Received: 22 October 2023 | *Revised:* 7 February 2024 | *Accepted:* 23 February 2024
DOI: 10.37917/ijeee.21.2.10 *Early View* | *December 2025*

Iraqi Journal for Electrical and Electronic Engineering
Original Article

Open Access
IJEEE University of Basrah College of Engineering

Utilizing Raspberry-Pi 3 to Implement Fuzzy Logic Controller Optimized by Genetic Algorithm

Amal Ibrahim Nasser*, Hasan M. Kadhim, Emad Ahmed Hussien
Electrical Engineering Department, Mustansiriyah University, Baghdad, Iraq

Correspondance
*Amal Ibrahim Nasser
Electrical Engineering Department, Mustansiriyah University,
Baghdad, Iraq
Email: amalalshemmiri@uomustansiriyah.edu.iq

Abstract
The development of Fuzzy Logic Controllers (FLC) with low error rates and cost effectiveness has been the subject of numerous studies. This paper study goals to the investigation and then implementation an FLC using the readily accessible and reasonably priced Raspberry Pi technology. The FLC used in this work has two inputs, one output, and five Membership Functions (MFs) for each input and output. The FLC goes through two processes, tweaking the MF parameters and tuning input/ output Scaling Factors. The tuning technique makes use of the Genetic Algorithm (GA). The whole set of the FLC probabilities is taken into account as the tuned FLC controller, and then transformed into a lookup table. The Center of Gravity (COG) approach is used to determine the output for the tuned FLC controller. The resulting table is converted into values of digital binary using a specific type of encoder, and then extraction of the set of Boolean functions to apply this tuned circuit. Finally, the Python 3 programming language is used to define the resultant Boolean functions on the Raspberry Pi platform, and then a decoder extracted the appropriate control action from the output. The Benefit of this method is the use of a digital numbering system to define the FLC, which is implemented on Raspberry Pi technology and allows for fuzzified high processing speed output per second. The controller speed has not been unaffected by the quantity for these fuzzy rules.

Keywords
Genetic Algorithm, Raspberry Pi, Python, Fuzzy Logic Controller.

I. Introduction and Background

Fuzzy Logic System (FLS) has high level of non-linear outputs, which make academic researchers to think how to linearize that output, because many traditional methods of linearizing have specific errors. Genetic Algorithms (GAs) have been appeared to be giving the linearization for FLS outputs. Researchers used GA in many FLS applications, such as optimizing the rules of FLS, generating optimized membership functions, setting optimized parameters for the membership functions, etc., [1–4]. Currently, fuzzy logic systems proved their capability to overcome various types of issues in different applications. In addition, there are many ascending curiosities for providing these applications the capabilities to learn. A couple well-known methods for hybridizing the system of fuzzy logic with modification approaches are adapted in the computer software domain: the systems of neuro-fuzzy and the systems of genetic fuzzy which hybridize the method for approximating cause of fuzzy logic systems alongside the abilities for learning the evolutionary algorithms and the neural networks [5, 6]. There are two types of Fuzzy Inference Systems (FIS). These types utilize the following renowned computational systems: The Multi-objective Fuzzy (MFS), the Neuro Fuzzy (NFS), the Genetic Fuzzy (GFS), the Evolving Fuzzy (EFS), and the Hierarchical Fuzzy (HFS) systems. Thes five fuzzy systems are connected together. The Multi-objective Fuzzy System (MFS) overcomes trade-offs with multiple objectives, such as the synchronous maximization for the accuracy and the illustration. The Neuro Fuzzy System (NFS)

merges the systems of Neural Network Learning (subset of Machine Learning (ML)) with the Fuzzy Inference Systems (FIS) to upgrade the approximation calculations and features. The Genetic Fuzzy System (GFS) has the self-guiding structure that utilizes the optimization process in the Evolutionary Algorithms (EA). The Evolving Fuzzy System (EFS) overcomes the curses of data streaming using the changes of this system gradually. The optimization methods are the following fuzzy system types: The Takagi–Sugeno–Kang and the Mamdani. The Hierarchical Fuzzy System (HFS) merges multiple low dimensional hierarchical approaches with fuzzy logic units in order to solve the dimensionality problems [4]. According to the search techniques of Gas' stochastic, optimization can be utilized without gradient data relying or getting Local Minima stuck [5] Fuzzy systems capable for solving a range of issues have been created using the combination of fuzzy logic (FL) and GAs. Researchers have begun to utilize the GA as a fuzzy matcher in FL systems in the 1990's [7–9]. Obviously, FL methods have proven to be quite effective in solving a variety of other issues. For instance, FL can be used to characterize vague and ambiguous information in case representation. Through the idea of progressive rules, FL can also be used for case adaptation [9]. The global minimization for the numerical functions' tasks is most significant for many fields of knowledge. The tasks are exploited for a variety of industries, including engineering, finance, management, and medicine. For the Self-Optimizing Fuzzy which has the ability for tackling a wide range of issues, FL and Ga approaches were combined into a generic model. New cases would then be solved using the current case base. The detection of occupancy is a very important machine learning issue. Several approaches are available for the detection of occupancy in the systems of vehicles, such as automobiles in particular. In the meanwhile, automobile safety is becoming very important and sufficient in its industry's field. For instance, car airbags are becoming a simple but significant tool to ensure safety whilst driving these cars. These airbags do come with their risks though, they are capable of severely injuring minors because of their tendency to generate a lot of power. The detection of the passengers' number in a vehicle and the sorting of each individual based on gender and age through image processing of the photos of these passengers. This is applied for the purpose of reducing the risk of airbag usage close to minors. At low speed (less than 30 km/hr), the cars are classified and each car is determined further. Haar Cascades method used for the aforementioned detection process; starts with face detection, and secondly determines their age classification [10]. This article system applied a Raspberry Pi kit, which is a small single-board industrial minicomputer. First design of the Raspberry-Pi kit was in United Kingdom [11,12] . The more popular version of that kit is the original models in spite of the anticipated models [13] , because it doesn't have any peripherals or cases. Also, the academic researchers and students may need specific accessories to accomplish their electronic circuits projects [14, 15]. Many algorithms look to accomplish the issues of the ways of single control, operation of cumbersome, and interconnection complexity among machines in smart home devices which are available in stores. Raspberry Pi kits could be designed for the motherboard of control systems, utilizing speech interaction, gesture recognition, face detection, posture detection, applications of mobile phones and multiple operation techniques of the devices for the home control, based on a light standard messaging protocol, defined self-contained systems for the smart home. The open-source software EMQ proxy could be used as a server to denote the local networking for the smart home by the house Local Area Network (LAN). The delay of transmitting the information should be short. The security should be very high. Traditional WIFI (such as ESP8266) module with the required hardware reduces the costs. The functions could be selected and customized. While GA tuning techniques have made significant contributions to enhancing system specifications, the quest for achieving optimal values has driven the motivation behind this research to pursue further optimization. In this research, self-improving fuzzy systems have been developed by combining genetic algorithms and fuzzy logic, which have proven the effectiveness for addressing a wide range of problems in many research fields. In addition to this introduction, the paper consists of the following titles: Section II provides a formulation of the problem; Section III derives the FLC by using the Raspberry Pi kit; Section IV offers the control methodology that we adopted; Section V for the result; and Section 6 make conclusions from the research results

II. Problem Formulation

In this work, the researchers used a closed loop control system, which contains an error signal (e). That error generates from the difference between the input signal and feedback of the output signal of the system. The rate of change for this error (ce) produces from a differentiation block of that error. The closed loop system contains a controller (e.g., Fuzzy Logic Controller) to get a control action to control the system plant. In this paper approach, the researchers added specific Scaling Factors (SFs). Two Scaling Factors (SF1 and SF2) have been added at the input side of the plant. Only one Scaling Factor (SF3) is added at the output side of the plant, Fig. 1. These SFs are being adjusted using genetic algorithm (GA). In this research, the genetic algorithm has two jobs:

1. Improving these scaling factors with the fixing parameters of the membership function base.
2. Adjusting the membership function base parameters

with fixing the scaling factors.

By try-and-error procedure, scaling factors parameters are obtained. These SF applied on the closed-loop control system for the FLC of the plant, Fig.1. To get an appropriate response, the inputs will be fuzzified, ruled, and deffuzified. The GA chromosomes will serve as the scaling factors (SF1, SF2, and SF3), and the integration of squared error (ISE). That ISE is the difference between the desired and obtained response. ISE serves as the fitness function, Fig.2. In order to obtain the proper base-parameters (x-axis parameters) for these MFs, these scaling factors must be settled using GA once more. The GA's chromosomes represent these parameters. When tuning process have been accomplished, the rules table can be written according to the following membership-base parameters: error (e), crisp error (ce) and control action (ca). The researchers build a prototype version for the system. There are many methods have been used to build the FLC. The most well-known and suited method was the implementation by the using of the FLC microprocessors and/or microcontrollers. 8-bit microprocessors/ microcontrollers are practical in the design of many control systems, but they are not suitable in the systems that need a high processing speed. For our system case, Raspberry Pi 3 platform is the suitable platform. Model B+ of that Raspberry Pi 3 kit contains 64-bit with a 1.4-GHz clock of processor. Before installing the Raspberry-Pi circuit, the inputs (e) and (ce) membership-base parameters must be converted. After that, the signals will be processed using the Raspberry Pi kit, which acts as the fuzzy logic controller (FLC). Finally, the output will translate again as a control action to analog signal using another special design decoder. The block diagram for the process is illustrated in Fig.3.

III. Fuzzy Logic Controller Using Raspberry Pi

Fuzzy Logic Controller (FLC) is a technique, which has the capability for emulating human expertise by employing a fuzzy rule-based system to convert linguistic control into an automated control algorithm. This makes FLC highly suitable for controlling systems represented by nonlinear mathematical models, as it eliminates the requirement for an explicit mathematical model. FLC design defined two fuzzy logic control inputs with five membership functions for each input (e and ce) and output (ca). The FLC design created the rules as well. For the e, ce and ca, each membership function has been encoded by a linguistic variable such as Zero (Z), Positive small (PS), Positive big (PB), Negative small (NS) and Negative big (PB), respectively as shown in Fig. 4. Generation of the appropriate control actions in the FLC, need an expert system that based upon the fuzzified in/out rules, Table I shows the rules that the system needs them to implement the FLC.

TABLE I.
Rules adopted for the FLC

	Change of the control error (ce)					
error(e)	**U**	**NB**	**NS**	**Z**	**PS**	**PB**
	NB	PB	PB	PB	PS	Z
	NS	PB	PB	PS	Z	NS
	Z	PB	PS	Z	NS	NB
	PS	PS	Z	NS	NB	NB
	PB	Z	NS	NB	NB	NB

To implement a fuzzy controller using the Raspberry Pi Kit which is programmed by the Python language, many types of the Raspberry-Pi kit were designed according to their generations. The first was the foundation (model A) and the second for trading (Model B). The last model was developed by Eben Upton (as CEO), called B+ and used for foundation and trading. Raspberry Pi buying and selling is chargeable for growing the technology even as the inspiration is a charity of the education to promote the coaching of the computer technology's basics in colleges, universities and schools; and in the developed countries. In 2018, Raspberry Pi 3 / B+ has been industrialized, which has the following specifications [16, 17] :

- 1.4 GHz quad-core processor with 64-bit.
- Built-in Bluetooth.
- USB and network boot.
- Ethernet of (300 Mbit / s).
- 2.4 / 5 GHz dual-band Wi-Fi of (100 Mbit / s).
- Power over Ethernet (PoE).

The SWI-Prolog programming language [18] is the main software that can download it in the Raspberry-pi 3 kit, but it is not possible to directly control a Raspberry-Pi pins (GPIO) using only Prolog command. It is possible to achieve that if Prolog combined with Python language to create a program named [19]. PySWIP can allow Python program to call Prolog commands in the same IDE. The programming language (Python 2.7) may be setting up within various types of operating systems [20]. In this paper the researchers set up Linux operating system that is installed in the Raspberry-Pi kit. Python programmed packages (library) help to implement the FLC. These packages are: numpy, scipy, matplotlib, and skfuzzy. The steps to implement a fuzzy controller using the Python programming language are as follows:

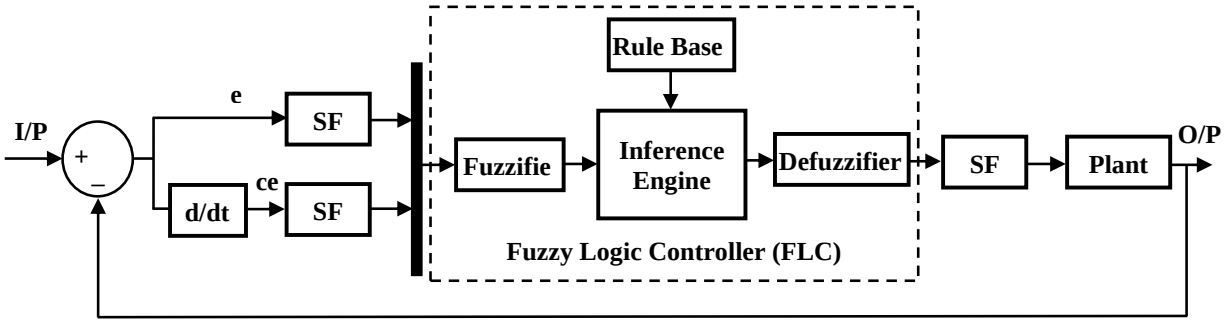

Fig. 1. FLC Structure; The System of the Feedback Control is Unity

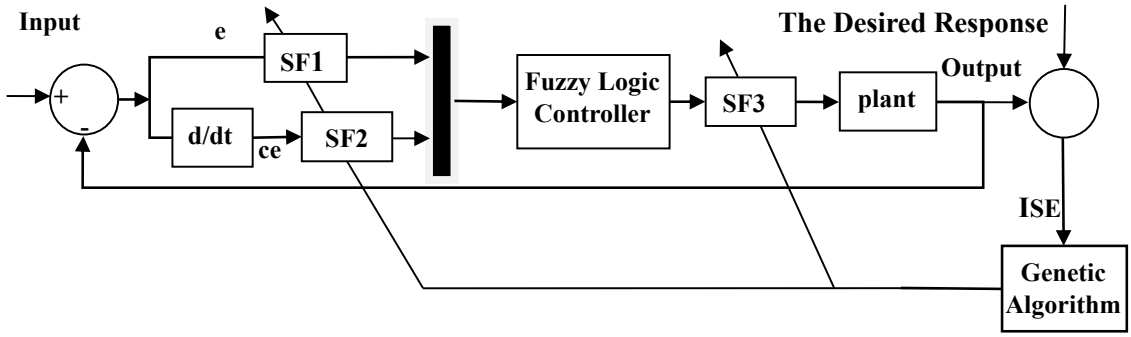

Fig. 2. The Genetic Algorithm, Which is Utilized for Obtaining the Most Optimal Three Scaling Factors

- Create universe variables for error (e). Create the change of that error (ce). Create the required control action (ca) for that.
- Utilize the skfuzzy library to generate fuzzy membership functions for e, ce, and ca.
- Verify the program's functionality by providing values for error (e) and change of error (ce), then calculate the degree of membership for each group using the membership-base parameters (e, ce, ca) function.
- Construct an expert system based on the fuzzified input/ output rules to generate appropriate control actions within the Fuzzy Logic Controller.
- Implement the FLC in the Python programming language by employing the generated rules.
- Utilize the matplotlib library to visualize the membership functions graphically.
- Install the skfuzzy software by cloning it from the GitHub repository and configuring it using the setup.py file.

Execute the Python scripts incorporating fuzzy logic to generate the appropriate control actions.

IV. Genetic Algorithm-Fuzzy Optimization

Generally, the purpose of combining fuzzy logic and genetic algorithm methods is to develop a flexible and self-optimizing fuzzy system that is capable of solving a variety of problems. This system would use a case database already in existence, to develop answers for fresh cases, increasing its adaptability and problem-solving skills [21, 22]. When the supplied function exhibits multiple local minima, each with their own reaction basin and often causing the outcome to rely on a starting point, problems start to arise. Unfortunately, nonlinear, discontinuous, multi-modal, high-dimensional, and complicated goal functions are where the majority of real problems emerge. Stochastic approaches appear to be a viable solution (and sometimes the only), for these problems. One of the most widely used methods for stochastic global optimization is the combination of GAs and simulated annealing [23]. The problem in that situation has to do with convergence speed and the GA approaches guarantee that it will be possible to obtain a global optimum under normal conditions. On the other hand, pure analytic methods provide results that guarantee their convergence to a global minimum with sure/certain event probability (equals 1), although the performance shown by the majority of implementations is not very promising.

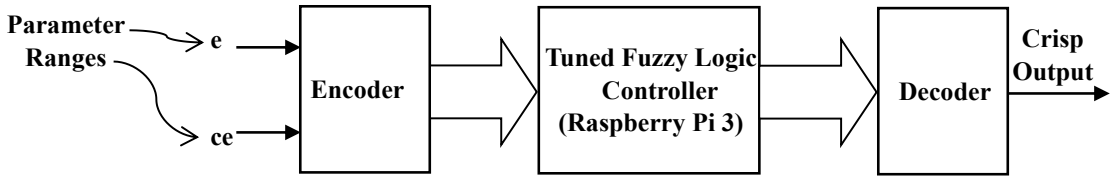

Fig. 3. Functional Block Diagram of the Fuzzy Logic Controller (Genetically Tuned) of Raspberry PI

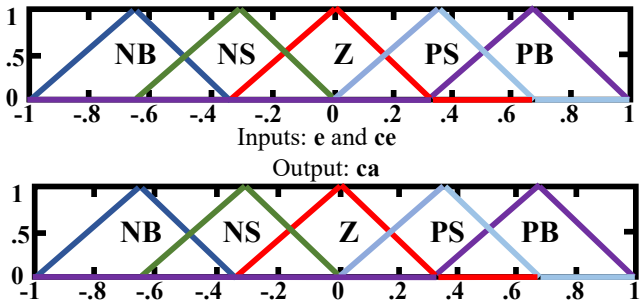

Fig. 4. The Controller's Inputs are (e, ce). The Output is (ca) Which is the Initial Membership Functions

This research strategy adheres to the following two phases to enhance output:

1. Genetic Algorithm Optimization of Scaling Factors: The first step is to use GA to optimize the scaling factors, which they are SF1, SF2, and SF3; while keeping the membership function base parameters fixed. The fitness function for GA is the performance index, which is the Integral of Square of Error (ISE) for the desired response with the step response. The GA operations (encoding, mutation, selection, and crossover) are applied to obtain the optimal solution that has minimum fitness.

2. Genetic Algorithm Optimization of Membership Function Base Parameters: The second step is to use GA to optimize the membership function base parameters while keeping the scaling factors fixed. The Fitness Function for GA is the same as in step 1. The GA operations are applied to obtain the optimal solution that has minimum fitness.

Integral of Square of Error (ISE) used for the performance index in this paper, which is defined as:

$$ISE = \int_0^T e^2(t)dt \tag{1}$$

Where T is the time that the error (e(t)) when they reach the steady state. $T = Ts$ can be used, where Ts is the settling time. So, the relation below is the used GA fitness function (fs) that we will use it to get minimum ISE:

$$fs = \frac{1}{ISE} \tag{2}$$

The GA processing are the encoding, the mutation, the selection, and the crossover. The processing is applied in order to get the optimal solution for the minimal fs. The survived chromosomes encoded to five bits (A1, A2, A3, A4, A5) those can be reduced using the Karnaugh-Map. These five bits will be the digital inputs to the Raspberry Pi. The outputs of the Raspberry Pi will be four Boolean functions (P1, P2, P3, P4) and those are the memberships-base parameters after the use of the Center of Gravity calculations. After that, they resulted Boolean functions (P1, P2, P3, P4) are programmed using Python 2.7 language that was stored within the Raspberry Pi 3 software. Finally, the decoder translates the resulted four bits outputs to its equivalent analog signal (crisp output) to derive the controlled plant. Fig. 5 illustrates the Encoder/ Decoder of the crisp output.

V. Results and Discussion (Case Study)

The research provides an example of implementing the new controller to control a plant with a specific transfer function and step response. The GA is used to minimize the ISE between the step response and the desired response. The results are presented. The implementation of the controller is to control a plant with the bellow transfer function:

$$P(s) = \frac{1}{s(s+1)} \tag{3}$$

With a step response contains the following parameters (Fig.6): Settling time (Ts) = 0.6 seconds, percentage overshoot (PO) = 32. To minimize the ISE between the desired and the current step response, Genetic Algorithm (GA) is used to

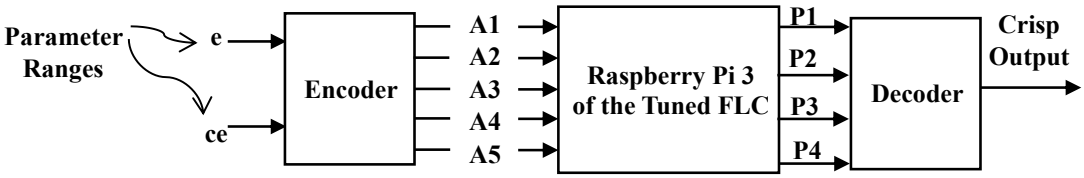

Fig. 5. The Encoder/ Decoder Representation of the Crisp Output

obtain the optimized input/ output of the scaling factors (i.e., SF1, SF2, and SF3), while keeping without any change for the parameters of the membership base. The GA is executed with the following parameters: 500 generations, a population size of 50, a chromosome length of 3 (for SF1, SF2, and SF3), a crossover probability of 0.95 (using simple crossover), and a mutation probability of 0.01 (using uniform mutation). The results of this step are: The first parameter SF1 is 1.0043, the second SF2 is 1.019, the third SF3 is -9.7989, and the ISE = 1.9362.

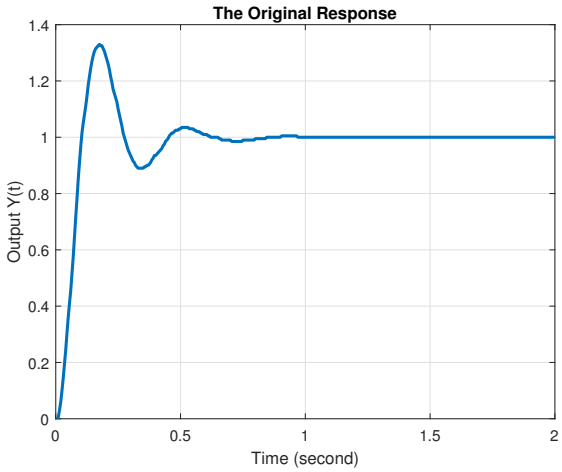

Fig. 6. The Required Step Response

For the next step, Genetic Algorithm has been used once more to get the membership-base parameters, while fixing that scaling factors (SF1, SF2, and SF3), utilizing the following parameters: 500 generations, a population size of 200, a chromosome length of 35 (for the membership-base parameters: e, ce, and ca), a mutation probability of 0.01 (using uniform mutation), and a crossover probability of 0.95 (simple crossover). The result is the membership functions for error (e), change of error (ce), and control action (ca), with ISE equals to 1.9362, as shown in Fig. 7, Fig. 8 and Fig. 9 illustrate the steps of GA training. Finally, Fig. 10 shows the obtained step-response

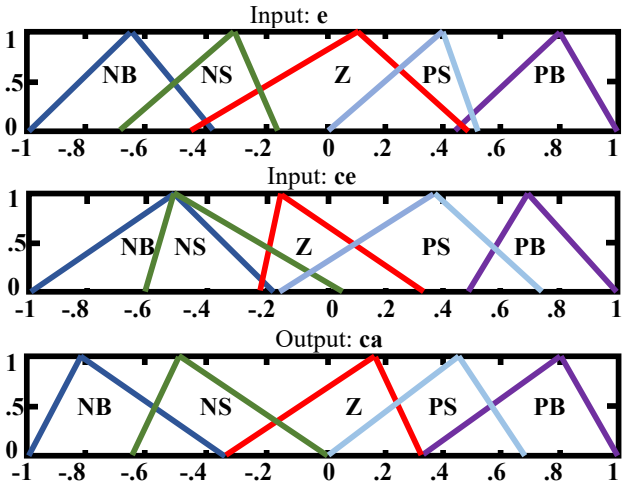

Fig. 7. The Optimized MFs Using Scales: e is 1.0043, ce is 1.019, and ca is -9.7989

using our GA. The resulted MFs for the two inputs (e and ce) will be divided into regions after which each region will choose from among all possible choices of (e / SF1) and (ce / SF2) to cover all probabilities as shown in Table II. Applying the mentioned values to our FLC to obtain the crisp output (ca) and multiply it by the scaling factor SF3 to get the values in column 3. Now the result (ca × SF3) will be approximated then encoded using a four-bit decimal to binary encoder to represent the variables (P1, P2, P3, P4). Finally, we will split every five rows in Table II to create five regions represents the input variables (A1, A2, A3, A4, A5). Lookup table could be built, which represents the input variables (A1, A2, A3, A4, A5) and the output variables (P1, P2, P3, P4) as shown in Table III.

Using Karnaugh-map to reduce the Boolean variables to get:

$$P1 = A2A3\overline{A4} + A2\overline{A4}A5 + A2A3\overline{A5} + A1A2 + A1A4 + A1A5 \quad (4)$$

TABLE II.
ENCODING OF (E), (CE) AND (CA)

Error (e) / SF1	Change of Error (ce) / SF2	Crisp Value (ca) × SF3	Approximate	Digital Conversion
-1 to -0.6	-1 to -0.6	0.694 × 9.7989 = 6.8	7	0111
-1 to -0.6	-0.6 to -0.2	0.707 × 9.7989 = 6.9	7	0111
-1 to -0.6	-0.2 to 0.2	0.583 × 9.7989 = 5.7	6	0110
-1 to -0.6	0.2 to 0.6	0.374 × 9.7989 = 3.6	4	0100
-1 to -0.6	0.6 to 1	0.0427 × 9.7989 = 0.4	0	0000
-0.6 to -0.2	-1 to -0.6	0.694 × 9.7989 = 6.8	7	0111
-0.6 to -0.2	-0.6 to -0.2	0.672 × 9.7989 = 6.58	7	0111
-0.6 to -0.2	-0.2 to 0.2	0.261 × 9.7989 = 2.5	3	0011
-0.6 to -0.2	0.2 to 0.6	0.0467 × 9.7989 = 0.45	1	0001
-0.6 to -0.2	0.6 to 1	-0.361 × 9.7989 = -3.53	-4	1100
-0.2 to 0.2	-1 to -0.6	0.683 × 9.7989 = 6.7	7	0111
-0.2 to 0.2	-0.6 to -0.2	0.536 × 9.7989 = 5.25	5	0101
-0.2 to 0.2	-0.2 to 0.2	-0.0528 × 9.7989 = -0.5	-1	1111
-0.2 to 0.2	0.2 to 0.6	-0.389 × 9.7989 = -3.8	-4	1100
-0.2 to 0.2	0.6 to 1	-0.711 × 9.7989 = -6.9	-7	1001
0.2 to 0.6	-1 to -0.6	0.48 × 9.7989 = 4.7	5	0101
0.2 to 0.6	-0.6 to -0.2	0.285 × 9.7989 = 2.8	3	0011
0.2 to 0.6	-0.2 to 0.2	-0.318 × 9.7989 =- 3.11	-3	1101
0.2 to 0.6	0.2 to 0.6	-0.625 × 9.7989 = -6.1	-6	1010
0.2 to 0.6	0.6 to 1	-0.711 × 9.7989 = -6.9	-7	1001
0.6 to 1	-1 to -0.6	0.0301 × 9.7989 = -0.3	0	0000
0.6 to 1	-0.6 to -0.2	-0.177 × 9.7989 = -1.73	-2	1110
0.6 to 1	-0.2 to 0.2	-0.655 × 9.7989 = -6.4	-6	1010
0.6 to 1	0.2 to 0.6	-0.719 × 9.7989 = -7.04	-7	1001
0.6 to 1	0.6 to 1	-0.711 × 9.7989 = -6.9	-7	1001

$$P2 = \overline{A1A2A3} + \overline{A4}A5 + A2A5 + \overline{A1A3}A4 + \overline{A1A2A4A5} + A2A3\overline{A4} \quad (5)$$

$$P3 = \overline{A2A3A5} + \overline{A1A2A4}A5 + \overline{A2}A4\overline{A5} + \overline{A1A2}A3A4 + \overline{A2}A3\overline{A4}A5 + A2A3\overline{A4A5} + \overline{A3}A4\overline{A5} \quad (6)$$

$$P4 = A2\overline{A5} + A2A4 + \overline{A1}A3A4 + A1A4A5 + A2\overline{A3A4} + \overline{A1A2A4}A5 \quad (7)$$

These Boolean functions will be built using Python 2.7 programming language, and then contained in Raspberry Pi 3 to get an optimal robust fuzzy-genetic controller.

VI. CONCLUSIONS

The paper discussed the use of genetic algorithms (GAs) in combination with Fuzzy Logic (FL) to create self-optimizing fuzzy systems that can deal with several problems. Also highlighted is the use of the Raspberry Pi kit, a small single-board microcomputer, to implement fuzzy controllers. GA was used to optimize the in/out scaling factors and membership base parameters for the fuzzy controller. It was concluded that the combination of FL with GA has proven effective in addressing many problems, and the use of the Raspberry Pi kit provides a suitable processor for implementing fuzzy controllers. The paper also stresses the importance of stochastic methods such as GA for solving complex optimization problems.

ACKNOWLEDGMENT

The article authors would like to thank the Mustansiriyah University / Engineering College, Electrical Engineering Department. They have encouraged us to complete the research. The university, the college, and the department are institutes for the Iraqi MoHE&SR.

CONFLICT OF INTEREST

The authors have no conflict of relevant interest to this article.

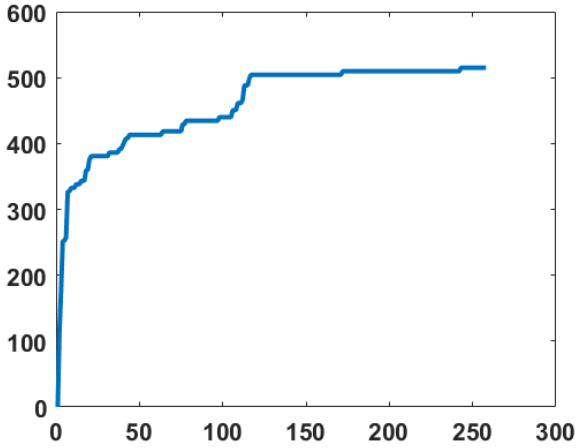

Fig. 8. Depicts the Relationship Between the Generation Throughout the GA (Horizontal Axis), Versus the Procedure Fitness (Vertical Axis)

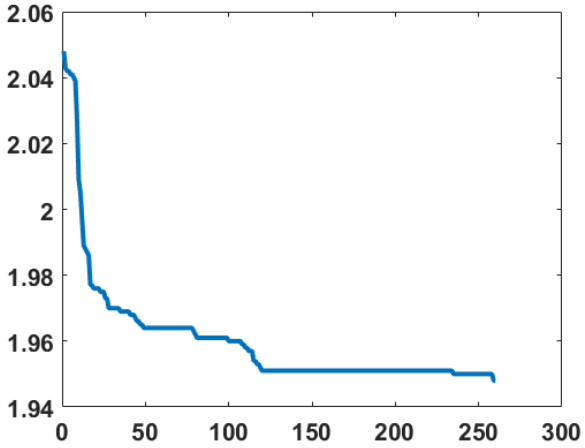

Fig. 9. ISE (Vertical Axis) Versus the Generation Curve (Horizontal Axis) for the GA Procedure

TABLE III.
ENCODED INPUTS/OUTPUTS LOOKUP TABLE

A1	A2	A3	A4	A5		P1	P2	P3	P4
0	0	0	0	0		0	1	1	1
0	0	0	0	1		0	1	1	1
0	0	0	1	0		0	1	1	0
0	0	0	1	1		0	1	0	0
0	0	1	0	0		0	0	0	0
0	0	1	0	1		0	1	1	1
0	0	1	1	0		0	1	1	1
0	0	1	1	1		0	0	1	1
0	1	0	0	0		0	0	0	1
0	1	0	0	1		1	1	0	0
0	1	0	1	0		0	1	1	1
0	1	0	1	1		0	1	0	1
0	1	1	0	0		1	1	1	1
0	1	1	0	1		1	1	0	0
0	1	1	1	0		1	0	0	1
0	1	1	1	1		0	1	0	1
1	0	0	0	0		0	0	1	1
1	0	0	0	1		1	1	0	1
1	0	0	1	0		1	0	1	0
1	0	0	1	1		1	0	0	1
1	0	1	0	0		0	0	0	0
1	0	1	0	1		1	1	1	0
1	0	1	1	0		1	0	1	0
1	0	1	1	1		1	0	0	1
1	1	0	0	0		1	0	0	1

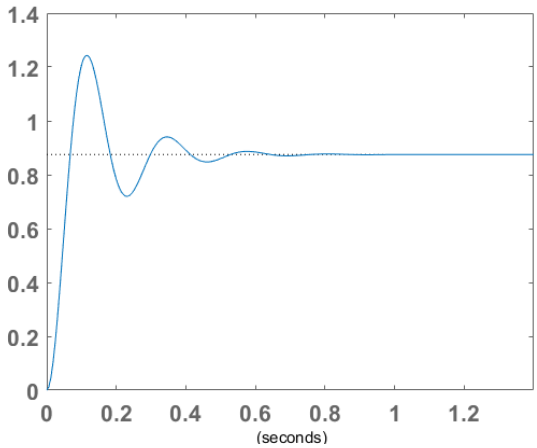

Fig. 10. Response of the GA.

REFERENCES

[1] O. Cordón, F. Herrera, F. Gomide, F. Hoffmann, and L. Magdalena, "Ten years of genetic fuzzy systems: current framework and new trends," in *Proceedings joint 9th IFSA world congress and 20th NAFIPS international conference (Cat. No. 01TH8569)*, vol. 3, pp. 1241–1246, IEEE, 2001.

[2] A. BASTIAN and I. HAYASHI, "An anticipating hybrid genetic algorithm for fuzzy modeling," *Journal of Japan Society for Fuzzy Theory and Systems*, vol. 7, no. 5, pp. 997–1006, 1995.

[3] O. Cord *et al.*, *Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases*, vol. 19.

World Scientific, 2001.

[4] V. Ojha, A. Abraham, and V. Snášel, "Heuristic design of fuzzy inference systems: A review of three decades of research," *Engineering Applications of Artificial Intelligence*, vol. 85, pp. 845–864, 2019.

[5] G. Warner, S. Wijesinghe, U. Marques, O. Badar, J. Rosen, E. Hemberg, and U.-M. O'Reilly, "Modeling tax evasion with genetic algorithms," *Economics of Governance*, vol. 16, pp. 165–178, 2015.

[6] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia tools and applications*, vol. 80, pp. 8091–8126, 2021.

[7] P. V. de Campos Souza, "Fuzzy neural networks and neuro-fuzzy networks: A review the main techniques and applications used in the literature," *Applied soft computing*, vol. 92, p. 106275, 2020.

[8] M. Mowbray, T. Savage, C. Wu, Z. Song, B. A. Cho, E. A. Del Rio-Chanona, and D. Zhang, "Machine learning for biochemical engineering: A review," *Biochemical Engineering Journal*, vol. 172, p. 108054, 2021.

[9] S. Das, D. Chakraborty, and L. T. Kóczy, "Linear fuzzy rule base interpolation using fuzzy geometry," *International Journal of Approximate Reasoning*, vol. 112, pp. 105–118, 2019.

[10] M. Vamsi and K. Soman, "In-vehicle occupancy detection and classification using machine learning," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–6, IEEE, 2020.

[11] S. Hodges, S. Sentance, J. Finney, and T. Ball, "Physical computing: A key element of modern computer science education," *Computer*, vol. 53, no. 4, pp. 20–30, 2020.

[12] T. S. Roy, S. Ghosh, R. Datta, and A. Santra, "Iot based home automation using raspberry pi," *International Journal of Computer Engineering and Technology*, vol. 10, no. 3, pp. 70–74, 2019.

[13] L. Zhu, P. Spachos, E. Pensini, and K. N. Plataniotis, "Deep learning and machine vision for food processing: A survey," *Current Research in Food Science*, vol. 4, pp. 233–249, 2021.

[14] G. Yan, J. Zeng, H. Sang, Y. Lin, J. Lin, and Q. Li, "New customizable smart home system design based on raspberry pi," in *International Workshop of Advanced Manufacturing and Automation*, pp. 91–99, Springer, 2022.

[15] M. Nykyri, M. Kuisma, T. J. Kärkkäinen, J. Hallikas, J. Jäppinen, K. Korpinen, and P. Silventoinen, "Iot demonstration platform for education and research," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, pp. 1155–1162, IEEE, 2019.

[16] J. R. Strickland, "Raspberry pi for arduino users," *Raspberry Pi for Arduino Users*, 2018.

[17] G. Flurry and G. Flurry, "Raspberry pi 3 model b+ setup," *Java on the Raspberry Pi: Develop Java Programs to Control Devices for Robotics, IoT, and Beyond*, pp. 21–48, 2021.

[18] R. Hosseini, K. Akhuseyinoglu, P. Brusilovsky, L. Malmi, K. Pollari-Malmi, C. Schunn, and T. Sirkiä, "Improving engagement in program construction examples for learning python programming," *International Journal of Artificial Intelligence in Education*, vol. 30, no. 2, pp. 299–336, 2020.

[19] B. A. Malloy and J. F. Power, "Quantifying the transition from python 2 to 3: An empirical study of python applications," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 314–323, IEEE, 2017.

[20] B. P. Ganthia, S. K. Barik, and B. Nayak, "Genetic algorithm optimized and type-i fuzzy logic controlled power smoothing of mathematical modeled type-iii dfig based wind turbine system," *Materials Today: Proceedings*, vol. 56, pp. 3355–3365, 2022.

[21] M. Dirik, "Prediction of nox emissions from gas turbines of a combined cycle power plant using an anfis model optimized by ga," *Fuel*, vol. 321, p. 124037, 2022.

[22] Y.-T. Hsiao and C.-Y. Chien, "Enhancement of restoration service in distribution systems using a combination fuzzy-ga method," *IEEE Transactions on Power Systems*, vol. 15, no. 4, pp. 1394–1400, 2000.

[23] Y. Li, J. Zhang, W. Liu, and S. Tong, "Observer-based adaptive optimized control for stochastic nonlinear systems with input and state constraints," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 12, pp. 7791–7805, 2021.