Evolutionary Computation Techniques Based Optimal PID Controller Tuning

Sulochana Wadhwani[#1], Veena Verma[*2]

[#]*Department of Electrical Engineering, RGPV*
MITS, Gwalior, Madhya Pradesh-474005, India

[*] *Department of Electrical Engineering, RGPV*
MITS, Gwalior, Madhya Pradesh-474005, India

***Abstract*—The main aim of this paper is to analyse the implementation of two Evolutionary Computation (EC) techniques viz. Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) for optimal tuning of PID controllers parameters and enumerate their advantages over the conventional tuning methodologies. The two techniques were implemented and analysed on a third order plant model of a DC servomotor with the aim of developing a position controller. The results obtained from GA and PSO algorithms were compared with that obtained from Ziegler Nichols method. It was found that the evolutionary computation techniques outperformed traditional tuning practices of Zeigler-Nichols at tuning of PID controllers.**

***Keywords*— PID Tuning, Evolutionary Computation, Evolutionary Algorithm, Genetic Algorithm, Swarm Intelligence, Particle Swarm Optimization**

I. INTRODUCTION

In process control industry, majority of control system loops are based on Proportional-Integral-Derivative (PID) controllers. PID controllers are being widely used in industry due to their well-grounded established theory, simplicity, maintenance requirements, and ease of tuning. The basic structure of the PID controllers makes it easy to regulate the process output. Therefore, efficient design and tuning methods leading to an optimal and effective operation of the PID controllers in order to regulate the different parameters of the plant are economically vital for process industries.

The main aim of this paper is to analyse the implementation of two evolutionary computation techniques viz. Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) for optimal tuning of PID controllers parameters and enumerate their advantages over the conventional tuning methodologies.

Genetic Algorithms (GA) are adaptive heuristic search based on evolutionary ideas of natural selection and genetics. Genetic Algorithms are effective and intelligent choices at finding the best solution among the space of all feasible solutions. Genetic Algorithms were used to evaluate the optimum PID controller gain values where performance indices ITAE, IAE, ISE and MSE were used as the objective functions. It was experimentally determined that the Integral of Absolute Magnitude of the error (IAE) performance criterion produces the most effective PID controller when compared with other performance criterion.

Particle swarm optimization (PSO) is a metaheuristic algorithm based on swarm behaviour observed in nature such as in bird flocking or fish schooling. It attempts to mimic the natural process of group communication of individual knowledge, to achieve some optimum property. PSO searches the space of an objective function by adjusting the trajectories of individual agents, called particles. Each particle traces a piecewise path which can be modelled as a time-dependent position vector.

The proposed methodologies were verified using a third-order physical plant (Armature-controlled DC servomotor position control system) where tuning algorithms were driven mainly by the acquired system data and the desired performance parameters specified by the user are successfully satisfied. Resultant improvements on the step response behaviour of DC servomotor position control system are shown for two cases.

This paper is organized as follows: system modelling of DC servomotor is presented in Section II, brief introduction to EC is discussed in Section III, the basics of PID controller and implementation of GA and PSO to optimize PID parameter is presented in Section IV and V respectively. In Section VI, simulated results obtained for the system considered are shown. At the end, conclusion of the present research work is given in Section VII.

II. SYSTEM MODELLING

As a reference we consider armature controlled DC servomotor as shown in Figure 1. In the point of control system, DC servo motor can be considered as linear SISO plant model having third order transfer function. The DC servomotors are found to have an excellent speed and position control. A simple mathematical relationship between the shaft angular position 'θ' and voltage input 'V_a' to the DC motor may be derived from physical laws.

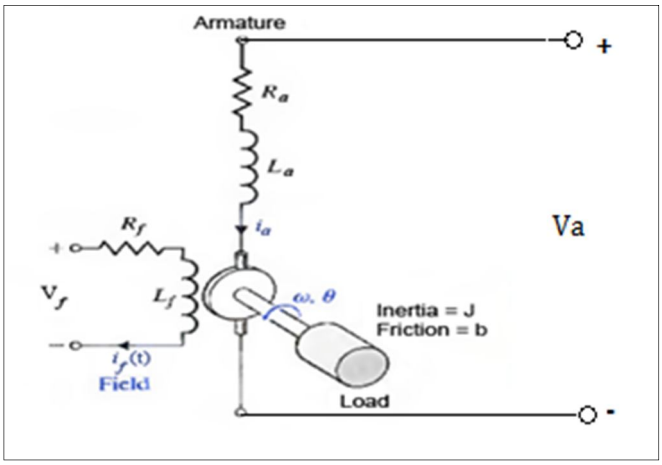

Fig.1 Schematic Diagram of armature controlled DC Servo motor

The dynamic behaviour of the armature current-controlled DC servomotor is given by the following equations [1]:

The air gap flux φ of the motor is proportional to the field current so that

$$\varphi = k_f\, i_f(t) \quad (1)$$

The torque developed by motor is assumed to be related linearly to air gap flux and the armature current as follows

$$T_m = k_1\, \varphi\, i_a(t) \quad (2)$$

$$or \quad T_m = k_1\, k_f\, i_f(t)\, i_a(t),$$

where $k_1 and\ k_f$ are constants.

When a constant field current is established in a field coil, the motor torque is

$$T_m = k_m i_a(t) \quad (3)$$

In Laplace transform notification,

$$T_m(s) = k_m I_a(s) \quad (4)$$

The armature current is related to the input voltage applied to the armature by

$$V_a(s) = R_a\, I_a(s) + L_a s\, I_a(s) + V_b(s) \quad (5)$$

where $V_b(s)$ is back emf voltage proportional to the motor speed. Therefore, we have

$$V_b(s) = k_b \omega(s) \quad (6)$$

where $\omega(s) = s\theta(s)$the transform of the angular speed and the armature is current is

$$I_a(s) = \frac{V_a(s) - k_b \omega(s)}{R_a + L_a s} \quad (7)$$

The motor torque is equal to the torque delivered to the load which may be expressed as

$$T_m(s) = T_l(s) + T_d(s) \quad (8)$$

where T_l is the load torque and T_d is the disturbance torque which is often negligible, so

$$T_l(s) = Js^2\theta(s) + bs\theta(s) \quad (9)$$

Therefore, the transfer function of the motor load combination, with $T_d = 0$, is:

$$\frac{\theta(s)}{V_a(s)} = \frac{k_m}{s\left((L_a s + R_a)(Js + b) + k_m k_b\right)} \quad (10)$$

Or

$$\frac{\theta(s)}{V_a(s)} = \frac{k_m}{L_a J s^3 + (L_a b + R_a J)s^2 + (R_a b + k_m k_b)s} \quad (11)$$

Here the angular displacement $\theta(s)$ is considered the output and the armature voltage $V_a(s)$ is considered the input. The block diagram representation is shown in figure 2.

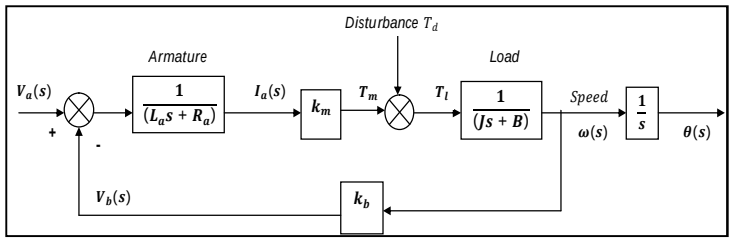

Fig.2 Block Diagram representation of a DC Servo motor

For the DC servomotor with parameters given in Appendix A, the overall transfer function of the system is given as:

$$\frac{\theta(s)}{V_a(s)} = \frac{0.01}{0.005s^3 + 0.06s^2 + 0.1001s} \quad (12)$$

III. EVOLUTIONARY COMPUTATION

Evolutionary computing is the collective name for a range of problem-solving techniques based on principles of biological evolution, which are being increasingly applied to a variety of problems, ranging from practical applications in industry and commerce to leading-edge scientific research. The idea in all these systems was to evolve a population of candidate solutions to a given problem, using operators inspired by natural genetic variation and natural selection.

Evolutionary computing techniques mostly involve metaheuristic optimization algorithms. The field includes: [9]

Evolutionary algorithms (EA):

- Genetic algorithm
- Genetic programming
- Evolutionary programming
- Differential evolution

Swarm intelligence (SI):

- Artificial Bee Colony Algorithm
- Ant colony optimization
- Particle swarm optimization

An EA simulates an evolutionary process on a population of individuals with the purpose of evolving the best possible approximate solution to the optimization problem at hand. Swarm intelligence is the collective behaviour of decentralized, self-organized systems, natural or artificial. The concept is employed in work on artificial intelligence.

Swarm Intelligence systems are typically made up of a population of simple objects interacting locally with one another and with their environment. Natural examples of SI include ant colonies, bird flocking, animal herding, bacterial growth, and fish schooling.

This paper mainly focuses Genetic Algorithm and Particle swarm Optimization for the optimal tuning of PID controller parameters as discussed in the following sections.

IV. GENETIC ALGORITHMS

A. *Overview*

Genetic Algorithms (GAs) are heuristic search techniques based on an artificial simulation of the mechanisms underlying the evolution of living beings: natural selection and genetic.

The simplest form of genetic algorithm involves three types of operators: selection, crossover, and mutation. GAs are population-based search methods that work through the following elements: populations of chromosomes, selection according to fitness, crossover to produce new offspring, and random mutation of new offspring.

The GA process consists in an iterative stepwise refinement of the performance of the individuals. The first step is the creation of a new population composed of individuals randomly generated. Then a fitness function evaluates and assigns to each individual a performance measure, or fitness value. The definition of the fitness function depends on the objective function. Then this population evolves for a number of iteration called generation until to satisfy a termination criterion.

B. *GA-Based PID Controller Optimization*

1) *GA Tuning Parameters*

The values in the Table 1 describe the GA settings used for this work.

TABLE 1: GA Tuning Parameters

PARAMETERS	**VALUES**
Lower bound [Kp Ki Kd]	[0 0 0]
Upper bound [Kp Ki Kd]	[100 100 100]
Stopping criteria (Iterations)	100
Population Size	40
Crossover Fraction	4
Mutation Fraction	0.08

2) *Steps in GA-Based PID Controller Optimization*

Step 1 % Establish initial population of individuals %
An initial random population having P(t) individuals is generated.

Step 2 % Evaluate the fitness of each individual in P(t)%
Evaluate all the individual solutions with the fitness function, which can be the inverse of error function.

Step 3 % Select some highly fit solutions%
Select P'(t+1) form intermediate population of fittest members from initial population P(t).

Step 4 % Apply crossover to selected solutions%
Pair off and mate individuals in P'(t+1) as parents and perform crossover operation to generate offsprings.

Step 5 % Apply mutation%
Perform mutation by slightly changing some random solution.

Step 6 Steps 2–5 are repeated until the predefined value of the function or the number of iterations has been reached. Record the optimized Kp, Ki and Kd values.

Step 7 Perform closed-loop test with the optimised values of controller parameters and calculate the time domain specification for the system.

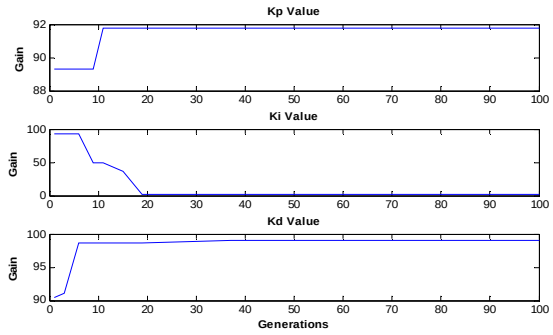

Fig.3 Convergence of Genetic Algorithm

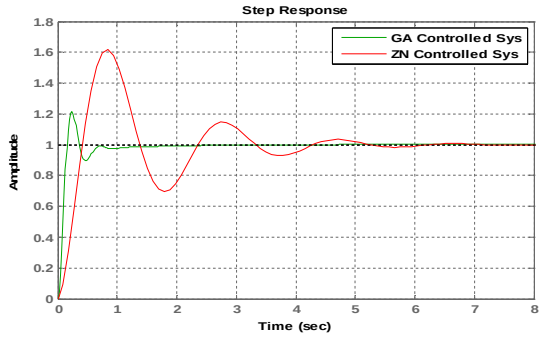

Fig.4 Step responses for GA and ZN tuned system

Figure 3 shows the convergence of genetic algorithm through various generations for the three PID parameters, Kp, Ki and Kd. The comparative output responses of the system tune using GA-based PID controller and conventionally tuned PID controller using Zeigler Nichols (ZN) method is shown in figure 4. The GA tuned system exhibits greatly reduced overshoot, rise time and settling time.

V. PARTICLE SWARM OPTIMIZATION

A. *Overview*

Particle swarm optimization (PSO) algorithm is a population-based evolutionary computation technique developed by the inspiration of the social behaviour in bird flocking or fish schooling. It attempts to mimic the natural process of group communication of individual knowledge, to achieve some optimum property. In this method, a population of swarm is initialized with random positions S_i and velocities V_i. At the beginning, each particle of the population is scattered randomly throughout the entire search space and with the guidance of the performance criterion, the flying particles dynamically adjust their velocities according to their own flying experience and their companions flying experience.

In PSO, each single solution is a "bird" in the search space; this is referred to as a "particle". The swarm is modelled as particles in a multidimensional space, which have positions and velocities. These particles have two essential capabilities: their memory of their own best position and knowledge of the global best [10].

Each particle remembers its best position obtained so far, which is denoted as *pbest* (P_i^t). It also receives the globally best position achieved by any particle in the population, which is denoted as *gbest* (G_i^t).

The updated velocity of each particle can be calculated using the present velocity and the distances from *pbest* and *gbest* as given by the following equations:

$$V_i^{t+1} = W^t \cdot V_i^t + C_1 \cdot R_1 \cdot (P_i^t - S_i^t) + C_2 \cdot R_2 \cdot (G_i^t - S_i^t) \quad (13)$$

$$S_i^{t+1} = S_i^t + V_i^{t+1} \quad (14)$$

$$W^t = (W_{max} - Iter) \times \left[\frac{(W_{max} - W_{min})}{Iter_{max}}\right] \quad (15)$$

The updated velocity and the position are given in (13) and (14), respectively. Equation (15) shows the inertia weight.

B. *PSO-Based PID Controller Optimization*

1) *PSO Tuning Parameters*

The values in the Table 2 describe the PSO settings used for this work

TABLE 2: PSO Tuning Parameters

PARAMETERS	VALUES
Lower bound [Kp Ki Kd]	[0 0 0]
Upper bound [Kp Ki Kd]	[100 100 100]
Stopping criteria (Iterations)	100
Population Size	40
Maximal velocity factor	0.2
Inertial weight factor [Min Max]	[0.4 0.9]
Acceleration constants [c1, c2]	[2 2]

2) *Steps in PSO-Based PID Controller Optimization* [2]

Step 1 % Assign values for the PSO parameters %
Initialize: swarm (N) and step size; learning rate (C_1, C_2) dimension for search space (D); inertia (W);
% Initialize random values and current fitness %
R_1=rand (D, N); R_2=rand (D, N);
current fitness =0*ones (N, 1).

Step 2 % Initialize Swarm Velocity and Position %
Current position =10*(rand (D, N)-0.2),
Current velocity =0.5*rand (D, N)

Step 3 Evaluate the objective function of every particle and record each particle's P_i^t andG_i^t. Evaluate the desired optimization fitness function in D-dimension variables.

Step 4 Compare the fitness of particle with its P_i^t and replace the local best value as given below.
for i=1: N
If current fitness (*i*) < local best fitness (*i*);
Then local best fitness = current fitness;
local best position = current position (*i*);
end
% same operation to be performed for G_i^t %.

Step 5 Change the current velocity and position of the particle group according to (13) and (14).

Step 6 Steps 2–5 are repeated until the predefined value of the function or the number of iterations has been reached. Record the optimized Kp, Ki and Kd values.

Step 7 Perform closed-loop test with the optimised values of controller parameters and calculate the time domain specification for the system.
If the values are within the allowable limit, consider the current Kp, Ki and Kd values. Otherwise perform the retuning operation for Ki, by replacing the optimised numerical values for K_p and K_d.

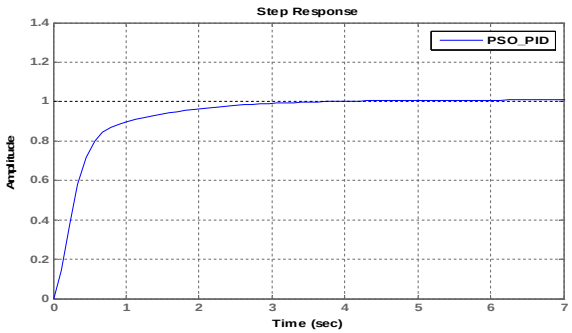

Fig.5 Step responses for GA and ZN tuned system

The output response of the system tune using PSO-based PID controller is shown in figure 5. The system exhibits nearly zero overshoot and remarkably reduced rise time.

V. SIMULATION RESULTS

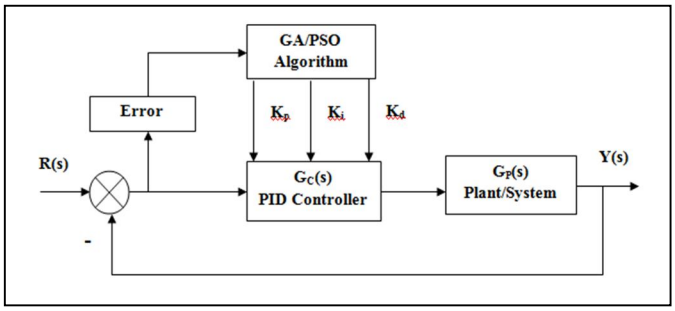

Fig.6 Block diagram for PID parameters tuning using PSO/GA.

In order to improve the performance of the dc motor under transient and steady state condition, a PID controller is inserted in the forward path as shown in Fig 6. The parameters of the PID controller are now adjusted by using conventional method i.e. Ziegler-Nichols method and the response obtained for the DC servomotor is evaluated. Further again the parameters of PID controller are obtained using evolutionary computation of GA and PSO and the system step responses are evaluated.
The controller gains were computed by using the classical Zeigler-Nichols rules and evolutionary computation techniques i.e. Genetic Algorithm and Particle swarm optimization. The controller gains obtained from the methods are listed in Table 3.

TABLE 3: Comparison of steady state responses

TITLE	ZN_PID	GA_PID	PSO_PID
Rise Time(sec)	0.2901	0.1070	0.9564
Settling Time (sec)	5.0139	1.0721	2.4892
% Overshoot	61.7409	21.3911	0.8190
Peak Time (sec)	0.8492	0.2395	8.0611
K_d	12.6486	99.0275	16.5836
K_p	72.0720	91.7813	20.4031
K_i	102.6374	1.3092	0.3758

Figure 7 shows the corresponding step responses of ZN, GA and PSO-based PID systems. It can be clearly seen that GA tuned system shows improved response with respect to the rise time and overshoot as compared to that of ZN tuned system. However, system tuned with PSO has further improved the overshoot of the considered system though the rise time and settling time exhibit slight increament.

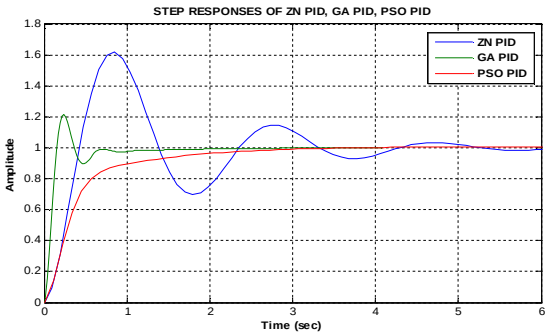

Fig.7 Comparitive step responses for GA, PSO and ZN tuned system

VI. CONCLUSIONS

Application of evolutionary computation techniques to the optimum tuning of PID controller led to a satisfactory close-loop response for the system under consideration. Comparison of the results as shown in Table 3 clearly reflects that the GA and PSO tuned PID outperformed the classical Zeigler-Nichols's tuning method in order to achieve minimum rise time, settling time overshoot and nearly zero steady state error. The simulation results show that the PSO-based PID controller tuning approach provides more improved performance as compared to GA-based PID controller for the considered system and hence, showed the superiority of the PSO.

REFERENCES

[1]. Dorf and Bishop, *Modern Control Systems*, 9th Ed., Prentice-Hall, Inc. 2001.

[2]. V. Rajinikanth and K. Latha, *Tuning and Retuning of PID Controller for Unstable Systems Using Evolutionary Algorithm*, Research Article, International Scholarly Research Network, Volume 2012, Article ID 693545, doi:10.5402/2012/693545

[3]. Ömer Gündoğdu, *optimal-tuning of PID controller gains using genetic algorithms*, Journal of Engineering Sciences, pp 131-135, 2005 11(1).

[4]. Bhawna Tandon, Randeep Kaur, *genetic algorithm based parameter tuning of pid controller for composition control system*, International Journal of Engineering Science and Technology, ISSN:0975-5462, Vol. 3 No. 8 August 2011, pp 6705-6711.

[5]. F. M. Amaral, Ricardo Tanscheit, Marco A. C. Pacheco, *Tuning PID Controllers through Genetic Algorithms*, WSES International Conference on Evolutionary Computation, Feb 12-14, 2001, pp 6121-6124.

[6]. Neenu Thomas, Dr. P. Poongodi, *Position Control of DC Motor Using Genetic Algorithm Based PID Controller*, Proceedings of the World Congress on Engineering 2009 (ISBN: 978-988-18210-1-0) Vol II WCE 2009, July 1 - 3, 2009, London, U.K.

[7]. Harinath Babu Kamepalli, The optimal basics for Gas, IEEE Potentials, 0278-6648/01/$10.00 © 2001 IEEE, April/May 2001 pp 25-27.

[8]. Tom V. Mathew, *Genetic Algorithm*, Lecture notes *http://www.civil.iitb.ac.in/tvm/2701_dga/2701-ga-notes/ gadoc/ gadoc.html*.

[9]. http://en.wikipedia.org/wiki/Evolutionary_computation

[10]. M. Molenaar, P. Nijdam, Y. Yan, W.A. Klop, *Tuning a PID controller: Particle Swarm Optimization versus Genetic Algorithms*, http://www.martinm.nl/attachments/028_paper_ tuning_ a_PID_controller.pdf

[11]. Sulochana Wadhwani, Veena Verma, Rekha Kushwah, "Design and Tuning of PID Controller Parameters based on Fuzzy Logic and Genetic Aalgorithm" , *Int. Conf. on Soft Computing, Artificial Intelligence, Pattern Recognition, Biomedical Engineering and Associated Technologies (SAP-BEATS)* 23-24 Feb ,*2013*", Department of Electrical Engineering, MBM Eng. College Jai Narain Vyas University,Jodhpur.

APPENDIX A: SYSTEM MODEL PARAMETERS

The parameters of the DC servomotor under consideration are as follows:

(J) moment of inertia of the rotor	0.01 kg.m^2
(b) motor viscous friction constant	0.1 N.m.s
(k_b) back emf constant	0.01 V/rad/sec
(k_m) motor torque constant	0.01 N.m/Amp
(R_a) electric resistance	1.0 Ohm
(L_a) electric inductance	0.5 H

NOMENCLATURE

i_a	Armature current in ampere,
i_f	Field current in ampere,
k_m	Motor torque constant,
k_b	Back emf constant,
V_a	Armature voltage in volts,
V_b	Back emf in volts,
J	Moment of inertia of rotor,
b	Viscous frictional constant of motor,
θ	Angular displacement of shaft in radians,
L_a	Armature inductance in henry,
R_a	Armature resistance in ohm
C	Positive acceleration constants (0–2)
gbest	Global best position
Gc(s)	Controller model
Gp(s)	Process model
IAE	Integrated absolute error
ISE	Integral squared error
ITAE	Integral time absolute error
Iter	Iteration
Kp	Proportional gain
Ki	Integral gain
Kd	Derivative gain
pbest	Local best position.
R	Random number (0-2)
R(s)	Reference input
S	Position of particle
V	Velocity of particle
W	Inertia weight of particle
Y(s)	Process out