electronics

Article

Tuning of PID Controller in PLC-Based Automatic Voltage Regulator System Using Adaptive Artificial Bee Colony–Fuzzy Logic Algorithm

Hüseyin Altınkaya [1,*] and Dursun Ekmekci [2,*]

[1] Department of Electrical-Electronics Engineering, Karabuk University, 78050 Karabuk, Türkiye
[2] Department of Computer Engineering, Karabuk University, 78050 Karabuk, Türkiye
* Correspondence: haltinkaya@karabuk.edu.tr (H.A.); dekmekci@karabuk.edu.tr (D.E.)

Abstract: The voltage control of synchronous generators, particularly under varying load conditions, remains a significant and complex challenge in the field of engineering. Although various control methods have been implemented for automatic voltage regulator (AVR) systems to control the terminal voltage of synchronous generators, the PID-based control method continues to be one of the most basic and widely used approaches. Determining the optimal values for the *Kp*, *Ki*, and *Kd* values is essential to ensuring efficient and rapid performance in a PID controller. This study presents PLC-based PID controller tuning using an adaptive artificial bee colony–fuzzy logic (aABC-FL) approach for voltage control in a micro-hydro power plant installed as an experimental setup. The real-time control and monitoring of the system was conducted using an S7-1200 programmable logic controller (PLC) integrated with a totally integrated automation (TIA) portal interface and a SCADA screen. The aABC-Fuzzy design was developed using the MATLAB/Simulink platform, with PLC-MATLAB communication established through OPC UA and the KEPServerEX interface. The results obtained from experiments conducted under different load conditions showed that the proposed aABC-FL PID significantly minimized settling time and overshoot compared to the classical PLC-PID. Additionally, the proposed method not only provided a good dynamic response but also proved to be robust and reliable for real physical AVR systems.

Keywords: artificial bee colony; fuzzy logic; PID; AVR; PLC

check for updates

Citation: Altınkaya, H.; Ekmekci, D. Tuning of PID Controller in PLC-Based Automatic Voltage Regulator System Using Adaptive Artificial Bee Colony–Fuzzy Logic Algorithm. *Electronics* **2024**, *13*, 5039. https://doi.org/10.3390/electronics13245039

Academic Editor: Maciej Ławryńczuk

Received: 24 October 2024
Revised: 12 December 2024
Accepted: 19 December 2024
Published: 21 December 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The control of terminal voltage in synchronous generators (SG) has been a challenging issue in both academic research and industrial applications for decades. This challenge arises from the nonlinear nature of SGs and the systems in which they operate, as well as the impact of unpredictable parameters on the system behavior. The fluctuations in load lead to variations in voltage, causing control circuits to adjust the excitation voltage or current of SGs to restore nominal values. Devices that regulate the terminal voltage of synchronous generators and maintain voltage variations within specified limits are called automatic voltage regulators (AVR). In parallel (synchronous) operating synchronous generators and power systems, the AVR not only adjusts the output voltage but also ensures the proper sharing of reactive power among the generators.

To improve the control capabilities of automatic voltage regulators (AVRs), several advanced control strategies have been proposed in the literature. A fractional high-order differential feedback controller (FHODFC) was proposed, which has been shown to provide a lower overshoot percentage, settling time, rise time, and peak time compared to other controllers [1]. In another study, a neural network predictive controller with an imperialist competitive algorithm was proposed for AVR [2]. Another proposed approach involves the use of a PIΛDND2N2 controller using the coyote optimization algorithm for an AVR in a power system [3].

Achieving a robust, stable, and dynamic response in AVR systems requires determining the optimal values for the proportional, integral, and derivative gains (*Kp*, *Ki*, and *Kd*). An appropriate tuning technique to obtain these optimal gains is essential for ensuring the stable performance of the system.

To enhance the robustness of PID controllers and optimize the proportional–integral–derivative gains in an automatic voltage regulator, various optimization methods, artificial intelligence techniques, and metaheuristic algorithms have been employed in the design and tuning processes of these control systems. A review of the literature shows that various methods have been proposed to optimize PID controllers in AVR systems, including the cuckoo search algorithm [4,5], yellow saddle goatfish algorithm [6], chaotic black widow algorithm [7], future search algorithm [8], nonlinear sine cosine algorithm [9], teaching–learning based optimization algorithm [10], whale optimization algorithm [11,12], coyote optimization algorithm [3], kidney-inspired algorithm [13], stochastic fractal search algorithm [14], particle swarm optimization algorithm [15], artificial bee colony algorithm [16], tree seed algorithm [17], grey wolf optimization algorithm [18], imperialist competitive algorithm [19], metaheuristic optimization [20], particle swarm optimization [21], Runge Kutta optimization [22], PSS and H∞ optimization [23], monarch butterfly optimization [24], Sugeno fuzzy-PID tuning [25], fractional order PID [26], hyper-heuristic approach [27], and sliding mode controller approach [28]. Furthermore, review studies have been conducted on controller design for AVR and the metaheuristic algorithms used for their optimization [29,30].

Automation control systems are widely used in science and technology. In industrial automation systems, PLCs are commonly preferred as controllers and are often considered the main component of the system. In a study, a fuzzy-PID controller was designed using MATLAB and PLC for voltage control in an AVR system [31]. In another study, a real concentrated solar power plant was modeled, with system control achieved through a PLC using fuzzy-PID [32]. Another study reported the control of a real physical system via PLC using deep reinforcement learning optimization [33]. In a study on process control applications using a PLC, a generalized predictive PI controller was proposed to enhance system performance [34].

An AVR system consists of four main components: an amplifier, an exciter, a generator, and a sensor. Each of these components is modeled as a first-order system, which is characterized by respective gains, K_a, K_e, K_g, and K_s, along with time constants, T_a, T_e, T_g, and T_s. In the literature, existing studies on AVR are observed to focus on the following topics: proposed controllers, proposed parameter optimization algorithms, superiority comparison, tuning methods, analysis methods, and performance indices. Comprehensive surveys and comparisons have been conducted on AVR-related studies [13,28–30]. These studies typically present block diagrams of uncontrolled, controlled, and metaheuristic-controlled AVR, along with their respective transient response, step response, and robustness graphs. In addition, they include pole-zero plots, Bode diagrams, robustness analysis, and frequency response analyses.

The same classical AVR model, transfer function, and similar or closely related parameter ranges are observed to be used in almost all studies. The primary focus has been on achieving better results with the applied metaheuristic method compared to those used in other studies.

In this study, the terminal voltage of a micro-hydro power plant (MHHP) installed as an experimental setup was controlled using both classical PID and adaptive artificial bee colony–fuzzy logic PID (aABC-FL PID). A task-sharing approach among the hardware was proposed for voltage control, in which more complex and challenging parts of the software were executed on a separate computer, thereby reducing the software load on the PLC. This study may serve as a valuable example for researchers aiming to perform complex operations using PLCs. The PID controller gains on PLC were tuned using the artificial bee colony–fuzzy logic algorithm, which was implemented in MATLAB/Simulink on a separate computer. The membership function (MF) parameters of the developed

fuzzy model were dynamically optimized using the proposed adaptive artificial bee colony (aABC) algorithm. The main contributions of this study can be listed as follows:

- The aABC-Fuzzy PID technique has been proposed for the first time in the PID controller design of a real physical AVR system, marking the most significant aspect of the originality of this study.
- In contrast to existing studies, rather than using the classical AVR model and transfer function, an experimental approach specific to the controlled system has been presented.
- Unlike many existing studies, a dynamic and adaptive control design has been implemented in a real physical system.
- A real-time application has been presented demonstrating the practical value and tuning methodology of the proposed algorithm using industrial equipment, such as a PLC.
- This study demonstrates that metaheuristic algorithms can be applied in industrial applications using simple controllers, such as PLCs.
- The aABC-fuzzy PID controller offers significant theoretical and practical advantages, including real-time performance, online parameter adjustment, and reliable convergence behavior.
- This approach can be applied to develop advanced real-time controllers for use in both academic and industrial applications.
- This approach can be extended to hybrid and large-scale power plants.

The paper is organized as follows: Section 2 provides materials and methods, offering detailed information about the experimental setup. Section 3 explains the proposed aABC-fuzzy PID. Section 4 discusses the results obtained from experimental studies. Finally, Section 5 presents the conclusions and future works.

2. Experimental Setup

The experimental setup consists of three 1 kW synchronous generators synchronized with each other, three single-nozzle Pelton turbines fabricated using a 3D printer, a water tank with a capacity of approximately one ton, a 5.5 kW centrifugal pump supplying the required flow and pressure, and an 11 kW frequency converter controlling the pump's speed. In addition, electrical components, such as network analyzers, synchronization devices, proportional valves, sensors, and contactors, are used in the setup. The design of the mechanical and electrical systems of the prototype, as well as its operational scenario, closely simulates a real MHPP. S7-1214C DC/DC/DC PLC has been used to control the voltage and frequency of the synchronous generators based on load and other control processes. Figure 1 illustrates the overall design of the experimental setup.



Figure 1. Experimental setup.

2.1. Working Principle of the Experimental Setup

In the prototype, water is drawn from a tank with an approximate capacity of one ton and is delivered to the proportional valves through a horizontal shaft, a four-stage pump driven by an asynchronous motor, and a connected piping system. An analog sensor is used to monitor the water pressure within the system. The centrifugal pump motor, which is controlled by a frequency converter connected to a PLC, stabilizes the water pressure. The proportional valves regulate the flow rate, directing the water through nozzles to the Pelton turbines at an increased velocity. As the water passes through the turbines, it loses energy and flows back into the tank. The mechanical energy generated in the turbines is then converted into electrical energy by the synchronous generators coupled with the turbine shaft. The excitation of synchronous generators is conducted by DC converters, which are controlled by a 0–5 V analog signal. The frequency of the SGs is managed by the PLC, which controls the proportional valves. The electrical energy generated is synchronized and integrated into the power grid using synchroscope devices. Throughout these processes, the PLC continuously monitors and controls the system using sensors and network analyzers. Additionally, real-time event monitoring is facilitated by PLC-SCADA TIA Portal V17 software. Figure 2 shows a block diagram of the system.

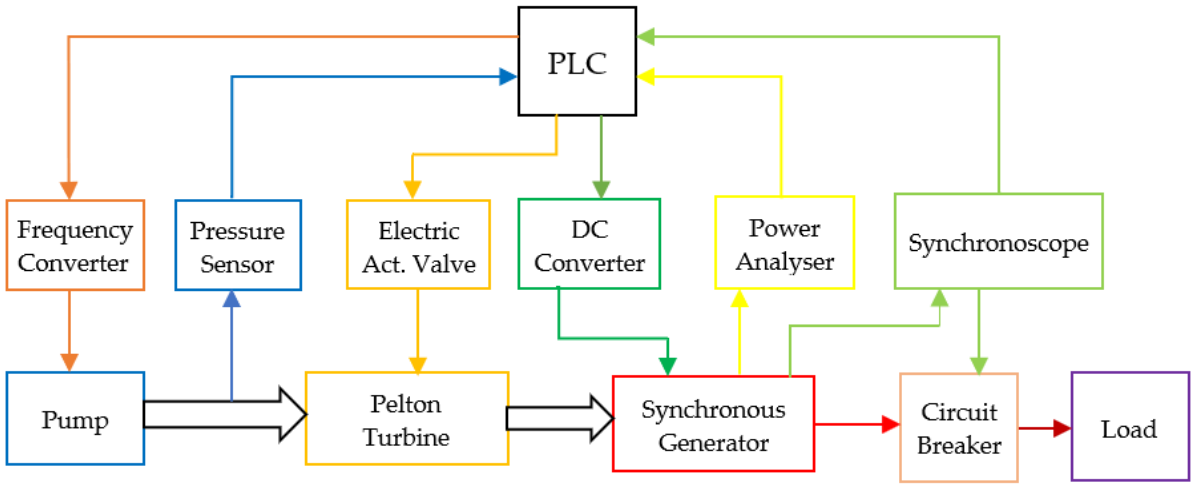


Figure 2. Block diagram of the system.

2.2. Voltage Control

The 0–5 V analog control inputs of the DC converters are connected to the analog outputs of the PLC to control the voltage produced by the SGs. This signal adjusts the excitation voltage, thereby regulating the output voltage of the generators. Prior to the synchronization of the SGs with each other, generator voltages need to match the busbar voltage. The process, which is detailed in the flowchart in Figure 3, ensures that this voltage matches between the generators and the busbar. This process is essential for successful synchronization and is terminated once synchronization is completed.

Here, I_{exc} is the excitation current, and $V_{L\text{-}N}$ is the phase-neutral voltage.

The nominal values of the SGs in the experimental setup are as follows:

- Power: 1 kW;
- Speed: 1500 rpm;
- Frequency: 50 Hz;
- Voltage: 380 V AC (Phase-neutral: 220 V);
- Current: 2.3 A AC;
- Excitation voltage: 72 V DC;
- Excitation current: 2.1 A DC.

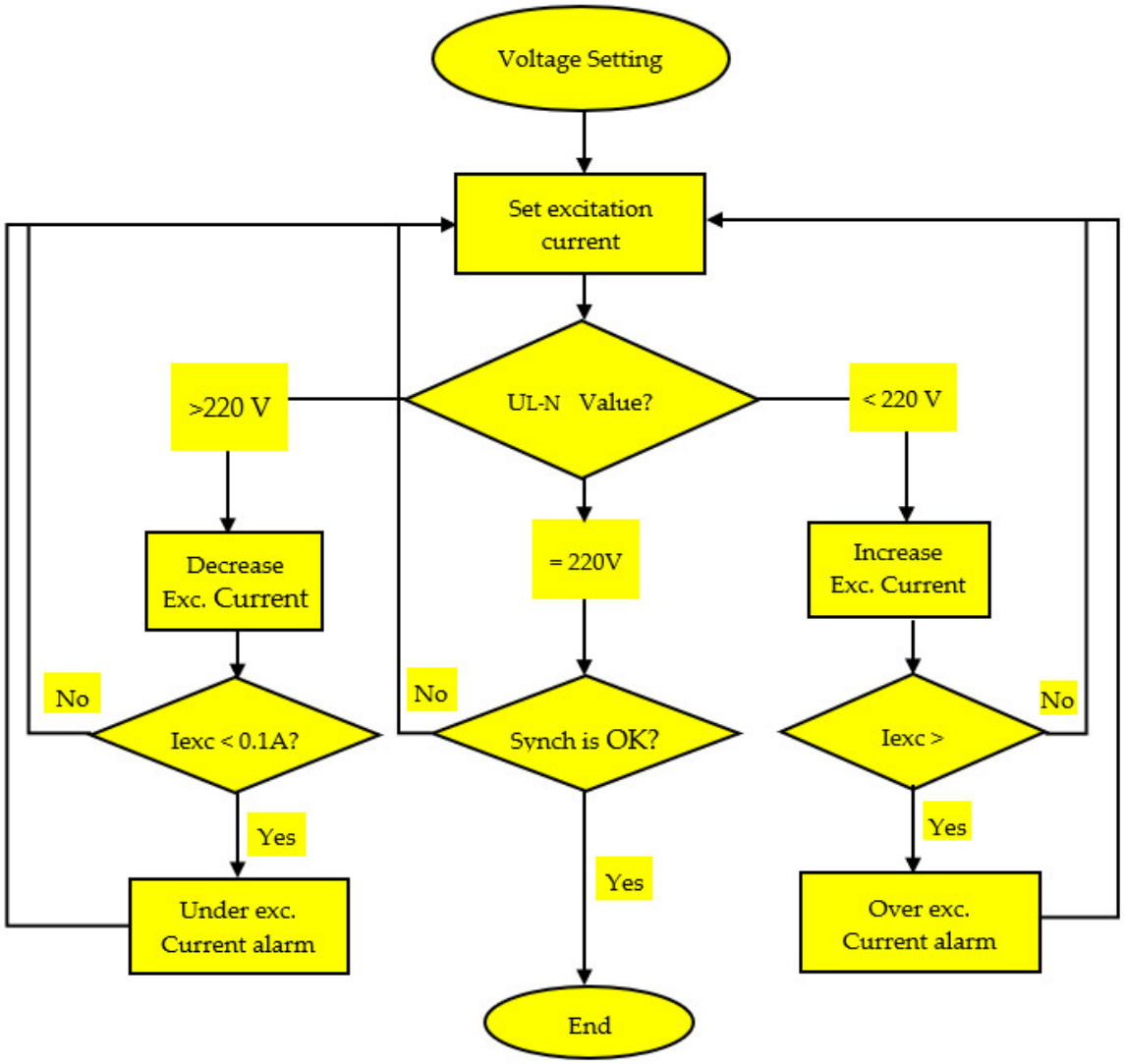


Figure 3. Voltage control flowchart.

The nominal current and voltage of the SGs are effective values.

The control of terminal voltages of the SGs under varying load conditions is achieved using the PID_Compact function. The PID algorithm works according to Equation (1).

$$y = K_p\left[(b \cdot w - x) + \frac{1}{T_I \cdot s}(w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - x)\right] \quad (1)$$

Here, y is the output value of the PID algorithm, K_p is the proportional gain, s is the Laplace operator, b is the proportional action weighting, w is the set point, x is the process value, T_I is the integral action time, T_D is the derivative action time, a is the derivative delay coefficient, and c is the derivative action weighting [35]. Figure 4 shows the PID block diagram of voltage control using the PLC.

The compact PID created for SG1 within the OB30 Cyclic Interrupt block in the TIA Portal V17 interface is shown in Figure 5.

PID_Compact obtains the measured value of the process in a control loop and, after processing it, compares the value with the setpoint. Using the obtained control deviation, PID_Compact calculates the output value, which is quickly adapted to the setpoint. For the PID controller, the calculated output value consists of three actions, namely, P, I, and D.

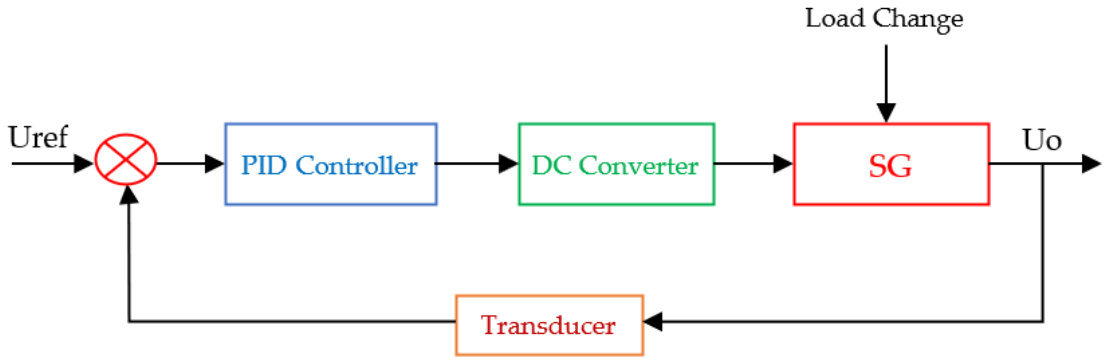


Figure 4. PID block diagram of voltage control.

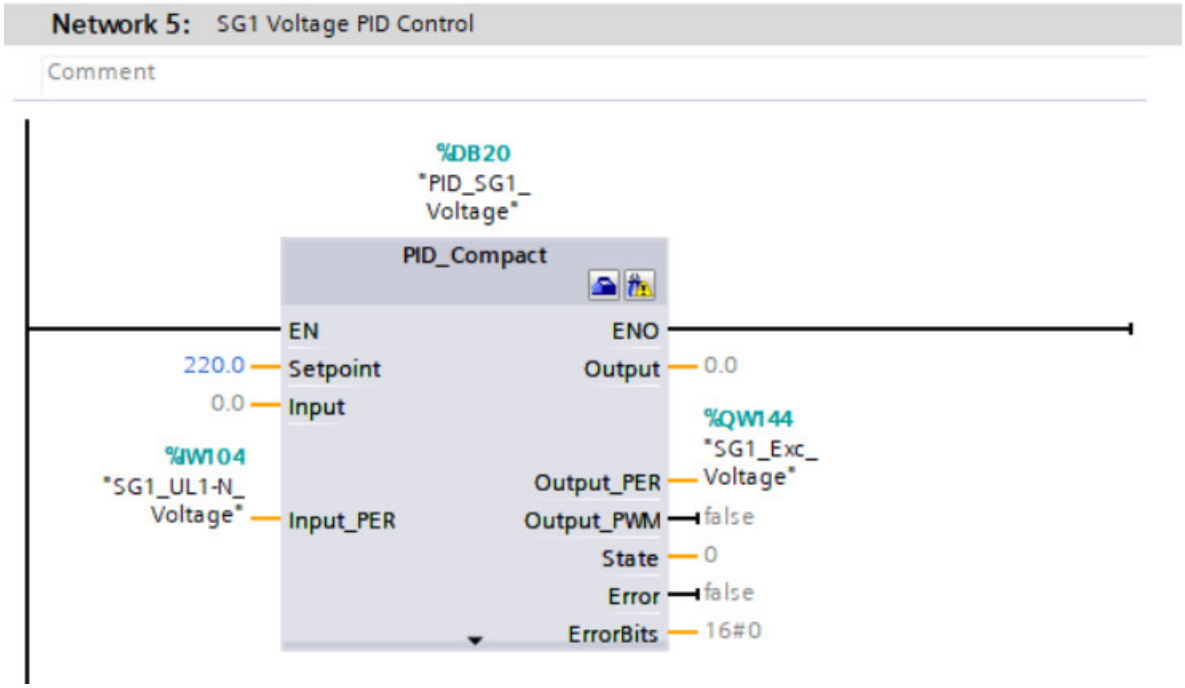


Figure 5. Compact PID in TIA Portal.

The proportional action responds directly to the control deviation by increasing in proportion with it. The integral action gradually increases the output value over time until the control deviation is fully balanced. With the rate of change in control deviation, the derivative action increases, helping in the rapid correction of the setpoint. It decreases as the rate of change drops in order to ensure stability. During pretuning, PID_Compact calculates the P, I, and D parameters for the controlled system, and during fine-tuning, PID_Compact is used to further tune the parameters.

The changes in the setpoint may weaken the proportional action weighting and are adjustable between 0.0 and 1.0. At 1.0, the proportional action is fully effective for setpoint change, while at 0.0, it has no effect. However, when the process value is changed, the proportional action remains fully effective. The changes in the setpoint may weaken the derivative action weighting and are adjustable between 0.0 and 1.0. At 1.0, the derivative action is fully effective for setpoint change, while at 0.0, it has no effect. However, when the process value is changed, the derivative action remains fully effective [35].

After completing other parts of the software according to the working scenario, PID Compact in the TIA Portal interface was used for voltage control. The configuration window within the PID Compact was opened, and the basic settings were configured. The controller type was set to voltage. In the Input/Output Parameters section, Input_PER

(analog) and Output_PER (analog) were selected because the output voltage is continuously measured through the analog output of a power analyzer, and the control signal for the voltage-regulating DC converter is also analog. The setpoint was configured as 220 V. The generator output voltage was continuously measured as an effective (RMS) value. In the PID Parameters section, the parameters were left at their default values, as shown in Figure 6. These default values were used as the initial parameter settings.

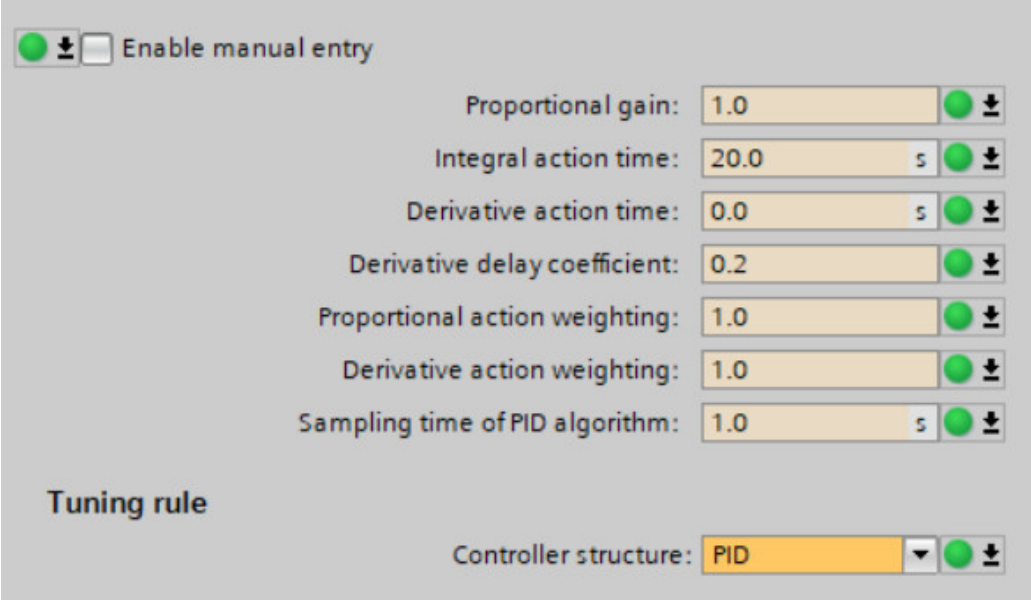


Figure 6. Default (initial) values of PID parameters.

After completing the configuration settings mentioned above, the real/physical system was commissioned. The commissioning window in PID Compact was opened, and both the pretuning and fine-tuning processes were carried out accordingly. While the system was running, the appropriate K_p, K_i, and K_d values for the system were automatically calculated on the PLC side after the pretuning and fine-tuning processes. Following this, the system was controlled based on the K_p, K_i, and K_d values calculated by the PLC. The values automatically calculated by the PLC after the pretuning and fine-tuning processes are shown in Figure 7.

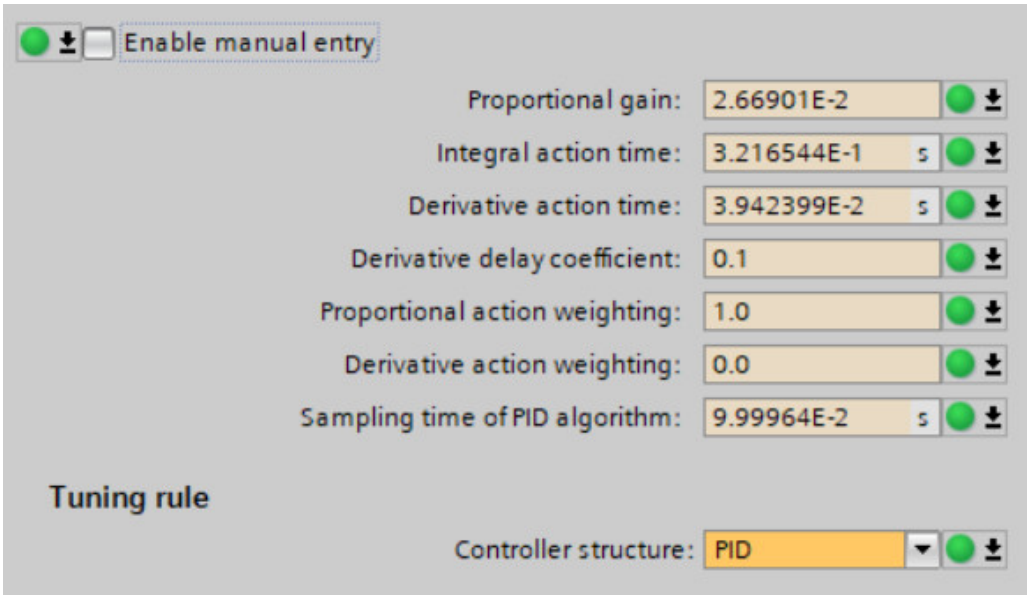


Figure 7. PID parameter values calculated by PLC.

3. Background of the Proposed Method

This section introduces the proposed method for the AVR system operating in a real-world environment. The method combines the effective performance of the fuzzy logic in control problems with the efficiency of the adaptive artificial bee colony (aABC) algorithm in numerical optimization problems (NOPs). In this context, the dynamic fuzzy system that tunes the PID parameters achieves the balance between accuracy and complexity through aABC [36], which is an improved version of the classical ABC [37]. This metaheuristic approach aims to create a robust model.

In studies that adjust PID parameters with the fuzzy logic system, a classical fuzzy-PID structure is generally preferred. However, in some studies, a fuzzy logic controller modeled with metaheuristic algorithms is used for the dataset obtained by system simulation, and the system is controlled by a static controller at runtime. In the proposed method, an adaptive fuzzy logic controller has been used, allowing the system to adapt to the environment in a shorter time period.

In [38], the fuzzy logic system and metaheuristic algorithms were used together for adaptive PID control. PID parameters were updated at runtime with a metaheuristic algorithm, while a fuzzy logic system was used to increase the efficiency of the algorithm. In the proposed method, a more robust fuzzy-PID modeled with a metaheuristic method has been created for adaptive control of the system.

In this study, the fuzzy logic system parameters are normalized to the range of [0, 1], and the metaheuristic method adjusts the membership function parameters of the fuzzy logic based on this sensitivity. Hence, the proposed method can also be adapted to the control of other nonlinear systems [39–43].

The proposed method is coded in MATLAB, and the application runs on a machine with 32 GB of RAM, an i7-1165MG7 2.80 processor, and a Windows 10 operating system.

The method aims to optimize the PID parameters (K_p, K_i and K_d) by considering the *error* (*e*) of the result produced by the system and the *change in the error* (*ce*) compared to the previous step, with the approach widely used in fuzzy-PID designs. To enhance the efficiency of the method, fuzzy variables were defined in the range of [0, 1], and values were assigned to the membership function (*MF*) parameters of these variables within the same range. Therefore, the lower and upper limits for aABC, which searches for the optimum values for these parameters, were determined as 0 and 1, respectively. In the preliminary stage of this study, the system was operated with random loads, and the errors and adjustment time data obtained with different PID parameters were saved for further analysis. The dataset prepared with approximately 500 experiments was analyzed, and the minimum and maximum values were determined for each variable (input–output) as presented in Table 1. Then, the dataset was normalized to the range of [0, 1].

Table 1. Minimum and maximum values for variables in the dataset.

	e	***ce***	K_p	K_i	K_d
Min	−120	−120	0	0	0
Max	120	120	5	3	1

The error and the change in error can take negative values. However, for more efficient design, it is advantageous to restrict the values of fuzzy variables and metaheuristic methods to the range of [0, 1]. In this context, note that the minimum values in columns *e* and *ce* in Table 1, which were used for scaling, are negative.

The proposed method is illustrated in Figure 8.

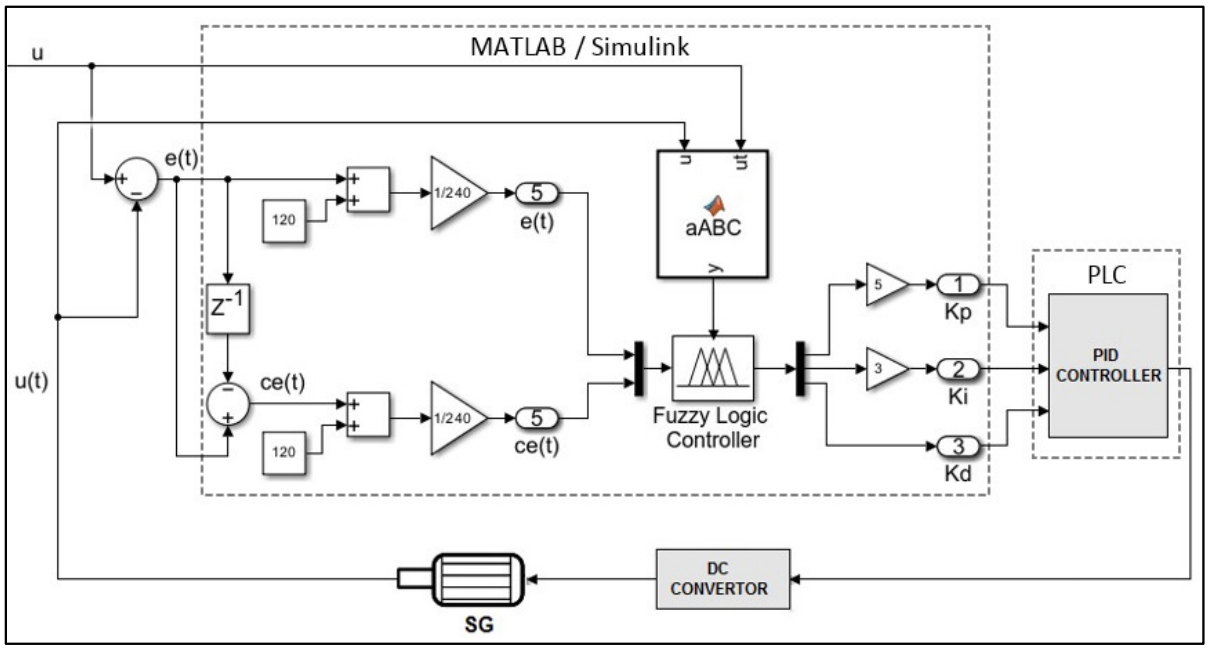


Figure 8. Block schema of the aABC-fuzzy-PID.

As shown in Figure 8, the output of the system is compared with the reference value, determined as 220 V, and the error is calculated accordingly. This error is then input into the aABC algorithm, which searches for the fuzzy logic parameters that optimize the PID parameters to minimize the error in the next step, thereby achieving a lower error than expected. After the new fuzzy model design, the current error and error change values are scaled to the [0, 1] range and input to the fuzzy model. The PID parameter values determined by the fuzzy model are then multiplied by the necessary coefficients and assigned to the PID controller. The details of the fuzzy logic and aABC techniques used in this method are described as follows.

3.1. Fuzzy Logic Side

In the fuzzy logic (FL) designed to tune the PID parameters of the method, *the error* (*e*) and *the change in error* (*ce*) are used as input variables, while the PID parameters to be tuned for these inputs (K_p, K_i, and K_d) are used as output variables. The Mamdani inference system was used for the relationship analysis between the inputs and outputs. The fuzzy model structure used in the method is presented in Figure 9.

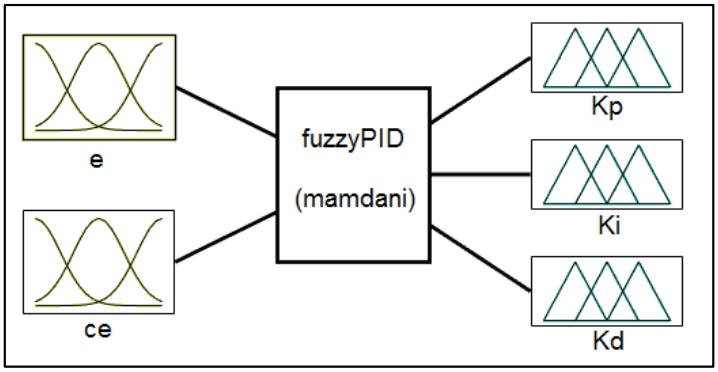
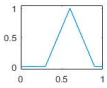
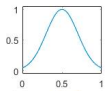
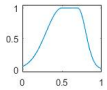


Figure 9. Structure of the FL side.

Each variable in the fuzzy model contains MFs named negative large (*NL*), negative medium (*NM*), negative small (*NS*), zero (*ZE*), positive small (*PS*), positive medium (*PM*), and positive large (*PL*). All MFs are designed in the same type as a triangle (*trimf*), Gauss

(*gaussmf*), and combined Gauss (*gauss2mf*), and the system performance is tested with these types in the preliminary process. The triangle type is an MF type commonly used in fuzzy-PID systems that can provide successful solutions. Its function has three parameters. The Gauss type has smooth transitions instead of sharp curves like the triangle type. In its function, the right and left sides of the MF are symmetrical to each other. In addition, the function has two parameters; thus, it is advantageous in terms of parameter optimization. The combined Gauss type is the combined form of two different Gauss types. The curves on the right and left sides of the MF are independent of each other. In this context, the function is more flexible than the triangle and Gaussian-type MFs, and more successful solutions have been obtained in the studies [44]. Details of the selected MFs are shown in Table 2.

Table 2. Membership functions used in the method.

Name	Shape	Formula
Trimf		$f(x;a,b,c) = \begin{cases} 0, & x \le a \\ \frac{x-a}{b-a}, & a \le x \le b \\ \frac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases}$
Gaussmf		$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$
gauss2mf		

The mathematical formulations of MFs are given in the formula column in Table 2 as *trimf* parameters: *a*, the starting point of the *triangle*; *b*, the value at which the membership degree reaches 1; and *c*, the third point at which the triangle is completed. In the *gaussmf* and *gauss2mf* functions, σ represents the standard deviation, and c represents the mean for the Gaussian function. For *gauss2mf*, the left curve of the function graph is determined by $\sigma 1$ and $c1$, while the right curve is determined by $\sigma 2$ and $c2$ (MATLAB code: $y = gauss2mf(x,[\sigma 1,c1,\sigma 2,c2])$) [45] (details are available in MATLAB R2021a).

The fuzzy model was designed as a rule-based system, and the rule table in Table 3 was used for each output. In the model tuning process, the rules were kept constant, while the optimal values were searched for MF parameters.

Table 3. Rule table used for all outputs of the system.

K_p, K_i, K_d		***e***						
		NL	***NM***	***NS***	***ZE***	***PS***	***PM***	***PL***
ce	***NL***	PL	PL	PL	PL	PM	ZE	ZE
	NM	PL	PL	PL	PL	PM	ZE	ZE
	NS	PM	PM	PM	PS	ZE	NM	NM
	ZE	PM	PM	PS	ZE	NS	NM	NM
	PS	PM	PM	ZE	NS	NM	NM	NM
	PM	ZE	ZE	NM	NL	NL	NL	NL
	PL	ZE	ZE	NM	NL	NL	NL	NL

3.2. Metaheuristic Side

On the fuzzy logic side of the proposed method, system performance is improved by optimizing the MF parameters while keeping the rules constant. In this context, a metaheuristic approach that can effectively adjust the numerical parameters, effectively change the numerical parameters, efficiently scan the search space, and offer fast convergence is required. Therefore, the aABC algorithm is preferred for tuning the fuzzy system

in the proposed method. The standard ABC is a metaheuristic method developed for NOPs [37]. It is favored over other metaheuristic methods due to its advantage of fewer parameters and easy applicability. The method has been successfully applied to NOP solutions in different fields of engineering with its original model, improved versions, and extensions [46,47]. The algorithm has also provided successful solutions in optimizing different artificial intelligence methods [48–50]. The aABC is an enhanced version of ABC that dynamically updates the ABC parameters during the search process, thus providing the algorithm with a more effective exploitation–exploration balance. In [36], the weights of the artificial neural network were optimized using the aABC algorithm. In this way, the efficiency of the algorithm for NOP was tested on an artificial intelligence technique by comparing it with the state-of-the-art methods.

In the standard ABC algorithm, there are two parameters, "*food*" and "*limit*". The food parameter represents the number of random solutions to be generated initially. The number of artificial employed bees and artificial onlooker bees that carry out the local search process is equal to the number of foods. The limit parameter indicates how many times the current solution can be used in the process of generating a new solution. An equal number of limit values is assigned to each food (any possible solution in the searching area). In the artificial employed bee and artificial onlooker bee procedures, if the new solution is better than the current one, the current solution in the solution matrix is replaced with the new solution, and the counter is reset for this new solution. If no better solution can be derived with the current solution in a limited number of trials, the current solution is removed from the solution matrix and a new random solution is generated.

In the aABC algorithm, the search is directed by considering the fitness values of the solutions. The fitness value ($fit(X_i)$) for solution i is calculated as shown in Equation (2).

$$fit(X_i) = \begin{cases} \frac{1}{1+f(X_i)} & if(f(X_i) \geq 0) \\ \frac{1}{1+abs(f(X_i))} & otherwise \end{cases} \quad (2)$$

Here, $f(X_i)$ represents the value calculated with the objective function of the problem for solution X_i.

In the standard ABC, an equal number of limits is assigned to each solution. In this context, if a better solution cannot be derived from any of the current solutions, the $limit_{sum} = food * limit$ number of trials is targeted. The same number of trials is targeted in the aABC algorithm, as well. However, in order to avoid spending excessive time on unsuccessful solutions and to be able to effectively scan the neighborhood of successful solutions, the total limit ($limit_{sum}$) is distributed according to the fitness values of the solutions. Therefore, in the aABC algorithm, the limits for all existing solutions are updated at each iteration. The current limit for solution i is calculated using Equation (3).

$$limit(i) = \frac{fit(X_i) * food * D}{\sum_{j=1} fit(X_j)} \quad (3)$$

Here, D represents the number of items (dimension) of any solution vector.

One of the main differences between aABC and ABC is that aABC adopts a greedy approach in the artificial onlooker bee procedure. In aACB, after the artificial employed bee phase is completed, the average fitness value for the existing solutions is calculated using Equation (4).

$$fit_{avg} = \frac{\sum_{i=1}^{food} fit(X_i)}{food} \quad (4)$$

In the artificial onlooker bee phase, solutions with better fitness values than the average fitness value are used when deriving new solutions.

The solution space scanning approach and solution generation strategy of the proposed method were tested on single-objective numerical optimization problems, and widely used

CEC 2019 benchmark functions were selected for statistical tests of aABC. CEC 2019 is a benchmark set known as the "100-Digit Challenge". The set consists of ten multimodal test functions that represent complex optimization problems. All functions are minimization functions and are scalable. Some functions (CEC01-CEC03) are taken as they are, while some (CEC04-CEC10) are shifted and rotated. Additionally, the functions allow contestants to adjust up to two parameters. Table 4 presents the solution search range, dimensions (D), and global optimum (best) for each CEC 2019 test problem.

Table 4. CEC2019 benchmark functions.

Id	Formulation	Range	D	Best
CEC01	Storn's Chebyshev Polynomial Fitting Problem	[−8192, 8192]	9	1
CEC02	Inverse Hilbert Matrix Problem	[−16,384, 16,384]	16	1
CEC03	Lennard–Jones Minimum Energy Cluster	[−4, 4]	18	1
CEC04	Rastrigin's Function	[−10, 10]	10	1
CEC05	Griewangk's Function	[−10, 10]	10	1
CEC06	Weierstrass Function	[−10, 10]	10	1
CEC07	Modified Schwefel's Function	[−10, 10]	10	1
CEC08	Expanded Schaffer's F6 Function	[−10, 10]	10	1
CEC09	Happy Cat Function	[−10, 10]	10	1
CEC10	Ackley Function	[−10, 10]	10	1

The performance of the aABC was compared with the results obtained by the algorithms in [51] for the same set of functions. The algorithm's performance was compared with various optimization algorithms, including the flower pollination algorithm for global optimization (FPA) [51], the grey wolf optimizer (GWO) [52], the Salp swarm algorithm (SSA) [53], the cuckoo search (CS) [54], the sine cosine algorithm (SCA) [55], and the adaptive flower pollination algorithm (AFPA) [56].

For a fair comparison, the parameter set of the aABC was adjusted to match that in [53]. The value of "food" for the ABC and aABC was set to 15 (note that in ABC algorithms, the colony size consists of the number of "artificial employed bees" and "artificial onlooker bees" equal to "food"). The "limit" is calculated as food*D in the ABC and aABC.

The ABC and aABC algorithms were run 30 times independently for each function, and the maximum cycle number (MCN) in each experiment was set to 500.

The outcomes of all competitive algorithms were evaluated based on the best, worst, mean, and standard deviation (SD) values in 30 independent experiments for each function. The results obtained by the algorithms are presented in Table 5.

Table 5. Results of competitive algorithms for CEC2019 benchmarks.

Function	Alg.	Best	Worst	Mean	SD
CEC01	FPA	1.66×10^{8}	1.34×10^{9}	4.94×10^{8}	2.36×10^{8}
	GWO	4.04×10^{4}	1.22×10^{9}	1.17×10^{8}	2.39×10^{8}
	SSA	1.87×10^{8}	4.03×10^{10}	7.73×10^{9}	1.04×10^{10}
	CS	1.00×10^{10}	1.00×10^{10}	1.00×10^{10}	0.00×10^{0}
	SCA	1.60×10^{8}	2.08×10^{10}	6.90×10^{9}	6.70×10^{9}
	AFPA	1.05×10^{6}	1.75×10^{8}	2.52×10^{7}	3.59×10^{7}
	ABC	1.90×10^{7}	2.58×10^{9}	5.78×10^{8}	1.12×10^{8}
	aABC	1.76×10^{7}	2.02×10^{9}	3.91×10^{8}	6.51×10^{7}
CEC02	FPA	1.80×10^{1}	3.18×10^{1}	2.14×10^{1}	3.35×10^{0}
	GWO	1.73×10^{1}	1.74×10^{1}	1.73×10^{1}	2.11×10^{-2}
	SSA	1.73×10^{1}	1.73×10^{1}	1.73×10^{1}	4.75×10^{-4}
	CS	1.73×10^{1}	1.73×10^{1}	1.73×10^{1}	2.86×10^{-4}
	SCA	1.74×10^{1}	1.77×10^{1}	1.75×10^{1}	8.77×10^{-2}
	AFPA	1.73×10^{1}	1.73×10^{1}	1.73×10^{1}	2.80×10^{-5}
	ABC	1.73×10^{1}	1.73×10^{1}	1.73×10^{1}	3.13×10^{-6}
	aABC	1.73×10^{1}	1.73×10^{1}	1.73×10^{1}	1.05×10^{-8}

Table 5. *Cont.*

Function	Alg.	Best	Worst	Mean	SD
CEC03	FPA	1.27×10^{1}	1.27×10^{1}	1.27×10^{1}	1.56×10^{-7}
	GWO	1.27×10^{1}	1.27×10^{1}	1.27×10^{1}	2.01×10^{-6}
	SSA	1.27×10^{1}	1.27×10^{1}	1.27×10^{1}	2.93×10^{-12}
	CS	1.27×10^{1}	1.27×10^{1}	1.27×10^{1}	9.81×10^{-11}
	SCA	1.27×10^{1}	1.27×10^{1}	1.27×10^{1}	1.00×10^{-4}
	AFPA	1.27×10^{1}	1.27×10^{1}	1.27×10^{1}	2.78×10^{-14}
	ABC	1.27×10^{1}	1.27×10^{1}	1.27×10^{1}	3.07×10^{-6}
	aABC	1.27×10^{1}	1.27×10^{1}	1.27×10^{1}	2.11×10^{-10}
CEC04	FPA	9.12×10^{1}	1.98×10^{2}	1.26×10^{2}	2.54×10^{1}
	GWO	2.35×10^{1}	1.43×10^{3}	1.60×10^{2}	3.41×10^{2}
	SSA	1.19×10^{1}	7.86×10^{1}	3.74×10^{1}	1.85×10^{1}
	CS	3.67×10^{1}	5.82×10^{2}	1.94×10^{2}	1.19×10^{2}
	SCA	4.94×10^{2}	2.25×10^{3}	1.40×10^{3}	4.08×10^{2}
	AFPA	2.06×10^{1}	4.09×10^{1}	2.92×10^{1}	5.41×10^{0}
	ABC	5.97×10^{1}	2.85×10^{2}	3.33×10^{2}	2.77×10^{2}
	aABC	1.19×10^{1}	6.92×10^{1}	2.98×10^{1}	1.99×10^{1}
CEC05	FPA	1.35×10^{0}	1.80×10^{0}	1.60×10^{0}	9.38×10^{-2}
	GWO	1.05×10^{0}	1.85×10^{0}	1.41×10^{0}	2.72×10^{-1}
	SSA	1.04×10^{0}	1.74×10^{0}	1.24×10^{0}	1.44×10^{-1}
	CS	1.04×10^{0}	2.00×10^{0}	1.44×10^{0}	2.50×10^{-1}
	SCA	2.06×10^{0}	2.38×10^{0}	2.21×10^{0}	7.55×10^{-2}
	AFPA	1.03×10^{0}	1.08×10^{0}	1.06×10^{0}	1.25×10^{-2}
	ABC	1.07×10^{0}	1.81×10^{0}	1.47×10^{0}	2.38×10^{-1}
	aABC	1.03×10^{0}	1.09×10^{0}	1.10×10^{0}	1.72×10^{-2}
CEC06	FPA	8.94×10^{0}	1.16×10^{1}	1.04×10^{1}	7.77×10^{-1}
	GWO	7.78×10^{0}	1.20×10^{1}	1.06×10^{1}	9.26×10^{-1}
	SSA	1.71×10^{0}	8.55×10^{0}	4.69×10^{0}	1.76×10^{-0}
	CS	7.73×10^{0}	1.01×10^{1}	9.18×10^{0}	6.53×10^{-1}
	SCA	9.20×10^{0}	1.18×10^{1}	1.08×10^{1}	6.45×10^{-1}
	AFPA	8.25×10^{0}	1.16×10^{1}	1.03×10^{1}	7.95×10^{-1}
	ABC	9.71×10^{0}	8.89×10^{0}	1.01×10^{1}	3.76×10^{-0}
	aABC	3.07×10^{0}	7.19×10^{0}	4.11×10^{0}	1.74×10^{-0}
CEC07	FPA	2.65×10^{1}	4.93×10^{2}	2.84×10^{2}	8.92×10^{1}
	GWO	-1.34×10^{2}	1.02×10^{3}	4.69×10^{2}	3.61×10^{2}
	SSA	-1.27×10^{2}	7.61×10^{2}	2.69×10^{2}	2.24×10^{2}
	CS	-1.50×10^{2}	3.63×10^{2}	1.41×10^{2}	1.17×10^{2}
	SCA	3.66×10^{2}	1.05×10^{3}	7.04×10^{2}	1.70×10^{2}
	AFPA	-4.56×10^{2}	1.81×10^{2}	7.83×10^{1}	6.45×10^{1}
	ABC	1.06×10^{2}	1.15×10^{3}	6.44×10^{2}	9.13×10^{2}
	aABC	-1.46×10^{2}	1.51×10^{2}	1.13×10^{2}	5.55×10^{2}
CEC08	FPA	4.96×10^{0}	6.24×10^{0}	5.66×10^{0}	3.27×10^{-1}
	GWO	3.39×10^{0}	6.83×10^{0}	5.09×10^{0}	9.83×10^{-1}
	SSA	2.49×10^{0}	6.38×10^{0}	5.05×10^{0}	7.80×10^{-1}
	CS	4.59×10^{0}	5.80×10^{0}	5.28×10^{0}	3.13×10^{-1}
	SCA	4.68×10^{0}	6.62×10^{0}	6.01×10^{0}	5.02×10^{-1}
	AFPA	2.94×10^{0}	5.63×10^{0}	4.86×10^{0}	8.36×10^{-1}
	ABC	3.01×10^{0}	7.22×10^{0}	5.91×10^{0}	6.07×10^{-1}
	aABC	2.38×10^{0}	5.10×10^{0}	4.11×10^{0}	2.41×10^{-1}
CEC09	FPA	3.70×10^{0}	6.28×10^{0}	4.72×10^{0}	6.46×10^{-1}
	GWO	2.61×10^{0}	7.70×10^{0}	4.52×10^{0}	1.02×10^{0}
	SSA	2.36×10^{0}	2.73×10^{0}	2.49×10^{0}	9.24×10^{-2}
	CS	2.56×10^{0}	3.09×10^{0}	2.78×10^{0}	1.31×10^{-1}
	SCA	1.11×10^{1}	3.89×10^{2}	1.09×10^{2}	8.22×10^{1}
	AFPA	2.42×10^{0}	4.72×10^{0}	3.02×10^{0}	5.23×10^{-1}
	ABC	2.81×10^{0}	4.08×10^{2}	1.29×10^{2}	6.13×10^{1}
	aABC	2.40×10^{0}	3.11×10^{0}	2.25×10^{0}	4.84×10^{-1}

Table 5. *Cont.*

Function	Alg.	Best	Worst	Mean	SD
CEC10	FPA	2.02×10^{1}	2.06×10^{1}	2.04×10^{1}	9.49×10^{-2}
	GWO	2.03×10^{1}	2.06×10^{1}	2.05×10^{1}	8.86×10^{-2}
	SSA	2.00×10^{1}	2.03×10^{1}	2.00×10^{1}	8.87×10^{-2}
	CS	2.01×10^{1}	2.03×10^{1}	2.03×10^{1}	5.31×10^{-2}
	SCA	2.03×10^{1}	2.07×10^{1}	2.05×10^{1}	7.70×10^{-2}
	AFPA	1.01×10^{1}	2.06×10^{1}	1.98×10^{1}	2.35×10^{0}
	ABC	2.03×10^{1}	2.07×10^{1}	2.04×10^{1}	6.09×10^{2}
	aABC	1.99×10^{1}	2.04×10^{1}	2.07×10^{1}	3.03×10^{1}

When Table 5 is evaluated from a general perspective, it is observed that the aABC algorithm achieves better results than other metaheuristics for CEC2019 test problems. The aABC produced the best solutions in five out of ten samples (CEC02, CEC03, CEC04, CEC05, and CEC08). It produced the second-best solutions for functions CEC06, CEC09, and CEC10 and the third-best solutions for functions CEC01 and CEC07. In this regard, it ranks the best overall, with an average of 1.7.

For parameter analysis, the aABC algorithm was tested on the CEC01 sample from CEC2019 benchmark functions with different parameter combinations. Since the algorithm runs in online mode with a small colony size, small values were assigned to the "food" parameter of the algorithm in the parameter analysis. The values 3, 5, and 7 and 10, 20, and 50 were assigned to the "food" and "limit" parameters, respectively. The obtained results are shown in Table 6.

Table 6. Results obtained for parameter analysis of CEC01.

Food	Limit	Best	Worst	Mean	SD
3	10	9.73×10^{10}	1.17×10^{12}	7.91×10^{11}	3.34×10^{10}
	20	3.04×10^{10}	4.89×10^{11}	9.07×10^{10}	1.01×10^{9}
	50	2.41×10^{10}	4.73×10^{11}	7.03×10^{10}	1.04×10^{9}
5	10	1.40×10^{10}	2.22×10^{11}	8.21×10^{10}	3.81×10^{9}
	20	2.17×10^{9}	5.74×10^{10}	6.63×10^{9}	7.48×10^{8}
	50	2.21×10^{9}	5.05×10^{10}	7.87×10^{9}	8.03×10^{8}
7	10	5.11×10^{9}	1.01×10^{10}	9.19×10^{9}	1.00×10^{9}
	20	2.46×10^{9}	7.47×10^{9}	4.12×10^{9}	4.11×10^{9}
	50	2.19×10^{9}	8.02×10^{9}	5.01×10^{9}	6.97×10^{9}

When the results in Table 6 are examined, it is observed that the aABC algorithm produces better results with the parameter values of food = 5 and limit = 20 compared to other parameter combinations.

3.2.1. Solution Vector

Since the fuzzy rules in the FL part of the method are fixed, only the optimal parameters for the membership functions are searched on the metaheuristic part. The previous section discussed the fuzzy model with *trimf*, *gaussmf*, and *gauss2mf* MF types, which were designed and tested in the preliminary study. From Table 2, which provides the details of the selected MF types, it is evident that each MF type has a different number of parameters. Therefore, solution vectors of various lengths were designed for each fuzzy model in the experiments. Figure 10 explains the vector designed for the fuzzy model with *gaussmf* membership functions in the aABC algorithm.

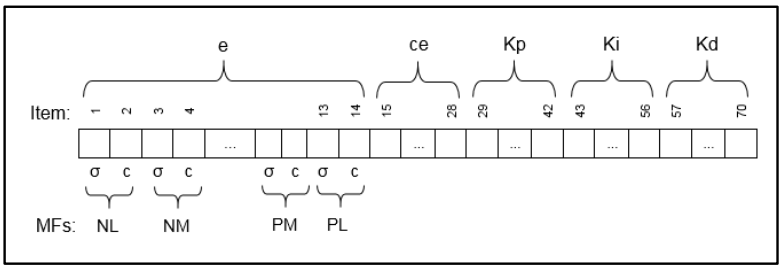


Figure 10. Structure of the solution vector used in the metaheuristic side of the method.

As explained in the previous sub-section, there are five variables in the fuzzy model of the system, two inputs and three outputs, and each variable has seven MFs. Therefore, the length of any solution vector to be generated on the aABC side of the system is "5*7*MF parameter". In this context, vectors with 70, 105, and 140 items were designed for fuzzy models with *gaussmf*, *trimf*, and *gauss2mf* types of MFs, respectively. Figure 8 shows the solution vector structure used in the design of the fuzzy model with *gaussmf*-type MFs.

3.2.2. Objective Function for the Offline Design

In the preliminary study, approximately 500 experimental results obtained with different PID parameters were saved. From these experiments, 200 solutions that achieved the shortest settling time with minimum error were selected. The training set (T) contained 75 samples representing the 200 selected solutions.

The results obtained by the designed fuzzy model for the PID parameters were compared with the targets in the training set (T), and the error was calculated for each sample. The total error (E), which represents the objective function, was calculated using Equation (5).

$$E = \frac{1}{2}\sum_{i=1}^{75} (y_1 - T_{i,3})^2 + (y_2 - T_{i,4})^2 + (y_3 - T_{i,5})^2 \quad (5)$$

Note that the first and second columns in the training set are the *error* (*e*) and *change in error* (*ce*), respectively.

3.2.3. Parameter Settings for the Offline Design

The aABC algorithm was executed 30 times independently for each version of the fuzzy model with different MF types using the parameters in Table 7.

Table 7. Parameter setting of aABC for the offline design of fuzzy-PID.

Name	Value
food	10
limit	100

In each experiment, the maximum number of cycles was set to 500. The minimum, maximum, mean, and standard deviation results of the errors obtained in 30 trials were recorded for each fuzzy model.

3.3. Experiments for the Offline Design

The proposed method was tested to design the optimal fuzzy-PID model according to the prepared dataset. In each experiment, the maximum number of cycles was set to 500, and the *minimum*, *maximum*, *mean*, and *standard deviation* results of the errors obtained in 30 trials for each fuzzy model were recorded accordingly. The results are presented in Table 4. Since the system was designed according to the values normalized in the range of [0, 1], the results in Table 8 represent the scaled values of the errors within that range.

Table 8. Results for the offline experiments.

	Min	Max	Mean	SD
trimf	0.0471	0.0516	0.0486	0.0019
gaussmf	0.0452	0.0473	0.0465	0.0006
gauss2mf	0.0464	0.0523	0.0484	0.0022

When the results in Table 5 are evaluated, it can be seen that the most successful results from the perspective of min, max, mean, and SD are obtained when the fuzzy model has *gaussmf*-type MFs. Since the *gaussmf* MF has fewer parameters than the others, it is easier to optimize the fuzzy model with this type of MF. In contrast, *trimf* has sharper curves than the other types. The solution space of a problem consisting of double solutions in the range of [0, 1] can be considered to have a smooth transition surface. *gauss2mf* is more advantageous than the *gaussmf* function in terms of optimization. Therefore, the aABC model with a larger population is thought to produce better solutions for *gauss2mf*-type MFs by executing more cycles. However, in an online optimization, the goal is to obtain better solutions in a shorter time, typically with a smaller population. Preliminary studies show that more successful solutions can be obtained with *gaussmf*. As a result, the fuzzy model with *gaussmf*-type MFs is preferred in online optimization.

3.4. Online Model of aABC-FL

Offline experiments conducted in the preliminary phase of this study provided valuable insights into the performance of the proposed fuzzy model and the aABC algorithm. The online model of the method was designed based on these findings. In this context, the online objective function for the aABC algorithm was defined, and the parameter design was determined to obtain results in a shorter time.

3.4.1. Online Objective Function for aABC-FL

In this study, the integral of the absolute value of error (IAE) technique, which considers recent errors, was selected as the online performance criterion for the aABC [57]. The IAE technique, which can equally weight recent errors and transient response, is widely preferred in the design of fuzzy systems with metaheuristic methods. If the IAE value calculated using Equation (5) for the solution vector is better than the previous IAE value, the fuzzy model is redesigned according to this vector.

$$IAE = \int_0^t |u - u(t)|dt = \int_0^t |e(t)|dt \tag{6}$$

Here, U is the desired voltage, and $u(t)$ is the actual voltage produced by the system. Therefore, in the online process, the function value of the solution i ($f(X_i)$) used in Equation (1) is the IAE value calculated with Equation (6).

3.4.2. Parameter Settings for the Online Design of aABC-FL

In the designed system, the response time from the PLC is slightly over 100 milliseconds. Therefore, the total time for solution generation by aABC and the redesign time of the fuzzy model should be completed in less than 100 milliseconds for each sample. Considering these criteria, the online parameter design for aABC is provided in Table 9.

Table 9. Parameter setting of aABC for the online design of fuzzy-PID.

Name	Value
food	5
limit	20

4. Experiment Results and Discussions

4.1. Experiment Results

Numerous experiments have been conducted on the MHPP, which was installed as an experimental setup, and the obtained results have been logged and converted into graphs. The graphs were obtained for both cases: when SG1 operates alone and when all three SGs operate together in synchronization. This section presents the results of the experiments involving sudden and large load fluctuations (with loads of 50%, 75%, and 100% of the nominal power of the SG) that significantly change the terminal voltages of the SGs, instead of small-power load fluctuations that have a very small effect on the terminal voltage value. In these experiments, the loads were switched on at the t = 0 point and in the pulse transition form.

The results of the experiments from no load to 500 W load (0–50% loading) and from 500 W load to no load (load shedding) are depicted in Figures 11 and 12, respectively. In the experiment with conventional PID (PLC-PID), the minimum voltage was 200.2 V (max undershoot was 19.8 V), and the maximum voltage was 234.3 V (max overshoot was 14.3 V). The voltage reached the set value in 10 and 13 s, respectively, when controlled by the classical PID controller.

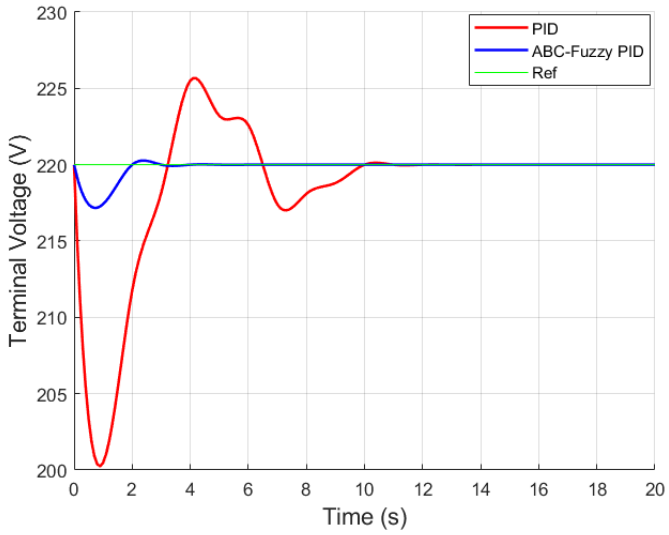


Figure 11. Loading experiment: 0–500 W.

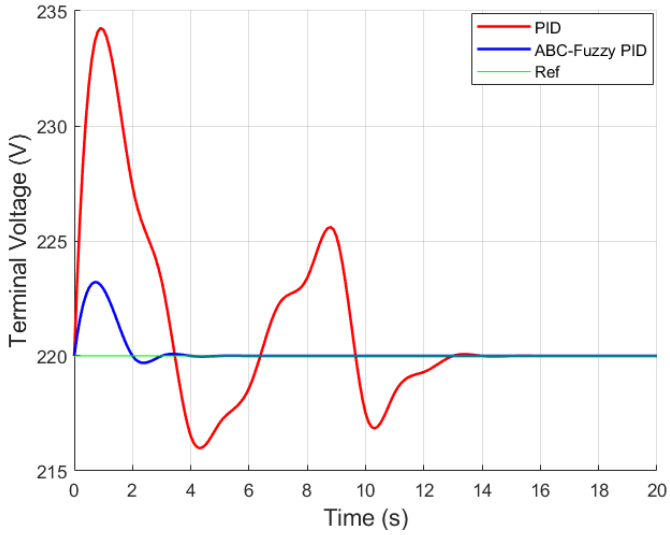


Figure 12. Load shedding experiment: 500–0 W.

In the experiment using the aABC-fuzzy-PID, the minimum voltage was 217.2 V (max undershoot was 2.8 V), and the maximum voltage was 223.2 V (max overshoot was 3.2 V). The voltage value reached the set value in 2 and 3 s in the loading and load shedding experiments, respectively.

Figures 13 and 14 illustrate the results of the experiments from no load to a 750 W load (0–75% loading) and from a 750 W load to no load (load shedding). In the experiment using conventional PID (PLC-PID), the minimum voltage was 259.7 V (max undershoot was 60.3 V), and the maximum voltage was 253.5 V (max overshoot was 33.5 V). The voltage value reached the set value in 14 and 15 s, respectively. In the experiment with the aABC-fuzzy-PID, the minimum voltage was 213.2 V (max undershoot was 6.8 V), and the maximum voltage was 230.4 V (max overshoot was 10.4 V). The voltage value reached the set value in 3 and 4 s in the loading and load shedding experiments, respectively.

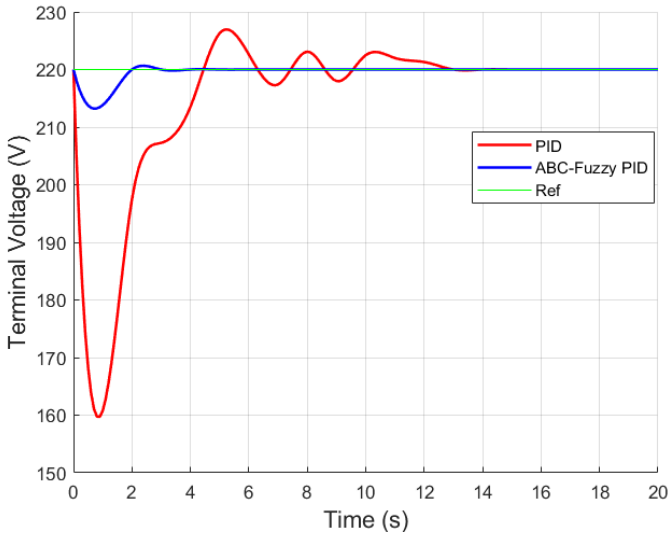


Figure 13. Loading experiment: 0–750 W.

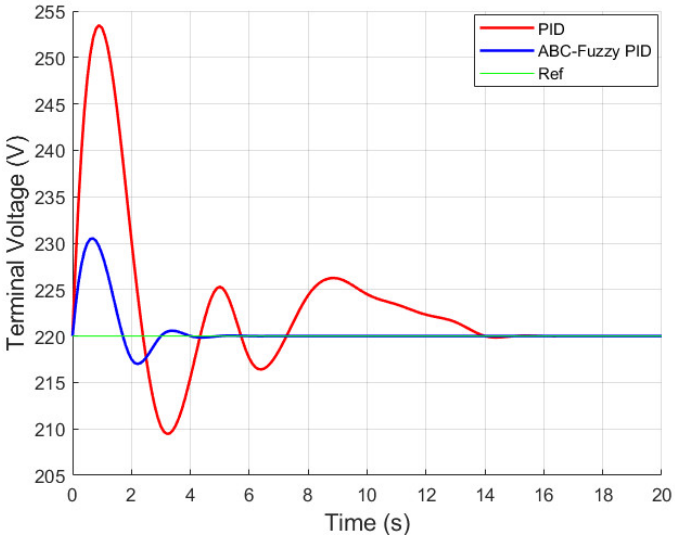


Figure 14. Load shedding experiment: 750–0 W.

The results of the experiments from no load to a 1000 W load (0–100% loading) and from a 1000 W load to no load (load shedding) are shown in Figures 15 and 16, respectively. The minimum voltage in the experiment with conventional PID was 140.6 V (max undershoot was 79.4 V), and the maximum voltage was 295.5 V (max overshoot was 75.5 V). The voltage value reached the set value in 17 and 18 s. In the experiment using the

aABC-fuzzy-PID, the minimum voltage was 205.5 V (max undershoot was 14.5 V), and the maximum voltage was 238 V (max overshoot was 18 V). The voltage value reached the set value in 4 and 4 s, respectively.

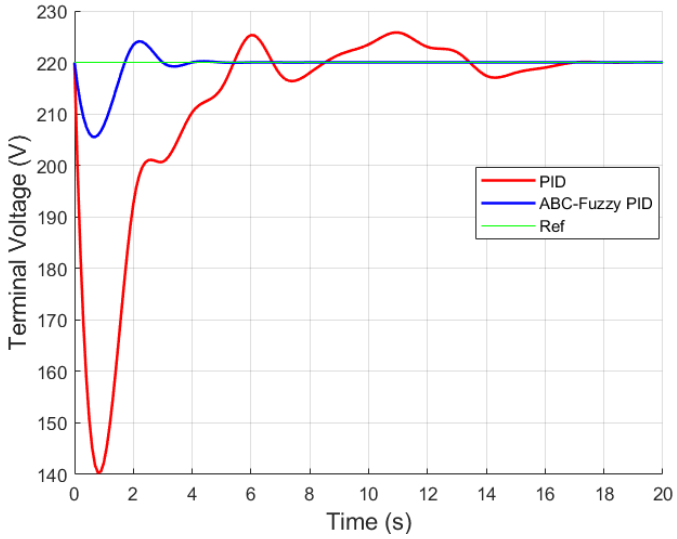


Figure 15. Loading experiment: 0–1000 W.

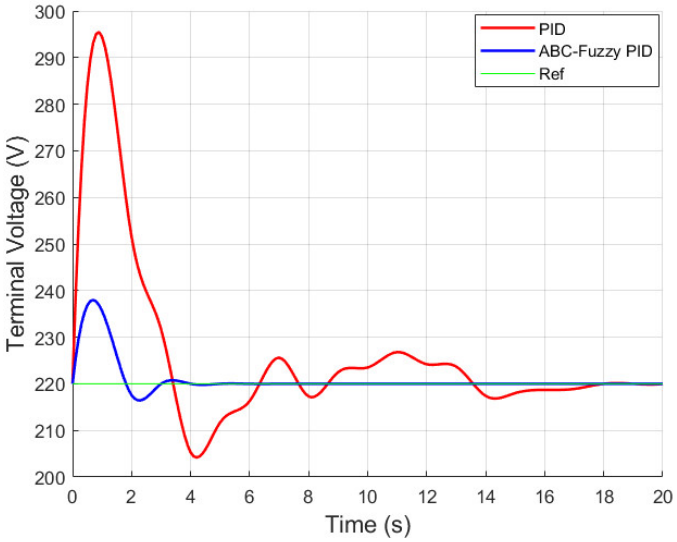


Figure 16. Load shedding experiment: 1000–0 W.

Figures 17 and 18 illustrate the results of the experiments of the synchronized SGs from no load to a 2000 W load (0–67% loading) and from a 2000 W load to no load (load shedding). Using traditional PID, the minimum and the maximum voltages were 169.6 V and 243.6 V (max undershoot and max overshoot were 50.4 V and 23.6 V), respectively. The voltage reached the set value in 13 and 14 s. With the aABC-fuzzy-PID, the minimum and the maximum voltages were 213.4 V and 227.8 V (max undershoot and max overshoot were 6.6 V and 7.8 V), respectively. The voltage reached the set value in 3 and 4 s, respectively.

The results obtained from the traditional PID and aABC-FL-PID in the experiments are given in Tables 10 and 11 for loading and load shedding, respectively.

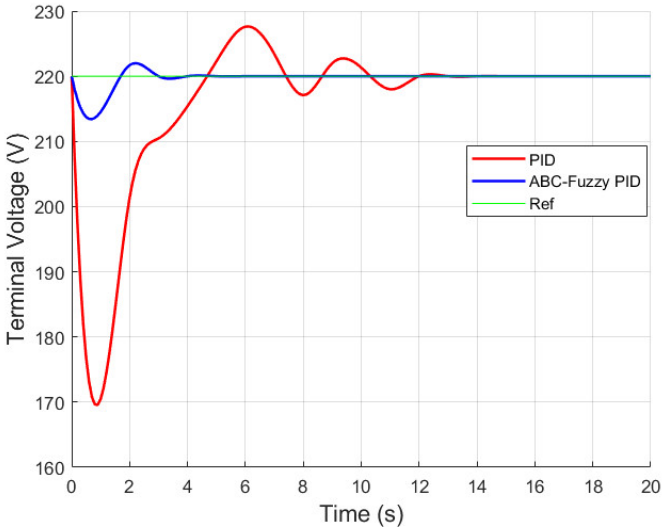


Figure 17. Loading experiment: 0–2000 W (synch).

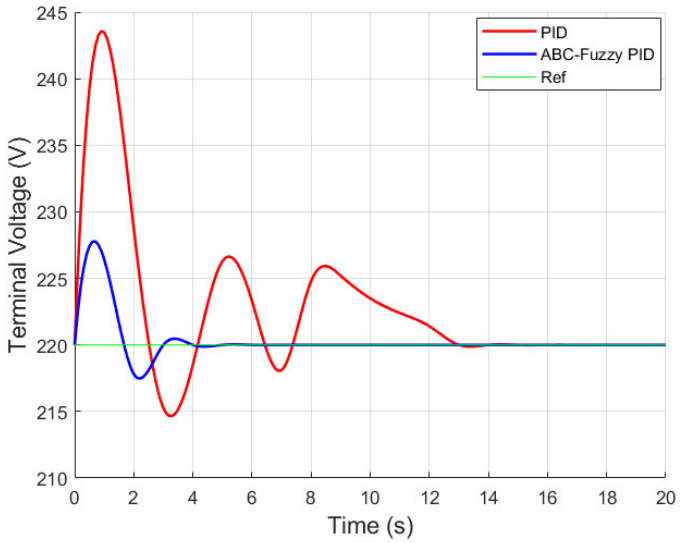


Figure 18. Load shedding experiment: 2000–0 W.

Table 10. PID and aABC-fuzzy-PID comparison for loading experiments.

Experiment	Settling Time (s)		Max Undershoot (V)	
	PID	aABC-FL-PID	PID	aABC-FL-PID
0–500 W	10	2	19.8 (9%)	2.8 (0.7%)
0–750 W	14	3	60.3 (27.4%)	6.3 (3.1%)
0–1000 W	17	4	79.4 (36.1%)	14.5 (6.6%)
0–2000 W (Synch)	13	3	50.4 (22.9%)	6.6 (3%)

Table 11. PID and aABC-fuzzy-PID comparison for load shedding experiments.

Experiment	Settling Time (s)		Max Undershoot (V)	
	PID	aABC-FL-PID	PID	aABC-FL-PID
0–500 W	10	2	19.8 (9%)	2.8 (0.7%)
0–750 W	14	3	60.3 (27.4%)	6.3 (3.1%)
0–1000 W	17	4	79.4 (36.1%)	14.5 (6.6%)
0–2000 W (Synch)	13	3	50.4 (22.9%)	6.6 (3%)

4.2. Discussions

Loading and unloading (load shedding) tests during the experiments were performed under challenging conditions, with sudden and large load fluctuations (50%, 75%, and 100% of the system's nominal power). In real power systems, such sudden load changes at these levels are observed to be rare. However, under more gradual or smaller load fluctuations (such as 10% or 20% of the nominal power), in our system, the output voltage reached the setpoint within 1 s or less.

In addition, according to the IEC 60038 standard, (International Electrotechnical Commission IEC Standard Voltages) voltage fluctuations of ±10% from the nominal voltage are permissible in power systems. Therefore, voltage values in the range of 198 V to 242 V are considered acceptable. Considering these values, even in the most challenging experiment (100% sudden load), the settling time can be considered acceptable if it is less than 3 s. Even in the 100% sudden load experiment conducted with ABC-fuzzy-PID, the voltage value was consistently maintained within the acceptable range. The settling times are often shorter than 1 s. Moreover, the SGs used in the experiments are not commercial generators but are designed for experimental purposes. It is expected that better results could be achieved with commercial SGs using the proposed controller.

5. Conclusions and Future Work

In this study, a novel and robust aABC-fuzzy-PID algorithm is proposed for the PID controller tuning of the AVR system in a micro-hydro power plant installed as an experimental setup. Unlike most studies in the literature, the implementations were conducted using real physical equipment. The real-time control and monitoring of the system were carried out using an S7-1200-type PLC and TIA Portal V17 software. This study demonstrated that the proposed metaheuristic algorithm can be used effectively with an industrial and simple controller, such as a PLC. The optimal *Kp*, *Ki*, and *Kd* values for PID control of the nonlinear AVR system under different load conditions were obtained adaptively and dynamically. PLC-MATLAB communication was established using the OPC UA protocol and the KEPServerEX interface. With the proposed aABC-fuzzy-PID, a significant reduction in settling time and maximum overshoot values were observed when compared to the classical PLC-PID. These results were presented in Figures 11–18 and Tables 10 and 11. The proposed aABC-fuzzy-PID controller holds theoretical and practical significance in both industrial and academic implementations.

In future studies, the application of the proposed control strategy to hybrid and large-scale power plants will be a valuable area of exploration. Event-triggered and predefined-time control methods can also be integrated into the proposed methods in further research. Moreover, the results obtained using different metaheuristic algorithms, multi-objective optimization, and hybrid approaches, like combining MPC or ANN with aABC tuning, will be discussed in future studies. Furthermore, we plan to obtain the equivalent circuits and mathematical model of the system and implement new control approaches using mixed uncertainties or polytopic uncertainty methods [58–60].

Author Contributions: Conceptualization, H.A. and D.E.; Methodology, H.A. and D.E.; Software, H.A. and D.E.; Writing—original draft, H.A. and D.E.; Writing—review & editing, H.A. and D.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Coordinatorship of Research Projects of the Karabuk University. Project Number: KBÜ-BAP-16/1-YL-142.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ayas, M.S. Design of an optimized fractional high-order differential feedback controller for an AVR system. *Electr. Eng.* **2019**, *101*, 1221–1233. [CrossRef]
2. Elsisi, M. Design of neural network predictive controller based on imperialist competitive algorithm for automatic voltage regulator. *Neural Comput. Appl.* **2019**, *31*, 5017–5027. [CrossRef]
3. Moschos, I.; Parisses, C. A novel optimal PIʎDND2N2 controller using coyote optimization algorithm for an AVR system. *Eng. Sci. Technol.* **2022**, *26*, 100991. [CrossRef]
4. Bingul, Z.; Karahan, O. A novel performance criterion approach to optimum design of PID controller using cuckoo search algorithm for AVR system. *J. Frankl. Inst.* **2018**, *355*, 5534–5559. [CrossRef]
5. Sikander, A.; Thakur, P. A new control design strategy for automatic voltage regulator in power system. *ISA Trans.* **2020**, *100*, 235–243. [CrossRef] [PubMed]
6. Oziablo, P.; Mozyrska, D.; Wyrwas, M. Fractional-variable-order digital controller design tuned with the chaotic yellow saddle goatfish algorithm for the AVR system. *ISA Trans.* **2022**, *125*, 260–267. [CrossRef]
7. Munagala, V.K.; Jatoth, R.K. Improved fractional PIλDμ controller for AVRsystem using Chaotic Black Widow algorithm. *Comput. Electr. Eng.* **2022**, *97*, 107600. [CrossRef]
8. Elsisi, M.; Soliman, M. Optimal design of robust resilient automatic voltage regulators. *ISA Trans.* **2021**, *108*, 257–268. [CrossRef]
9. Suid, M.H.; Ahmad, M.A. Optimal tuning of sigmoid PID controller using Nonlinear Sine Cosine Algorithm for the Automatic Voltage Regulator system. *ISA Trans.* **2022**, *128*, 265–286. [CrossRef]
10. Chatterjee, S.; Mukherjee, V. PID controller for automatic voltage regulator using teaching–learning based optimization technique. *Electr. Power Energy Syst.* **2016**, *77*, 418–429. [CrossRef]
11. Mosaad, A.M.; Attia, M.A.; Abdelaziz, A.Y. Whale optimization algorithm to tune PID and PIDA controllers on AVR System. *Ain Shams Eng. J.* **2019**, *10*, 755–767. [CrossRef]
12. Ozgenc, B.; Ayas, M.S.; Altas, I.H. A hybrid optimization approach to design optimally tuned PID controller for an AVR System. In Proceedings of the 2020 International Congress on Human Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 26–27 June 2020. [CrossRef]
13. Ekinci, S.; Hekimoğlu, B. Improved Kidney-Inspired Algorithm Approach for Tuning of PID Controller in AVR System. *IEEE Access* **2019**, *7*, 39935–39947. [CrossRef]
14. Çelik, E. Incorporation of stochastic fractal search algorithm into efficient design of PID controller for an automatic voltage regulator system. *Neural Comput. Appl.* **2018**, *30*, 1991–2002. [CrossRef]
15. Al Gizi, A.J.H. A particle swarm optimization, fuzzy PID controller with generator automatic voltage regulator. *Soft Comput.* **2019**, *23*, 8839–8853. [CrossRef]
16. Gozde, H.; Taplamacioglu, M.C. Comparative performance analysis of artificial bee colony algorithm for automatic voltage regulator (AVR) system. *J. Frankl. Inst.* **2011**, *348*, 1927–1946. [CrossRef]
17. Köse, E. Optimal Control of AVR System With Tree Seed Algorithm-Based PID Controller. *IEEE Access* **2020**, *8*, 89457–89467. [CrossRef]
18. Kuri, R.K.; Paliwal, D.; Sambariya, D.K. Grey Wolf Optimization Algorithm based PID controller design for AVR Power system. In Proceedings of the 2019 2nd International Conference on Power Energy, Environment and Intelligent Control (PEEIC), Greater Noida, India, 18–19 October 2019.
19. Yinggan, T.; Zhao, L.; Han, Z.; Bi, X.; Guan, X. Optimal gray PID controller design for automatic voltage regulator system via imperialist competitive algorithm. *Int. J. Mach. Learn. Cyber* **2016**, *7*, 229–240. [CrossRef]
20. Gozde, H. Robust 2DOF state-feedback PI-controller based on meta-heuristic optimization for automatic voltage regulation system. *ISA Trans.* **2020**, *98*, 26–36. [CrossRef]
21. Zamani, M.; Ghartemani; Sadati, M.K.; Parniani, M. Design of a fractional order PID controller for an AVR using particle swarm optimization. *Cont. Eng. Pract.* **2009**, *17*, 1380–1387. [CrossRef]
22. Izci, D.; Ekinci, S. An improved RUN optimizer based real PID plus second-order derivative controller design as a novel method to enhance transient response and robustness of an automatic voltage regulator. *e-Prime Adv. Electr. Eng. Electron. Energy* **2022**, *2*, 100071. [CrossRef]
23. Verrelli, C.M.; Marino, R.; Tomei, P.; Damm, G. Nonlinear robust coordinated PSS-AVR control for a synchronous generator connected to an infinite bus. *IEEE Trans. Autom. Control* **2021**, *67*, 1414–1422. [CrossRef]
24. Sambariya, D.K.; Gupta, T. Optimal design of PID controller for an AVR system using monarch butterfly optimization. In Proceedings of the IEEE International Conference on Information, Communication, Instrumentation and Control (ICICIC—2017), Indore, India, 17–19 August 2017.
25. Al Gizi, A.J.H.; Mustafa, M.W.; Al-geelani, N.A.; Alsaedi, M.A. Sugeno fuzzy PID tuning, by genetic-neutral for AVR in electrical power generation. *Appl. Soft Comput.* **2015**, *28*, 226–236. [CrossRef]
26. Ortiz-Quisbert, M.E.; Duarte-Mermoud, M.A.; Milla, F.; Castro-Linares, R.; Lefranc, G. Optimal fractional order adaptive controllers for AVR applications. *Electr. Eng.* **2018**, *100*, 267–283. [CrossRef]
27. Gutierrez, D.F.Z.; Rivera, G.H.V.; Cervantes, J.G.A.; Amaya, I.; Duarte, J.M.C. Designing heuristic-based tuners for fractional-order PID controllers in automatic voltage regulator systems using a hyper-heuristic approach. *Fractal Fract.* **2024**, *8*, 223. [CrossRef]

28. Furat, M.; Gücü, G.G. Design, implementation, and optimization of sliding mode controller for automatic voltage regulator system. *IEEE Accsess* **2022**, *10*, 55650–55674. [CrossRef]
29. Çavdar, B.; Sahin, E.; Sesli, E. On the assessment of meta-heuristic algorithms for automatic voltage regulator system controller design: A standardization process. *Electr. Eng.* **2024**, *106*, 5801–5839. [CrossRef]
30. Sivanandhan, A.; Thriveni, G. Optimal design of controller for automatic voltage regulator performance enhancement: A survey. *Electr. Eng.* **2023**, *106*, 3705–3720. [CrossRef]
31. Al Gizi, A.J.H.; Mustafa, M.W.; Al Zaidi, K.M.A.; Al-Zaidi, M.K.J. Integrated PLC-fuzzy PID Simulink implemented AVR system. *Electr. Power Energy Syst.* **2015**, *69*, 313–326. [CrossRef]
32. Dettori, S.; Iannino, V.; Colla, V.; Signorini, A. An adaptive Fuzzy logic-based approach to PID control of steam turbines in solar applications. *Appl. Energy* **2018**, *227*, 655–664. [CrossRef]
33. Lawrence, N.P.; Forbes, M.G.; Loewen, P.D.; McClement, D.G.; Backström, J.U.; Gopaluni, R.B. Deep reinforcement learning with shallow controllers: An experimental application to PID tuning. *Control. Eng. Pract.* **2022**, *121*, 105046. [CrossRef]
34. Briones, O.; Alarcón, R.; Rojas, A.J.; Sbarbaro, D. Tuning Generalized Predictive PI controllers for process control applications. *ISA Trans.* **2022**, *119*, 184–195. [CrossRef]
35. Available online: https://support.industry.siemens.com/cs/document/100746401/pid-control-with-pid_compact-for-simatic-s7-1200-s7-1500?dti=0&lc=en-WW (accessed on 10 May 2024).
36. Hamdan, A.M.K.; Ekmekci, D. Prediction of monkeypox infection from clinical symptoms with adaptive artificial bee colony-based artificial neural network. *Neural Comput. Appl.* **2024**, *36*, 13715–13730. [CrossRef]
37. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization*; Technical Report for Erciyes University: Kayseri, Turkey, 2005.
38. Ahmed, H.; Asarry, A.; Hairuddin, A.A.; Hassan, M.K.; Liu, Y.; Onwudinjo, E.C.U. Online DE Optimization for Fuzzy-PID Controller of Semi-Active Suspension System Featuring MR Damper. *IEEE Access* **2022**, *10*, 129125–129138. [CrossRef]
39. Huang, H.C.; Chuang, C.C. Artificial Bee Colony Optimization Algorithm Incorporated With Fuzzy Theory for Real-Time Machine Learning Control of Articulated Robotic Manipulators. *IEEE Access* **2020**, *8*, 192481–192492. [CrossRef]
40. Nikolic, M.; Šelmic, M.; Macura, D.; Calic, J. Bee Colony Optimization metaheuristic for fuzzy membership functions tuning. *Expert Syst. Appl.* **2020**, *158*, 113601. [CrossRef]
41. Guo, J.; Li, Y.; Du, B.; Wang, K. Multi-objective fuzzy partial disassembly line balancing considering preventive maintenance scenarios using enhanced hybrid artificial bee colony algorithm. *Expert Syst. Appl.* **2024**, *237*, 121681. [CrossRef]
42. Chen, C.H.; Jeng, S.Y.; Lin, C.J. Adaptive Fuzzy Neural Network Based on a Multi-Strategy-Based Artificial Bee Colony for Mobile Robot Control. *Mathematics* **2020**, *8*, 1223. [CrossRef]
43. Kumar, N.K.; Gopi, R.S.; Kuppusamy, R.; Nikolovski, S.; Teekaraman, Y.; Vairavasundaram, I.; Venkateswarulu, S. Fuzzy Logic-Based Load Frequency Control in an Island Hybrid Power System Model Using Artificial Bee Colony Optimization. *Energies* **2022**, *15*, 2199. [CrossRef]
44. Marjani, A.; Babanezhad, M.; Shirazian, S. Application of adaptive network-based fuzzy inference system (ANFIS) in the numerical investigation of Cu/water nanofluid convective flow. *Case Stud. Therm. Eng.* **2020**, *22*, 100793. [CrossRef]
45. Akay, B.; Karaboga, D.; Gorkemli, B.; Kaya, E. A survey on the Artificial Bee Colony algorithm variants for binary, integer and mixed integer programming problems. *Appl. Soft Comput.* **2021**, *106*, 107351. [CrossRef]
46. Devadason, J.R.; Hepsiba, P.S.; Solomon, D.G. Case studies on the applications of the artificial bee colony algorithm. *Sādhanā* **2024**, *49*, 152. [CrossRef]
47. Öztürk, Ş.; Ahmad, R.; Akhtar, N. Variants of Artificial Bee Colony algorithm and its applications in medical image processing. *Appl. Soft Comput.* **2020**, *97*, 106799. [CrossRef]
48. Kasihmuddin, M.S.B.M.; Mansor, M.A.B.; Alzaeemi, S.A.; Sathasivam, S. Satisfiability Logic Analysis Via Radial Basis Function Neural Network with Artificial Bee Colony Algorithm. *Int. J. Interact. Multimed. Artif. Intell.* **2021**, *6*, 164. [CrossRef]
49. Shaibani, M.J.; Emamgholipour, S.; Moazeni, S.S. Investigation of robustness of hybrid artificial neural network with artificial bee colony and firefly algorithm in predicting COVID-19 new cases: Case study of Iran. *Stoch. Environ. Res. Risk Assess.* **2022**, *36*, 2461–2476. [CrossRef] [PubMed]
50. Ratnayake, R.M.C.; Antosz, K. Development of a Risk Matrix and Extending the Risk-based Maintenance Analysis with Fuzzy Logic. *Procedia Eng.* **2017**, *182*, 602–610. [CrossRef]
51. Singh, P.; Mittal, N. An efficient localization approach to locate sensor nodes in 3D wireless sensor networks using adaptive flower pollination algorithm. *Wirel. Netw.* **2021**, *27*, 1999–2014. [CrossRef]
52. Yang, X.S. Flower Pollination Algorithm for Global Optimization. In *Unconventional Computation and Natural Computation*; UCNC 2012. Lecture Notes in Computer Science; Durand-Lose, J., Jonoska, N., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7445. [CrossRef]
53. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
54. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
55. Yang, X.S.; Deb, S. Cuckoo Search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
56. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]

57. Sun, Z.; Xing, R.; Zhao, C.; Huang, W. Fuzzy auto-tuning PID control of multiple joint robot driven by ultrasonic motors. *Ultrasonics* **2007**, *46*, 303–312. [CrossRef]
58. Zhang, Q.H.; Lu, J.G. Robust stability of fractional-order systems with mixed uncertainties: The $0 < \alpha < 1$ case. *Commun. Nonlinear Sci. Numer. Simul.* **2023**, *126*, 107511.
59. Zhang, Q.H.; Lu, J.G. Novel admissibility and robust stabilization conditions for fractional-order singular systems with polytopic uncertainties. *Asian J. Control.* **2024**, *26*, 70–84. [CrossRef]
60. Zhang, Q.-H.; Lu, J.-G.; Xu, J.; Chen, Y.-Q. Solution Analysis and Novel Admissibility Conditions of SFOSs: The $1 < \alpha < 2$ Case. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *52*, 5056–5067.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.