The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM)
ISSN: 2602-3199

The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM), 2023

Volume 24, Pages 110-118

IConTech 2023: International Conference on Technology

User-Defined Autogenerated Fuzzy Solvers for Embedded Applications

Alexei Evgenievich Vassiliev
Saint Petersburg State Marine Technical University

Ye Min Htet
Saint Petersburg State Marine Technical University

Htut Shine
Saint Petersburg State Marine Technical University

Anton Victorovich Vegner
Saint Petersburg State Marine Technical University

Abstract: A characteristic modern trend in control systems is the use of artificial intelligence methods (including fuzzy methods) in embedded applications. The presence of limitations in the resources of such systems leads to the need to choose the best combination of fuzzifier, inference system, and defuzzifier, which requires appropriate tools. In this paper, based on the principles of fuzzy logic and fuzzy sets, we propose an integrated development environment (IDE) that generates assembly codes for microcontrollers, which will be used as fuzzy controllers in embedded applications. The user of this IDE will have the option to choose the desired fuzzifier, inference system, and defuzzifier based on the established indicators of accuracy, speed, and the amount of required memory. Our interest is to improve the efficiency of developing systems with fuzzy data processing by creating specialized integrated development environments. An example of the practical implementation of such a tool is given in this article.

Keywords: Fuzzy controller, Intelligent control system, On-board microcontroller, Embedded application.

Introduction

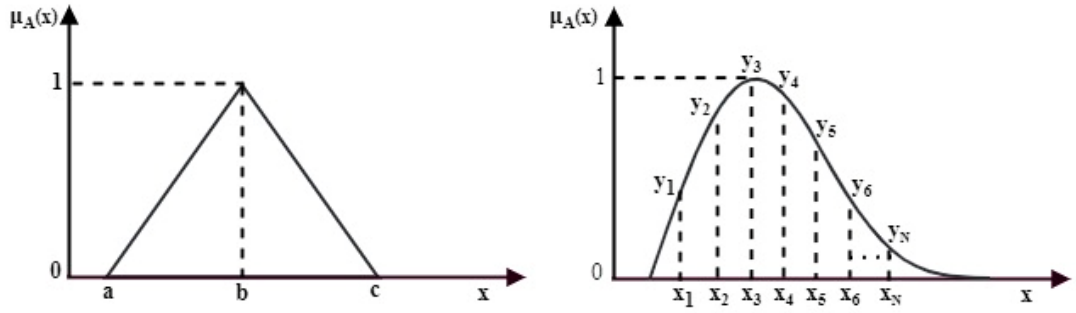
Decision-making systems based on the methods of fuzzy computing theory are widely used in control systems, including embedded control systems (Piegat, 2013). The fuzzy computing paradigm is based on the concepts of the intensity of manifestation of sets of properties of objects (processes) and the degree of validity of statements operating with these sets. When designing a fuzzy solver, the developer specifies sets of values (“memberships”) of input signals X, sets of values (“memberships”) of output signals Y, as well as the statements (“decision rules”) of the form “If (Set of X) Then (Set of Y) ". The fuzzy solver consists of three subsystems: the fuzzification subsystem - "fuzzifier" (for each input signal, by its instantaneous value, forms the intensities of its correspondence to each of the given input memberships), the inference subsystem - "solver" (for each rule, according to the values of instantaneous intensity of its input memberships forms the intensity of manifestation of the output memberships), and the defuzzification subsystem - “defuzzifier” (it calculates the resulting value of each output signal from the intensity of output memberships).

Component Models of Fuzzy Solvers and Methods of Their Implementation

First of all, it should be noted that the mathematical apparatus of the theory of fuzzy sets is quite diverse: there are various ways of defining membership functions (singleton, linear, nonlinear), various methods of processing complex conditions, various types of logical operations (Ramot et al., 2003; Vassiliev, 2018; Van Leekwijck & Kerre, 1999; Šaletić et al., 2002). The listed elements of the mathematical apparatus have different degrees of complexity, require different amounts of resources for their implementation, and at the same time provide a different degree of adequacy of the fuzzy models being formed. Estimates of the "quality to resource capacity" ratios are known for some implementations of fuzzy systems, however, the theory of formal synthesis of optimal fuzzy solvers is still in its infancy (Ghosh & Dubey, 2013; Kim et al., 2000). Based on the principle of information processing stages in fuzzy solvers, we will give a brief overview of common ways to describe the three subsystems listed above that are part of a fuzzy solver.

Fuzzification Subsystem

Membership functions of input variables can be described by piecewise linear or non-linear functions (Fig. 2).



(a) (b)

Figure 1. Membership functions of input variables: (a) piecewise linear and (b) nonlinear

The first type of description makes it possible to efficiently use the memory of a fuzzy solver, since three points are enough to store information about a piecewise linear term, however, to determine the degree of membership of the input variable (Fig. 1(a)), it is necessary to perform calculations according to triangular membership function formula (1).

$$\mu_A(x) = \begin{cases} 0, & if\ x \le a \\ \dfrac{x-a}{b-a}, & if\ a \le x \le b \\ \dfrac{c-x}{c-b}, & if\ b \le x \le c \\ 0, & if\ x \ge c \end{cases} \quad (1)$$

In addition, in the case of fuzzification of piecewise linear description, the results of fuzzy calculations will be approximate values. The second type of description allows one to describe an arbitrary form of a membership function (2) with the highest possible accuracy, but requires either memory costs for point-by-point storage of the scan of the membership function (in this case, the maximum fuzzification rate is ensured), or the calculation of an externally specified function that analytically describes the form of the membership function (in this case, additional time for calculations and memory is consumed to store and execute code of the function).

$$\mu_A(x) = \{(y_1, x_1), (y_2, x_2), (y_3, x_3), \dots, (y_N, x_N)\} \quad (2)$$

Inference Subsystem

The “If” parts of the rules can operate on several input variables, and the conditions for these variables to correspond to their membership functions can be combined by various logical expressions. So, for example, if the particular conditions of a certain rule are combined by the “OR” operation, the degree of membership of the entire rule is determined by the greatest degree of membership among these particular conditions, and if by the "AND" operation, then the least. In addition, particular conditions can be hierarchically grouped among themselves utilizing logical operations (Fig. 2). A variety of such operations provides an increase in the degree

of flexibility of the description, but increases the amount of memory required to store the rule base; in addition, when the description of the rule becomes more complicated, the time of its processing in the fuzzy solver increases.

1	IF (X1 is mf1) AND (X2 is mf1)	THEN (Y1 is mf1)
2	IF (X2 is mf1) AND (X1 is mf2 OR X3 is mf3)	THEN (Y2 is mf2) AND (Y3 is mf3)
.		
.		
.		
N	IF (X1 is mf3 AND X2 is mf2) OR (X3 is mf3 AND X4 is mf4)	THEN (Y3 is mf2) AND (Y4 is mf3)

Figure 2. Example of rule base for a fuzzy solver

Defuzzification Subsystem

Defuzzification is a procedure or process of finding the crisp (non-fuzzy) value for each of the output linguistic variables of the set (Driankov et al., 1993). The procedure that extracts crisp output value from a fuzzy output set is called defuzzification (Grum, 2008). Currently, there are a significant number of defuzzification methods, as noted in Van Leekwijck& Kerre (1999). Among them, the simplest way to perform the defuzzification procedure is to choose the corresponding number of a maximum output membership function. Variants of this method are FOM (first of maximum), MOM (middle of maximum), and LOM (last of maximum) which are called maxima methods. However, regardless of their computational simplification, they generally ignore the rules that are triggered below the maximum level of the membership, and hence, this makes their results less accurate (Mamdani et al., 1984). As such, the center of gravity (COG) and the weighted average method (WAM) have been mostly used to come up with crisp controller outputs. These methods give more accurate results than maxima methods, but their computability is much more complex.

Let us illustrate with an example of a fuzzy approximation function to see clearly how the results will be obtained according to the defuzzification methods they used. Here we will use the cross-section method for the fuzzy solver to approximate the function (Vassiliev, 2018). Let's consider the application of the cross-section method in the example of solving the problem of fuzzy approximation of the nonlinear function below.

$$F(x1, x2) = 128 + \left[cos\left[\frac{(x1 - 127.5) \cdot \pi}{128}\right] + sin\left[\frac{(x2 - 127.5) \cdot \pi}{128}\right]\right] \cdot 32 \quad (3)$$

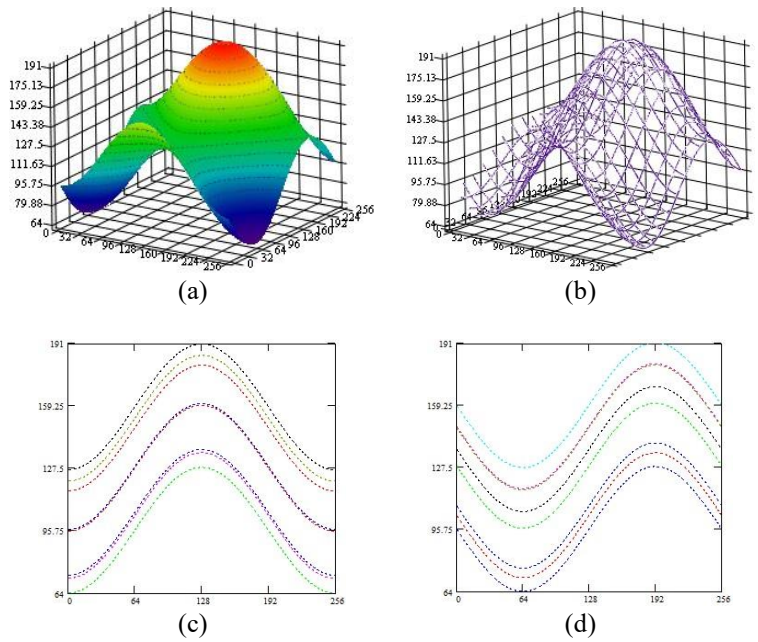


Figure 3. The approximated surface (a, b), and the projections of its sections on the plane x1-z (c) and on the plane x2-z (d)

Analysis of a given functional dependence (in some cases, for example, when conducting heuristic analysis, it is convenient to use a graphical representation of the dependence, rather than its mathematical notation) allows us to conclude that the desired graph is a cosine wave function (the argument of which is the variable x1) experiencing a vertical shift (the intensity of this shift depends according to the sinusoidal law on the variable x2) — see Figure 3.

According to the cross-section method, sections of the original function are constructed parallel to the planes that limit the space of its permissible values (in this case 0 to 256). Each new shape of the section is described by a pair of membership functions (direct and inverse; in general, the membership functions of these inputs will be nonlinear. In this way, we get a pair of membership functions for each input: membership functions 'cos' and 'not_cos' for input variable x1 and membership functions 'sin' and 'not_sin' for input variable x2 (Fig. 4).

In our case, the value of the cosine function decreases to a minimum value as it approaches the center of the first half-period and increases to a maximum value as it approaches the center of the second half-period (passing through the median value at the edges and in the center of the interval on which it is defined).

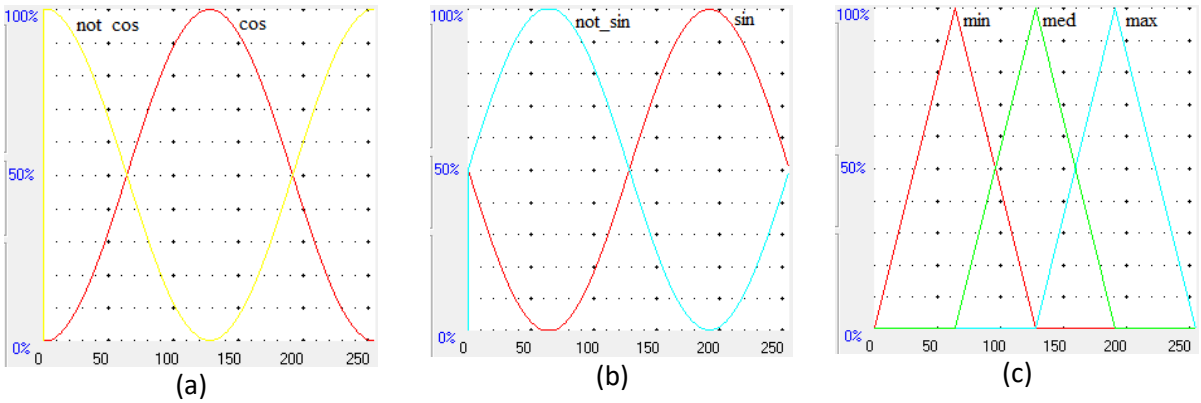


Figure 4. Membership functions of (a) input variable x1, (b) input variable x2, and (c) output variable y

So, the output value varies from minimum to maximum according to the sinusoidal law (depending on the measure of proximity of the current value of the variable x1 to the edges and the center of its interval), but in two different ranges (depending on the measure of proximity of the current value of the variable x2). This allows us to set the membership functions of the output variable and link them by rules with functions describing changes in input variables.

1	IF (x1 is cos) AND (x2 is sin)	THEN (y is max)
2	IF (x1 is cos) AND (x2 is not_sin)	THEN (y is med)
3	IF (x1 is not_cos) AND (x2 is sin)	THEN (y is med)
4	IF (x1 is not_cos) AND (x2 is not_sin)	THEN (y is min)

Figure 5. Rule base for fuzzy approximation

After setting all membership functions and the inference rules, we then implement and debug this fuzzy inference system into assembly codes, load them into the microcontroller, and then, test and record their operational characteristics by using different defuzzification methods. Here we use an x51-compatible 8-bit microcontroller with 12 MHz crystal oscillator. Table 1 shows approximation errors of the results from the MATLAB fuzzy toolbox and microcontroller. By comparing the approximation errors of our fuzzy solver to MATLAB, it can be seen that the microcontroller (fuzzy solver) executes reliable and exact output values for the fuzzy approximation function.

And then let us compare some options for the user of fuzzy solver. Table 2 shows various results for various defuzzification methods (each of them possesses the same input-output membership functions and inference rules) and clearly, we can see that each method has different accuracies, different execution times, and different memory sizes.

As we can see in Table 2, the maxima methods have very less computational time due to their simplicity of method of choosing crisp output values. However, approximation errors of these methods are significantly high

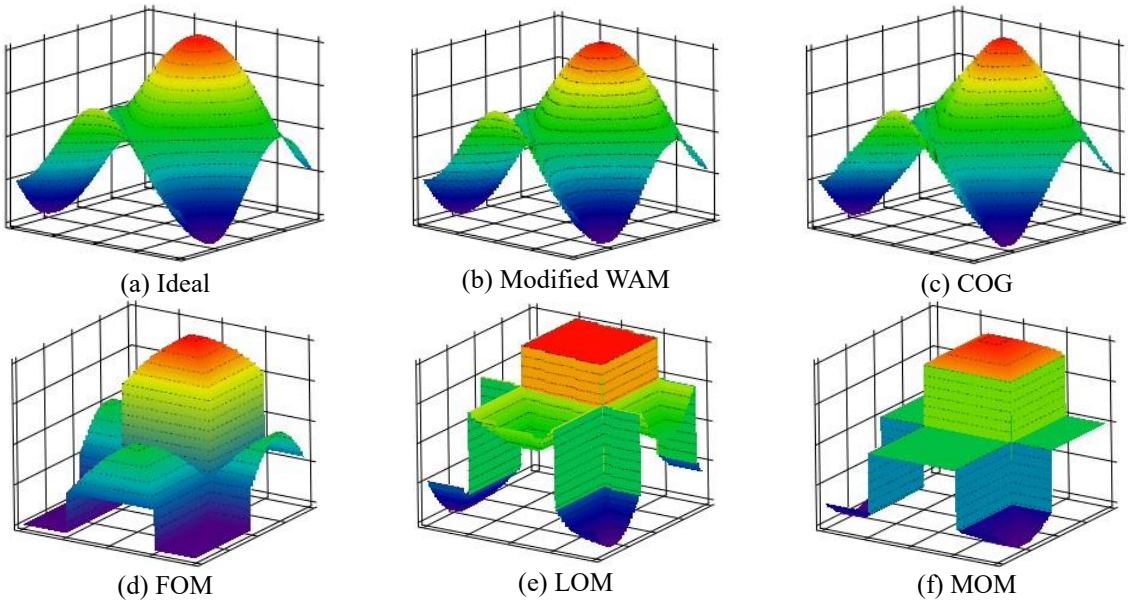
compared to the COG method. This is because they ignore rules that are triggered below the maximum level of membership function and hence, these methods are discontinuous because an arbitrarily small change in the input values of the fuzzy system can cause the output value to switch to another, more plausible result (Saade & Diab, 2004). This process can be seen in Figure 6 as their approximated surfaces are mostly discontinuous and suddenly change very roughly at certain points.

Table 1. Comparison of approximation errors with the ideal function

Defuzzification Methods	MATLAB			Microcontroller		
	Minimum Error	Maximum Error	Approximation Error (%)	Minimum Error	Maximum Error	Approximation Error (%)
FOM $(\min(x_m))$	-32.06	62.647	24.472 %	-29.86	63.547	24.823 %
LOM $(\max(x_m))$	-60.9	31.492	23.789 %	-61.827	32.419	24.152 %
MOM $\left(\frac{\sum x_i \in M^{(x_i)}}{\lvert M \rvert}\right)$	-44.81	44.537	17.504 %	-43.86	45.292	17.692 %
COG $\left(\frac{\sum(\mu(x_i)\cdot x_i)}{\sum \mu_i}\right)$	-9.382	11.9	4.648 %	-9.598	11.9	4.648 %

Table 2. Experimental results of fuzzy solver

Defuzzification Methods	Code Size	Execution Time	Total Machine Cycles	Approximation Error (%)
FOM $(\min(x_m))$	2262 bytes	1.7 ms/cycle	1700	24.823 %
LOM $(\max(x_m))$	2283 bytes	2 ms/cycle	2000	24.152 %
MOM $\left(\frac{\sum x_i \in M^{(x_i)}}{\lvert M \rvert}\right)$	2366 bytes	3.7 ms/cycle	3700	17.692 %
COG $\left(\frac{\sum(\mu(x_i)\cdot x_i)}{\sum \mu_i}\right)$	2678 bytes	2.7 ms/cycle	2700	4.648 %
Modified WAM $\frac{\sum(\prod_i^{k\in 1:M}\mu_k)Y_{max_i}}{\sum(\prod_i^{k\in 1:M}\mu_k)}$	2598 bytes	4.2 ms/cycle	4200	0.976 %



(a) Ideal (b) Modified WAM (c) COG

(d) FOM (e) LOM (f) MOM

Figure 6. Comparison of surfaces of the fuzzy approximated function

COG method takes into account all the rules, but it does not allow control actions in the extreme limits of the range of output signal. This is because the Center of Gravity (COG) defuzzification method evaluates the area

under the scaled membership functions only within the range of the output linguistic variable, the resulting crisp output values cannot cover the full range (Talon & Curt, 2017). As shown in Figure 6, the approximated surface of the COG method is very similar to the ideal one, since it gives continuous action, but the edges slightly differ which makes the approximation error 4.648 %. But, compared to maxima methods, COG has good accuracy and not much execution time.

MOM and COG are the most famous and frequently used defuzzification methods, but still, we see that they have some approximation errors compared to the ideal function, and, suppose the user is not satisfied with these errors and wants to increase the quality of accuracy. Then we consider the new defuzzification method. Vassiliev et al. (2023) proposed this method and this is a modified method of weighted average of maximums. According to this method, the weight of the next rule will be determined not by the minimum degree of validity of the conditions that make up the "IF" part, but by the product of all their values. Hence, software implementation of this method becomes more complex and it takes most execution time compared to others, but it increases the accuracy significantly which is 0.976 % approximation error. Another feature of this method is that it can only be used for symmetrical and singleton membership functions but not asymmetrical memberships. Again, the centroid method can be used for both symmetrical and asymmetrical membership functions, but not singletons. So, depending on the type of membership functions we choose to use, we need to choose a suitable defuzzification method accordingly.

Again, the code sizes of each method are not significantly different because the code size only depends on the algorithm of each method and, in this case, we use the same fuzzification method and the same inference rules. So, the code sizes of each method differ only by their defuzzification method. As mentioned above, here, input membership functions are nonlinear and are defined as values in look-up tables of assembly language programs. Since we use 256 values for one membership function, a single membership will take 256 bytes of memory. So, if we need a lot of membership functions and have some limits for the memory size of the microcontroller we chose, to implement the membership functions as piecewise linear will be the option for users.

Thus, since there is no single optimal combination of options for implementing the stages of fuzzy information processing, and the theory of synthesis of optimal fuzzy solvers (which could analytically substantiate the variant of such a combination for each specific task) is in the process of becoming, to expand the possibilities of developing fuzzy solvers and achieve suboptimal indicators of their quality, tools are needed that provide the developer with the opportunity to choose the method of implementing information processing for each stage of the fuzzy solver, and the combined use of these implementations.

Therefore, an urgent task is to develop methods and tools for the automated generation of fuzzy solvers with combinations of options for fuzzification, inference, and defuzzification subsystems specified by the developer, which guarantee the operability of the fuzzy solver and the adequacy of the a priori assessment of its performance characteristics.

Structure of the Instrumental System

Let's consider the variant of the structurally functional organization of the automated synthesis system for the software implementation of fuzzy solvers (Fig. 7) and the stages and steps of its application. The first stage (steps 1–4) ensures the formation and replenishment of libraries of ready-made software solutions for fuzzifiers, solvers, and defuzzifiers. The idea of each new prototype of a fuzzy computer component is developed and analyzed in the simulation environment (stage 1), then it is implemented and debugged in the form of program code for the microcontroller (step 2), loaded into the microcontroller (step 3) to test and record the values of real operational characteristics - accuracy, speed, and memory costs, after which it replenishes the corresponding library of software implementations (step 4).

The second stage (steps 5–7) provides automated generation of a software implementation of a fuzzy solver that meets the specified requirements for a specific control system. From the total set of requirements, those related to the fuzzy decision-making system are determined (step 5), from the triad of libraries of ready-made solutions of fuzzifiers, solvers, and defuzzifiers, solutions that satisfy the given constraints are selected, aggregated into a fuzzy solver and analyzed as a whole (step 6), after which they are combined with other software components to obtain the resulting software part of the control device (step 7).

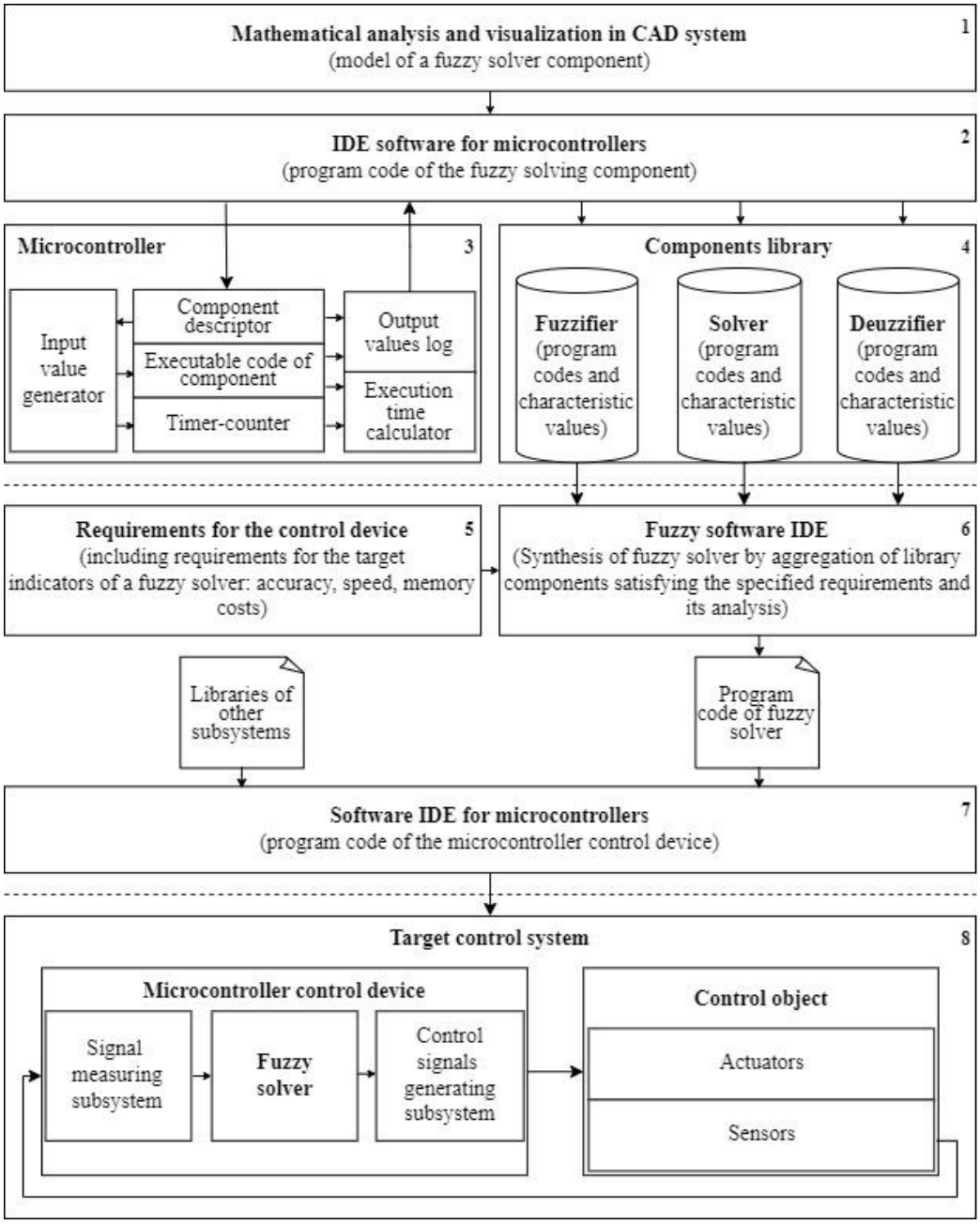


Figure 7. The process of designing control systems with auto-generated fuzzy solver

Third stage (stage 8) - the target microcontroller device with a fuzzy computer is put into operation, which serves as a source of new ideas for improving fuzzy computers.

A Variant of Implementation of the Instrumental System

For experimental purposes, the authors of the article have developed and continue to improve a prototype of an instrumental system for the automated generation of software implementations of fuzzy solvers with established performance characteristics – FuzzyWizard-51 (Fig. 8). The user of the system - the developer of fuzzy solvers - can declare libraries of ready-made solutions and perform the necessary manipulations with them (add, exclude, and modify program modules and their descriptions), make the necessary settings for the tool environment, and at the stage of generating the program code of the target fuzzy solver - set the required restrictions on the execution time, the amount of code and data for each of the three types of modules, and select them according to the selected criteria. After analyzing the found library modules that satisfy the constraints of interest to the developer, the automatic assembly of the program code of the resulting fuzzy solver is carried out.

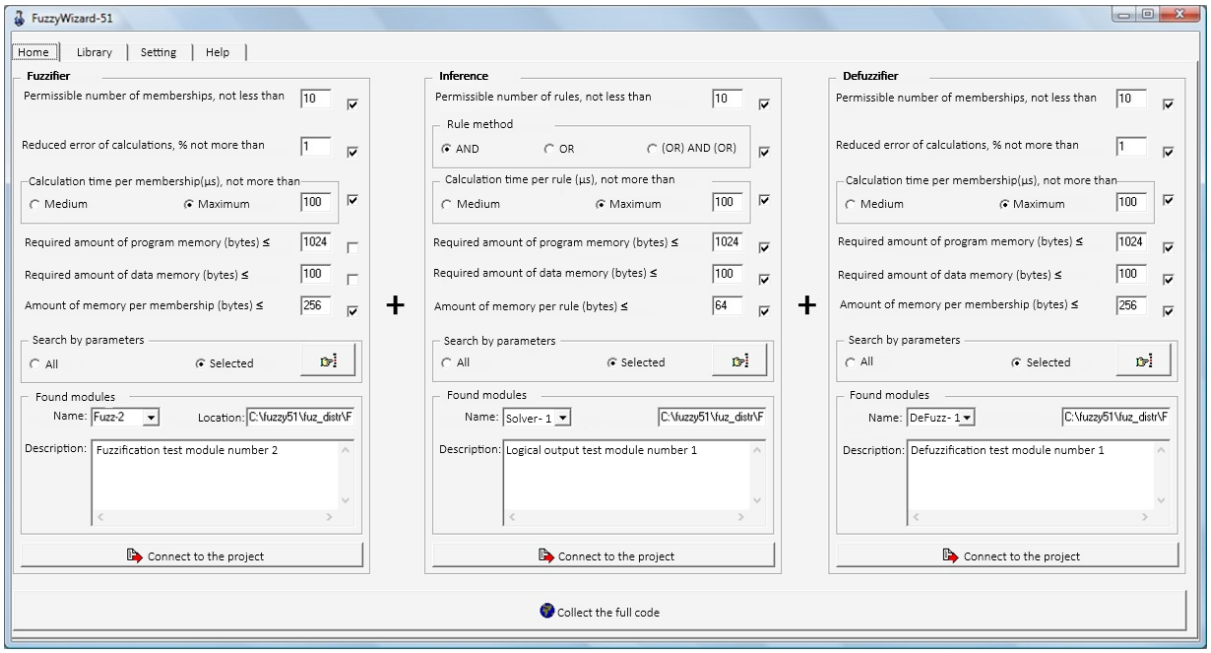


Figure 8. Appearance of the main window of Fuzzy Wizard-51

Conclusion

Thus, the methods and means of automating the design of fuzzy solvers with specified performance characteristics proposed by the authors contribute to the expansion of the methodological and technological base of instrumental support for intelligent decision-making and control systems based on applications of fuzzy set theory.

Scientific Ethics Declaration

The authors declare that the scientific ethical and legal responsibility of this article published in EPSTEM journal belongs to authors.

Acknowledgements or Notes

* This article was presented as an oral presentation at the International Conference on Technology (www.icontechno.net) held in Antalya/Turkey on November 16-19, 2023.

References

Driankov, D., Hellendoorn, H., & Reinfrank, M. (1993). *An introduction to fuzzy control*. Springer eBooks.

Ghosh, S., & Dubey, S. K. (2013). Comparative analysis of K-Means and fuzzy C-Means algorithms. *International Journal of Advanced Computer Science and Applications, 4*(4).

Grum, J. (2008). Book review: Fuzzy controller design, theory and applications by Z. Kovacic and S. Bogdan. *International Journal of Microstructure and Materials Properties*, *3*(2-3), *465.*

Kim, Y. H., Ahn, S. C., & Kwon, W. H. (2000). Computational complexity of general fuzzy logic control and its simplification for a loop controller. *Fuzzy Sets and Systems, 111*(2), 215–224.

Mamdani, E. H., Efstathiou, H., & Sugiyama, K. (1984). Developments in fuzzy logic control. *23rd Conf. On Decision and Control*, *7*(1), 1-13.

Piegat, A. (2013). *Fuzzy modeling and control.* Physica.

Ramot, D., Friedman, M., Langholz, G., & Kandel, A. (2003). Complex fuzzy logic. *IEEE Transactions on Fuzzy Systems*, *11*(4), 450–461.

Saade, J. J., & Diab, H. (2004). Defuzzification methods and new techniques for fuzzy controllers. *Iranian Journal of Electrical and Computer Engineering*, *3*(2), 161–174.

Šaletić, D. Z., Saletic, Velasevic, D., & Mastorakis, N. E. (2002). Analysis of basic defuzzification techniques. Retrieved from https://www.researchgate.net/publication

Talon, A., & Curt, C. (2017). Selection of appropriate defuzzification methods: Application to the assessment of dam performance. *Expert Systems with Applications, 70,* 160–174.

Van Leekwijck, W., & Kerre, E. (1999). Defuzzification: Criteria and classification. *Fuzzy Sets and Systems, 108*(2), 159–178.

Vasiliev A.E. (2018). *Embedded automation and computer technology systems* (p.590). Microcontrollers. – M.: Hotline-Telecom.

Vassiliev, A.E, Vegner, A. V., Golubeva, D. E., Dotsenko, A. S., & Карпенко, В. А. (2023). Increasing the quality indicators of the functioning of fuzzy solvers at the defuzzification stage. *Journal of Communications Technology and Electronics*, *68*(7), 810–818.

Author Information

Alexei Evgenievich Vassiliev
Saint Petersburg State Marine Technical University
Saint Petersburg, Russia
Contact e-mail: avasil@*smtu.ru*

Ye Min Htet
Saint Petersburg State Marine Technical University
Saint Petersburg, Russia

Htut Shine
Saint Petersburg State Marine Technical University
Saint Petersburg, Russia

Anton Victorovich Vegner
Saint Petersburg State Marine Technical University
Saint Petersburg, Russia

To cite this article:

Vassiliev, A.E, Htet, Y.M., Shine, H. ,& Vegner, A. V (2023). User-defined autogenerated fuzzy solvers for embedded applications. *The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM), 24*, 110-118.