A new Displacement based PSO algorithm Optimised PI Controller for Speed Control of a DC Motor

N Ratana Devi* and Robert L Biate

Abstract— Manufacturing, plant processing, industrial automation, mills power servos, robotics and many more field of industries and automobiles all are possible because of drive systems run by the DC motors. Besides its salient features, in a motor control system, hundreds of problems are faced such as change in load dynamics, speed control or noise parameters. In our research work we have designed a new Displacement based Particle Swarm Optimisation algorithm (DPSO) to control the speed of a DC motor. Our proposed DPSO consider the previous and current velocities of the particles to evaluate the next position of the particles. We have used DPSO to optimise the PI controller parameters. Along with analysis of DPSO, we have done a comparative test with Standard PSO and accelerated PSO optimised PI controller. Design and simulation of controllers are done in LabVIEW VI.

Index Terms— Optimisation, Soft Computing, PSO, APSO, DPSO and LabView

1 INTRODUCTION

Motor drives have been in use for long time and its an efficient way of transferring mechanical energy into desirable output in industries. Although there are two types of motor drives currently being used in every industry but DC motors can be considered much better than AC motors especially when considering transportation equipment because of their maximum torque producing quality at stalls which is very poor in AC motors. Also energy recovery mechanism observed in DC motors is much better than in AC motors [1]. Moreover, dc motors provide a low horsepower rating at a much cheaper rate than AC drives [2]. To achieve maximum productivity, every single thing of a machine should be taken into account and analysed accordingly. In motor control systems, hundreds of problems are faced such as change in load dynamics. The most important affecting factor will be noise parameter because its various and unpredictable affecting the functioning of the machine [3]. Similarly, another main factor is speed which should be monitored constantly according to the requirement for a desirable and reliable output. A DC motor as the name indicates is a motor initiated usually by direct current and is converted into mechanical energy according to the requirement. DC motors are ruling the world due to their extensive use in modern technologies and in almost every industry such as to operate steel rolling mills, electric screw drivers, sewing machines, hard disk drives, air compressors, reciprocating machine etc. [4]. In these applications, the motor should be precisely controlled to give the desired performance. The controllers of the speed that are conceived for goal to control the speed of DC motor to execute one variety of tasks, is of several conventional and numeric controller types, the controllers can be: proportional integral (PI), proportional integral derivative (PID) and Fuzzy Logic Controller (FLC) or the combination between them: Fuzzy Neural Networks, Fuzzy-Genetic Algorithm, Fuzzy-Ants Colony, Fuzzy-Swarm [5]. The proportional – integral – derivative (PID) controller operates the majority of the control system in the world. It has been reported that more than 95% of the controllers in the industrial process control applications are of PID type as no other controller match the simplicity, clear functionality, applicability and ease of use offered by the PID controller [8], [9]. PID controller provides robust and reliable performance for most systems if the PID parameters are tuned properly.

The major problems in applying a conventional control algorithm (PI, PD, PID) in a speed controller are the effects of nonlinearity in a DC motor. The nonlinear characteristics of a DC motor such as saturation and friction could degrade the performance of conventional controllers [6], [7]. Generally, an accurate nonlinear model of an actual DC motor is difficult to find and parameter obtained from systems identification may be only approximated values. Introduction of soft computing for system optimisation, results in better system performance. Particle Swarm Optimisation algorithm and Fuzzy Logic are one of those soft computing method. Many researchers have worked on PSO algorithm to control the system and also developed many of its variants. Fuzzy logic becomes the latest trend in control system because of its easy implementation and linguistic rule makings.

Vikrant Vishal *et al.*, have done a comparative study of optimisation methods: GA, Accelerated Particle Swamp

- *N Ratana Devi is currently pursuing master degree program in Electrical engineering at National Institute of Technology (NIT), Manipur, India, E-mail: ratanade482@gmail.com*
- *Robert L Biate is currently Teaching Assistant , Department of Electrical Engineering at National Institute of technology(NIT),Manipur, India, E-mail: robhmar@nitmanipur.ac.in*

Optimisation (APSO), Differential Evolution (DE) and Cuckoo Search (CS). The authors have applied these methods to optimize the PID gain parameters for speed control of DC motor by minimising Integral Time Absolute Error. Their research works show that performance of CS tuned controller is superior to other studied algorithms. [10]

Adel A. A. El-Gammal and Adel A. El-Samahy, have used PSO technique to set the gains of a PID speed controller to minimise the rise time, settling time, overshoot and steady state error by combining these multiple objectives into a single objective function. The single optimized value compromise between all multiple objectives. Their research has a drawback it cannot be used for systems with multiple conflicting objective function. [11]

Luis Brito Palma *et al.* their proposed control applies an on-line optimisation approach based on serialization of the parallel PSO optimisation scheme, using a cost function based on the Harris index and on the control error. The proposed control scheme is suitable for auto-tuning and automatic controller design. [15]

In our research work we have proposed a new variant of PSO i.e. Displacement based PSO. Unlike standard PSO it doesn't determine the next position of the particle using only the velocity of particle instead DPSO evaluates the displacement of the particles using the third equation of motion $2as = (v^2 - u^2)$, where 'a' is the acceleration, 's' is the displacement, 'v' and 'u' are the final and initial velocities, respectively. PI controller parameters 'K_p' and 'K_i' are optimized using DPSO. In this paper, we have analysed our proposed DPSO for Separately Excited DC motor speed control using LabVIEW VI and compared with other PSO variants.

.

2 Mathematical Modelling of Separately Excited DC Motor (SEDCM)

Our research work aims at controlling the speed of a separately excited DC motor rather than self-excited DC motor. We need variable speed drives in our everyday industries such as automotive, petrochemical, food and beverage etc. However, position control of machine drive is also important but once a position is adjusted by some mechanism then its need not to be changed accordingly again and again. However, speed of an object needs to be changed as required such as of motor used in blender. Sometimes it is required to blend the mixture at high speed and sometimes at medium or low speed. Therefore, a technique should be devised for variable speed control rather than variable position control. The major reason of working on separately excited DC motor is that initiation of the motor is independent of internal circuitry of the machine. This gives us an advantage of generating output as desired by varying input supplied voltage with accurate and better speed control as compared to self-excited DC motors.

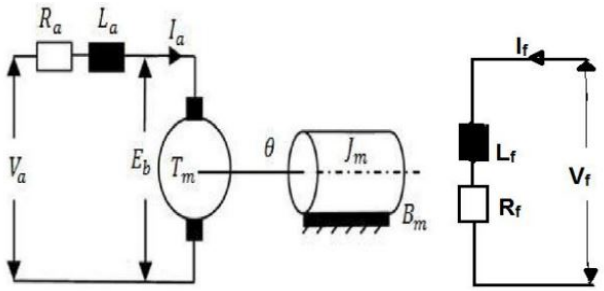

Figure: 1 SEDCM

Where,

V_a	:	the armature voltage. (In volt)
E_b	:	back emf of the motor (In volt)
I_a	:	the armature current (In ampere)
I_f	:	the armature current (In ampere)
R_a	:	the armature resistance (In ohm)
L_a	:	the armature inductance (In Henry)
T_m	:	the mechanical torque developed (In Nm)
J_m	:	moment of inertia (In kg/m²)
B_m	:	friction coefficient (In Nm/(rad/sec))
θ	:	angular displacement (rad)
ω	:	angular velocity (In rad/sec)

We know that

$$\omega \propto (V_a - I_a R_a)/\varphi$$

$$\Rightarrow \quad \omega = (V_a - I_a R_a)/K_a\varphi \qquad (2.1)$$

Where φ = Field flux per pole

Ka= Armature constant = PZ/2πa

P = No. of pole

Z = Total no. of armature conductor

a = No. of parallel path

From the equation (2.1) it is clear that for DC motor there are basically 3 methods to control the speed.

They are: -

1- Variation of resistance in armature circuit.

2- Variation of field flux.

3- Variation of armature terminal voltage.

In servo applications, the DC motors are generally used in the linear range of magnetization curve. Therefore, the air gap flux φ is proportional of field current, i.e.

$$\varphi = K_f I_f \qquad (2.2)$$

Where K_f is constant.

The torque T_m of the motor is proportional to the product of air gap flux and armature current, i.e.

$$T_m = K_1 K_f I_f I_a \qquad (2.3)$$

Where K_1 is constant.

In armature controlled dc motor, the field current is kept constant, so

$$T_m = K_T I_a \quad (2.4)$$

Where K_T is known as the motor torque constant. The motor back *emf* being proportional to speed is given

$$E_b = K_b \frac{d\theta}{dt} \quad (2.5)$$

Where K_b is the back *emf* constant.

The differential equation of the armature circuit is

$$L\frac{dI_a}{dt} + RI_a + E_b - E_a = 0 \quad (2.6)$$

The torque equation is

$$J\frac{d^2\theta}{dt^2} + B\frac{d\theta}{dt} = T_m = K_T I_a \quad (2.7)$$

Taking the Laplace transform of equations, assuming initial conditions as zero.

$$E_b(s) = K_b s\theta(s)$$

$$(Ls + R)I_a(s) = E_a(s) - E_b(s)$$

$$(Js^2 + Bs)\theta(s) = T_m(s) = K_T I_a(s)$$

So, the final transfer function will be

$$\frac{\theta(s)}{E_a(s)} = \frac{K_T}{s[(R + sL)(Js + B) + K_T K_B]}$$

Or

$$G(s) = \frac{\omega(s)}{E_a(s)} = \frac{K_T}{[(R + sL)(Js + B) + K_T K_B]}$$

Block diagram can be realize using equation

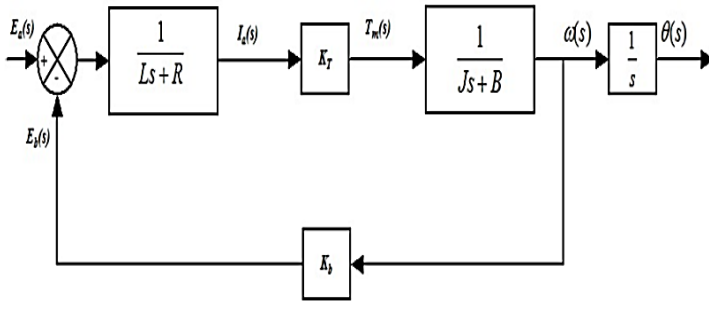

Figure 2. SEDCM Block Diagram

3 Particle Swarm Optimisation

PSO algorithm is swarm intelligence based evolutionary computation method. The individual in swarm is a volume-less particle in multidimensional search space. The position in search space represents potential solutions of optimization problem, and the velocity determines the direction and step of search. The particle flies in search space at definite velocity which is dynamically adjusted according to its own flying experience and flying experience of their companions, i.e., constantly adjusting its approach direction and velocity by tracing the best position found so far by particle themselves and that of the whole swarm, which forms positive feedback of swarm optimization. Particle swarm tracks the two best current positions, moves to better region gradually, and finally arrives to the best position of the whole space search.

Standard PSO (SPSO): Standard PSO has two primary operators: Velocity update and Position update. During each generation every particle are accelerated towards their previous best position and global best position. At each iteration a new velocity value for each particle is calculated based on its current velocity, location from its previous best position, and location from the global best position. The new velocity value is then used to calculate the next position of the particle in the search space. This process is then iterated a set number of times, or until a minimum error is achieved [11] [12]

Suppose in PSO, a swarm consists N number of particles moving around in a D dimensional search space, the i^{th} particle denote as

$$X_i = (x_{i1}, x_{i2}, x_{i3} \ldots\ldots x_{iD})$$

Whose previous best solution P_{best} is presents as

$$P_i = (p_{i1}, p_{i2}, p_{i3} \ldots\ldots p_{iD})$$

And the current velocity is described by

$$V_i = (v_{i1}, v_{i2}, v_{i3} \ldots\ldots v_{iD})$$

Finally, the best solution of whole swarm G_{best} also called global best is represented as

$$P_g = (p_{g1}, p_{g2}, p_{g3} \ldots\ldots p_{gD})$$

At each time step, each particle moves towards P_{best} and G_{best} locations. The fitness function evaluates the performance of the particles to determine whether the best fitting solution is achieved or not.

The detailed operation of PSO is given below:

STEP 1: Initialization Randomly The positions and velocities of the particles are set within a pre-defined range.

STEP 2: Fitness Function The fitness of each particle of the swarm is evaluated.

STEP 3: Update Velocity After every iteration, the particles velocities are updated according to the following equation

$$v_{id} = v_{id} + c_1 * rand * (p_{id} - x_{id}) + c_2 * rand * (p_{gd} - x_{id})$$

Where c_1 and c_2 are two positive constants, called cognitive learning rate and social learning rate respectively; *rand* is a random function in the range [0, 1]. The velocity of the particle is limited in $[V_{min}, V_{max}]$. Since the original formula of PSO lacks velocity control mechanism, it has a poor ability to search at a fine grain. In 1998, to overcome this limitation Elberhart and Shi designed a time decreasing inertia factor. In PSO, a parameter called inertia weight is introduced in the original equation for balancing the global and local search. Then, the velocity equation will be as follows:

$$v_{id} = w * v_{id} + c_1 * rand * (p_{id} - x_{id}) + c_2 * rand * (p_{gd} - x_{id}) \quad (3.1)$$

Where w is inertia factor which balances the global wide-range exploitation and the local nearby exploration abilities of the swarm.

STEP 4: Update Position After every iteration the positions of the particles are updated according to the following equation

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (3.2)$$

STEP 5: Update Memory Best solution P_{best} and global best solution G_{best} values are updated, according to the given conditions

$$P_{best} = p_i \text{ if } f(p_i) > f(P_{best})$$
$$G_{best} = p_g \text{ if } f(p_g) > f(G_{best})$$

Where *f(x)* is the objective functions subjective to minimisation.

STEP 6: Stop Iteration From step 2 to step 5, the algorithm will repeat itself until or unless certain stop conditions are met, such as a pre-defined number of iterations.

Once terminated, the algorithm reports the values of P_{best} and G_{best}.

Accelerated PSO (APSO): A simplified version which could accelerate the convergence of the algorithm is to use the global best only. Thus, in the accelerated particle swarm optimization (APSO) [13], [14], the velocity vector is generated by a simpler formula

$$v_{id} = w * v_{id} + c_1 * rand + c_2 * rand * (p_{gd} - x_{id}) \quad (3.3)$$

To increase the convergence, we can directly update the positions on the particles using the following equation:

$$x_{id}(t+1) = (1-\beta)x_{id}(t) + \beta * p_{gd} + \alpha\epsilon_n \quad (3.4)$$

This simple version will give the same order of convergence. Typically, α = 0.1L ~ 0.5L where L is the scale of each variable, while β = 0.1 ~ 0.7 is sufficient for most applications. It is worth pointing out that velocity does not appear in equation (3.4), and there is no need to deal with initialization of velocity vectors. Therefore, APSO is much simpler. Comparing with many PSO variants, APSO uses only two parameters, and the mechanism is simple to understand.

Proposed Displacement PSO (DPSO): The standard PSO uses the velocity of the particle for their position update. In our proposed displacement based PSO we are using the third equation of motion for our position update:

$$2as = v^2 - u^2$$

Where, *v*= final velocity: *u*=initial velocity; *a*=acceleration; *s*= displacement

Velocity update is same as the Standard PSO equation (3.1). DPSO particle position updating steps are as follows:

1. Update the velocity of each particle using equation:

$$v_{id} = w * v_{id} + c_1 * rand * (p_{id} - x_{id}) + c_2 * rand * (p_{gd} - x_{id})$$

2. Calculate acceleration using

$$a = (v-u)/t$$

Where $t \neq 0$ and it value can be freely set. Here, we will analyse the performance by varying it.

3. Calculate displacement of all particles using the third equation of motion

$$s = (v^2 - u^2)/2a$$

Optimisation of PI controller: In our research work Standard PSO, APSO and DPSO are used to optimize the control parameters of PI controller to control the speed of DC motor efficiently. Optimization of PI controller firstly need the design of optimization goal and then the optimization to be searched. The objective function is to minimise the overshoot. For our research work we are using ISE in our fitness function.

ISE (Integral Square Error), $ISE = \int_0^{\infty} e^2(t)dt$

Objective: ***To eliminate overshoot***

Fitness Function: $\alpha * ISE + \beta * maxOvershoot$

Where α and β are scaling factor (Depends upon the choice of designer)

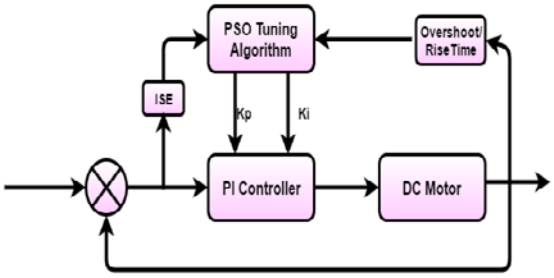

Figure 3. PSO Optimised PI

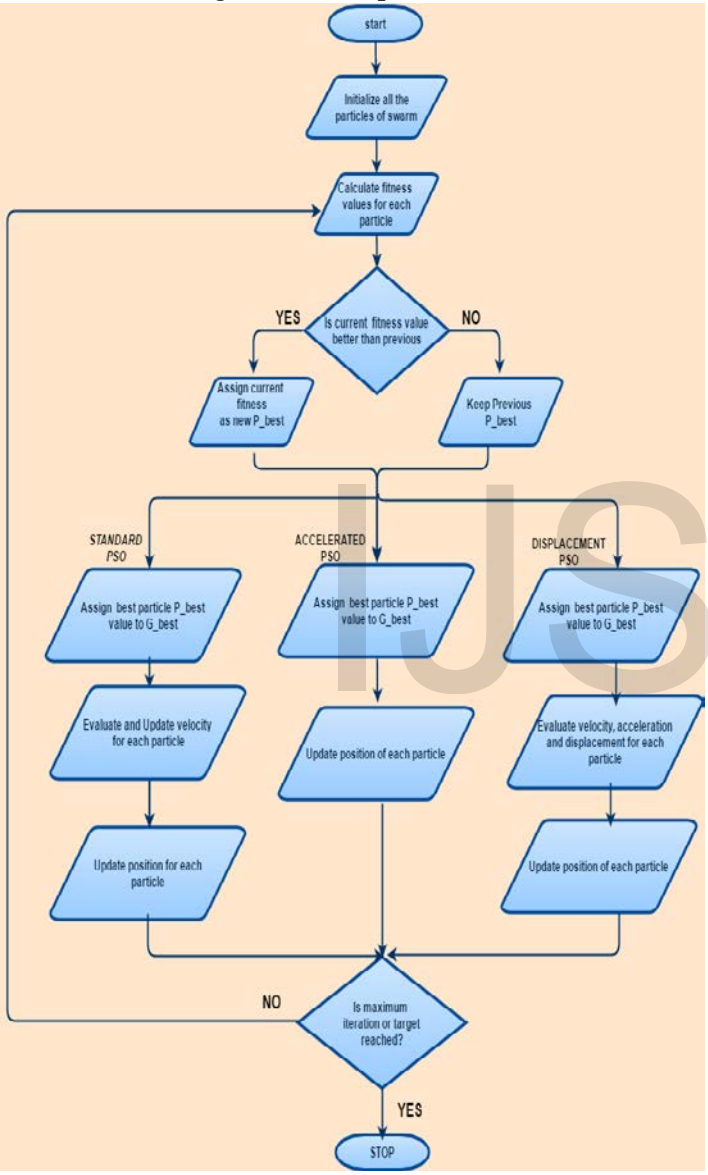

Figure 4: PSO Variants Algorithm

4 Simulation and Results

For our simulation purpose we have used the following parameters of SEDCM

TABLE 1: System Parametrs

Va	240V
R	3.3Ω
L	4.64 μH
K_b	0.0028
K_T	0.0028
B	1.86E6 Nm/rad
J	9.64E6 kg-m^2

We have designed SPSO, APSO and proposed DPSO using the MATHScript of LabVIEW software and the simulation is done on LabVIEW VI. Figure 5 shows the simulation block.

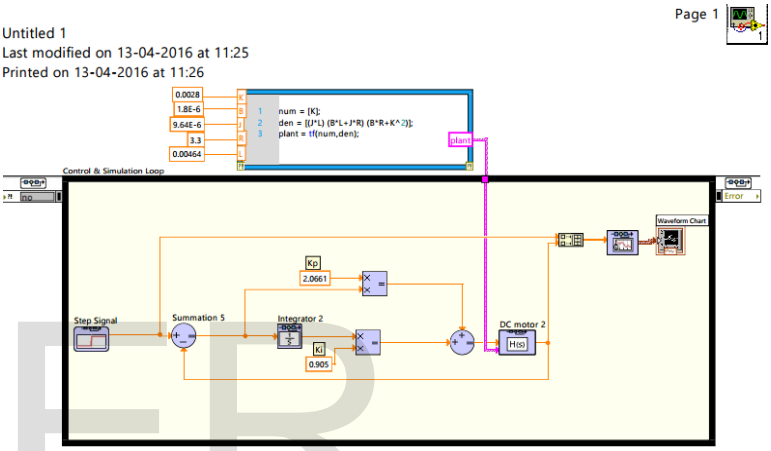

Figure 5: Simulation Block

In my research work, the predefined numerical coefficients of PSO are set as: acceleration constant c1=0.12, c2=1.2, random functions will be in the range, rand () = [0 1] and the inertia weight w=0.9, and population size of swarm n: 100. Algorithm is run for 300 iterations. For DPSO, we are using't' =1, in our research we found that at't'=1 DPSO gives the best optimize value. The fitness function vs Iteration plot is shown in figure 6.

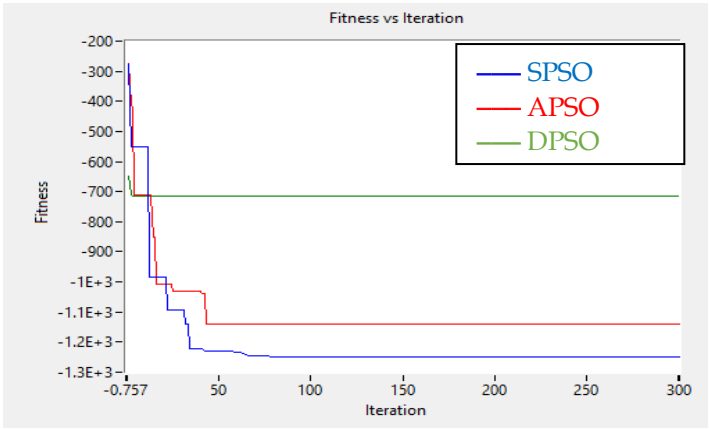

Figure 6: Fitness value VS Iteration

The optimized PI parameters value for different PSOs are given in table 2

TABLE 2: Optimised PI parameters

PSO Variant	Kp	Ki
SPSO	2.2747	1.0716
APSO	2.3283	1.054
DPSO	4.2135	0.1489

Taking unit step as input value we have run our simulation for optimized Kp and Ki value obtained from the PSOs algorithm. The system output to different optimized PI controller parametrs are shown in figure 7.

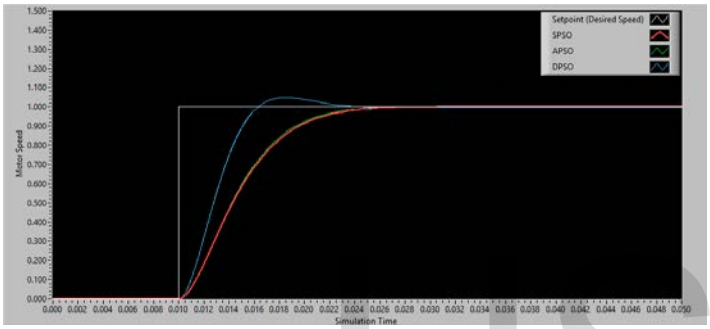

FIGURE 7: SYSTEM OUTPUT

TABLE 3: System Transient Responses

TYPE	Kp	Ki	Overshoot	RiseTime	SettlingTime
SPSO	2.2747	1.0716	0	0.009	0.0014
APSO	2.3283	1.054	0	0.012	0.018
DPSO	4.2135	0.1489	4.3%	0.0052	0.013

5 CONCLUSION AND SCOPE

Our proposed DPSO is based on the displacement of the particle rather than only considering the velocity of the particle like standard PSO. The displacement of the particles is evaluated using the third equation of motion. We have designed and simulated our proposed DPSO in LabVIEW software.

From the table 3, we can conclude that our proposed DPSO has little extra overshoot (4.3%) as compared to other mentioned PSOs (0%), but it gives more effective result by providing lesser rise time and settling time. Even though the convergence of our proposed DPSO is little delaying than SPSO and APSO, but it optimised PI more effectively. Also, it takes less number of iteration to get the best fitness value. For the same fitness function, DPSO proves to be more efficient in optimization.

For future work we can perform online DPSO algorithm optimized PI controller and we can try to increase its convergence by reducing algorithm steps.

ACKNOWLEDGEMENT

I sincerely show my gratiutude and respect to Mr. Robert L. Biate (Teaching Assistant, NIT Manipur) for his kind and helpful assistance in this research work.

7.3 References

[1] Bansal, U.K. and Narvey, R. (2003) Speed Control of DC Motor Using Fuzzy PID Controller. Advance in Electronic and Electric Engineering, 3, 1209-1220.

[2] Afrasiabi, N. and Yazdi, M.H. (2013) DC Motor Control Using Chopper. Global Journal of Science, Engineering and Technology, 8, 67-73.

[3] Atri, A. and Ilyas, Md. (2012) Speed Control of DC Motor Using Neural Network Configuration. International Journal of Advanced Research in Computer Science and Software Engineering, 2, 209-212.

[4] Saleem, F.A. (2013) Dynamic Modeling, Simulation and Control of Electric Machines for Mechatronics Applications. International Journal of Control, Automation and Systems, 1, 30-42.

[5] BomedieneAlloua, AbdellahLaoufBrahimGasbaoui and ABdessalamAbderrahamani, "NeuroFuzzy DC Motor Speed Control Using Particle Swarm Optimization," Leonaro Electronic Journal of Practices and Technologies ISSN,1583-1078.

[6] B.J. Chalmers, "Influence of saturation in brushless permanent magnet drives." IEE proc. B, Electr.Power Appl, vol.139, no.1, 1992.

[7] C.T. Johnson and R.D. Lorenz, "Experimental identification of friction and its compensation in precise, position controlled mechanism." IEEE Trans. Ind, Applicat, vol.28, no.6, 1992.

[8] J. Zhang, N. Wang and S. Wang, "A developed method of tuning PID controllers with fuzzy rules for integrating process," Proceedings of the American Control Conference, Boston, 2004, pp. 1109-1114.

[9] K.H. Ang, G. Chong and Y. Li, "PID control system analysis, design and technology," IEEE transaction on Control System Technology, Vol.13, No.4, 2005, pp. 559-576

[10] V. Vishal et al, "Comparitive study of some optimization technique", Advance Computing Conference (IACC), IEEE International, pp. 1342-1347, doi: 10.1109/IAdCC.2014.6779522, 2014

[11] A. A. A. El-Gammal and A. A. El-Samahy, "A modified design of PID controller for DC motor drives using Particle Swarm Optimization PSO," *2009 International Conference on Power Engineering, Energy and Electrical Drives*, Lisbon, 2009, pp. 419-424. doi: 10.1109/POWERENG.2009.4915157

[12] Riccardo Poli et. Al., "An Overview of Particle Swarm Optimization", Springer Science, 2007

[13] R. Eberhat and J. Kennedy, "A New Optimizer Using Particle Swarm theory," *Proceeding of 6th International*

Symposium on Micro Machine and Human Science, pp. 39-43, Japan, 1995

[14] J. Kennedy and R. Eberhat, "Particle Swarm Optimization," *Proceeding of IEEE International Conference on Neural Network,* Vol. 4, pp. 1942-1948, Perth, W.A. 1995

[15] L. B. Palma, F. V. Coito, B. G. Ferreira and P. S. Gil, "PSO based on-line optimization for DC motor speed control," *2015 9th International Conference on Compatibility and Power Electronics (CPE)*, Costa da Caparica, 2015,pp.301-306. doi: 10.1109/CPE.2015.7231090