the above data, the following results were obtained, sorted for each object in Table 1: [0, 0.6666667, 0.4]. The second code test was performed on the functions corresponding to the output variable, which was entered with the following values [0, 0.5, 0.8] and the configuration presented in Table 2.

Table 2. Presentation of the objects corresponding to the membership functions for the output variable.

Function type	Parameters
Trapezoidal	[0, 0, 0.1, 0.5]
Triangular	[0.1, 0.5, 0.9]
Trapezoidal	[0.5, 0.9, 1, 1]

With this input and configuration, the code presents two outputs: the first graph in Figure 2 corresponds to the evaluation of the minima of each function and its respective inputs, while the

second graph corresponds to the cumulative value of each function and will assist in determining the centroid.

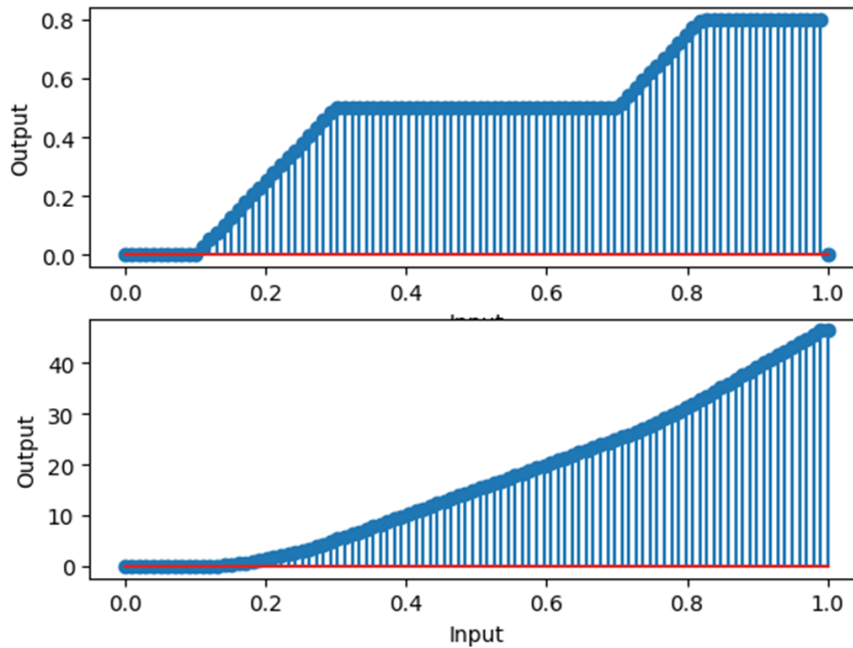


Figure 2. (top) Minima of the membership functions and their inputs. (bottom) Cumulative of the minima.

Using the cumulative values, the centroid of the minima can be found, and in this case, it is set to 0.6565. The centroid value is the controller's final output. However, to evaluate the code's complete behavior, a new controller was created with the parameters described in Table 3.

Table 3. Parameters for the controller developed as a final test of the proposed code.

Functions for the input variable		Functions for the output variable	
Function type	Parameters	Function type	Parameters
Triangular	[0, 0, 0.3]	Triangular	[0, 0, 0.3]
Triangular	[0.2, 0.5, 0.8]	Triangular	[0.3, 0.5, 0.7]
Triangular	[0.7, 1, 1]	Triangular	[0.7, 1, 1]

With these parameters, a sweep of different input values was performed, and a control curve shape was found to explain the proposed controller's behavior. This would help make its implementation easier, mainly if it is intended to be implemented on devices with reduced capabilities or in processes where processing speed could be vital. The control curve is presented in Figure 3.

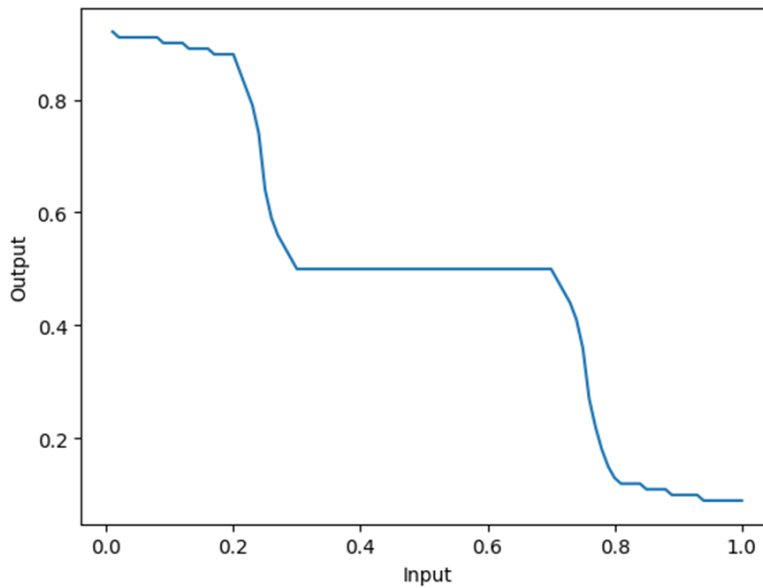


Figure 3. Control curve for the fuzzy controller with the parameters described in Table 3.

5. CONCLUSIONS

This work successfully demonstrated the feasibility of developing a single-variable fuzzy controller entirely from scratch in Python, without reliance on pre-existing libraries. This foundational development, specifically tailored for one input and one output variable with triangular membership functions and simple inference rules, proves to be a robust and highly adaptable tool. Its minimal computational overhead makes it particularly well-suited for implementation on devices with limited processing capabilities, where efficiency and execution speed are paramount. Such resource-constrained environments often benefit significantly from custom, lightweight solutions that avoid the overhead of comprehensive external libraries.

The object-oriented design employed for the membership functions ensures that the code is not only clean and maintainable but also highly reusable and extensible. This modularity is a critical advantage, as it allows for straightforward expansion to accommodate more complex scenarios. This project represents a pivotal first step towards the creation of more sophisticated multivariable fuzzy controllers capable of handling a broader range of inputs and outputs, as well as incorporating more intricate and adaptive rule sets. Future research can build upon this foundation to explore various defuzzification methods beyond the centroid, integrate learning algorithms for automated rule generation, and apply this controller to diverse real-world applications in areas such as embedded systems, robotics, and industrial automation where precise control under uncertainty is required. The successful execution of this "from scratch" approach opens promising avenues for developing tailored fuzzy logic solutions for specific control challenges.

REFERENCES

- [1] Kouro, S., & Musalem, R. (2002). Control mediante lógica difusa. *Técnicas Modernas en Automática*, 1, 1-7.
- [2] Kassir, E. E. (2015). *Sistemas De Control Difuso*. Eslava Zuluaga, AF.
- [3] Tremante, P., & Brea, E. (2014). Una visión de la teoría difusa y los sistemas difusos enfocados al control difuso. *Ingeniería Industrial. Actualidad y Nuevas Tendencias*, 4(12), 121-136.

- [4] Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3), 338-353.
- [5] Zadeh, L. A. (1969). *Toward a theory of fuzzy systems* (No. NASA-CR-1432). NASA.
- [6] Zadeh, L. A. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(1), 28-44.
- [7] Klir, G. J., & Yuan, B. (1995). *Fuzzy sets and fuzzy logic: Theory and applications*. Prentice Hall.
- [8] Kuhlman, D. (n.d.). *A Python Book: Beginning Python, Advanced Python, and Python Exercises*. Archived from the original (PDF) on June 23, 2012.
- [9] Van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). NumPy: Array processing with Python. *Computing in Science & Engineering*, 13(2), 22-34.
- [10] Oliphant, T. E. (2007). *Python for scientific computing*. *Computing in Science & Engineering*, 9(3), 10-20.
- [11] Fuchs, C., Spolaor, S., Nobile, M. S., & Kaymak, U. (2020, July). pyFUME: a Python package for fuzzy model estimation. In *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (pp. 1-8). IEEE.
- [12] Avelar, E., Castillo, O., & Soria, J. (2020). Fuzzy logic controller with fuzzylab python library and the robot operating system for autonomous robot navigation: a practical approach. In *Intuitionistic and Type-2 Fuzzy Logic Enhancements in Neural and Optimization Algorithms: Theory and Applications* (pp. 355-369). Cham: Springer International Publishing.
- [13] Kumar, G. K. S., & Prasad, R. M. (2018). Design and implementation of fuzzy logic controller for mobile robot navigation. *International Journal of Pure and Applied Mathematics*, 118(2), 527-536.
- [14] Chicco, D., Spolaor, S., & Nobile, M. S. (2023). Ten quick tips for fuzzy logic modeling of biomedical systems. *PLoS Computational Biology*, 19(12), e1011700.

AUTHOR

Ricardo Francisco Martínez-González , He was born in Veracruz, Veracruz, Mexico, on January 2, 1983. He earned his Bachelor's degree in Electronic Engineering from the Instituto Tecnológico de Veracruz. He pursued his Master's and Ph.D. degrees in Science, specializing in Electronics, at the Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE). He has worked as a software platform and mobile application developer for private companies. Currently, he works at the Tecnológico Nacional de México campus Veracruz, in the Department of Electrical-Electronic Engineering, where he has directed various research projects in areas such as artificial vision, control, instrumentation, artificial intelligence, steganography, energy efficiency, among others. He is the author of several scientific articles in the aforementioned areas.

