HITTITE JOURNAL OF SCIENCE AND ENGINEERING

e-ISSN: 2148-4171
Volume: 12 • Number: 2
June 2025

Optimization-Based Tuning of PI Controller Parameters for DC Motor Speed Control

Ahmet Top
Fırat University, Faculty of Technology, Department of Electrical and Electronic Engineering, Elazığ, Türkiye.

Corresponding Author
Ahmet TOP
E-mail: atop@firat.edu.tr Phone: +90 0424 237 00 00
RORID[1]: https://ror.org/05teb7b63

Article Information
Article Type: Research Article
Doi: https://doi.org/10.17350/HJSE19030000354
Received: 0 1.12.2024
Accepted: 10.04.2025
Published: 30.06.2025

Cite As
Top A. Optimization-Based Tuning of PI Controller Parameters for DC Motor Speed Control . Hittite J Sci Eng. 2025;12(2):81-90.

Peer Review: Evaluated by independent reviewers working in at least two different institutions appointed by the field editor.
Ethical Statement: Not available.
Plagiarism Checks: Yes - iThenticate
Conflict of Interest: Authors declare no conflict of interest.

CRediT AUTHOR STATEMENT
Ahmet Top: Conceptualization, Data curation, Formal Analysis, Investigation, Methodology, Resources, Supervision, Writing – review and editing.

Optimization-Based Tuning of PI Controller Parameters for DC Motor Speed Control

Ahmet TOP

Fırat University, Faculty of Technology, Department of Electrical and Electronic Engineering, Elazığ, Türkiye.

Abstract

Direct current (DC) motors are widely used in industrial applications due to their numerous advantages, such as high efficiency, cost-effectiveness, and adaptability. Therefore, accurate control of these motors is equally crucial. The most popular controller for regulating the speed of a DC motor is the conventional Proportional-Integral-Derivative (PID) controller. However, determining the parameters of a DC motor, developing a mathematical model, and subsequently identifying or experimentally selecting control parameters is a laborious and time-consuming process. In this study, the coefficients of the PI controller used for speed regulation of a DC motor were determined using the Particle Swarm Optimization (PSO) and Sine Cosine Algorithm (SCA) methods. The study was conducted experimentally for three different reference values and four distinct control methods, with the resulting data visualized using MATLAB. Step, sinus with offset, and sinus without offset signals were selected as reference values. The control methods employed included open-loop control, PI control, PSO-PI control, and SCA-PI control. When the results of open-loop control and optimization-based PI control were compared, it was observed that steady-state errors decreased by 91.25% and 90.41% for step reference with PSO and SCA, respectively; by 84.7% and 80.58% for sinus with offset reference with PSO and SCA, respectively; and by 76.72% and 74.75% for sinus without offset reference with PSO and SCA, respectively. Additionally, the motor demonstrated a more stable tracking of the reference values. When the PI control results were compared with PSO-PI and SCA-PI control, the steady-state error was found to decrease by an mean of 9.74% for the same reference values.

Keywords: DC motor control, PID controller, PSO,SCA, Arduino

INTRODUCTION

Motors that can be powered directly by a DC (direct current) source and convert electrical energy into mechanical energy are known as DC motors. They come in a variety of forms, including Brushed DC Motor [1], Servo Motors [2], Stepper Motors [3], Brushless DC (BLDC) Motors [4], and more. Their prices are also relatively low compared to AC Motors [5]. Furthermore, DC motors can be operated with simple and stable control algorithms. High efficiency and high initial torque in the event of abrupt load increases are further benefits [6]. On the other hand, brushless DC motors have become popular as a replacement for DC motors, which have drawbacks like the need for frequent maintenance, rapid mechanical wear of the outputs, glare, noise pollution, and the impact of the brush on efficiency [7]. Because of their benefits, including their lack of noise, low maintenance requirements, quick dynamic response, excellent torque characteristics, and effective operation, brushless DC motors may be preferred [8].

These days, DC motors are used extensively and are considered crucial to support human activity. Examples of frequently utilized systems are industrial applications [9], running a conveyor machine to transport an object [10], pumping water from below the surface to the top [11], utilizing a fan to cool the room [12], robotics [13,14], and electric vehicles [15]. Because of this, it's critical to employ a suitable control method when controlling motor speed, particularly when facing variable and non-linear loads and input disturbances [16]. The DC motor system has been fitted with many controllers, including the PI Controller [17], PID Controller [18,19], ANFIS Hybrid PID Controller [20], Fractional Order PID [21], Fuzzy Logic Controller [22], Model Reference Adaptive Control (MRAC) [23], and Integral state feedback [24]. The most popular among these is the PID controller [25-27]. PID has several benefits, including an easy-to-implement hardware or software architecture, robust resistance, good stability, and simplicity of structure [28,29]. It is used in numerous systems, including temperature control [30], airplane systems [31], robots [32], etc. The main problem often discussed in PID is parameter tuning [33,34]. In other words, to obtain optimal system performance, the proportional gain (Kp), integral gain (Ki), and derivative gain (Kd) parameter values are determined [35,36]. One of the techniques used for this is parameter determination by trial and error [37]. However parameters are hard to set with this method, as the parameter search takes a long time, the control precision is poor, and the parameters that are employed are not ideal.

Many smart methods for adjusting PID parameters have been employed by researchers recently, including Learning-Based Optimization [38], Artificial Bee Colony Algorithms [39,40], Gray Wolf Optimization [41], Firefly Algorithms [42], Differential Evolution [43], Genetic Algorithms [44], Sine Cosine Algorithms [45,46], and Water Wave Optimization [47]. When compared to other methods, parameter optimization utilizing the PSO method yields stable results, according to some research references [48].

The majority of research published in the literature on DC motor control has been performed in simulation. In experimental studies, only step references were used. It has not been shown that PSO or SCA can be effective in time-varying references for motors. In this study, the speed control of a DC motor with a PSO-based PI and SCA-based PI controller has been carried out experimentally for more than one reference. Step, offset sine and non-offset sine signals were selected as references. Open loop control, PI control, PSO-PI, and SCA-PI control were used as control methods. The reason for selecting the sinus reference is that different values of reference data can be obtained in real time. In the experiments conducted using the Arduino Due development board, the performance of the motor was first examined by performing open-loop control. Then, the steady-state error of the system was measured by performing PI control with different parameter values. In cases where the error value was greater than the specified threshold value, parameter determination was performed with PSO/SCA for a certain number of iterations. Error rates

were re-evaluated by performing PI control again with these parameters. Since step reference is generally chosen in the literature, step, offset sine, and non-offset sine were preferred as references in this study. The results obtained with the Arduino IDE serial port were transferred to the MATLAB program and the graphs drawn here. The results obtained show that PSO and SCA are quite effective in obtaining PI parameters to reduce the steady-state error in motor speed.

MATERIAL AND METHOD

PID Control

Proportional (P), Integral (I), and Derivative (D) controllers, or PID controllers for short, are typically used to regulate the speed of the DC motor. The PID controller is widely recognized as the most commonly utilized method in various nonlinear control systems [49]. The PID controller concept is essentially a straightforward three-term controller that reduces steady-state error and enhances stability [50]. For many control problems, this controller provides the most effective and straightforward solution, addressing both transient and permanent state responses. The PID controller's transfer function is typically expressed as the "gain notation" given by Equation (1) or the "time constant notation" given by Equation (2).

$$T(s) = K_p + K_i\frac{1}{s} + K_d s \quad (1)$$

$$T(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \quad (2)$$

Where T_i is the integral time constant, T_d is the derivative time constant. K_p is the proportional gain, K_i is the integral gain, and K_d is the derivative gain, When any of the gain values are set to zero, the controller type may change. For example, if K_d=0, it becomes a PI controller. While proportional control increases the response speed of the system, integral control corrects the steady-state error as it takes the integral of the error. Derivative control, on the other hand, detects rapid changes by taking the derivative of the error, that is, the slope of the error signal, and is thus effective in transient overshoots. Because the motor speed's transient state would not be considered and the steady-state error would be addressed, PI control was employed in this study.

Particle Swarm Optimization (PSO)

Particle Swarm Optimization is a swarm-based optimization algorithm inspired by the social behavior of birds and fish and developed by transferring it to computer simulations. It was introduced by Eberhart and Kennedy in 1995 [51]. Each individual in this approach is referred to as a particle [52].

The general procedure of the PSO algorithm is as follows: In the first stage, the initial positions of the particles are generated randomly. In the second stage, the fitness values of these positions are calculated using a fitness function. The fitness function is used to obtain these fitness values. In the third stage, the best position of each particle for the relevant iteration ($p_{best,i}$) and the position of the best of the swarm so far (g_{best}) are identified. To make this determination, the fitness values calculated in the second stage are used. In the fourth stage, the speed and position of each particle are updated according to $p_{best,i}$, and g_{best}. A weight coefficient (inertia weight) is used to update the speed. In the fifth stage, the best solution obtained through these updates is stored in memory and the algorithm proceeds to the next iteration. This iterative process continues until the stopping criterion is met. Equations 3 and 4 show the speed and position update equations, and Figure 1(c) shows the workflow diagram of the algorithm.

$$v_{t+1} = wv_t + c_1 r_1(p_{\text{best}} - p_t) + c_2 r_2(g_{\text{best}} - p_t) \quad (3)$$

$$p_{t+1} = p_t + v_{t+1} \quad (4)$$

where v_{t+1} is the current speed of the particle, w is the inertia weight, v_t is the previous speed of the particle, r_1 ,r_2 are random numbers generated within the range (0,1), c_1,c_2 are the learning coefficients, p_t is the previous position of the particle, p_{t+1} is the updated position of the particle [53].

Among the studies on PID controllers, the commonly used fitness functions are integral of absolute error (IAE), integral of time-weighted absolute error (ITAE), integral of squared error (ISE), and integral of time-weighted squared error (ITSE) [54]. These fitness functions are expressed in Equations 5-8. Because the ISE and IAE criteria treat all errors equally, without accounting for time, they can result in a response with a long settling time and a relatively small overshoot [55]. To address this issue, an integral of time-weighted absolute error (ITAE) is used as a fitness function in this paper [56].

$$\text{IAE} = \int_0^T \left|e(t)\right| dt \quad (5)$$

$$\text{ITAE} = \int_0^T t\left|e(t)\right| dt \quad (6)$$

$$\text{ISE} = \int_0^T e^2(t)\, dt \quad (7)$$

$$\text{ITSE} = \int_0^T te^2(t)\, dt \quad (8)$$

Where t is the time and $e(t)$ is the difference between the set point and the controlled variable.

Sine Cosine Algorithm (SCA)

The Sine Cosine Algorithm (SCA) is a newly developed population-based metaheuristic optimization technique introduced by Mirjalili. It mimics the behavior of mathematical sine and cosine functions [57]. In recent years, various versions of SCA, such as USMN-SCA [58], MTV-SCA [59], IC-SCA [60], and others, have been extensively studied and applied in DC motor control [57,61] as well as in renewable energy systems [62] and buck converter [63] optimizations. Solutions are updated based on the sine or cosine function, as represented in Equation 9.

$$R_i^{t+1} = \begin{cases} R_i^t + r_1 \cdot \sin(r_2) \cdot \left|r_3 Y_i^t - R_i^t\right| & r_4 < 5 \\ R_i^t + r_1 \cdot \cos(r_2) \cdot \left|r_3 Y_i^t - R_i^t\right| & r_4 \geq 5 \end{cases} \quad (9)$$

Here, Ri represents the current solution position, t denotes the current iteration, and Y_i indicates the target solution. The

parameter r_1 is referred to as the transformation parameter, which determines the region of the next solution. The range of the sine and cosine functions in Equation 9 is adaptively adjusted using Equation 10.

$$r_1 = b\left(1 - \frac{t}{T}\right) \quad (10)$$

where t is the current iteration, T is the total number of iterations, and b>0 is a constant. In Equation 9, $r_2 \in [0,2\pi]$ is a random variable used to determine the movement direction of the next solution (i.e., either towards or away from Y_i). Additionally, r_3 provides random weights as a stochastic factor to either increase (r_3>1) or decrease (r_3>1) the influence of Y_i on defining the distance. The term r_4 in Equation 9 is used to switch between sine and cosine functions [57]. To ensure accurate comparison with PSO, the time-weighted absolute error is used as a fitness function in this algorithm.

System Software

The Arduino Due development board was used to create the optimization and control algorithm and to facilitate bidirectional information exchange with the hardware. The software was written in the Arduino IDE (Integrated Development Environment) interface according to the workflow diagram in Figure 1. Specifically, Figure 1(a) shows the main program, Figure 1(b) shows the test subprogram, Figure 1(c) shows the PSO subprogram, and Figure 1(d) shows the SCA subprogram. After initially performing the necessary assignments and adjustments, the algorithm generates random parameters for speed control of the motor with the PI controller. These parameters are sent to the test subprogram and ensuring the motor operates for a specific period. During this period, the transient state is disregarded, and the absolute value of the difference between the steady-state reference speed (V_{ref}) and the measured motor speed (V_{est}) is taken. After completing the subprogram, these error values are divided by the number of samples as in Equation 11 and the mean absolute error (MAE) value is determined.

$$\text{MAE} = \frac{1}{N}\sum_{k=1}^{N}\left|e\left(k\right)\right| \quad (11)$$

where e is the error value, N is the total number of samples, and k is the number of samples. In this study, the MAE value was set to 1 rpm for sine without offset, while it was accepted as 0.5 rpm for other reference values. This is because the errors at the zero crossing points in the sine without offset are high. According to the algorithm, the system will control motor speed with randomly determined parameters if the MAE value is less than the threshold value. However, if it exceeds the threshold value, the optimization subprogram will be executed for the specified number of iterations. For the parameters found with optimization, the error status is rechecked by returning to the test subprogram, and motor control is provided using these parameters. The motor continues to move until the button specified for exit is pressed. PI parameter information, reference speed data, and instantaneous motor speed data were taken from the Arduino IDE serial port and entered into the MATLAB program, where the data was graphed.

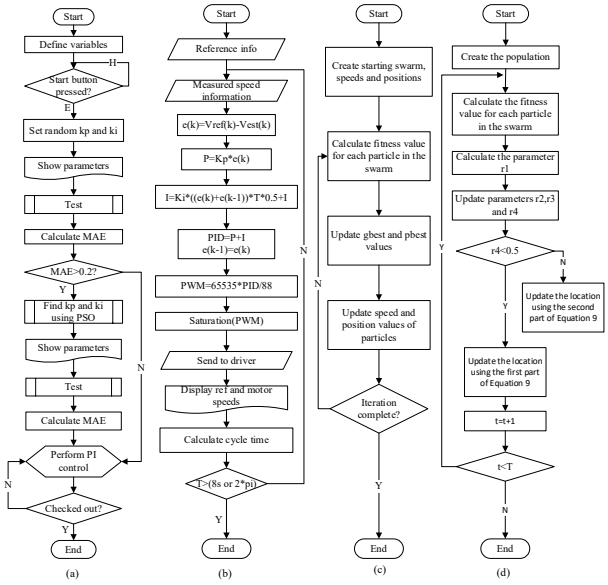

Figure 1 System software workflow diagram: a) main program, b)test subprogram, c) PSO subprogram and d) SCA subprogram

Hardware

Direct current motors have advantages such as excellent torque and speed characteristics, quick response to dynamic changes, high power-to-weight ratio, no requirement for current excitation, and low operating noise. For these reasons, they are widely used in industrial applications and robotics [64]. In this study, a 12 V DC motor with a reducer with a conversion ratio of 131.1:1 and an encoder with a resolution of 64 PPR was used. To generate voltage for this motor based on PWM (Pulse Width Modulation) values received from an Arduino Due, a SparkFun Monster Motor Driver Module containing two VNH2SP30-E H-bridge integrated circuits was used. This driver module allows the control of motors with a maximum voltage of 16 V, a continuous current draw of 14 A, and a maximum PWM frequency of 20 kHz

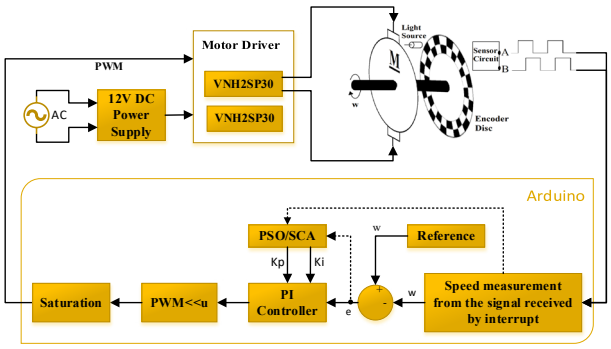

Figure 2 General block diagram of the system

Figure 2 illustrates the general schematic structure of the system, while Figure 3 shows the physical setup. The motor is powered by a 12 V voltage source, whereas the Arduino is powered via a computer connection. To enhance the precision of motor operation, the Arduino's PWM output resolution was increased from 8 bits to 16 bits, and the PWM frequency was set to 15 kHz.

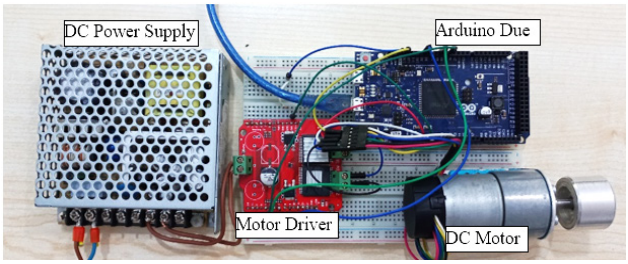

Figure 3 System setup

The motor encoder is powered by the Arduino. The signals from the encoder are transferred to the interrupt pin, where instantaneous speed measurements are conducted within the interrupt subprogram in the Arduino. The measured speed value is then compared with the reference value in the program to calculate the error value.

EXPERIMENTAL RESULTS

In this study, the results were analyzed for three distinct reference values and four different control methods for the speed control of a DC motor. Step, offset sine, and non-offset sine signals were chosen as reference values. For these references, motor speed control was implemented using open-loop control, a PI controller, a PSO-based PI controller, and an SCA-based PI controller. The steady-state MAE (Mean Absolute Error) was calculated by disregarding the transient states for the step and offset sine references, whereas for the non-offset sine reference, the MAE of the entire signal was considered to highlight the differences at zero-crossing points. Initially, as shown in Figure 4, the open-loop speed control results of the motor were obtained for a step reference value of 25 rpm.

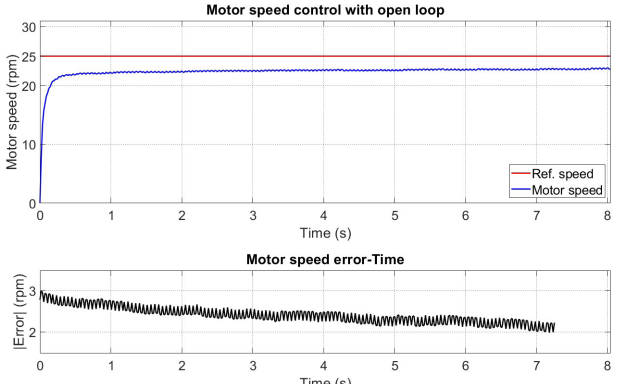

Figure 4 Motor speed control with an open loop for 25 rpm setpoint

When the open loop speed control of the motor was examined, it was observed that the motor reached a steady state in 0.66 seconds without overshoot, However, it remained constant at an mean of 22.60 rpm during the steady state, resulting in a steady state error of 2.40 rpm.

Closed loop control was achieved using the PI controller based on feedback received from the DC motor encoder. Since the motor parameters were unknown, the PI coefficients could not be determined directly. Consequently, the results obtained with different integral and proportional coefficients for PI control are shown in Figure 5.

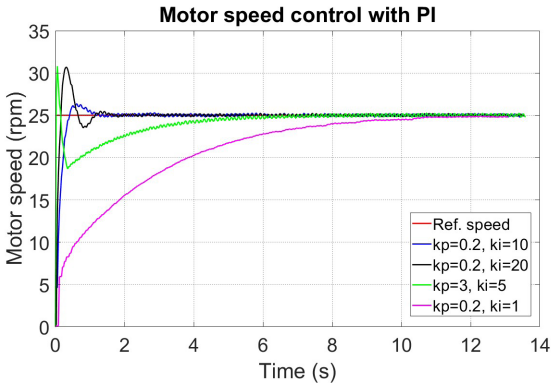

Figure 5 Motor speed control with different PI parameters for 25 rpm setpoint

When the results in Figure 5 are examined, it is observed that when the proportional and integral coefficients are low, the system does not exceed the reference value, but takes a long time to reach the steady state. Conversely, increasing the proportional coefficient and integral coefficient can accelerate the system response, but may lead to overshoots in transient states. To further improve this result, the appropriate parameter should either be calculated through mathematical modeling or determined experimentally via trial and error. However, since both methods are time-consuming when applied to different motors, PSO and SCA were used to find the appropriate parameters.

In the created algorithm, the MAE is first calculated using randomly determined parameters over a specific time or period. If this error value exceeds the threshold, adjustments are made using PSO or SCA. The motor speed results for PI parameters determined using PSO and SCA are shown in Figures 6 and 7, respectively. To ensure clarity, error graphs were plotted from the moment the speed reached the reference value, while the MAE was calculated only after the steady-state time was achieved.

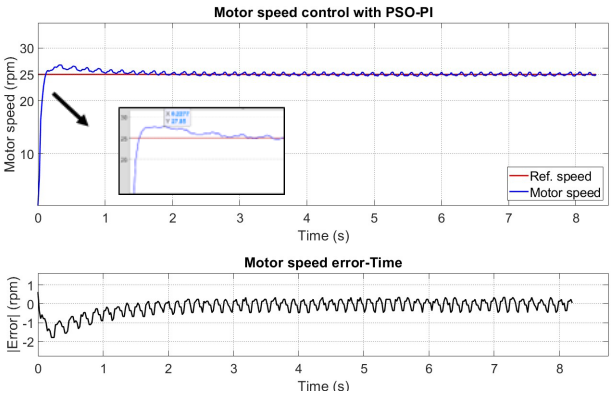

Figure 6 Motor speed control with PSO-PI for 25 rpm setpoint

The PI parameters obtained through PSO are kp=0.83 and ki=3.31. Upon evaluating the performance with these parameters, the motor exhibits a rise time of 0.1242 seconds,

an overshoot of 7.12% at 0.3312 seconds, and reaches a steady state after 1.7595 seconds. Subsequently, the motor operates with an average steady-state error of 0.21 rpm.

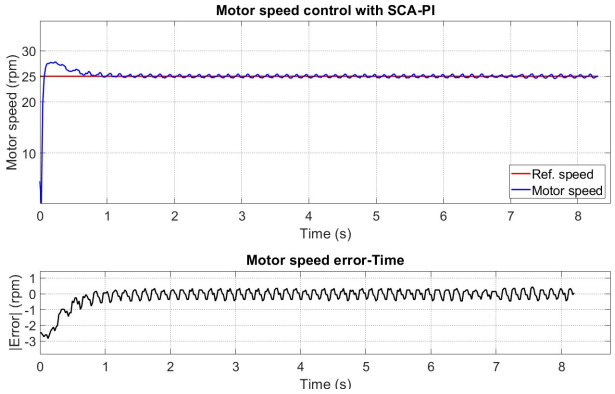

Figure 7 Motor speed control with SCA-PI for 25 rpm setpoint

The PI parameters found with SCA are kp=1 and ki=8.37. Upon analyzing the results with these parameters, the motor exhibits a rise time of 0.062 seconds, an overshoot of 7.4% at 0.22 seconds, and reaches steady state after 1.09 seconds. Subsequently, the motor operates with an average steady-state error of 0.23 rpm.

When the same control applications were applied to the reference value of 30+15Sin(wt), the results depicted in Figures 8-11 were obtained.

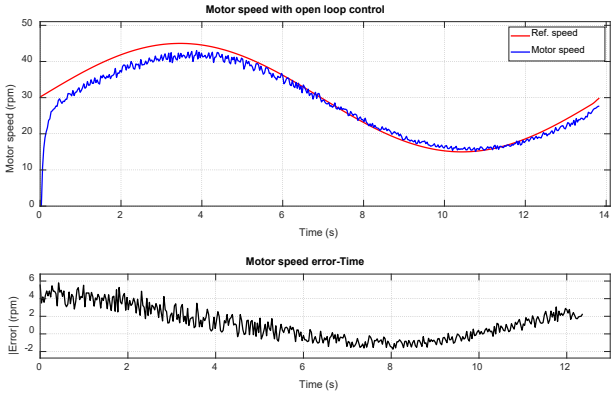

Figure 8 Motor speed control with an open loop for 30+15Sin(wt) rpm setpoint

In open-loop control, the system began tracking the reference value at 0.39 seconds, without exhibiting any overshoot during the transient state. The motor then continued its movement with an mean absolute speed of 1.70 rpm. Furthermore, as illustrated in the graph, the reference signal is followed with a noticeable delay.

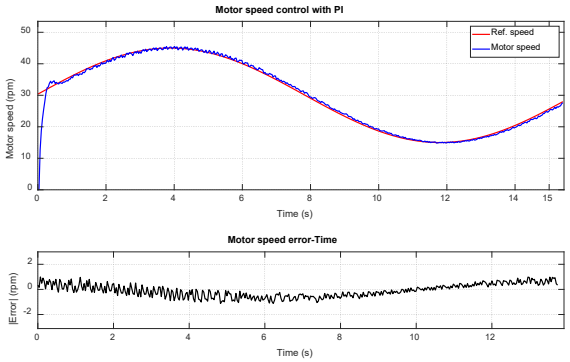

Figure 9 Motor speed control with PI for 30+15Sin(wt) rpm setpoint

For the PI control, the user set kp=0.5 and ki=23.75., and The results obtained with these parameters are shown in Figure 9. The motor speed reached its rise time at 0.27 seconds and exhibited an overshoot of 5.77% at 0.37 seconds. The motor successfully followed the sine reference with an average error of 0.443 rpm.

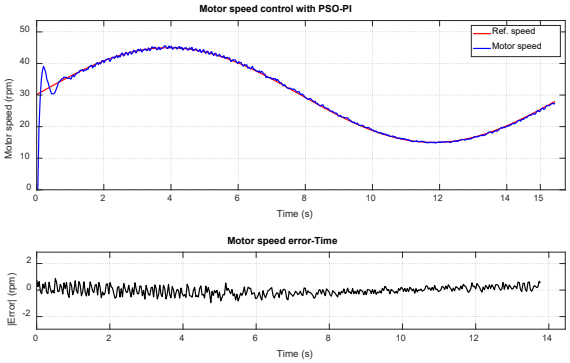

Figure 10 Motor speed control with PSO-PI for 30+15Sin(wt) rpm setpoint

After ten iterations, PSO algorithm determined the proportional parameter to be 0.92 and the integral coefficient to be 60.03. Upon evaluating the performance of the PI control with these coefficients, the motor exhibited an MAE of 0.268 rpm. The system achieved a rise time of 0.102 seconds and a peak time of 0.204 seconds, before reaching steady state at 1.29 seconds. During the peak time, the motor experienced an overshoot of 24.56%.

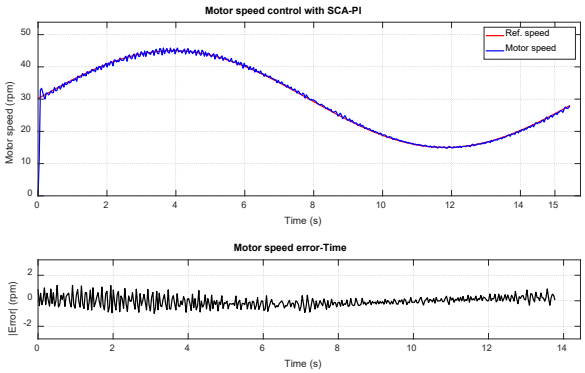

Figure 11 Motor speed control with SCA-PI for 30+15Sin(wt) rpm setpoint

After ten iterations, SCA found the proportional parameter to be 7.28 and the integral coefficient to be 55.9. When the results of the PI control performed with these coefficients were examined, the motor moved with an MAE of 0.338 rpm. The system reached a rise time of 0.068 seconds and a peak time of 0.102 seconds, before attaining steady state at 0.476 seconds. During the peak time, the motor experienced an overshoot of 8.34%.

For bidirectional motion control of the motors, the reference signal 25Sin(2wt) was applied to the motor, and control was performed using open-loop, PI, PSO-PI, and SCA-PI methods. The results obtained for each control method are presented in Figures 12, 13, 14, and 15, respectively. Two periods were considered for each control method, and the mean absolute errors were calculated. Due to the low pulse per revolution (PPR) of the encoder, speed information cannot be obtained with high accuracy, especially at low speeds, as the period of the encoder signal increases. Consequently, errors may occur at the zero-crossing points of the signal.

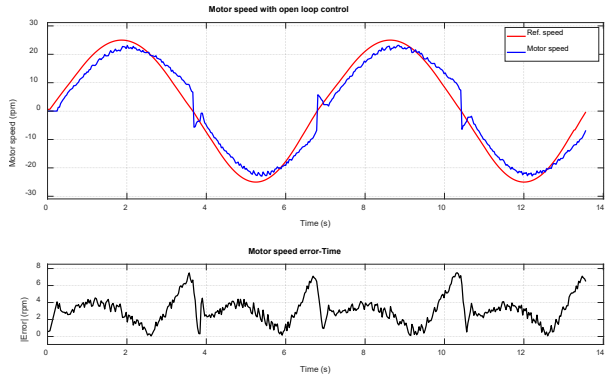

Figure 12 Motor speed control with an open loop for 25Sin(2wt) rpm setpoint

In open-loop control, the motor followed the reference with a delay of 0.22 seconds and an mean error of 3.054 rpm.

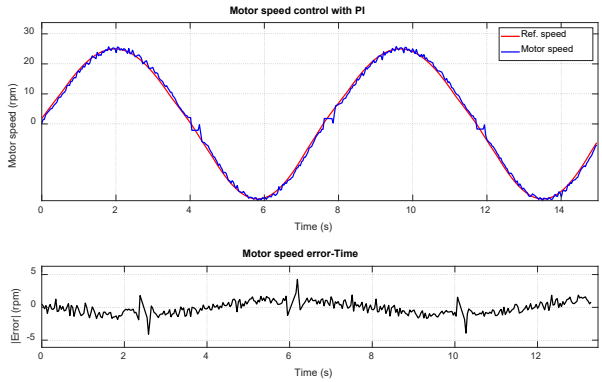

Figure 13 Motor speed control with PI for 25Sin(2wt) rpm setpoint

The experimental values produced by the user are Kp = 2.8 and Ki = 55.12. The results obtained from the closed-loop PI controller of the motor with these values are shown in Figure 13. Upon examining these graphs, it is observed that the motor follows the reference value with an mean error of 0.787 rpm, with no delay in the signal. However, jumps are present at the zero-crossing points.

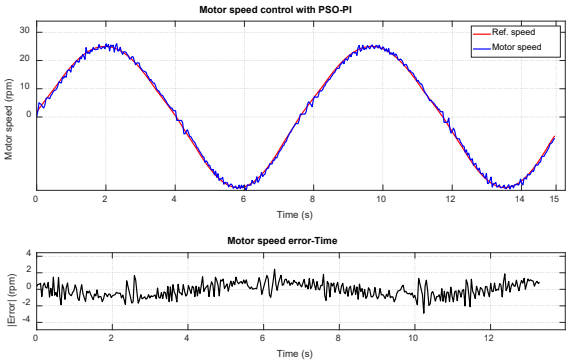

Figure 14 Motor speed control with PSO-PI for 25Sin(2wt) rpm setpoint

Since the MAE exceeded the threshold value, the parameters were recalculated using Particle Swarm Optimization (PSO), resulting in kp = 7.23, and ki = 69.21. Upon examining the control results obtained with these values, it is observed that the reference is tracked with an error of 0.712 rpm, with no delay in the signal. Additionally, the zero-crossing points occur with reduced error compared to previous results.

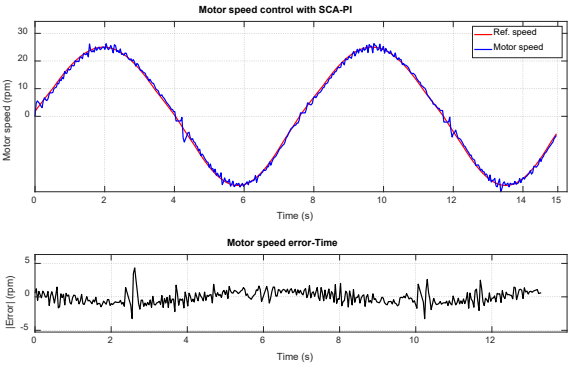

Figure 15 Motor speed control with SCA-PI for 25Sin(2wt) rpm setpoint

Since the MAE was above the threshold value, the parameters were calculated using SCA, yielding kp = 8.87, and ki = 63.35. When the control result obtained with these values is examined, it is observed that the reference is tracked with an error of 0.771 rpm, there is no delay in the signal and zero crossing points occur with less error. The results obtained with the controllers for all three references are given in Figures 16, 17, and 18.

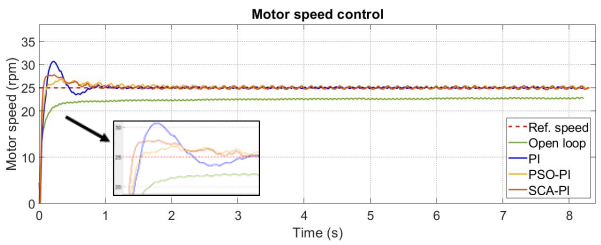

Figure 16 Motor speed control for 25rpm setpoint

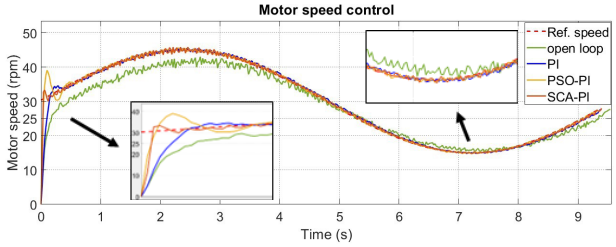

Figure 17 Motor speed control for 30+15Sin(wt) rpm setpoint

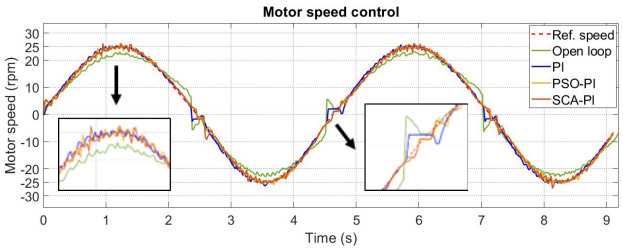

Figure 18 Motor speed control for 25Sin(2wt) rpm setpoint

When analyzing the above results, it is observed that in the DC motor controls performed with the PI parameters obtained through experimental studies using SCA and PSO, SCA provides superior performance in the transient state, while PSO proves to be more effective in reducing steady-state error.

CONCLUSION

In this study, the Particle Swarm Optimization (PSO) algorithm and the Sine Cosine Algorithm (SCA) were employed to determine the optimal proportional-integral (PI) controller parameters for the speed control of a DC motor. The experimental study utilized using three different reference values: step reference, offset sine, and non-offset sine. Additionally, bidirectional control of the motor was achieved with the non-offset sine reference. A comparative analysis was conducted on the data obtained from open-loop control, PI control with experimentally determined parameters, and PI control using the proposed optimization methods. The results demonstrated that the PI controller when optimized with the proposed methods, produced effective outcomes.

An analysis of the experimental results presented in Table 1 and the associated graphs revealed steady-state errors of 2.40 rpm, 1.70 rpm, and 3.05 rpm for step, offset sine, and non-offset sine reference values, respectively, under open-loop control. Furthermore, the motor speed failed to reach the reference speed in the open-loop control scenario. For PI control with experimentally determined coefficients, it was observed that some parameter configurations yielded fast responses accompanied by overshoots, while others resulted in slower responses with no overshoot.

When the PI parameters were optimized using PSO, the steady-state error values for the step, offset sine, and non-offset sine reference cases were reduced to 0.21 rpm, 0.26 rpm, and 0.71 rpm, respectively. Similarly, with SCA-based optimization, the error values were 0.23 rpm, 0.33 rpm, and 0.77 rpm for the same reference cases. Notably, in the DC motor speed control implementations using PSO-optimized parameters, the mean absolute error was consistently below, indicating that the motor successfully tracked the reference values with high accuracy.

Table 1 Steady-state errors of control methods according to different setpoints

Control Method / Reference	Open loop	PI	PSO-PI	SCA-PI
e_{ss}(rpm) for Step Reference	2.40	0.2 (for kp=0.8, ki=5)	0.21	0.23
e_{ss}(rpm) for Sinus with offset ref.	1.70	0.44	0.26	0.33
e_{ss}(rpm) for Sinus without offset ref.	3.05	0.78	0.71	0.77

In the operation with a sinusoidal reference without offset, it was determined that the fluctuations at the zero-crossing points of the motor speed were caused by the low resolution of the DC motor encoder. Since feedback information was obtained with fewer samples for this reference value, the instantaneous speed was measured with greater oscillation. In contrast, more stable results were obtained with a fixed reference due to the higher sampling rate used during the measurement. Since the steady-state error was considered as the primary control performance criterion and derivative control was not included to improve the transient regime, overshoots during the transition phase were not evaluated as a success metric. Experimental studies demonstrated that 10 iterations for 10 particles reduced the error to an acceptable level; therefore, the mean absolute error (MAE) was not reassessed in either optimization result. However, to enhance usability in different systems, the optimization cycle can be repeated until the error from the PSO and SCA outputs reaches an acceptable level, or the optimization process can continue until the desired error level is achieved instead of relying on a fixed number of iterations.

References

1. Chotai J, Narwekar K. Modelling and position control of brushed DC motor. In: International Conference on Advances in Computing, Communication and Control (ICAC3); 2017. p. 1–5.
2. Kumar, P., Chatterjee, S., Shah, D., Saha, U.K., & Chatterjee, S. On comparison of tuning method of FOPID controller for controlling field controlled DC servo motor. Cogent Engineering, 2017; vol. 4, no. 1.
3. Shekhawat AS, Rohilla Y. Design and control of two-wheeled self-balancing robot using Arduino. In: Proceedings of the International Conference on Smart Electronics and Communication (ICOSEC); 2020. p. 1025–1030.
4. Shekhar S, Saha PK, Thakura PR. Optimal PID tuning of BLDC drive using LQR technique. In: Proceedings of the 2019 IEEE International Conference on Intelligent Systems and Green Technology (ICISGT); 2019. p. 57–61.
5. Köse F, Kaplan K, Ertunç M. PID ve bulanık mantık ile DC motorun gerçek zamanda STM32F407 tabanlı hız kontrolü. In: Otomatik Kontrol Ulusal Toplantısı; 2013 Sep 26–28; Malatya, Türkiye.

6. Siemens Training Education Program. STEP 2000 Series. Basics of DC drives and related products.
7. Oguntoyinbo, O. PID control of brushless dc motor and robot trajectory planning and simulation with matlab/simulink, Vaasan Ammattikorkeakoulu University Of Applied Sciences Degree Programme of Inform. 2009.
8. Yedamale P. Brushless DC motor fundamentals. Chandler (AZ): Microchip Technology Inc.; 2003.
9. Okoro, I. S., & Enwerem, C. O. Robust control of a DC motor. Heliyon, 2020; vol. 6, no. 12, https://doi.org/10.1016/j.heliyon.2020.e05777.
10. Dezaki, M. L., Hatami, S., Zolfagharian, A., & Bodaghi, M. A pneumatic conveyor robot for color detection and sorting. Cognitive Robotics. 2022; vol. 2, pp. 60–72, https://doi.org/10.1016/j.cogr.2022.03.001.
11. Ahmed, M. M., Hassanien, W. S., & Enany, M. A. Modeling and evaluation of SC MPPT controllers for PVWPS based on DC motor. Energy Reports. 2021; vol. 7, pp. 6044–6053, https://doi.org/10.1016/j.egyr.2021.09.055.
12. Hendra, H., Pebriyanto, S., Hernadewita, H., Hermiyetti, H., & Yoserizal, Y. Applying Programmable Logic Control (PLC) for Control Motors. Blower and Heater in the Rubber Drying Processing, Jurnal Ilmiah Teknik Elektro Komputer dan Informatika. 2021; vol. 7, no. 1, pp. 131–141, https://doi.org/10.26555/jiteki.v7i1.20514.
13. Auzan, M., Hujja, R. M., Fuadin, M. R., & Lelono, D. Path Tracking and Position Control of Nonholonomic Differential Drive Wheeled Mobile Robot, Jurnal Ilmiah Teknik Elektro Komputer dan Informatika. 2021; vol. 7, no. 3, pp. 368–379. https://doi.org/10.26555/jiteki.v7i3.21017.
14. Maarif, A., Puriyanto, R. D., & Hasan, F. R. T. Robot Keseimbangan Dengan Kendali Proporsional-Integral-Derivatif (PID) dan Kalman Filter. IT Journal Research and Development. ITJRD. 2020; vol. 4, no. 2, pp. 117–127.
15. Ashokkumar, R., Suresh, M., Sharmila, B., Panchal, H., Gokul, C., Udhayanatchi, K. V. et al. A novel method for Arduino based electric vehicle emulator. International Journal of Ambient Energy. 2021; 43(1), pp. 4299-4304. https://doi.org/10.1080/01430750.2020.1860129.
16. Kocaoğlu S, Kuşçu H. PIC ile DC motorun hız ve konum kontrolü için gerekli PID parametrelerinin belirlenmesi ve bir uygulama. In: Otomatik Kontrol Ulusal Toplantısı (TOK); 2012.
17. Chaouch S, Hasni M, Boutaghane A, Babes B, Mezaache M, Slimane S, Djenaihi M. DC-motor control using Arduino-Uno board for wire-feed system. In: 2018 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM); 2018 Oct. p. 1–6.
18. Gasparesc G. PID control of a DC motor using LabVIEW interface for embedded platforms. In: 12th International Symposium on Electronics and Telecommunications (ISETC); 2016. p. 145–148.
19. Adel Z, Hamou AA, Abdellatif S. Design of real-time PID tracking controller using Arduino Mega 2560 for a permanent magnet DC motor under real disturbances. In: 2018 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM); 2018 Oct. p. 1–5.
20. Guo, Y., & Mohamed, M. E. A. Speed Control of Direct Current Motor Using ANFIS Based Hybrid P-I-D Configuration Controller, IEEE Access. 2020; vol. 8, pp. 125638–125647.
21. Hekimoglu B. Optimal Tuning of Fractional Order PID Controller for DC Motor Speed Control via Chaotic Atom Search Optimization Algorithm. IEEE Access. 2019; vol. 7, pp. 38100–38114.
22. Tir Z, Malik O, Hamida MA, Cherif H, Bekakra Y, Kadrine A. Implementation of a fuzzy logic speed controller for a permanent magnet DC motor using a low-cost Arduino platform. In: 2017 5th International Conference on Electrical Engineering - Boumerdes (ICEE-B); 2017 Oct. p. 1–4.
23. Akbar, M. A., Naniwa, T., & Taniai, Y. Model reference adaptive control for DC motor based on Simulink. In: 2016 6th International Annual Engineering Seminar (InAES); 2016 Aug. p. 101–106.
24. Ahmad M, Khan A, Raza MA, Ullah S. A study of state feedback controllers for pole placement. In: 2018 5th International Multi-Topic ICT Conference (IMTIC); 2018 Apr. p. 1–6.
25. Somwanshi, D., Bundele, M., Kumar, G., & Parashar, G. Comparison of fuzzy-PID and PID controller for speed control of DC motor using LabVIEW. in Procedia Computer Science. 2019; vol. 152, pp. 252–260.
26. Varshney, A., Gupta, D., & Dwivedi, B. Speed response of brushless DC motor using fuzzy PID controller under varying load condition. Journal of Electrical Systems and Information Technology. 2017; vol. 4, no. 2, pp. 310–321.
27. Ma'arif, A., Nabila, H., & Wahyunggoro, O. Application of Intelligent Search Algorithms in Proportional-Integral-Derivative Control of Direct-Current Motor System. İn: The 2019 Conference on Fundamental and Applied Science for Advanced Technology; 2019, vol. 1373, no. 1, pp. 1–10.
28. Joseph, S. B., Dada, E. G., Abidemi, A., Oyewola, D. O., & Khammas, B. M. Metaheuristic algorithms for PID controller parameters tuning: review, approaches and open problems. Heliyon. 2022; vol. 8, no. 5. https://doi.org/10.1016/j.heliyon.2022.e09399.
29. Borase, R. P., Maghade, D. K., Sondkar, S. Y., & Pawar, S. N. A review of PID control, tuning methods and applications. International Journal of Dynamics and Control. 2021; vol. 9, pp. 818–827. https://doi.org/10.1007/s40435-020-00665-4.
30. Kherkhar, A., Chiba, Y., Tlemçani, A., & Mamur, H. Thermal investigation of a thermoelectric cooler based on Arduino and PID control approach. Case Studies in Thermal Engineering. 2022; vol. 36. https://doi.org/10.1016/j.csite.2022.102249.
31. Xu, F., Liang, X., Chen, M., & Liu, W. Robust Self-Learning PID Control of an Aircraft Anti-Skid Braking System. Mathematics. 2022; vol. 10, no. 8, p. 1290, 2022. https://doi.org/10.3390/math10081290.
32. Božek, P., & Nikitin, Y. The Development of an Optimally-Tuned PID Control for the Actuator of a Transport Robot. Actuators. 2021; vol. 10, no. 8, p. 195. https://doi.org/10.3390/act10080195.
33. Sun, J., Zhou, H., Ma, X., & Ju, Z. Study on PID tuning strategy based on dynamic stiffness for radial active magnetic bearing. ISA Transactions. 2018; vol. 80, pp. 458–474.
34. Fan, Y., Shao, J., Sun, G., & Shao, X. Improved Beetle Antennae Search Algorithm-Based Lévy Flight for Tuning of PID Controller in Force Control System. Mathematical Problems in Engineering. 2020; vol. 2020. https://doi.org/10.1155/2020/4287315.
35. Potnuru, D., Mary, K. A., & Babu, C. S. Experimental implementation of Flower Pollination Algorithm for speed controller of a BLDC motor. Ain Shams Engineering Journal. 2019; vol. 10, no. 2, pp. 287–295. https://doi.org/10.1016/j.asej.2018.07.005.
36. Alagoz, B. B., Deniz, F. N., & Koseoglu, M. An efficient PID-based optimizer loop and its application in De Jong's functions minimization and quadratic regression problems. Systems & Control Letters. 2022; vol. 159. https://doi.org/10.1016/j.sysconle.2021.105090.

37. Farag W. Complex Trajectory Tracking Using PID Control for Autonomous Driving. International Journal of Intelligent Transportation Systems Research. 2019; vol. 18, no. 2, pp. 356–366. https://doi.org/10.1007/s13177-019-00204-2.
38. Ulusoy, S., Nigdeli, S. M., & Bekdaş, G. Novel metaheuristic-based tuning of PID controllers for seismic structures and verification of robustness. Journal of Building Engineering. 2021; vol. 33. https://doi.org/10.1016/j.jobe.2020.101647.
39. Du, H., Liu, P., Cui, Q., Ma, X., & Wang, H. PID Controller Parameter Optimized by Reformative Artificial Bee Colony Algorithm. Journal of Mathematics. 2022; vol. 2022, pp. 1–16. https://doi.org/10.1155/2022/3826702.
40. Wang, H., Du, H., Cui, Q., & Song, H. Artificial bee colony algorithm based PID controller for steel stripe deviation control system. Science Progress. 2022; vol. 105, no. 1, pp. 1–22. https://doi.org/10.1177/00368504221075188.
41. Bhookya, J., Kumar, M. V., Kumar, J. R., & Rao, A. S. Implementation of PID controller for liquid level system using mGWO and integration of IoT application. Journal of Industrial Information Integration. 2022; vol. 28. https://doi.org/10.1016/j.jii.2022.100368.
42. Kommula, B. N., & Kota, V. R. Direct instantaneous torque control of Brushless DC motor using firefly Algorithm based fractional order PID controller. Journal of King Saud University - Engineering Sciences. 2020; vol. 32, no. 2, pp. 133–140. https://doi.org/10.1016/j.jksues.2018.04.007.
43. Rodríguez-Molina, A., Villarreal-Cervantes, M. G., Álvarez-Gallegos, J., & Aldape-Pérez, M. Bio-inspired adaptive control strategy for the highly efficient speed regulation of the DC motor under parametric uncertainty. Applied Soft Computing. 2019; vol. 75, pp. 29–45. https://doi.org/10.1016/j.asoc.2018.11.002.
44. Gani, M. M., Islam, M. S., & Ullah, M. A. Optimal PID tuning for controlling the temperature of electric furnace by genetic algorithm. SN Applied Sciences, 2019; vol. 1, no. 8, pp. 1–8, https://doi.org/10.1007/s42452-019-0929-y.
45. Bhookya, J., & Jatoth, R. K. Optimal FOPID/PID controller parameters tuning for the AVR system based on sine–cosine-algorithm.Evolutionary Intelligence. 2019; vol. 12, no. 4, pp. 725–733. https://doi.org/10.1007/s12065-019-00290-x.
46. Hekimoğlu B. Sine-cosine algorithm-based optimization for automatic voltage regulator system. Transactions of the Institute of Measurement and Control. 2018; vol. 41, no. 6, pp. 1761–1771 https://doi.org/10.1177/0142331218811453.
47. Zhou, Y., Zhang, J., Yang, X., & Ling, Y. Optimization of PID Controller Based on Water Wave Optimization for an Automatic Voltage Regulator System. Information Technology and Control, 2019; vol. 48, no. 1, pp. 160–171. https://doi.org/10.5755/j01.itc.48.1.20296.
48. Rahayu, E. S., Ma'arif, A., & Cakan, A. Particle swarm optimization (PSO) tuning of PID control on DC motor. International Journal of Robotics and Control Systems. 2022; Vol 2, No 2.
49. Kushwah, M., & Patra, A. Tuning PID controller for speed control of DC motor using soft computing techniques-A review. Advance in Electronic and Electric Engineering. 2014; 4(2), 141-148.
50. Ang, K. H., Chong, G., & Li, Y. PID control system analysis, design, and technology. IEEE transactions on control systems technology. 2005 13(4), 559-576.
51. Eberhart R, Kennedy J. A new optimizer using particle swarm theory. In: MHS'95 Proceedings of the Sixth International Symposium on Micro Machine and Human Science; 1995 Oct. p. 39–43.
52. Yıldırım, M. Y., & Akay, R. Mobil robotlar için çok engelli ortamlarda hızlı yol planlama. Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi. 2021; 36(3), pp.1551-1564.
53. Garip, Z., Karayel, D., & Çimen, M. E. Parçacık Sürü Optimizasyon Tabanlı Mobil Robotlarda Global Yol Planlama. Journal of Smart Systems Research. 2021; Volume: 2 Issue: 1, pp. 18-26.
54. Ulusoy, A., & Güneş, M. Mobil robot kolunun PSO ile stabilizasyonu. Kahramanmaraş Sütçü İmam Üniversitesi Mühendislik Bilimleri Dergisi. 2019; 22(4), 288-297.
55. Sahib, M. A., & Ahmed, B. S. A new multiobjective performance criterion used in PID tuning optimization algorithms. Journal of Advanced Research. 2016; Vol.7, No.1, pp. 125-134.
56. Idir, A., Kidouche, M., Bensafia, Y., Khettab, K., & Tadjer, S. A. Speed control of DC motor using PID and FOPID controllers based on differential evolution and PSO. Int. J. Intell. Eng. Syst, 2018; Vol.11, No.4.
57. Ekinci, S., Hekimoğlu, B., Demirören, A., & Eker, E. Speed control of DC motor using improved sine cosine algorithm based PID controller. In 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT). IEEE. 2019, October; (pp. 1-7).
58. Wu, C., Chen, L., Xiong, H., & Hu, J. USMN-SCA: A Blockchain Sharding Consensus Algorithm With Tolerance for an Unlimited Scale of Malicious Nodes. IEEE Transactions on Network and Service Management. 2024; Volume: 22 Issue: 2.
59. Nadimi-Shahraki, M. H., Taghian, S., Javaheri, D., Sadiq, A. S., Khodadadi, N., & Mirjalili, S. MTV-SCA: multi-trial vector-based sine cosine algorithm. Cluster Computing. 2024; 27(10), 13471-13515.
60. Shinde, V., Jha, R., & Mishra, D. K. Improved Chaotic Sine Cosine Algorithm (ICSCA) for global optima. International Journal of Information Technology. 2024; 16(1), pp. 245-260.
61. Kumar, M. S., Gopisetti, S., & Sujatha, P. Optimal PI controller parameter setting for torque ripple minimization in SVPWM–DTC based BLDC motor drive using sine cosine algorithm. Engineering Research Express. 2024; 6(4), 045359.
62. Yousef, A. M., Ebeed, M., Abo-Elyousr, F. K., Elnozohy, A., Mohamed, M., & Abdelwahab, S. M. Optimization of PID controller for hybrid renewable energy system using adaptive sine cosine algorithm. International Journal of Renewable Energy Research-IJRER. 2020; pp. 670-677.
63. Nanyan, N. F., Ahmad, M. A., & Hekimoğlu, B. Optimal pid controller for the dc-dc buck converter using the improved sine cosine algorithm. Results in Control and Optimization. 2024; 14, 100352.
64. Purnama HS, Sutikno T, Alavandar S, Subrata AC. Intelligent control strategies for tuning PID of speed control of DC motor—a review. In: 2019 IEEE Conference on Energy Conversion (CENCON); 2019 Oct. p. 24–30.