INTERNATIONAL SCIENTIFIC AND VOCATIONAL JOURNAL (ISVOS JOURNAL)

Vol.: 9 Issue: 1 Date: 30.06.2025 Received: 02.05.2025 Accepted: 23.06.2025 Final Version: 27.06.2025

ISVOS Journal, 2025, 9(1): 118-129 – DOI: 10.47897/bilmes.1688286

Performance Optimization of PID Controllers for DC Machine Drives Using PSO, ACO, and Hybrid PSO-ACO Algorithms

Ahmed Emin Yörük [a], Nuri Alper Metin [b], Murat Lüy [c,1]

[a] Electrical - Electronics Engineering, Kırıkkale University, Kırıkkale, Türkiye
ORCID ID: 0000-0002-1372-1734
[b] Electrical - Electronics Engineering, Kırıkkale University, Kırıkkale, Türkiye
ORCID ID: 0000-0002-9962-917X
[c] Electrical - Electronics Engineering, Kırıkkale University, Kırıkkale, Türkiye
ORCID ID: 0000-0002-2378-0009

Abstract

PID controllers are utilised extensively in the domain of electric motors and drives. The values of the PID controller have a direct impact on the controller's characteristics. Establishing optimal values is imperative to enhance the efficacy of control mechanisms. Consequently, a multitude of optimization algorithms have been developed. Employing these algorithms facilitates the optimisation of the controller's optimal values with greater efficiency, requiring less experience and a shorter timeframe. In this study, the parameters of the PID controller employed in the motor drive developed for a direct current (DC) motor are optimised by three distinct heuristic optimisation methods: The following optimization methods are used: Particle Swarm Optimisation (PSO), Ant Colony Optimisation (ACO), and PSO-ACO, which is a combination of these two methods. The execution of simulations is conducted within the MATLAB environment, with a subsequent comparative analysis of control performances. This study proposes a pioneering optimisation approach that integrates the PSO and ACO algorithms. The PID controller attains the reference value in the most efficient timeframe through this methodology. The simulation results show that the PSO-ACO method demonstrates optimal performance, followed by PSO and ACO.

Keywords: "PID Control, PSO, ACO, PSO-ACO, DC Machine."

1. Introduction

DC machines are widely used in industry, robotics, and automatic control systems due to their high controllability, linear speed-torque characteristics, and simple structure. Achieving precise and stable control of a DC motor's speed or position requires an effective control mechanism. In this context, Proportional-Integral-Derivative (PID) controllers, one of the classical control methods, are frequently preferred in DC motor applications due to their simplicity, ease of implementation, and broad applicability. However, the performance of a PID controller depends heavily on the accurate tuning of its parameters (K_p, K_i, K_d). Manual tuning of these parameters can be time-consuming and may limit system performance. To address this issue, heuristic and metaheuristic optimization algorithms have been increasingly adopted to automatically and optimally determine PID parameters, thereby improving performance criteria such as response time, stability, and error levels. Metaheuristic optimization algorithms have demonstrated significant practical applicability in a wide range of industrial domains due to their adaptability, robustness, and capacity to solve complex, nonlinear, and multimodal problems without requiring gradient information. Industries such as manufacturing, energy systems, automotive, aerospace, robotics, and process control frequently encounter high-dimensional optimization challenges where traditional deterministic or analytical approaches become infeasible or inefficient. In such contexts, metaheuristic algorithms such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Genetic Algorithms (GA) enable efficient parameter tuning, design optimization, fault diagnosis, and real-time control. For instance, in motor control applications, metaheuristics are used to optimize PID parameters dynamically, ensuring minimal error, reduced overshoot, and faster response under variable load and environmental conditions. Their ability to escape local minima and explore vast search spaces makes them especially suitable for systems with uncertainty, noise, or dynamic constraints. Furthermore, their algorithmic simplicity and parallelizable structure facilitate easy implementation on embedded systems, industrial controllers, and real-time platforms. As industrial systems increasingly demand autonomous adaptation and data-driven decision-making, the integration of metaheuristic optimization methods continues to gain traction as a key enabler in smart manufacturing and Industry 4.0 frameworks. Closed-loop techniques are used extensively in system control under current

[1] Corresponding Author
E-mail Address: mluy@kku.edu.tr

models. Usually, these configurations include sensors that give real-time feedback on state variables, a controller, and the process to be controlled. These setups seek to minimize the negative consequences of internal dynamics and external disturbances, guaranteeing the system reaches its intended output. Because PID control is so important in preserving control stability, it is extensively used among several techniques in industrial applications, especially for controlling the speed and position of a DC machine [1].

The precision of the proportional (K_p), integral (K_i), and derivative (K_d) gains determines how effective a PID controller is. Many conventional tuning methods depend on time-consuming manual trial-and-error processes that might not produce the best performance. Based on observed oscillatory behavior in the system, one often used technique in literature, the Ziegler–Nichols approach, establishes parameter values [2]. This approach, which assumes linearity, thus has limited relevance in systems with nonlinear or time-varying properties. Heuristic optimization techniques have become valuable tools for automatically tuning PID parameters to solve such constraints. These techniques present a strong substitute in complex control environments since they seek to outperform conventional methods in performance and provide resilience against unforeseen disturbances. This study conducts detailed investigations on tuning a PID control algorithm designed for a DC machine. Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) methods were applied separately in the initial phase. Subsequently, a hybrid algorithm named PSO-ACO is developed by combining the strengths of both techniques. The hybrid PSO-ACO approach is designed to combine the global search efficiency of PSO with the local refinement strength of ACO, with the objective of enhancing convergence speed and robustness in controller tuning. Simulation studies carried out in the MATLAB environment compared the performance of the proposed methods based on the ITAE (Integral of Time-Weighted Absolute Error) criterion, with the primary objective of achieving faster convergence to the reference value [3].

1.1. Literature Rewiev

The application of heuristic algorithms to the tuning of PID controllers for direct current (DC) motor control has recently demonstrated a certain degree of success. The responses produced by Yıldırım et al. from PSO were both fast and accurate, significantly lowering overshoot and settling time [4]. Utilising a hybrid PSO-MRAC approach, Oche et al. achieved zero overshoot and enhanced resilience under perturbations [5]. Compared to standalone approaches, a more balanced transient response can be achieved by utilizing a GA–PSO hybrid approach proposed by Beremeh et al [6]. By offering lower overshoot and faster settling times in both simulations and hardware implementations, Najem et al. found that ACO outperformed PSO [7]. Güven et al. presented a modified Jellyfish Search method that guarantees almost instantaneous, error-free performance and minimizes ITAE [8]. Finally, Ekinci et al. outperformed conventional methods by applying the Mountain Gazelle Optimizer to reach zero overshoot with a settling time of less than 0.1 seconds [9]. Literature rewiev is given in Table 1.

Table 1. Literature Rewiev

Study (Year)	Optimization Algorithm(s)
[4]	PSO
[5]	PSO-MRAC
[6]	GA–PSO
[7]	ACO, PSO
[8]	Modified Jellyfish Search (JMS)
[9]	Mountain Gazelle Optimizer (MGO)
This Study	PSO, ACO, PSO-ACO

2. Materials and Methods

2.1. System Description

Fig. 1 shows a speed control system for a DC motor where different heuristic optimization approaches help to fine-tune the PID controller's parameters. The system runs by computing the motor's actual speed output against the intended reference speed. The PID controller receives this error and uses its proportional, integral, and derivative components to generate a control signal. The aim is to change the motor's operation so that its speed almost matches the reference input. Above the PID block, three optimization techniques—particle swarm optimization (PSO), ant colony optimization (ACO), and their hybrid PSO-ACO approach—are used to ascertain the most appropriate values for the PID gains. Aiming to minimize the integrity of time-weighted absolute error (ITAE), these techniques use it as the performance criterion to improve dynamic response. By constantly adjusting the control parameters, integrating these algorithms lets the system reach the intended speed with better accuracy and faster response.

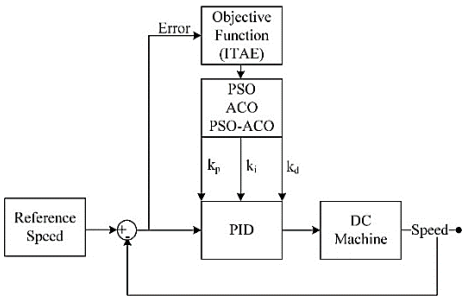


Fig. 1. Block diagram of System Description

2.2. PID Control

The PID control algorithm is a feedback-based control method commonly used in industrial applications. This technique continuously evaluates the difference (error) between the desired system state and the actual measured output, and shapes the control signal accordingly. In the control process, the system output is influenced by three key components—proportional, integral, and derivative—which operate in coordination with one another. The proportional component generates an immediate corrective response proportional to the current error magnitude. The integral component addresses long-term or persistent deviations by calculating the accumulated error over time, thereby helping the system converge to the desired setpoint. On the other hand, the derivative component predicts potential future instability by analyzing the rate of change of the error and contributes to reducing oscillations, particularly during transient responses [10]. The PID control equation is provided in Equation 1.

$$u(t) = K_p \cdot e(t) + K_i \cdot \int e(t)dt + K_d \cdot \frac{de(t)}{dt} \quad (1)$$

Fig. 2 shows the basic structure of a PID controller, which functions as a control system. In this system, the error is calculated as the difference between the reference speed and the current speed, and this error passes through three components: Proportional (P), Integral (I), and Derivative (D). Each component processes the error differently: the Proportional component responds based on the magnitude of the error, the Integral component accumulates past errors, and the Derivative component considers the rate of change of the error. Combining these three components generates an appropriate control signal that drives the system output toward the desired value [11].

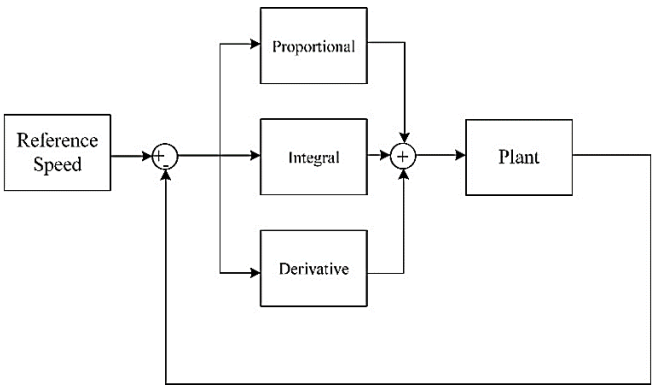


Fig. 2. Block diagram of PID control

2.3. Optimization Methods

Swarm Intelligence refers to the collective behavioral patterns that emerge from individuals acting in harmony with one another. This concept is frequently encountered in artificial intelligence research and was first introduced in 1989 by Gerardo Beni and Jing Wang in the context of cellular robotic systems [12]. Although simple rules guide each agent in such systems and lack predefined instructions for behavior, the overall interactions between individuals result in complex behaviors that can be described as 'intelligent,' even if the agents are unaware of the global effect [13]. Numerous swarm intelligence algorithms have been developed by drawing inspiration from nature. These include Stochastic Diffusion Search [14], Ant Colony Optimization (ACO) [15], Particle Swarm Optimization (PSO) [16], Genetic Algorithm (GA) [17], and Differential Evolution Algorithm [18].

2.3.1. Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) was first proposed in 1992 by Marco Dorigo as part of his doctoral dissertation, initially developed as a heuristic algorithm to find the optimal path between two points. Dorigo introduced this original approach under the name Ant System. In 1999, Hoos and Stützle further refined the algorithm's structure and presented an improved version called the Max-Min Ant System [19]. The ant colony optimization algorithm discussed in this article is also based on the work published by Dorigo and Stützle in 1999 [20]. Dorigo and Stützle later provided a comprehensive overview of the algorithm's detailed structure in a 2004 publication [21].

Regarding the philosophy behind the Ant Colony Algorithm, in 1959, French entomologist Pierre-Paul Grasse observed that ants respond to specific signals he referred to as “significant stimuli.” During his observations, he noticed that the responses of ants to these stimuli could generate new and critical triggers not only for the individuals that produced them but also for other colony members. To define this unique form of communication, which is activated through the actions of worker ants, Grasse introduced the term “stigmergy” [22]. Stigmergy is one of the core concepts of swarm intelligence and refers to an indirect coordination mechanism that arises without direct communication between agents and their actions. Its fundamental principle is that traces left in the environment by one action trigger subsequent actions. The same agent or another independent agent can initiate the follow-up action in this mechanism. As a result, sequential actions are built incrementally, leading to the emergence of coherent and systematic activity patterns. Stigmergy, as a self-organizing approach, enables the formation of complex and intelligent structures without requiring direct communication during planning and control stages. This provides a foundation for effective collaboration, even among simple agents with limited memory or cognitive capacity [23].

A practical example of this natural coordination mechanism can be observed using pheromones within ant colonies. In many ant species, during the food search, ants deposit a chemical substance called pheromone onto the ground; other ants detect this substance and are led to prefer paths with higher pheromone concentrations. Since ants are more likely to choose routes with stronger pheromone trails, more individuals increasingly use these paths. As a result, the pheromone concentration along the selected path intensifies, enabling the colony to identify the most efficient route to the target. The flowchart of the ACO algorithm is illustrated in Fig. 3.

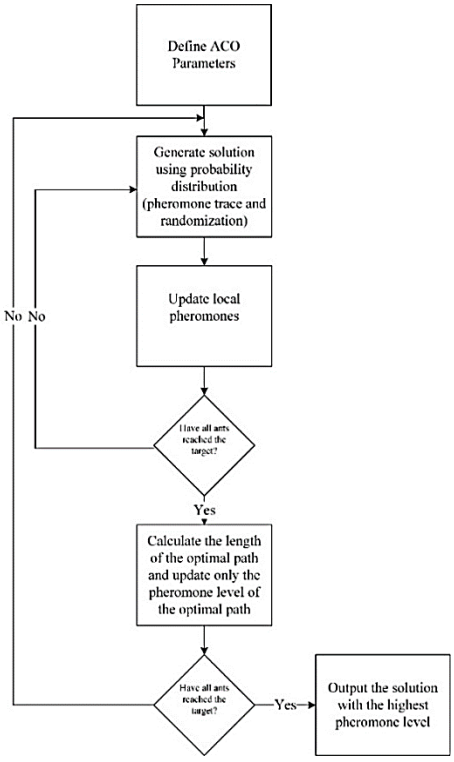


Fig. 3. Flowchart of ACO

The parameters used in the ACO hybrid algorithm are as follows: n is the number of nodes in the problem, m is the number of ants in the system, and d_{ij} is the distance between nodes i and j. T is the number of iterations, and T is the maximum number of iterations. ρ vaporization coefficient, τ_{ij} represents the pheromone density between nodes i and j, and η_{ij} represents the visibility value between nodes i and j. ρ_{ij}^{k} is the probability that ant k moves from node i to node j, $L^{k}(t)$is the distance travelled by ant k in iteration t, and $L_{thebest}(t)$is the shortest distance found in iteration t. α is the importance coefficient of pheromone, and β is the importance coefficient of distance. Finally, Q is the pheromone update constant. The ACO algorithm arrives at the result by applying the following procedure: The initial pheromone values m, n, m, d_{ij}, ρ, α, β, Q, and τ_{ij} required for the result

are determined. Ants are placed randomly. All ants complete the cycle to select the next episode based on the local search probability shown in Equation 2 [24].

$$\rho_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}\,[\eta_{ij}]^{\beta}}{\sum[\tau_{ij}]^{\alpha}\,[\eta_{ij}]^{\beta}} \tag{2}$$

The action distance is calculated for each ant in the algorithm. The local pheromone replenishment is given in Equations 3 and 4.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t+1) \tag{3}$$

$$\Delta\tau_{ij}^{k}(t+1) = \begin{cases} \dfrac{1}{L^{k}(t+1)} & \text{if ant used path } i-j \\ 0 & \text{another situations} \end{cases} \tag{4}$$

The most accurate result is calculated, and the global pheromone defeat is shown in Equations 5 and 6.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t+1) \tag{5}$$

$$\Delta\tau_{ij}^{k}(t+1) = \begin{cases} \dfrac{1}{L_{thebest}(t+1)} & \text{ifit belongs to best tour} \\ 0 & \text{another situations} \end{cases} \tag{6}$$

The cycle continues from step 2 until the maximum number of iterations (T) or another defined qualification criterion is reached.

2.3.2. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a heuristic method developed in 1995 by J. Kennedy and R. C. Eberhart, inspired by the natural behavior of bird flocks [25]. In nature, birds searching for food follow the individual closest to the food source and is considered the leader; however, a new bird that detects the food may take over the leadership. During this process, the communication among birds becomes a fundamental component of identifying the leader and maintaining coordination within the flock. Each bird represents a particle in the algorithm, and its current position corresponds to a potential solution to the target function. As the birds move, every position they occupy is evaluated, and the fitness value of the resulting solution is determined. This fitness value acts as a measure of proximity to the desired result. If a satisfactory solution is found according to predefined criteria, the search process is terminated; otherwise, particles continue to explore the search space to find the optimal value. The general block diagram of PSO is presented in Fig. 4 [26].

Before the algorithm starts, a swarm is initialized, and each particle is assigned an initial velocity and position. Within this swarm, the initial leader is identified. Particles move within a defined velocity range and update their positions each iteration. This velocity parameter can be constant or dynamic. At every new position, the objective function is evaluated for all particles, and a new leader representing the best solution is designated. The leader's position is checked to determine whether it meets the optimization criteria; if so, the search process is completed. If not, the particles continue moving through the search space [27].

What distinguishes PSO from classical optimization techniques is its simplicity, stemming from the limited number of parameters to be adjusted and the fact that it does not require derivative information. The process begins with a randomly generated initial swarm, defined by position, velocity, and direction data. The fitness value for each particle in the swarm is computed. Each particle's best position is recorded based on its own experiences. The best global position within the entire swarm is selected among these individual results. Velocity and position are updated based on previous steps. The process is terminated if the target is deemed reached; otherwise, the algorithm iterates, starting again from step two. The PSO flowchart is shown in Fig. 4 [28].

At the core of this methodology lies the concept of a swarm, with particles representing individual members within it. Each particle repositions itself reactively to align with the optimal location within the collective, drawing on its previous experiential performance. Taking the currently best-optimized position in the swarm as a reference, all other particles update their motion vectors accordingly. The acceleration dynamics of this approach evolve based on randomness and are typically characterized by an iterative process in which particles tend to move toward positions superior to those they have previously attained. This process is repeated until the final objective is achieved.

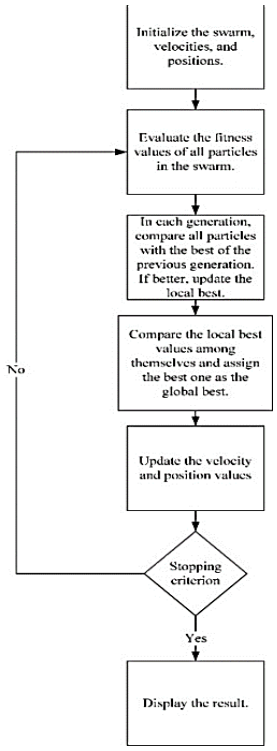


Fig. 4. Flowchart of PSO

$$v_{k+1}^{i} = wv_{k}^{i} + c_1 rand \frac{(p^{i} - x_{k}^{i})}{\Delta t} + c_2 rand \frac{(\rho_{k}^{g} - x_{k}^{i})}{\Delta t} \quad (7)$$

Equation 7 gives the velocity (V) equation of the i-th particle at time step k+1; here, W represents the inertia weight, c_1 functions as the particle's acceleration coefficient, and c_2 is the social acceleration coefficient associated with the swarm. p^{i} denotes the best-known position of the respective particle, whereas ρ_{k}^{g} indicates the globally best-known position across the entire swarm. These values are multiplied by random scaling factors and added together to compute the particle's next velocity vector. This, in turn, determines the new position, expressed as x_{k+1}^{i}. The process follows a cyclical structure, following the steps below [29].

2.3.3. PSO-ACO Optimization Algorithm

The PSO-ACO hybrid algorithm introduced in this study distinguishes itself from other hybrid approaches in the literature through its unique integration mechanism, which combines the global search capabilities of Particle Swarm Optimization (PSO) with the local refinement strength of Ant Colony Optimization (ACO) in a complementary and dynamic fashion. Unlike conventional hybrid methods that often operate sequentially or prioritize one algorithm over the other, the proposed method continuously incorporates pheromone-based directional vectors from ACO into the velocity update equation of PSO during each iteration. This allows poorly performing particles not only to be repositioned based on the swarm's best experience but also to be guided by pheromone trails, leading to more efficient exploration and exploitation of the solution space. Additionally, the adaptive ACO sampling applied to underperforming agents enables the algorithm to escape local minima and maintain population diversity. These features result in faster convergence, improved stability, and superior optimization performance compared to other methods that either lack real-time interaction between components or use static hybridization schemes.

PSO-ACO starts with parameters such as population size (n_{Agents}) and number of iterations (n_{Itern}). While the PSO directs the solution search by updating the velocities and positions of each agent with the personal best solution ($p_{Best(i)}$) and the global best solution (g_{Best}), the ACO marks the good regions in the solution space with pheromones and guides the PSO in these regions. At each iteration, ACO calculates the direction vector and adds it to the PSO's velocity update equation to find the solution more quickly. Pheromone update releases more pheromones for better solutions, which helps to select better paths. Furthermore, ACO sampling is applied for poorly performing agents, allowing these agents to explore different solution paths. As a result, at the end of each iteration, the best solution is updated considering both the cost value and the pheromone contributions [30].

ACO sampling is a technique that enables poorly performing agents to discover new solution paths using pheromones. If an agent's personal best solution ($p_{Best(i)}$) is twice as poor as the global best solution (g_{Best}), ACO sampling is applied for that agent. The agent is reset in this case, and a new solution is proposed based on the pheromone matrix. This allows agents with

poor performance to explore broader solution spaces and potentially find better outcomes. This method enhances the efficiency of the algorithm and contributes to faster convergence. PSO allows each particle (or agent) to move through the solution space and continue its search for solutions. Each agent updates its new position considering its previous best solution ($p_{Best(i)}$) and the best solution in the population (g_{Best}). The rate update equation of PSO is given in Equation 8.

$$v_i^{t+1} = w \cdot v_i^t + c_1 \cdot r_1 \cdot \left(p_{Best(i)} - x_i^t\right) + c_2 \cdot r_2 \cdot (g_{Best} - x_i^t) + c_3 \cdot r_3 \cdot (\tau_{dir} - x_i^t) \quad (8)$$

v_i^t: speed of particle i, x_i^t: current position of particle i, $p_{Best(i)}$: previous best solution of particle i, g_{Best}: global best solution of the population. τ_{dir}: Direction vector suggested by ACO (pheromone contribution), w: Inertia coefficient c_1, c_2, c_3: Attraction coefficients (for the personal best solution, global best solution, and ACO direction). r_1, r_2, r_3: Defined as random numbers. PSO aims to reach the optimal point in the solution space by updating the velocity and position of the agents. Here, the direction information (τ_{dir}) provided by the ACO is also added to the update equation of PSO [31].

The PSO-ACO algorithm begins with parameters such as population size (n_{Agents}) and number of iterations (n_{Iter}). While PSO directs the search process by updating each agent's velocity and position based on its personal best solution ($p_{Best(i)}$) and the global best solution (g_{Best}), ACO marks promising regions of the solution space with pheromones and guides PSO towards these regions. In each iteration, ACO computes a direction vector added to the PSO velocity update equation, allowing the solution to be found more rapidly. Pheromone updates deposit more pheromones for better solutions, helping the algorithm prioritize better paths. ACO sampling is also applied to poorly performing agents, enabling them to explore alternative solutions. As a result, at the end of each iteration, the best solution is updated by considering both the cost value and pheromone contributions [32].

2.3.4. Objective Function

In optimization problems, the objective function is the target value that algorithms seek to maximize or minimize to attain a system's best accuracy or performance. Standard metrics include ITAE (Integral of Time-weighted Absolute Error), which evaluates dynamic system performance by penalizing errors more heavily over time; ITSE (Integral of Time-weighted Squared Error), which emphasizes larger errors and promotes faster system stabilization; ISE (Integral of Squared Error), which minimizes the overall error and is effective when minor errors need to be penalized; and IAE (Integral of Absolute Error), which accumulates absolute errors over time to promote smoother responses [33].

These performance functions quantitatively express the amount of error occurring over time. Minimizing these functions during the optimization process means determining the PID controller parameters in a way that ensures system performance with minimal error. The Integral of Time-weighted Absolute Error (ITAE) is a commonly used criterion for evaluating the performance of control systems. This method integrates the absolute value of the error multiplied by time, penalizing errors that occur later more heavily. This characteristic helps the system reach a steady state more quickly and with less oscillation. The ITAE criterion is chosen to optimize dynamic performance indicators such as settling time and overshoot. The objective function used is presented in Equation 9 [34].

$$f_{ITAE} = \int_0^T t \cdot |e(t)|\, dt \quad (9)$$

2.4. DC Machine

Fig. 5 shows a circuit representing a DC machine's electrical and mechanical model. In the circuit, a series connection is formed with a source U_i resistance R and inductance L. Current Ia flows through this circuit and determines the voltage U_α in the electrical part of the motor. The mechanical part of the motor is characterised by torque T, speed w, and torque J. In addition, b_w (coefficient of friction), a term representing the motor's friction as it rotates, also affects the behaviour of the motor. This circuit is a basic model used for speed control and dynamic analysis of the motor [35].

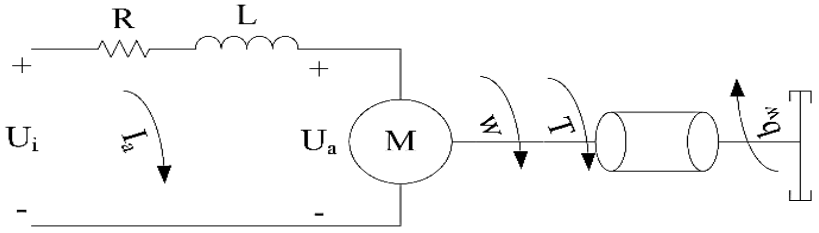


Fig. 5. Circuit of DC Machine

Equations 10 and 11, which describe the dynamic behavior of the DC motor, represent both the electrical and mechanical properties of the motor. Equation 10 describes the rotational motion of the motor, where J is the motor's inertia, b is the viscous friction coefficient, w is the angular velocity, Ki_α is the electromagnetic moment, and T is the external load torque. This equation states that the moment produced by the motor must balance the effects of load torque and friction. Equation 11 is based on Kirchhoff's voltage law. It describes how the current through the motor varies with the difference between the applied voltage and the back EMF (Kw), where L is inductance, R is resistance, I_α is current, and U_α is the applied voltage. When used together, Equations 10 and 11 form a basic mathematical model for the control and analysis of the motor [36].

$$j\frac{dw}{dt} + bw = Ki_\alpha - T \tag{10}$$

$$L\frac{di_\alpha}{dt} + Ri_\alpha = U_\alpha - Kw \tag{11}$$

3. Simulation Results

In this study, a control system using a PID controller optimised by heuristic techniques aims to control the speed of a DC machine. The process starts with a reference speed, which is mapped to the actual speed of the DC motor to produce an error signal. This error is fed into an objective function such as ITAE (Integral of Time Weighted Absolute Error), which evaluates system performance. Using this error-based performance index, optimisation algorithms such as PSO, ACO, and PSO-ACO repeatedly tune the PID controller parameters K_p, K_i, and K_d. The PID controller uses these optimal values and generates a control signal for the DC machine. The motor's output speed is fed back to the system to form a closed-loop control system to reduce the error and achieve exact speed tracking. This system is designed in the MATLAB environment.

A DC machine's mechanical and electrical characteristics are given in Table 2. Specifying 2.45 ohms, the armature resistance indicates the opposition to current flow within the motor windings. Given as 0.035 H, the inductance reflects the motor's capacity to oppose changes in current. The back EMF constant is 1.2 Vs/rad, representing the voltage produced as the motor shaft rotates. At 0.222 kg·m², the rotor's moment of inertia gauges the motor's resistance to rotational speed changes. Finally, listed as 0.0005 Nms/rad, the damping coefficient characterizes the effect of mechanical friction opposing motion. Designing a reasonable control system and simulating the dynamic behavior of the motor depend on these fundamental values.

Table 2. DC Machine Mechanical and Electrical Characteristics

DC Motor Parameters	Value
Armature Resistance	2.45 Ω
Inductance (L)	0.035 H
EMF Constant (K)	1.2 Vs/rad
Rotor Inertia (J)	0.022 kg·m²
Friction Coefficient (B)	0.0005 Nms/rad

This study evaluated three different optimization techniques: Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and a hybrid method combining both, PSO-ACO. Each algorithm was executed over 50 iterations using a population size of 30. The search space is defined by specific upper and lower bounds for a three-dimensional parameter set: [50, 20, 5] as the upper bound and [1, 0.1, 0.1] as the lower bound. Identical experimental conditions were maintained across all algorithms to compare their performances fairly. According to these parameter ranges, the K_p, K_i, and K_d values are given in Table 3.

Table 3. Optimized PID Parameters (K_p, K_i, K_d) for Each Algorithm

Optimization Algorithms	K_p	K_i	K_d
PSO	0.01	10	0.001
ACO	0.01	10	0.0011
PSO-ACO	0.01	10	0.00235065451759030

Fig. 6 shows a three-stage load torque profile that varies over time. Initially, between 0 and 3 seconds, the load torque is approximately 0.05 N·m. Then, from 3 to 6 seconds, the torque increases sharply to around 0.3 N·m. Finally, after 6 seconds, it reaches 0.5 N·m and remains constant at that level. This type of profile is commonly used to evaluate how a system responds to varying load conditions.

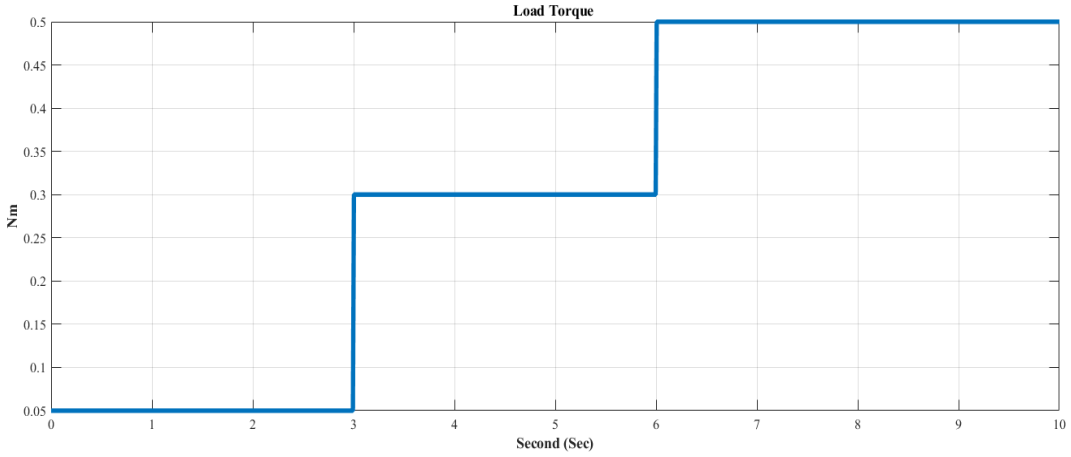
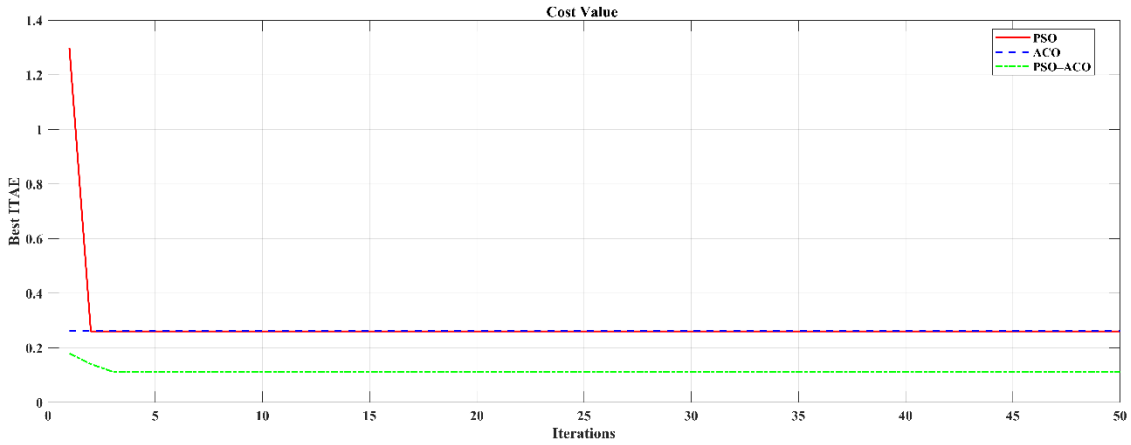


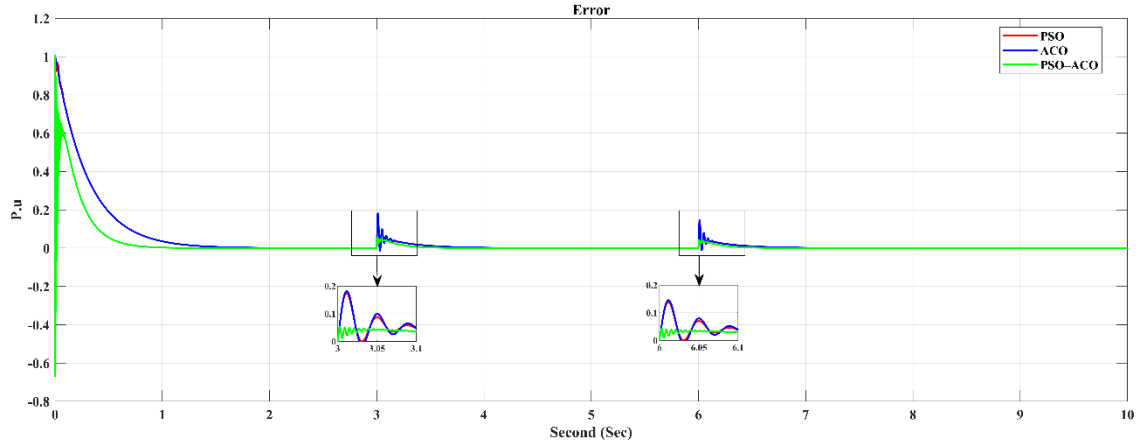
Fig. 6. Time-Varying Load Torque Profile Used for System Evaluation

The performance evaluation based on the speed response, error curve, and cost function results shown in Fig. 7 demonstrates the superior efficiency of the PSO-ACO algorithm in PID parameter optimisation. PSO-ACO reaches the reference speed faster and with fewer overshoots compared to the other methods, PSO and ACO, resulting in a faster and more stable system response. The error curve also supports this advantage and shows that the PSO-ACO algorithm rapidly reduces the tracking error and converges to zero with minimum fluctuation, thus providing high accuracy and control precision. Furthermore, the cost function plot emphasising the excellent convergence efficiency of the PSO-ACO algorithm shows that lower objective function values are achieved in fewer iterations. This performance improvement is due to the capacity of the PSO-ACO algorithm to successfully combine the local refinement capabilities of the ACO method with the global search capabilities of the PSO method. Although PSO effectively avoids local minima by covering large search spaces, it is prone to premature convergence. In contrast, ACO's pheromone-based reinforcement mechanism is particularly effective in improving solutions in promising regions. By harmoniously combining these two complementary behaviours, PSO-ACO can more precisely find and refine high-quality solutions and effectively balance exploration and exploitation. Thanks to this synergy, PSO-ACO demonstrates superior optimisation performance by maintaining population diversity, avoiding stalling, and converging to near-optimal PID parameters more efficiently than individual methods.

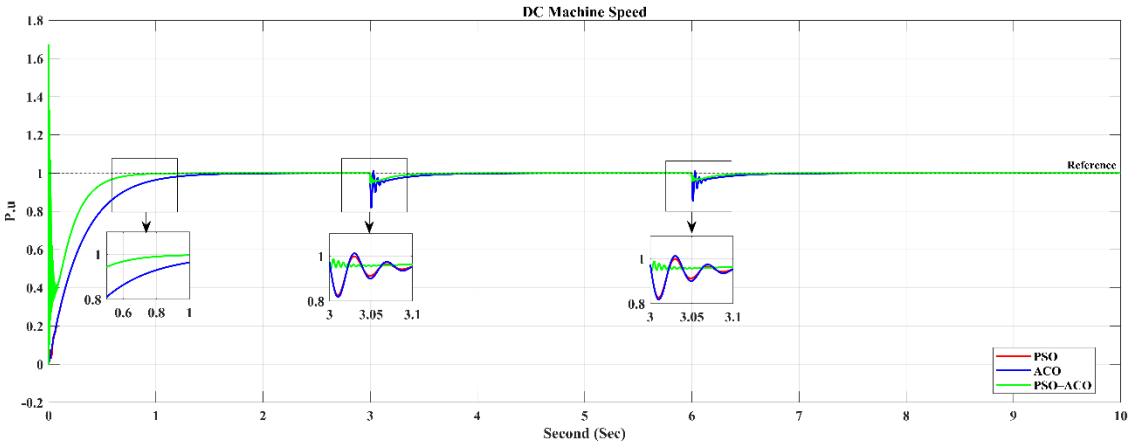
Table 4 compares the control performances of different optimisation algorithms, PSO, ACO, and the combination of these two methods, PSO-ACO. According to the results of the analyses, it is observed that the PSO-ACO algorithm exhibits a significant difference in terms of temporal response characteristics. While PSO and ACO methods generally offer similar performance parameters, the PSO-ACO method shows substantial deviations in specific metrics. When ITAE is considered, it is determined that the PSO-ACO algorithm provides lower error accumulation than other methods. This result indicates that the system can reach the reference value more efficiently. However, the significant overshoot rate observed in the PSO-ACO method suggests that the system response contains faster but more abrupt changes. The time to reach equilibrium for all algorithms is similar, indicating that the control strategies exhibit similar behaviour in terms of stability. However, regarding initial performance and overall system dynamics, it is concluded that the PSO-ACO method outperforms the other two algorithms.



(a)



(b)



(c)

Fig. 7. Each optimization algorithm a) Cost vs Iteration, b) Error, c) Speed

Table 4. Optimization Algorithms Performance Values

Optimization Algorithms	ITAE	Final Error	Overshoot	Rise Time
PSO	0.2605	0.0000	0.0000	0.6540
ACO	0.2615	0.0000	0.0129	0.6570
PSO-ACO	0.1130	0.0000	0.6695	0.0010

4. Conclusion

The performance of heuristic optimisation algorithms in tuning PID controller parameters for DC motor speed control is comparatively investigated concerning transient criteria such as overshoot and rise time. DC motors are frequently favoured in industrial contexts due to their linear speed-torque characteristics, high controllability, and wide range of applications. However, implementing an effective control strategy is imperative for the motor to attain the reference speed value with both rapidity and stability. PID controllers assume paramount importance at this juncture due to their uncomplicated configuration. Nevertheless, the efficacy of these controllers is contingent upon the precision of the parameter settings. The simulation results show that heuristic and hybrid optimisation methods, including PSO-MRAC, GA-PSO, and the proposed PSO-ACO, exhibit superior performance compared to classical tuning techniques. In particular, the Mountain Gazelle Optimiser (MGO) algorithm developed by Ekinci et al. (2025) offers a remarkable transient response with zero overshoot and a rise time of 0.0478 seconds [9]. However, the PSO-ACO algorithm developed in this study has been shown to achieve the most successful results, with an extremely low rise time of 0.0010 seconds and a low overshoot rate of 0.6695%. The findings indicate that hybrid swarm intelligence-based optimisation methods can significantly enhance the system's transient response speed and tracking accuracy, particularly in real-time DC motor control applications. Table 5 is given Comparison of Performance of the Study with Heuristic Optimisation Methods in the Literature.

Table 5. Comparison of Performance of the Study with Heuristic Optimisation Methods in the Literature

Study (Year)	Optimization Algorithm(s)	Overshoot (%)	Rise Time (s)
[4]	PSO	1.86%	0.045
[5]	PSO-MRAC (Hybrid)	0%	-
[6]	GA–PSO (Hybrid)	5.29%	0.30
[7]	ACO	0.68%	-
[8]	Modified Jellyfish Search (JMS)	~0%	-
[9]	Mountain Gazelle Optimizer (MGO)	0%	0.0478
This Study (2025)	PSO-ACO (Hybrid)	0.6695%	0.0010

Simulation results confirm that the hybrid PSO-ACO method exhibits superior performance, particularly in terms of rapid convergence and enhanced control precision. A more detailed examination of the transient response characteristics reveals that PSO-ACO effectively minimizes tracking error in the early phase of the system's operation and rapidly stabilizes the motor speed. In real-world control scenarios, such responsiveness is critical for applications requiring high agility and minimal delay, such as robotics, automated manufacturing, or precision motion control systems. The rapid rise time of 0.0010 s achieved by PSO-ACO indicates the algorithm's capacity to initiate control action almost immediately after receiving a reference input, which is a crucial requirement for time-sensitive applications. Additionally, its ability to reduce steady-state error without introducing significant oscillations or prolonged overshoots demonstrates improved robustness in transient conditions. While the overshoot is marginally higher than PSO or ACO alone, the hybrid approach compensates for this by reaching the desired value more quickly and with fewer oscillations during settling. Although PSO offers better exploration capabilities and ACO excels in local solution refinement, the proposed hybrid model successfully merges these advantages to balance exploration and exploitation. Furthermore, the study emphasizes the need for future validation via hardware implementations to assess the real-time performance and robustness of the proposed algorithms under physical uncertainties and disturbances. From a practical standpoint, PSO-ACO appears to be a highly promising candidate for industrial control systems that demand both speed and reliability. Nevertheless, in applications where overshoot must be strictly minimized—such as delicate positioning systems or safety-critical operations—pure ACO or further-tuned hybrid variants may be more appropriate. Ultimately, the choice of optimization approach should consider system constraints, performance priorities, and environmental dynamics

The PSO-ACO-based optimisation approach developed in this study has been shown to achieve successful results in the context of DC motor speed control. However, subsequent studies are planned to verify the performance of the proposed method by applying it to different motor types (e.g., brushless DC motors or synchronous motors). Furthermore, the objective is to develop more robust control algorithms by considering dynamic effects such as load variations, parameter uncertainties, and external disturbances in motor drive systems. Furthermore, developing more sophisticated hybrid methods is envisaged, combining diverse meta-heuristic algorithms (e.g., Grey Wolf Optimiser, Whale Optimisation Algorithm) within the PSO-ACO framework, complemented by comparative performance analyses. Ultimately, the efficacy and dependability of the algorithm in practical applications will be ascertained through the execution of tests on real-time hardware (HIL- Hardware-in-the-Loop).

References

[1] H. O. Erkol, "GA ve PSO ile Kontrol Parametrelerinin Optimizasyonu," Karaelmas Fen ve Mühendislik Dergisi, vol. 7, no. 1, pp. 179–185, 2017.

[2] J. G. Ziegler and N. B. Nichols, "Optimum Settings for Automatic Controllers," Journal of Dynamic Systems, Measurement and Control, vol. 115, no. 2B, pp. 759–765, 1993.

[3] R. C. Panda, C. Yu, and H. Huang, "PID Tuning Rules for SOPDT Systems: Review and Some New Results," ISA Transactions, vol. 43, no. 2, pp. 283–295, 2004.

[4] Ş. Yıldırım, M. S. Bingol, and S. Savaş, "Tuning PID controller parameters of the DC motor with PSO algorithm," International Review of Applied Sciences and Engineering, vol. 15, no. 3, pp. 281–286, 2024.

[5] J. E. Oche, H. A. Bashir, and T. J. Shima, "PSO-optimized model reference adaptive PID controller for precise DC motor speed control," Nigerian Journal of Technological Development, vol. 21, no. 4, pp. 135–144, Dec. 2024, doi: 10.4314/njtd.v21i4.2473.

[6] R. C. Beremeh et al., "A hybrid optimization scheme for tuning fractional order PID controller parameters for a DC motor," International Journal of Science and Research Archive, vol. 13, no. 2, pp. 2779–2789, 2024, doi: 10.30574/ijsra.2024.13.2.2291.

[7] A. Najem, A. Moutabir, and A. Ouchatti, "Simulation and Arduino hardware implementation of ACO, PSO, and FPA optimization algorithms for speed control of a DC motor," International Journal of Robotics and Control Systems, vol. 4, no. 3, pp. 1186–1206, 2024, doi: 10.31763/ijrcs.v4i3.1483.

[8] A. F. Güven, O. Ö. Mengi, M. A. Elseify, and S. Kamel, "Comprehensive optimization of PID controller parameters for DC motor speed management using a modified jellyfish search algorithm," Optimal Control Applications and Methods, vol. 46, no. 1, pp. 320–342, Jan. 2025, doi: 10.1002/oca.3218.

[9] S. Ekinci et al., "Advanced control parameter optimization in DC motors and liquid level systems," Scientific Reports, vol. 15, no. 1394, Jan. 2025, Doi: 10.1038/s41598-025-85273-y.
[10] M. Moghaddas, M. R. Dastranj, N. Changizi, and M. Rouhani, "PID Control of DC Motor Using Particle Swarm Optimization (PSO) Algorithm," Journal of Mathematics and Computer Science, vol. 1, no. 4, pp. 386–391, Dec. 2010, doi: 10.22436/jmcs.001.04.16.
[11] K. J. Åström and T. Hägglund, "The Future of PID Control," Control Engineering Practice, vol. 9, no. 11, pp. 1163–1175, Nov. 2001, doi: 10.1016/S0967-0661(01)00062-4.
[12] G. Beni and J. Wang, "Swarm Intelligence in Cellular Robotic Systems," Proceedings of NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, Jun. 1989.
[13] P. Erdoğmuş, "Doğadan Esinlenen Optimizasyon Algoritmaları ve Optimizasyon Algoritmalarının Optimizasyonu," DÜBİTED, vol. 4, no. 1, pp. 293–304, 2016.
[14] J. M. Bishop, "Stochastic Searching Networks," Proc. 1st IEE Int. Conf. on Artificial Neural Networks, pp. 329–331, London, UK, 1989.
[15] M. Dorigo and T. Stützle, Ant Colony Optimization, Cambridge, MA: MIT Press, 2004.
[16] K. E. Parsopoulos and M. N. Vrahatis, "Recent Approaches to Global Optimization Problems Through Particle Swarm Optimization," Natural Computing, vol. 1, no. 2–3, pp. 235–306, 2002, doi: 10.1023/A:1016568309421.
[17] J. Holland, "Genetic Algorithms," Scientific American Journal, 1992, pp. 66–72.
[18] R. Storn and K. Price, "Differential Evolution — A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," Journal of Global Optimization, vol. 11, no. 4, pp. 341–359, 1997.
[19] T. Stützle and H. H. Hoos, "MAX–MIN Ant System," 1999.
[20] M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," Artificial Life, vol. 5, no. 2, pp. 137–172, 1999.
[21] M. Dorigo and T. Stützle, Ant Colony Optimization, Cambridge, MA: MIT Press, 2004.
[22] M. Dorigo, M. Birattari, and T. Stützle, "Ant Colony Optimization," IEEE Computational Intelligence Magazine, vol. 1, no. 4, pp. 28–39, 2006, doi: 10.1109/MCI.2006.329691.
[23] D. Sandoval, I. Soto, and P. Adasme, "Control of direct current motor using Ant Colony optimization," 2015 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), Santiago, Chile, 2015, pp. 79–82, doi: 10.1109/Chilecon.2015.7400356.
[24] F. Ulu, G. Tabansız Göç, and F. Çavdur, "Floyd-Warshall ve Karınca Kolonisi Optimizasyonu Algoritmaları ile Depo Rota Planlaması," Verimlilik Dergisi, vol. 59, no. 2, pp. 337–354, 2025. [Online]. Available: https://doi.org/10.51551/verimlilik.1539618.
[25] F. Marini and B. Walczak, "Particle Swarm Optimization (PSO): A Tutorial," Chemometrics and Intelligent Laboratory Systems, vol. 149, pp. 153–165, 2015.
[26] C. A. Coello Coello and M. Reyes-Sierra, "Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art," International Journal of Computational Intelligence Systems, vol. 2, no. 3, pp. 287–306, 2006.
[27] M. Lüy and N. A. Metin, "PID Control Medium Size Wind Turbine Control with Integrated Blade Pitch Angle," Internatıonal Scıentıfic And Vocatıonal Journal (Isvos Journal), vol. 6, no. 1, pp. 22-31, Jun. 2022. [Online]. Available: https://doi.org/10.47897/bilmes.1091968
[28] D. Wang, D. Tan, and L. Liu, "Particle Swarm Optimization Algorithm: An Overview," Soft Computing, vol. 22, no. 2, pp. 387–408, 2018, doi: 10.1007/s00500-017-2534-6.
[29] A. T. Kiani et al., "An Improved Particle Swarm Optimization with Chaotic Inertia Weight and Acceleration Coefficients for Optimal Extraction of PV Models Parameters," Energies, vol. 14, no. 11, p. 2980, 2021, doi: 10.3390/en14112980.
[30] B. Nagaraj and P. Vijayakumar, "A Comparative Study of PID Controller Tuning Using GA, EP, PSO, and ACO," Journal of Automation, Mobile Robotics & Intelligent Systems, vol. 5, no. 2, pp. 42–48, 2011.
[31] P. Thakur, J. Mishra, C. Yadav, and K. Tiwari, "PSO–ACO PID Techniques for Stability Enhancement," International Journal of Novel Research and Development (IJNRD), vol. 3, no. 6, pp. 75–80, Jun. 2018.
[32] N. S. R. Krishnan and D. Devaraj, "Comparison of Bees Algorithm, Ant Colony Optimisation, and Particle Swarm Optimisation for PID Controller Tuning," Proc. 11th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 1975–1980, 2009, doi: 10.1145/1500879.1500912.
[33] M. R. H. Mojumder and N. K. Roy, "Review of Meta-Heuristic Optimization Algorithms to Tune the PID Controller Parameters for Automatic Voltage Regulator," arXiv preprint, arXiv:2409.00538, Aug. 2024.
[34] A. Yıldız, M. E. Karslıoğlu, and S. Yıldız, "Effects of objective function in PID controller design for an AVR system," Journal of Engineering Sciences and Design, vol. 8, no. 2, pp. 398–407, 2020.
[35] A. A. El-Gammal and A. A. El-Samahy, "A Modified Design of PID Controller for DC Motor Drives Using Particle Swarm Optimization," International Conference on Power Engineering, Energy and Electrical Drives, pp. 419–424, Lisbon, 2009.
[36] "DC Motor Speed: System Modeling," Control Tutorials for MATLAB and Simulink, University of Michigan. [Online]. Available: https://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling.