e-ISSN: 2582-7464

Modern Optimization Techniques Based PID Controller Tuning for the Speed Control of a DC Motor

Subrata Pandey

Department of Electrical Engineering, National Institute of Technical Teachers' Training and Research, Kolkata, India
Email: subratapandey.ee@gmail.com

Abstract

In this paper, the optimal configuration of the Proportional Integral Derivative (PID) controller for the speed control of a DC motor are determined and compared using five optimization algorithms. The five optimization algorithms are respectively Ant Lion Optimization (ALO), Grey Wolf Optimizer (GWO), Moth-Flame Optimization (MFO), Multi-Verse Optimization (MVO) and Salp Swarm Optimization (SSO). The objective function uses The Integral of Time multiplied by Absolute Error (ITAE) as the performance index. A comparison of all these methods is done using the following step response parameters - steady-state error, settling time, maximum overshoot and rise time. ALO performed best among all the optimization algorithms.

Keywords

Ant Lion Optimization (ALO), DC Motor, Moth Flame Optimization (MFO), PID tuning, Salp Swarm Algorithm (SSA).

INTRODUCTION

The DC motor has been used in many industrial applications for a prolonged period because, it poses some good characteristics such as high starting torque, wide ranges of speed control and varieties of speed torque profiles. Because of these characteristics, it has been used in many industries such as steel rolling mills, paper mills, Robotics and the textile industry. Due to this wide range of usages, its speed control using different meta-heuristic algorithms has been used for the performance evaluations and comparisons of these algorithms, as a real-world engineering application.

Proportional Integral Derivative (PID) [1] is the predominant controller in the industries because of its simplicity of understanding and ease of implementation. This three-term controller provides good system stability when its parameters are correctly tuned. But tuning the parameters of this controller for a desired system performance parameter and stability is a trivial task.

So far, the most frequently used methods are classical/analytical. Examples of these methods are Internal Model Control, Ziegler-Nichols and Cohen-Coon method [2]. However, previous comparisons of classical methods to optimization methods such as [3], [4] proved that optimization methods are far superior in performance.

Optimization techniques copy natural phenomenon so that it has a structured procedure for solving a problem, which, one cannot solve analytically. Although, it does not guarantee the best result every time, the result is most optimized for a given performance index and it derives so by using repeated iterations. However, as it uses iterations for getting the results, it is time-consuming. In addition, the search results are not consistent when retaken. Therefore, it will give different values every time. But, advancements in technology have reduced the computing time and also previous use of varieties of optimization techniques for tuning PID controllers had shown far better results than that of the classical/analytical methods. Many optimization techniques are available for PID controllers. Some examples are Artificial Bee Colony [5], Atom Search Optimization [6], Bat Algorithm [7]. Differential Evolution [8], Flower Pollination Algorithm [9], Firefly Algorithm [10], Genetic Algorithm [11], Gravitational Search Algorithm [12], Grey Wolf Optimization [13], Salp Swarm Algorithm [14], Moth Flame Optimization [15], Multi-Verse Optimization [16], Ant Lion Algorithm [17] etc.

Many researchers previously used different optimization techniques [18] [19] [20] [21] [22] for tuning the parameters of the PID controller to an optimal value so that it improves the controllability of the dc motor. This paper also has a similar motivation. It compares some recent optimization techniques to adjust a PID controller parameters for the control of dc motor speed and determines the best optimization algorithm for the control from the results.

The algorithms that have been used in this paper are Grey Wolf Optimizer (GWO) [13], Salp Swarm Optimization (SSO) [14], Moth-Flame Optimization (MFO) [15], Multi-Verse Optimization (MVO) [16] and Ant Lion Optimization (ALO) [17]. These are some of the most recent optimization techniques in the literature. These algorithms are applied to select the control setting for a PID controller used to regulate the speed of a DC motor. The effectiveness of the algorithms will be observed and analyzed using MATLAB.

PRINCIPLE OF OPTIMIZATION ALGORITHMS

This section presents a short introduction of ALO, GWO, MFO, MVO and SSO algorithms.

Antlion Optimization

The Ant Lion Optimization (ALO) method [17] is an evolutionary optimization technique that is inspired by the behavior of antlions, a type of insect that preys on ants. The technique have found application across a multitude of domains, ranging from engineering design to social network analysis.

In general, the antlion optimization method is based on the concept of a sinkhole, which is a type of pitfall trap that antlions use to capture their prey. In this technique, the prey (a vector of parameters) is moved randomly in the search space, and evaluated according to a predefined fitness function. This process is repeated until the prey is located in the sinkhole, which is considered the optimal point in the search space.

One of the advantages of the antlion optimization method is that it is relatively simple to implement, and can be applied to a wide variety of problems. In addition, it is suitable for problems with multiple constraints.

Grey Wolf Optimizer

Grey Wolf Optimization (GWO) [13] is an optimization algorithm used for solving nonlinear optimization problems. It was developed by Mirjalili and Lewis in 2014 and is based on the behavior of grey wolves in nature. Unlike other meta-heuristic algorithms, GWO does not use any additional parameters, which makes it easier to implement.

GWO algorithm uses a set of solutions for exploring the problem space. These solutions are called wolves. These wolves search for the optimal solution by examining the fitness of the population and updating their positions based on the success of their peers. This approach mimics the way in which wolves interact with each other in the wild, where weaker members of the pack are driven away by the stronger wolves until only the fittest survive.

GWO uses three types of wolves to search for the optimal solution. The alpha wolf is the leader of the pack and is responsible for searching the most promising areas of the problem space. The beta and delta wolves act as followers, helping the alpha wolf to explore more of the problem space and select promising solutions.

GWO is suitable for many different types of optimization problems, such as regression, classification, and feature selection. It is also computationally efficient, making it a good choice for real-time optimization applications.

Overall, Grey Wolf Optimization is a powerful algorithm that can be used to solve a range of different optimization problems. It is easy to implement and computationally efficient, making it a good choice for real-time optimization applications.

Moth Flame Optimization

Moth Flame Optimization (MFO) [15] algorithm is developed by researchers at the University of Sheffield. It is inspired by the behavior of moths circling around a flame and it draws its foundation from the collective intelligence of a swarm. The MFO algorithm is based on the principles of swarm intelligence and is used to solve complex optimization problems such as finding the global optimum of a given function.

MFO works by having a number of moths search for an optimizable space to find a global optimum. Each moth is a solution within the search domain and is initialized with a random solution vector. The moths are then evaluated using an objective function. The moths then adjust their solution vector according to their evaluation and the values of other moths in the same search space.

The adjustment of the solution vector is done using a number of parameters such as the intensity of the flame and the attraction of the opposite sex. The intensity of the flame and the attraction of the opposite sex determine the distance that a moth moves in the search space during each iteration. The MFO algorithms aim is the discernment of the global optimum within the provided function.

MFO stands as a straightforward yet highly effective algorithm for tackling intricate optimization challenges. It is capable of finding the global optimum of a given function in a short amount of time. It also provides high accuracy as it can take into account multiple parameters when adjusting the solution vector. It is also a parallelizable algorithm, meaning that it can be executed by multiple processors or computers in parallel.

The main disadvantage of MFO is that it does not scale well for large optimization problems.

Multi-Verse Optimization

Multi-Verse optimization (MVO) [16] is a type of evolutionary optimization technique that uses multiple universes, or populations, of solutions. It provides a powerful way to explore a large and complex search space for a given problem and can be applied to optimization tasks such as finding the minima or maxima of a function or system of equations.

The multi-verse optimization technique works by creating multiple universes, or populations, of solutions, and each universe is then tested and optimized. The different universes are then compared and the best solutions are selected and combined to create a new universe of solutions. This process is repeated until a satisfactory solution is found.

The main advantage of multi-verse optimization is its ability to explore a wide range of solutions, giving it the potential to find a better solution than traditional optimization techniques. Additionally, it also allows for a more efficient search as the same computation can be used in multiple universes, saving time and resources.

Salp Swarm Algorithm

Salp Swarm Optimization (SSA) [14] is inspired by the natural behavior of salps, small barrel-shaped ocean creatures. It is used to solve complex optimization problems in various areas, e.g., machine learning, robotics, image processing, and engineering.

SSA is based on the homing behavior of salps, which includes the ability to move from one location to another in search of food or a favorable environment. This behavior is modeled using a population of particles that search the solution space, guided by an evaluative function and a number of control parameters.

At each iteration, the particles move towards the best solution in their respective neighborhoods, according to the parameters set. This is done by adjusting the direction, speed, and position of the particles using a number of operations, such as reflection, expansion, contraction, and randomization.

By repeating this process until a suitable solution is found, the SSA can accurately locate the most optimal remedy for the issue. It is incredibly efficient, and can often produce better results than methods like genetic algorithms and simulated annealing.

SYSTEM DESCRIPTION

A DC motor with separate excitation circuit is used in this experiment. Following is the method for obtaining the open loop transfer function of the DC motor.

The back emf, $e_b(t)$ for a fixed value of flux is proportional to the angular speed, $\omega(t)$ of the motor and hence can be written as –

$$e_b(t) = K_b \frac{d\theta(t)}{dt} = K_b \omega(t) \quad (1)$$

Here,

$e_b(t)$= The back emf

$\theta(t)$= The angular displacement

$\omega(t)$= The angular speed

K_b= The back emf constant

The armature voltage $e_a(t)$ controls the speed of the DC motor. The differential equation for the armature voltage is –

$$e_a(t) = L_a \frac{di_a(t)}{dt} + R_a i_a(t) + e_b \quad (2)$$

Here,

$e_a(t)$= The armature voltage

L_a= The armature inductance

$i_a(t)$= The armature currents

R_a= The armature resistance

$e_b(t)$ = The back emf

For a zero-load torque, the torque produced by the armature current can be expressed as –

$$T(t) = J\frac{d\omega(t)}{dt} + B\omega(t) = K_m i_a(t) \quad (3)$$

Here,

$T(t)$= The torque produced by the armature current

J= The moment of Inertia of the motor

$\omega(t)$= The angular speed

B= The frictional coefficient of motor

K_m= The torque constant

$i_a(t)$= The armature currents

The Laplace transform of (1), (2) and (3) with zero initial conditions are as follows –

$$E_b(s) = K_b \Omega(s) \quad (4)$$

$$E_a(s) = (L_a s + R_a) I_a(s) + E_b(s) \quad (5)$$

$$T(s) = (Js + B)\Omega(s) = K_m I_a(s) \quad (6)$$

So, the open loop transfer function of a DC motor describing the relationship between the input voltage and the output speed can be written as –

$$\frac{\Omega(s)}{E_a(s)} = \frac{K_m}{(L_a s + R_a)(Js + B) + K_b K_m} \quad (7)$$

The parameter values of Eq. (7) are as follows.

$K_m = K_b = 0.01$ V/m/s

$J = 0.01$ kg-m^2

$B = 0.1$ N-m-s

$L = 0.5$ H

$R = 1\ \Omega$

So, the final expression of the transfer function of a DC motor that relates the motor speed with the input voltage is –

$$\frac{\Omega(s)}{E_a(s)} = \frac{0.01}{(0.5s+1)(0.01s+0.1)+0.0001} \quad (8)$$

THE PID CONTROLLER

The Proportional-Integral-Derivative (PID) Controller is widely prevalent in usage in the industry today, and is employed in a wide range of fields, from aerospace to robotics, to automated manufacturing.

A PID Controller is a form of a feedback control system. It works by comparing the desired output of a system to the actual output and then adjusting the system parameters to reduce the error between the two. The controller takes the form of a mathematical equation that takes in three input variables, the Proportional (P) term, the Integral (I) term, and the Derivative (D) term, and produces a control output.

The Proportional term is the simplest of the three and is used to measure the error of the system. This term is the basis of the control loop and is responsible for producing the majority of the output. The Integral term works by accumulating the error over time and serves to counteract any persistent offset in the system. The Derivative term is used to measure the rate of change of the error and is used to reduce the overshoot and oscillations of the system.

The transfer function of the controller is given as follows –

$$C(s) = K_p + \frac{K_i}{s} + K_d s \quad (9)$$

Here,

$C(s)$ = The controller transfer function

K_p = The proportional gain

K_i = The integral gain

K_d = The derivative gain

The PID Controller has several advantages, one of the most important being its simplicity. Its mathematical equation is easy to understand, and it is relatively quick to implement. Additionally, it is extremely versatile and can be used in a wide range of applications.

The PID Controller can be used for many different purposes, from controlling the temperature of a furnace to controlling the position of a robot, to regulating the speed of an electric motor. It can also be used in closed-loop control systems, such as those used in commercial aircraft, or to control the attitude of a satellite in orbit around the Earth.

The PID Controller is an incredibly useful and versatile tool in control engineering and is used in a wide range of applications. Its ease of use and flexibility make it an invaluable part of many automated systems.

EXPERIMENTAL SETUP

At first, five different values of K_p, K_i and K_d were determined using the five different optimization techniques mentioned in section 2. These values were then used to determine the transfer function models of five different PID controllers with the help of the standard PID controller transfer function model given in (9). The DC motor transfer function model given in (8) was then placed in series with each one of these controller transfer functions. These series connected systems serve as the forward gain for the unity feedback negative closed-loop transfer functions. The unit step responses of these systems were then determined. The block diagram of the proposed system is given in Fig. 1 below.

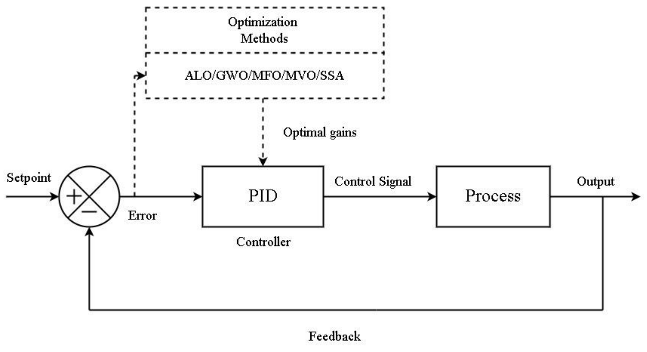

Figure 1. Block Diagram of the Proposed System (ALO- Ant Lion Algorithm, GWO-Grey Wolf Optimizer, MFO- Moth Flame Optimization, MVO- Multi-Verse Optimization, SSA-Salp Swarm Algorithm.)

The responses were then compared against different parameters such as percentage of overshoot, rise time, settling time and percentage of error to determine the best controller and the best optimization technique among the five. The computation time for each of the optimization techniques were also determined and compared to facilitate the result of the comparison.

WORKING OF THE OPTIMIZATION ALGORITHMS

Optimization algorithm selects the optimal values of control parameters to minimize the given function which is called a cost function.

Here, the control parameters are K_p, K_i, K_d and the integral of time-weighted absolute error (ITAE) is the objective function.

The formula for the ITAE is given as follows-

$$ITAE = \int t|e|dt \tag{10}$$

Here $|e|$ is the absolute value of the error and t is the time. The absolute error is amplified over passing of time in ITAE. As a result, errors that occur later are weighted significantly more strongly than those that occur earlier. Compared to the other tuning techniques, ITAE tuning results in systems that settle much more quickly. The cost function's inclusion of a time multiplication term increases the consequences for oscillation at later points in the time response. Consequently, it effectively shortens the closed-loop system's settling time.

For each optimization method the number of search agents, the maximum number of iterations, lower bound and upper bounds were kept exactly the same to get an accurate result. The values of the tuning parameters for each algorithm are given below.

- The number of search agents = 10
- The maximum number of iterations = 100
- The lower bound of PID control parameters = 0
- The upper bound of PID control parameters = 200

The experiments were carried out using the MATLAB R2014a environment on a PC with Corei3, 2 Giga Hertz processor and 4 Giga Bytes of RAM.

RESULTS AND DISCUSSION

The Table 1 shows the PID parameter values determined using the optimization techniques and their plot of convergence is shown in Fig. 2. The convergence rate of an algorithm is an important indicator of its performance. Higher convergence rate means better cost function minimization. Higher accuracy with lesser number of iterations is desirable of a good optimization algorithm.

The values of the cost function ITAE for each iteration up to the 100th iteration is shown in Fig 2 for all five optimization methods.

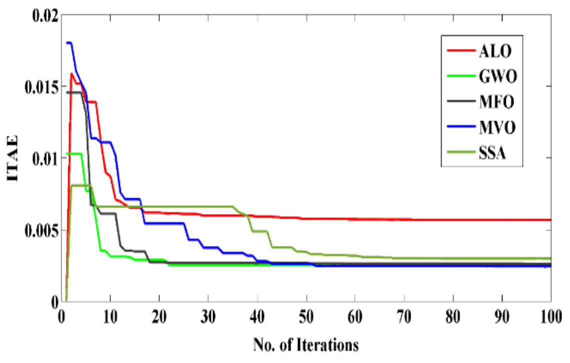

Figure 2. Convergence behaviour of optimization algorithms for ITAE

The ITAE value of MVO, MFO and GWO are approximately the same and are closer to zero than ALO and SSA. As the ITAE is an error value so, values with lower

ITAE are better performers than with the higher values. So, MVO, MFO and GWO performed better than ALO and SSA in terms of accuracy. However, error values of all these optimization techniques are very low and hence they can be neglected.

As can be seen in Fig. 2, in terms of convergence rate, performance wise, the optimization techniques could be ordered as ALO, MFO, GWO, MVO and SSA. Where ALO performed the best in terms of number of iterations whereas SSA took the highest numbers of iterations to reach its final value. Higher convergence rate also means the algorithm would need lesser computation time because a greater number of iterations need more computation time. As the error value of all the techniques are very low hence, all of them can be considered approximately the same.

The computation time of all these methods was also calculated along with the tuning parameters and given in Table 1. Among all, the computation time of MVO is the lowest needing around 261 seconds. On the other hand, GWO took the highest computation time of around 286 seconds. ALO lies somewhat in the middle of these two values. For ALO only around 20 iterations would have been sufficient for reaching its final value, which is one fifth of the number of iterations it performed. Hence, it would require lesser computation time. Therefore, the performance of the ALO algorithm is also good in terms of computation time.

Table 1. Optimal control parameters obtained from different optimization methods for G(s)

Controller	K_p	K_i	K_d	Computation time (seconds)
PID-ALO	170.4719	199.9964	11.7159	282.735863
PID-GWO	119.5194	199.9296	9.5584	286.332469
PID-MFO	117.6548	198.522	9.0034	273.597890
PID-MVO	119.554	200.0000	9.6300	261.409764
PID-SSA	126.4679	200.0000	9.6654	267.141877

The PID controller are integrated with the values derived through the optimization methods and the DC motor's reaction to the step voltage input is measured. The measured output is provided in Fig. 3 as well as in Table 1. In the result four key data points were included. These data points are – the percentage of overshoot (OS, %), the rise time (t_r), the settling time (t_s), and the error (E). The percentage of overshoot is the percentage by which the maximum peak value exceeds the desired response of the system. Lower the percentage of overshoot better the system is. The rise time is the time taken by any response to reach 90% percent of the final value from 10% of its final value. A good system has a low rise time. The settling time is the time required by the response to reach and stay within 2% of its final value. For a system with good performance this also needs to be as low as possible.

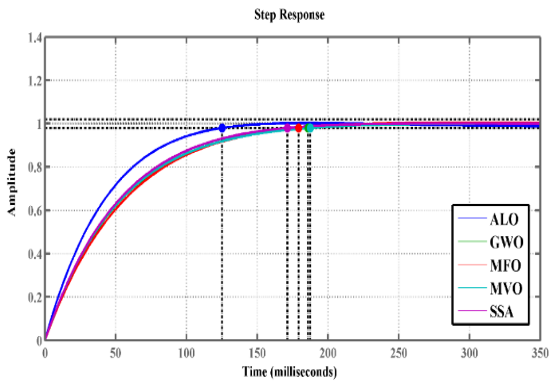

Figure 3. Step responses of different tuning methods

All of the overshoot values of all the five algorithms here are below 1% and hence all of them are approximately the same.

In terms of rising time, ALO performed best with a value of 80.2 milli-seconds whereas MFO has the highest rising time with a value of 111 milli-seconds. The values of GWO, MVO and SSA are 110,110 and 104 respectively and lies in between the highest and the lowest values. Here also the difference of time between ALO and others are significantly higher compared to the difference in time between GWO, MFO, MVO and SSA themselves. Hence, performance of ALO is significantly better in terms of rising time.

Of all the algorithms ALO had the least settling time of 125 milli-seconds, which can be seen in both Fig. 3 and in Table 2. MVO had the highest settling time of 188 milli-seconds. The settling time of the GWO, MFO and SSA lies in between the highest and the lowest value. The difference in the settling time between ALO and the others was far greater than that of the difference among GWO, MFO, MVO and SSA themselves. Hence, ALO outperformed all other techniques with a greater margin compared to rest of the optimization techniques.

The error values of all the optimization techniques were very low and hence can be ignored altogether.

Table 2. Time domain performance of different algorithms for speed control of dc motor.

Controller	OS, %	t_r(milli-sec)	$t_s(\pm 2\%)$ (milli-sec)	E
PID-ALO	0.226	80.2	125	0.0057
PID-GWO	0.101	110	186	0.0025
PID-MFO	0.396	111	179	0.0026
PID-MVO	0.073	110	188	0.0024
PID-SSA	0.0532	104	171	0.0030

CONCLUSION

Even today, PID controller is widely prevalent in usage in the industry and for decades tuning its parameters for obtaining the optimal output has been an important research problem. In this paper, the ALO, GWO, MFO, MVA and SSO algorithms were used for determining the optimal values of PID controller parameters for the DC Motor speed control. ITAE cost function were utilized for fine tuning the parameters. Convergence curves for varieties of optimization techniques were plotted and analyzed. Then, the computation time for each of these algorithms was also determined. At last, output of the DC motor speed control for a step voltage input was analyzed in terms of the rise time, the percentage of overshoot along with the error and the settling time. Overall, ALO performed better compared to other algorithms.

Acknowledgement

I extend my profound gratitude to my parents, Shree Prosanta Kumar Pandey and Smt. Tapasi Panday, as well as my elder brother Shree Partha Pandey. Their unwavering support has been instrumental on this remarkable journey of mine.

REFERENCES

[1] S. Bennett, "Development of the PID Controller," *IEEE Control Syst*, vol. 13, no. 6, 1993, doi: 10.1109/37.248006.

[2] K. Astrom, *PID Controllers, 2nd Edition*. 1995.

[3] J. M. S. Ribeiro, M. F. Santos, M. J. Carmo, and M. F. Silva, "Comparison of PID controller tuning methods: Analytical/classical techniques versus optimization algorithms," in *2017 18th International Carpathian Control Conference, ICCC 2017*, 2017. doi: 10.1109/CarpathianCC.2017.7970458.

[4] M. I. Solihin, L. F. Tack, and M. L. Kean, "Tuning of PID Controller Using Particle Swarm Optimization (PSO)," *Int J Adv Sci Eng Inf Technol*, vol. 1, no. 4, 2011, doi: 10.18517/ijaseit.1.4.93.

[5] D. Karaboga, "An idea based on Honey Bee Swarm for Numerical Optimization," *Technical Report TR06, Erciyes University*, no. TR06, 2005.

[6] W. Zhao, L. Wang, and Z. Zhang, "Atom search optimization and its application to solve a hydrogeologic parameter estimation problem," *Knowl Based Syst*, vol. 163, 2019, doi: 10.1016/j.knosys.2018.08.030.

[7] X. S. Yang, "A new metaheuristic Bat-inspired Algorithm," in *Studies in Computational Intelligence*, 2010, vol. 284. doi: 10.1007/978-3-642-12538-6_6.

[8] 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, 1997, doi: 10.1023/A:1008202821328.

[9] X. S. Yang, "Flower pollination algorithm for global optimization," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7445 LNCS. doi: 10.1007/978-3-642-32894-7_27.

[10] X. S. Yang, "Firefly algorithms for multimodal optimization," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5792 LNCS, pp. 169–178, 2009, doi: 10.1007/978-3-642-04944-6_14/COVER/.

[11] X.-S. Yang, "Genetic Algorithms," *Nature-Inspired Optimization Algorithms*, pp. 77–87, 2014, doi: 10.1016/B978-0-12-416743-8.00005-1.

[12] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A Gravitational Search Algorithm," *Inf Sci (N Y)*, vol. 179, no. 13, 2009, doi: 10.1016/j.ins.2009.03.004.

[13] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, Mar. 2014, doi: 10.1016/J.ADVENGSOFT.2013.12.007.

[14] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems," *Advances in Engineering Software*, vol. 114, pp. 163–191, Dec. 2017, doi: 10.1016/J.ADVENGSOFT.2017.07.002.

[15] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl Based Syst*, vol. 89, pp. 228–249, Nov. 2015, doi: 10.1016/J.KNOSYS.2015.07.006.

[16] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-Verse Optimizer: a nature-inspired algorithm for global optimization," *Neural Computing and Applications 2015 27:2*, vol. 27, no. 2, pp. 495–513, Mar. 2015, doi: 10.1007/S00521-015-1870-7.

[17] S. Mirjalili, "The Ant Lion Optimizer," *Advances in Engineering Software*, vol. 83, pp. 80–98, May 2015, doi: 10.1016/J.ADVENGSOFT.2015.01.010.

[18] P. Pal and R. Dey, "Optimal PID Controller Design for Speed Control of a Separately Excited DC Motor: A Firefly Based Optimization Approach," *The International Journal of Soft Computing, Mathematics and Control*, vol. 4, no. 4, 2015, doi: 10.14810/ijscmc.2015.4404.

[19] J. Agarwal, G. Parmar, R. Gupta, and A. Sikander, "Analysis of grey wolf optimizer based fractional order PID controller in speed control of DC motor," *Microsystem Technologies*, vol. 24, no. 12, 2018, doi: 10.1007/s00542-018-3920-4.

[20] M. A. Ibrahim, A. K. Mahmood, and N. S. Sultan, "Optimal PID controller of a brushless DC motor using genetic algorithm," *International Journal of Power Electronics and Drive Systems*, vol. 10, no. 2, 2019, doi: 10.11591/ijpeds.v10.i2.pp822-830.

[21] R. K. Achanta and V. K. Pamula, "DC motor speed control using PID controller tuned by jaya optimization algorithm," in *IEEE International Conference on Power, Control, Signals and Instrumentation Engineering, ICPCSI 2017*, 2018. doi: 10.1109/ICPCSI.2017.8391856.

[22] M. O. Mustafa, "Optimal Parameter Values of Pid Controller for DC Motor Based on Modified Particle Swarm Optimization With Adaptive Inertia Weight," *Eastern-European Journal of Enterprise Technologies*, vol. 1, 2021, doi: 10.15587/1729-4061.2021.225383.