INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2020, VOL. 66, NO. 4, PP. 737-743
Manuscript received August 14, 2019; revised October, 2020. DOI: 10.24425/ijet.2020.134035

Design of a Predictive PID Controller Using Particle Swarm Optimization

Norhaida Mustafa, and Fazida Hanim Hashim

Abstract—**The proportional-integral-derivative (PID) controller is widely used in various industrial applications such as process control, motor drives, magnetic and optical memory, automotive, flight control and instrumentation. PID tuning refers to the generation of PID parameters (K_p, K_i, K_d) to obtain the optimum fitness value for any system. The determination of the PID parameters is essential for any system that relies on it to function in a stable mode. This paper proposes a method in designing a predictive PID controller system using particle swarm optimization (PSO) algorithm for direct current (DC) motor application. Extensive numerical simulations have been done using the Mathwork's Matlab simulation environment. In order to gain full benefits from the PSO algorithm, the PSO parameters such as inertia weight, iteration number, acceleration constant and particle number need to be carefully adjusted and determined. Therefore, the first investigation of this study is to present a comparative analysis between two important PSO parameters; inertia weight and number of iteration, to assist the predictive PID controller design. Simulation results show that inertia weight of 0.9 and iteration number 100 provide a good fitness achievement with low overshoot and fast rise and settling time. Next, a comparison between the performance of the DC motor with PID-PSO, with PID of gain 1, and without PID were also discussed. From the analysis, it can be concluded that by tuning the PID parameters using PSO method, the best gain in performance may be found. Finally, when comparing between the PID-PSO and its counterpart, the PI-PSO, the PID-PSO controller gives better performance in terms of robustness, low overshoot (0.005%), low minimum rise time (0.2806 seconds) and low settling time (0.4326 seconds).**

Keywords—**proportional-integral-derivative (PID) controller, particle swarm optimization (PSO) algorithm, optimization, predictive PID**

I. INTRODUCTION

THE proportional-integral-derivative (PID) controller is one of the earlier control strategies used to control the speed and position in various control applications. PID controllers are known for their simplicity and high performance in different operating conditions, thus making them a popular choice in industrial applications. PID controllers are often used in industrial control systems such as process control, motor drives, magnetic and optical memory, automotive control, flight control and instrumentation for its reliability in tuning the control parameters to optimum control values. Additionally, the PID controller is known for its simple structure, simple design, low maintenance, stable steady-state error and ease of use. However, regardless of its widespread usage, one of its major

There are several methods for tuning the PID controllers to obtain optimum values for the PID parameters. The conventional methods are Ziegler-Nichols method, Ziegler-Nichols reaction curve method, Cohen Coon reaction curve method and Tyreus-Luyben. The Ziegler-Nichols method although most of the time gives satisfactory tuning parameters, it sometimes tends to create a high overshoot [2]. To increase the capability of conventional PID parameter tuning techniques, intelligent approaches using heuristic algorithms have been recommended by researchers in the field. Algorithms such as genetic algorithms (GA), differential evolutionary (DE) algorithm, ant colony optimization (ACO), biogeography based optimization (BBO) and the particle swarm optimization (PSO) are among the popular algorithms used to acquire the PID tuning parameters [3].

Previous studies have adopted PSO for tuning PID controllers for various reasons; from conventional control applications to modern design applications. Some of the latest design applications include utilizing PSO tuned PID for controlling the camera position in unmanned aerial vehicle (UAV) [4], controlling a twin rotor multi-input multi-output (MIMO) [5], controlling a DC-DC boost converter in a photovoltaic (PV) system [6], controlling the speed control of a hybrid electric vehicle [7], controlling a nonlinear double-pendulum overhead crane [8], etc. It can be seen that although PID is prevalent in control systems for decades, with the help of soft computing techniques such as the PSO, it could successfully improve the performance of the conventional controller [9].

In this paper, a predictive PID controller parameter tuning method using the PSO algorithm is proposed and applied to a DC motor system. The main advantage of using PSO algorithm is its ability in generating an auto tuning method to find the best PID parameters, K_p, K_i, and K_d, without complex mathematical descriptions. A detailed study is presented where PSO algorithm is implemented using various inertia weights from 0.4 to 0.9 with the iteration number of 60. The iteration number from 30 to 100 with inertia weights of 0.9 has also been studied and evaluated. A comparative study in controlling the DC motor system is carried out between PID tuned with PSO, PID with a gain of 1 and a DC motor system with the absence of a PID controller. Finally, a comparative study for the performance of

This work was supported by the Universiti Kebangsaan Malaysia grant scheme number GUP-2017-074.

Norhaida Mustafa is with Mara Japan Industrial Institute (norhaida.mustafa@mara.gov.my).

Fazida Hanim Hashim* is with Universiti Kebangsaan Malaysia (fazida@ukm.edu.my).

PID-PSO and proportional-and-integral-PSO (PI-PSO) is also evaluated.

The remaining parts of the paper is organized as follows. Section 2 presents the system architecture consisting of an overview of the PID controller and PSO algorithm. A PSO based PID controller design is presented in Section 3. Section 4 discusses the simulation results and section 5 concludes the findings of the study.

II. System Architecture

A. *Overview of the PID Controller*

PID controller has been widely used in many control applications due to its simplicity, ease of use and good performance. The PID controller tuning algorithm combines three separate parameters; the proportional (K_p), integral (K_i) and derivative (K_d) values with a control loop feedback mechanism. The proportional, K_p, value determines the reaction to the current error, the integral, K_i, value determines the reaction based on the sum of recent errors, and the derivative, K_d, value determines the reaction based on the rate at which the error has been changing [10]. Fig. 1 illustrates the core architecture of a PID controller.

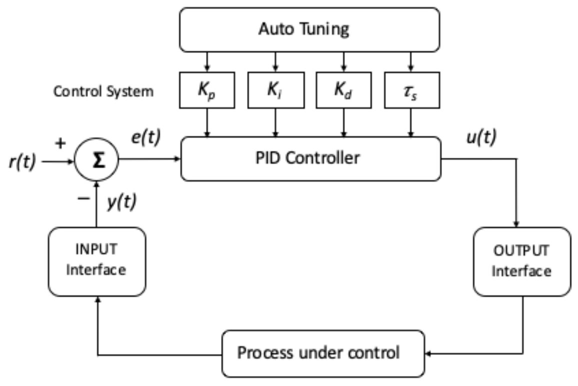


Fig. 1. A control loop with a PID controller [11]

As depicted in Fig. 1, the error voltage, *e(t),* is the difference between reference voltage, *r(t),* and the real output voltage, *y(t).* Error voltage enters PID controller and a control variable, *u(t),* comes out of the controller. A PID controller attempts to minimize the error between a real output voltage, *y(t),* and its reference voltage, *r(t),* through feedback controller by adjusting the control inputs. The control variable, *u(t),* is proportional to the error, the sum of all the previous errors, and the change rate of the error at the instant [12]. According to [13], the PID controller parameters carry the following characteristics:

1) The proportional (K_p) – provides an overall control action proportional to the error signal through the all pass gain factor.
2) The integral (K_i) – reduces steady state errors through low frequency compensation by an integrator.
3) The derivative (K_d) – improves transient response through high frequency compensation by a differentiator.

For optimum performance, K_p, K_i and K_d are mutually dependent in tuning. By tuning the three parameters in the PID controller algorithm, the controller can provide a control action designed for a system's specific process requirements. The PID controller is described in equation (1) as:

$$u(t) = k_p e(t) + k_i \int e(t)dt \ + k_d \frac{de(t)}{dt} \tag{1}$$

Where *e(t) = r(t) – y(t)* is the error representing the difference between the reference voltage and the real output voltage.

For a simple feedback control system with a PID controller, the transfer function of the PID controller is described by equation (2):

$$G(s) = K_p + \frac{K_i}{s} + sK_d \tag{2}$$

In this paper, an optimum tuning method for the parameters K_p, K_i and K_d has been determined using optimization techniques utilizing PSO algorithm by minimizing relevant performance measurements. The system performance of a PID controller can be measured using the performance index shown by [13]. By using this technique, the parameters of a PID controller can be adjusted to meet the required specifications to meet the optimum design requirement of the system. The performance of the PID controller can be evaluated using four basic parameters including rise time, overshoot, setting time and steady state error. The definitions of these parameters are as the following [12]:

1) Rise time (t_r): The time taken to rise beyond 90% of the reference for the first time.
2) Overshoot/undershoot (δ%): The difference between the peak value and the steady state value.
3) Settling time (t_s): The time taken for the output voltage to reach the specified accuracy.
4) Steady-state error (e_{ss}): The difference between the steady-state output and the reference voltage.

The study of a conventional PID controller has been done in 2009 by Xiaodong et al. [14]. According to their findings, the first derivative represents the change of speed of the error while the second derivative represents the acceleration of the error. By restraining the acceleration of the error from getting bigger, a second derivative is added to get a faster system response, a lower overshoot and an increment in system stability. This in turn, gives the controller a better control of the parameters. Nevertheless, increasing the parameters to higher order derivatives could lead to other problems, such as increased time, increased complexity in parameters setting and amplified noise interference [10].

B. *Particle Swarm Optimization (PSO)*

PSO is a robust stochastic optimization technique based on the movement and cooperation of swarms. PSO is an evolutionary algorithm stimulated by the social behavior of birds in a flock. The PSO algorithm was first introduced in 1995 by Kennedy and Eberhart, and was further expanded in 1997 [12].

The main theory behind PSO algorithm is similar to how birds are able to prey for food in a limited area. While a flock of birds are searching for food from one place to another, there is always one bird that can scent the food source better than the rest. While observing the food source, the birds transmit useful information to each other, and because of this, the birds will eventually flock to the place where food can be found. By taking every bird as a particle, this evolutionary algorithm is named particle swarm optimization. Each particle keeps track of its own parameters, with the most important parameter being its current position (in an *n* dimensional vector). Another parameter of importance is the particle's current velocity which keeps track of the current speed and direction of travel by the particles. Each particle has a current best solution fitness value which is obtained by evaluating the error function of the particle's current position. This value is referred to as personal best, 'p_{best}'. Another best solution of fitness value found by any particle in the community is called global best, 'g_{best}'. Each particle tries to alter its position using the information such as the current positions, the current velocities, the distance between the current position and 'p_{best}', the distance between the current position and the 'g_{best}' [13]. In each iteration, every particle updates its own velocity (speed) and position by tracking the local optimum and the global optimum. The position vector of a particle with respect to the origin of the search space represents a trail solution of the search problem. In the beginning, a population of particles is initialized with random positions marked by the vectors x_i and random velocities v_i [2]. The equations are presented for the i^{th} dimension of the position, $x_{i,m}^{(t+1)}$, velocity of the i^{th} particle, $v_{i,m}^{(t+1)}$, and the weighting function, *w*:

$$v_{i,m}^{(t+1)} = wv_{i,m}^{(t)} + c_1 \times rand() \times \left(pbest_{i,m} - x_{i,m}^{(t)}\right) + c_2 \times rand() \times \left(gbest_m - x_{i,m}^{(t)}\right) \quad (3)$$

$$x_{i,m}^{(t+1)} = x_{i,m}^{(t)} + v_{i,m}^{(t+1)} \quad (4)$$

$$w = w_{max} - \frac{(w_{max}-w_{min})\times iter}{iter_{max}} \quad (5)$$

Parameters c_1 and c_2 are two positive constants. The function *rand()* is a random function between 0 and 1, while *m* represents the iteration number. Equation (3) is used to calculate the particle's new velocity from its own best experience (position) and the group's best experience according to its previous velocity and the distances of its current position. The particle will then update its new position according to equation (4). Equation (5) determines the inertia weight to balance between the global search and local search capability by weighing the contribution of the previous velocity. When the inertia weight decreases from 0.9 to 0.4, the search will be narrowed down from a large area to a small area. The inertia weight is limited from 0.9 to 0.4 according to linear decrement which forces the search to start with a bigger area and locate the position with the most optimal solution. The speed of the particle will slow down as *w* is decreasing [15]. The performance of each particle is calculated according to a pre-defined fitness function.

Some of the advantages of the PSO algorithm are:

1) Less complexity in terms of the number of parameters to accommodate.
2) Efficient memory capability where every particle remembers its own previous best value as well as the neighborhood best.
3) Fast convergence since only the best particle can transmit information to other particles.

III. Methods

In this paper, a PID controller using the PSO algorithm is developed to improve the performance of a DC motor system. The proposed method is referred to as the PSO-PID controller for the rest of the paper. The PSO algorithm is mainly utilized to determine three optimal PID controller parameters K_p, K_i, and K_d to obtain excellent step response output for the control system. Fig. 2 shows the flow diagram for the methods adopted in this study. The PSO algorithm is developed and integrated with the PID controller and DC motor. Next, the effects of the inertia weights and iteration number is analysed before comparing the performance of a DC motor with PID-PSO, DC motor with PID with a gain of 1, and DC motor without any PID controller. The simulation is concluded and the results are discussed.

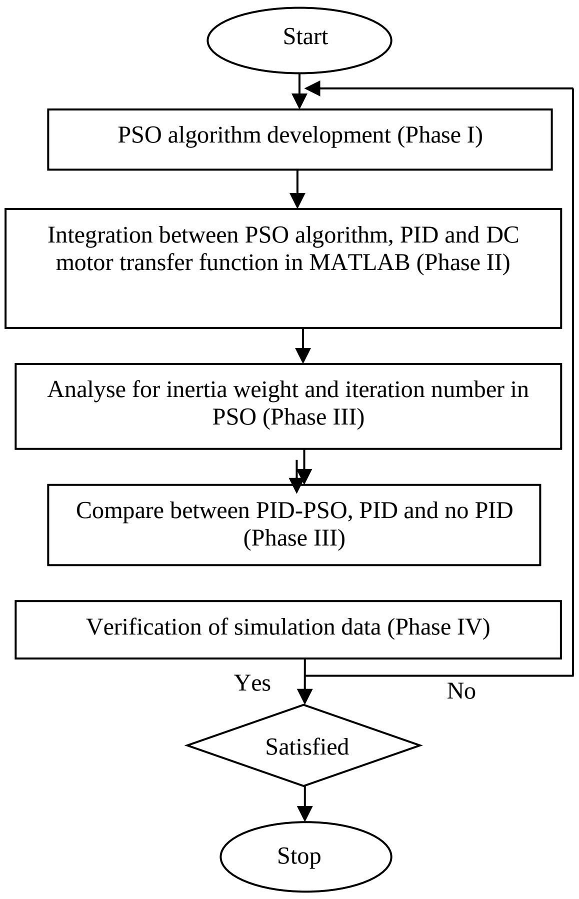
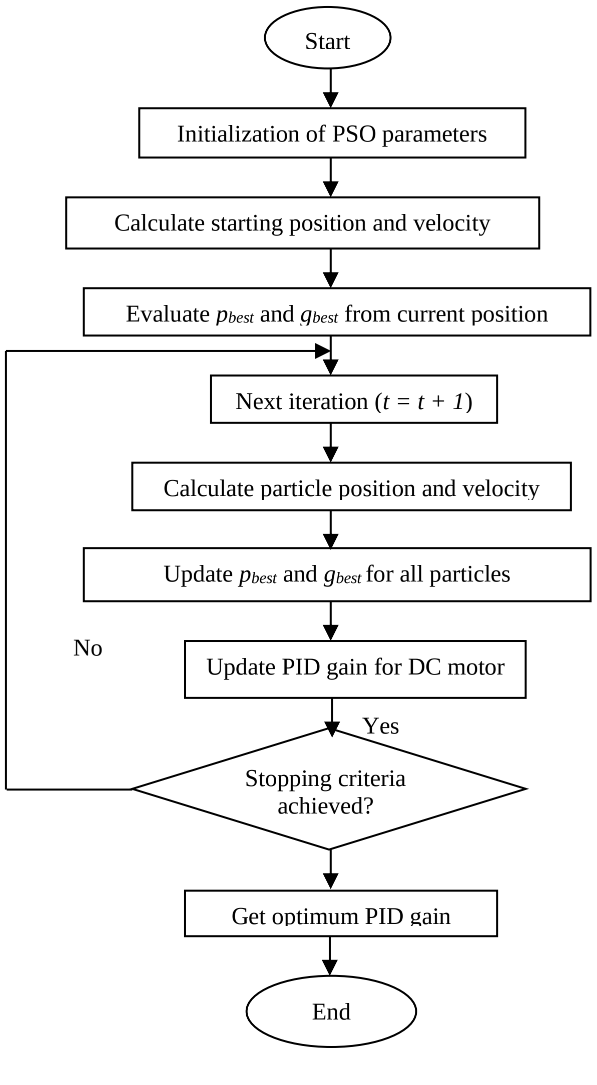

Start
PSO algorithm development (Phase I)
Integration between PSO algorithm, PID and DC motor transfer function in MATLAB (Phase II)
Analyse for inertia weight and iteration number in PSO (Phase III)
Compare between PID-PSO, PID and no PID (Phase III)
Verification of simulation data (Phase IV)
Yes
No
Satisfied
Stop

Fig. 2. Research methods flow diagram

A. PID-PSO controller

PSO algorithm is known for its fast and stable convergence rate, compared to its counterpart such as GA and ACO. It is a promising tool for parameter tuning in control systems such as the PID controllers.

The process flow for the implementation of the PID-PSO is shown in Fig. 3, while Fig. 4 illustrates the design of the PID-PSO controller. The process of designing a good PID controller relies on finding the optimum tuning parameters for the PID controller. At the early stage of the simulation, the PSO parameters are initialized; for example, the number of particles are assigned such that, n=54, c_1=c_2=2, w=0.9, and with number of iteration=100. With this, a group of artificial birds is initialized with arbitrary positions, x_i, and velocities, v_i. At the early searching stage, each bird in the swarm is scattered randomly throughout the D dimensional search space. During the optimization search, each particle remembers its best position attained so far, p_{best}, and also obtains the global best position information achieved by any particle in the population, g_{best}. After the K_p, K_i and K_d parameters are obtained and fed into the PID controller, it will then update the gain for the DC motor. The feedback from the DC motor will then be used to calculate the error to be fed into the PID controller.



Start
Initialization of PSO parameters
Calculate starting position and velocity
Evaluate p_{best} and g_{best} from current position
Next iteration (t = t + 1)
Calculate particle position and velocity
Update p_{best} and g_{best} for all particles
No
Update PID gain for DC motor
Yes
Stopping criteria achieved?
Get optimum PID gain
End

Fig. 3. Process flow of the PID-PSO controller

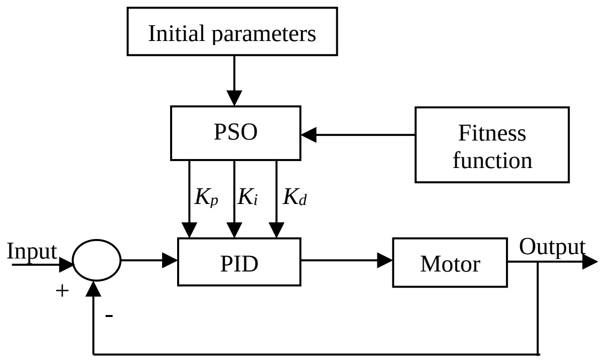


Fig. 4. Design of the PID-PSO controller

TABLE I
PARAMETERS FOR ANALYSIS OF INERTIA WEIGHT

Parameter	Value
Number of particles, n	54
Number of iteration, i	60
c_1	2
c_2	2
Inertia weight, w	0.4 – 0.9
Initial velocity, v	0
PID transfer function	$G_{PID} = \frac{K_d s^2 + K_p s + K_i}{S}$
DC motor transfer function	$G_{mdc} = \frac{1}{s^2 + 2s + 3}$

B. Fitness Function

The overall performance for the convergence speed, efficiency and PSO optimization algorithm accuracy depends on the fitness function to control the searching of the optimal parameters. For this study, the fitness function is measured according to equation (6).

$$F = min\ [(a) + min(c) + min(b) + min(d)] \quad (6)$$

where a = rise time, b = settling time, c = overshoot, d = undershoot.

IV. RESULTS AND DISCUSSION

In this paper, a PID controller utilizing the PSO algorithm is developed to improve the performance of a DC motor system. The analysis is divided into four parts; analysis on the effect of the inertia weight, analysis on the effect of the number of iteration, a comparison between a DC motor with PID-PSO, DC motor with PID of gain 1, and DC motor without PID controller, and lastly the comparison between PID-PSO and PI-PSO.

A. Analysis 1: Analysis of inertia weight

The 2nd order transfer function of a DC motor is considered. A PSO based PID controller for tuning the PID parameters is proposed with the design shown in Fig. 4. The parameters of the PSO algorithm are shown in Table I.

The final optimized PID parameters with various *w* values are presented in Fig. 5. The graph shows the three best values among a series of multiple trials. These values are individually evaluated using the process model in Fig. 3. Fig. 6 shows the performance of the PID controller when the inertia weight is varied from 0.9 to 0.4 with number of iteration 60.

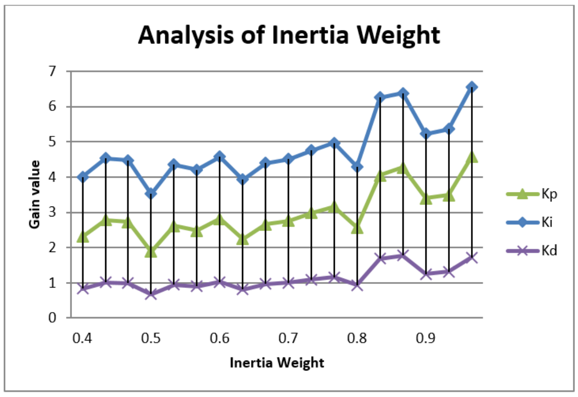


Fig. 5. Optimized PID parameter analysis on inertia weight

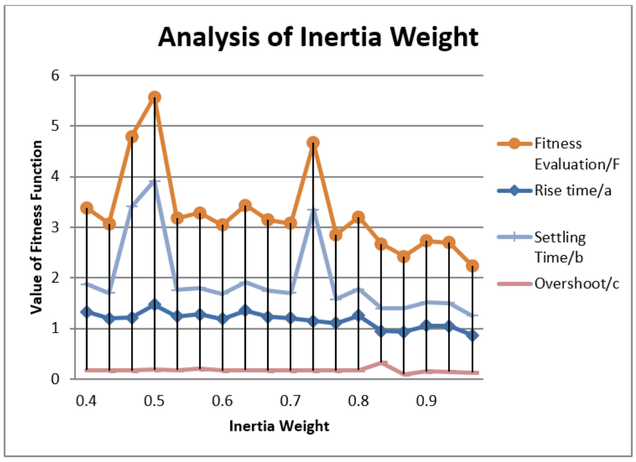


Fig. 6. Process performance with various inertia weights

From Fig. 6, for *w*=0.9, it is observed that even though the total iteration taken by the best three values is considerably large, it provides better performance compared to other values considered in this study. From Fig. 5, the PID gain for *w*=0.9 is not stable and it is due to the low iteration number used for this analysis (*i*=60). Thus, in the next analysis, the iteration number is increased to 100 with *w*=0.9.

B. Analysis 2: Analysis on number of iteration

The parameters used for the second analysis is tabulated in Table II. The final optimized PID parameters with various number of iteration are shown in Fig. 7. This graph presents three best values among a series of multiple trials. These values are individually evaluated using the process model in Fig. 3. Fig. 8 shows the performance of the PID controller when the number of iteration is varied from 30 to 100 with inertia weight of 0.9.

TABLE II
PARAMETERS FOR ANALYSIS OF NUMBER OF ITERATION

Parameter	Value
Number of particles, *n*	54
Number of iteration, *i*	30-100
c_1	2
c_2	2
Inertia weight, *w*	0.9
Initial velocity, *v*	0
PID transfer function	$G_{PID}=\dfrac{K_d s^2+K_p s+K_i}{S}$
DC motor transfer function	$G_{mdc}=\dfrac{1}{s^2+2s+3}$

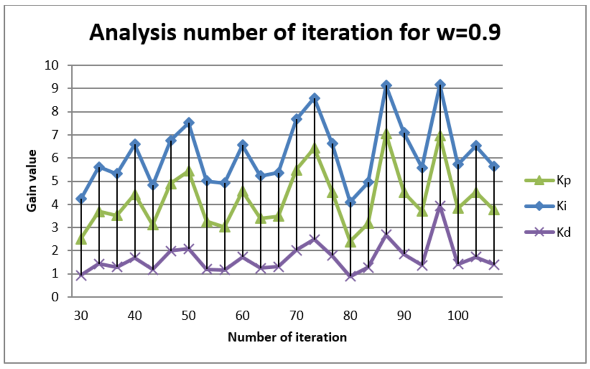


Fig. 7. Optimized PID parameter on analysis number of iteration

From Fig. 7, for *w*=0.9 and number of iteration 100, it is observed that the PID gain value is more stable compared to other iteration numbers. The average performance of the fitness function is also much lower as shown in Fig. 8.

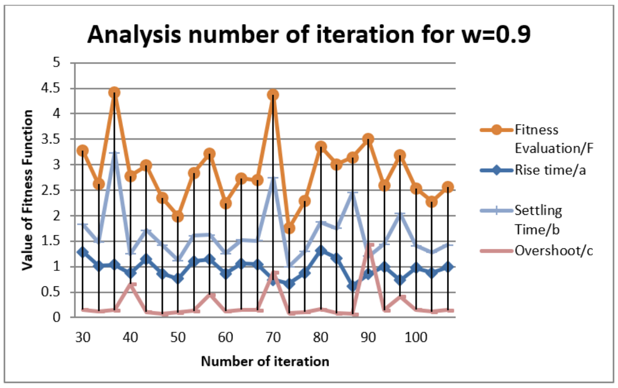


Fig. 8. Process performances with various number of iteration

From the second analysis it is shown that in order to get a stable PID gain with a good fitness function performance, the optimized parameters for the PSO are *n*=54, c_1=c_2=2, *w*=0.9 and *i*=100.

TABLE III
PERFORMANCE OF DC MOTOR WITH PID-PSO, PID OF GAIN 1, AND WITHOUT PID

Parameter	PID-PSO	PID with gain 1	Without PID
K_p	3.8381	6.3735	N/A
K_i	5.7297	7.3835	N/A
K_d	1.423	1.3467	N/A
Rise time (s)	0.977	0.622	0.82
Settling time (s)	1.41	3.09	4.04
Overshoot (%)	0.153	6.53	16.3
Fitness function	2.54	10.24	21.16

C. *Analysis 3: Comparison of DC motor with PID-PSO, PID with gain 1 and without PID*

The fitness function performance of the DC motor system with PID-PSO, with PID with gain of 1, and without PID is shown in Fig. 9. The figure shows that there is a slight difference in rise time between PID-PSO and PID without PSO but the overshoot is obviously reduced from 6.53 to 0.153 with the implementation of PSO. Table III summarizes the output for PID gain and system performance with and without PSO and PID.

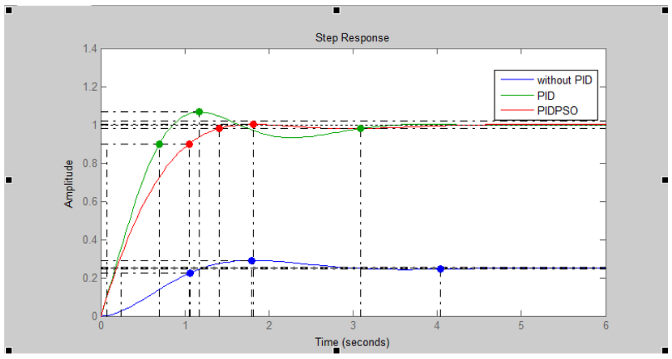


Fig. 9. Performance of fitness function

D. *Analysis 4: Comparison between PID-PSO and PI-PSO*

In order to emphasize the advantage of the proposed PID-PSO controller, the analysis has been done to the transfer function of the DC motor implemented by Kanojiya et al. in 2012 [16], $G_1(s) = \frac{0.015}{0.01s^2+0.14s+0.40015}$, to the PI-PSO controller, PI-ZN and PI-MZN. The performance of the rise time, settling time and overshoot output is shown in Fig. 10. It can be seen that there is a slight difference for all parameter performances between PI-PSO and PID-PSO but by using Ziegler-Nichols method, the overshoot is obviously high compared to the PSO method. Table IV summarizes the outputs for PI-ZN, PI-MZN, PI-PSO and PID-PSO.

TABLE IV
PERFORMANCE CRITERIA FOR PI-ZN, PI-MZN, PI-PSO AND PID-PSO

Author	Algorithm	PID parameters			PID performance		
		K_p	K_i	K_d	Rise time (s)	Sett. time (s)	Overshoot (%)
Kanojiya et al. [10]	PI-ZN	N/A	N/A	N/A	0.3120	2.27	27.9
	PI-MZN	N/A	N/A	N/A	0.0740	0.439	14.5
	PI-PSO	N/A	N/A	N/A	0.3907	0.6467	0.042
This study	PID-PSO	43.4703	132.4802	0.1083	0.2806	0.4326	0.0050

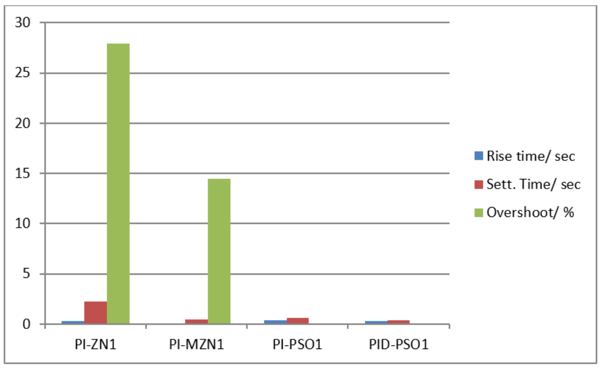


Fig. 10. Comparison performance for PI-ZN, PI-MZN, PI-PSO and PID-PSO

CONCLUSION

This paper proposes a predictive PID controller for a DC motor using PSO algorithm for optimizing the PID parameters. Extensive simulations have been done to study the effect of inertia weights and number of iterations for the PSO parameters. From the conducted simulations, it is found that the optimized value for the PSO parameters are w=0.9 and number of iteration=100 with number of particles 54. A comparison between the performance of the DC motor with PID-PSO, with PID of gain 1, and without PID were also discussed. From the analysis, it can be concluded that by tuning the PID parameters using the PSO method, the best gain in performance may be found. Finally, when comparing between the PID-PSO and its counterpart, the PI-PSO, the PID-PSO controller gives better performance in terms of robustness, low overshoot, low minimum rise time and low settling time.

REFERENCES

[1] S. J. Bassi, M. K. Mishra, & E. E. Omizegba, “Automatic Tuning Of Proportional-Integral-Derivative (PID) Controller Using Particle Swarm Optimization (PSO) Algorithm,” *International Journal of Artificial Intelligence & Applications*, vol. 2(4), 2011, pp. 25–34.

[2] S. Malik, P. Dutta, S. Chakrabarti & A. Barman, "Parameter Estimation of a PID Controller using Particle Swarm Optimization Algorithm," *Internation Journal of Advanced Researh in Computer and Communication Engineering*, vol. 3(3), 2014, pp.5827–5830.

[3] M. I. Solihin, L. F. Tack, & M. L. Kean, "Tuning of PID Controller Using Particle Swarm Optimization (PSO)," in *Proc. of the International Conference on Advanced Science, Engineering and Information Technology*, Bangi, Malaysia, 2011, pp. 458-461.

[4] R. J. Rajesh & C. M. Ananda, "PSO tuned PID controller for controlling camera position in UAV using 2-axis gimbal," in *Proc. 2015 International Conference on Power and Advanced Control Engineering (ICPACE)*, Bangalore, 2015, pp. 128-133.

[5] P. Biswas, R. Maiti, A. Kolay, K. Das Sharma & G. Sarkar, "PSO based PID controller design for twin rotor MIMO system," in *Proc. of The 2014 International Conference on Control, Instrumentation, Energy and Communication (CIEC)*, Calcutta, 2014, pp. 56-60.

[6] E. Sahin, M. S. Ayas & I. H. Altas, "A PSO optimized fractional-order PID controller for a PV system with DC-DC boost converter," in *Proc. 2014 16th International Power Electronics and Motion Control Conference and Exposition*, Antalya, 2014, pp. 477-481.

[7] P. Kaur, V. Kumar & R. Sharma, "Speed control of hybrid electric vehicle using PSO based fractional order PID controller," in *Proc. 2016 1st India International Conference on Information Processing (IICIP)*, Delhi, 2016, pp. 1-6.

[8] H. I. Jaafar, Z. Mohamed, N. A. M. Subha, A. R. Husain, F. S. Ismail, L. Ramli, M. O. Tokhi & M. A. Shamsudin, "Efficient control of a nonlinear double-pendulum overhead crane with sensorless payload motion using an improved PSO-tuned PID controller," *Journal of Vibration and Control*, 2019, vol. 25(4), pp. 907-921.

[9] M. A. A. Aziz, M. N. Taib & R. Adnan, "The functionality validation of PSO algorithm in optimizing PID controller's parameters using multi-dimensional test functions," in *Proc. 2016 7th IEEE Control and System Graduate Research Colloquium (ICSGRC)*, Shah Alam, 2016, pp. 203-208.

[10] A. H. Mary, "Generalized PID Controller Based On Particle Swarm Optimization," *Iraqi Journal of Computers, Communication and Constrol & Systems Engineering*, 2011, vol. 11, pp.114–122.

[11] S. Karthikeyan, P. Rameshbabu & B. J. Robi, "Design of Intelligent PID Controller Based on Particle Swarm Optimization in FPGA," *International Journal of Power Control Signal and Computation (IJPCSC)*, 2012, vol. 4(2), pp. 66-70.

[12] X. Li, M. Chen, & Y. Tsutomu, "A Method of Searching PID Controller's Optimized Coefficients for Buck Converter Using Particle Swarm Optimization," in *Proc. of the IEEE 10th International Conference on Power Electronics and Drive Systems (PEDS)*, 2013, pp. 22–25.

[13] M. W. Iruthayarajan, & S. Baskar, "Optimization of PID Parameters Using Genetic Algorithm and Particle Swarm Optimization," in *Proc. of the International Conference on Information and Communication Technology in Electrical Sciences (ICTES 2007)*, 2007, pp. 81–86.

[14] Z. Xiaodong, L. Yongqiang & X. Anke, "The Study of the Generalized PID Algorithm," in *Proc. of the International Forum on Computer Science-Technology and Applications*, Chongqing, 2009, pp. 255-258.

[15] Q. Bai, "Analysis of Particle Swarm Optimization Algorithm," *Computer and Information Science*, 1998, vol. 3(1), pp. 180–84.

[16] R. G. Kanojiya & P. M. Meshram, "Optimal tuning of PI controller for speed control of DC motor drive using particle swarm optimization," in *Proc. of the International Conference on Advances in Power Conversion and Energy Technologies (APCET)*, 2012, Mylavaram, Andhra Pradesh, pp. 1-6.