A Comparative Study of PID Controller Tuning Using GA, EP, PSO and ACO

Submitted 9th August 2010; accepted 22nd February 2011.

B. Nagaraj, P. Vijayakumar

Abstract:

Proportional – Integral – Derivative control schemes continue to provide the simplest and effective solutions to most of the control engineering applications today. However PID controller are poorly tuned in practice with most of the tuning done manually which is difficult and time consuming. This article comes up with a hybrid approach involving Genetic Algorithm (GA), Evolutionary Programming (EP), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). The proposed hybrid algorithm is used to tune the PID parameters and its performance has been compared with the conventional methods like Ziegler Nichols and Cohen Coon method. The results obtained reflect that use of heuristic algorithm based controller improves the performance of process in terms of time domain specifications, set point tracking, and regulatory changes and also provides an optimum stability. Speed control of DC motor process is used to assess the efficacy of the heuristic algorithm methodology.

Keywords: *Ant colony algorithm, evolutionary programming, genetic algorithm particle swarm optimization and soft computing.*

1. Introduction

PID controller is a generic control loop feedback mechanism widely used in industrial control systems. It calculates an error value as the difference between measured process variable and a desired set point [3]. The PID controller calculation involves three separate parameters proportional integral and derivative values .The proportional value determines the reaction of the current error, the integral value determines the reaction based on the sum of recent errors, and derivative value determines the reaction based on the rate at which the error has been changing the weighted sum of these three actions is used to adjust the process via the final control element.

The goal of PID controller tuning is to determine parameters that meet closed loop system performance specifications, and the robust performance of the control loop over a wide range of operating conditions should also be ensured. Practically, it is often difficult to simultaneously achieve all of these desirable qualities. For example, if the PID controller is adjusted to provide better transient response to set point change, it usually results in a sluggish response when under disturbance conditions [11].

On the other hand, if the control system is made robust to disturbance by choosing conservative values for the PID controller, it may result in a slow closed loop response to a set point change. A number of tuning techniques that take into consideration the nature of the dynamics present within a process control loop have been proposed [4]. All these methods are based upon the dynamical behavior of the system under either open-loop or closed-loop conditions.

In this paper, heuristic approach to optimally design a PID controller, for a DC motor is proposed. A comparison between the results obtained by the heuristic methods and conventional methods via simulation of the DC motor is presented in results and comparison section. The parameters of a DC motor used in this paper are listed in Table 1.

Table 1. Parameters of the DC Motor.

Parameters	Values & Units
R (Resistance of the stator)	21.2 Ω
K_b (Back electromotive force constant)	0.1433 Vs rad^{-1}
D (Viscous coefficient)	1*10^{-4} kg m s/rad
L (Inductance of the stator)	0.0524 H
K_t (Motor torque constant)	0.1433 kg m/A
J (Moment of inertia)	1*10^{-5} kg m S^2/rad

The characteristic equation of DC motor can be represented as,

$$v_{app}(t) = Ldi(t) + \frac{Ri(t)}{dt} + V_{emf}(t) \tag{1}$$

$$V_{emf} = K_b \cdot w(t) \tag{2}$$

$$T(t) = K_t \cdot i(t) \tag{3}$$

$$T(t) = J\frac{dw(t)}{dt} + D \cdot w(t) \tag{4}$$

Where $V_{app}(t)$ is the applied voltage, $w(t)$ is the motor speed, L is the inductance of the stator, $i(t)$ is the current of the circuit, R is the resistance of the stator, $V_{emf}(t)$ is the back electromotive force, T is the torque of the motor, D is the viscous coefficient. J is the moment of inertia, K_t is the motor torque constant, and K_b is the back electromotive force constant.

From the characteristics equations of the motor, the transfer function is obtained,

$$G(S) = \frac{0.1433}{5.2e-007\,s^2 + 0.000217\,s + 2.265} \quad (5)$$

The purpose of this paper is to investigate an optimal controller design using the evolutionary Programming, Genetic Algorithm, Particle Swarm Optimization and Ant Colony Optimization. The block diagram of a control system with unity feedback employing soft computing PID control action is shown in Figure 1[7].

The general equation of PID controller is,

$$Y(t) = [k_p e(t) + K_d \frac{d(e)}{d(t)} + K_i \int_0^t e(t)d(t)] \quad (6)$$

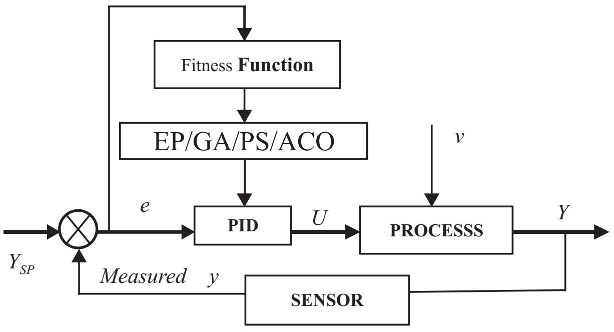

Fig 1. Block diagram of Intelligent PID controller.

The initial values of PID gain are calculated using conventional Z – N method. Being hybrid approach, optimum value of gain are obtained using heuristic algorithm.

The advantages of using heuristic techniques for PID are listed below,

i. Heuristic Techniques can be applied for higher order systems without model reduction [7].
ii. These methods can also optimize the design criteria such as gain margin, Phase margin, Closed Loop Band Width (CLBW) when the system is subjected to step & load change [7].

Heuristic techniques like Genetic Algorithm (GA), Evolutionary Programming (EP), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) methods have proved their excellence in giving better results by improving the steady state characteristics and performance indices.

2. GA based tuning of the controller

The optimal value of the PID controller parameters K_p, K_i, K_d is to be found using GA. All possible sets of controller parameters values are particles whose values are adjusted to minimize the objective function, which in this case is the error criterion, and it is discussed in detail. For the PID controller design, it is ensured the controller settings estimated results in a stable closed-loop system [1].

Genetic Algorithms are a stochastic global search method that mimics the process of natural evolution. It is one of the methods used for optimization. John Holland formally introduced this method in the United States in the 1970 at the University of Michigan. The continuing performance improvement of computational systems has made them attractive for some types of optimization. The genetic algorithm starts with no knowledge of the correct solution and depends entirely on responses from its environment and evolution operators such as reproduction, crossover and mutation to arrive at the best solution [1]. By starting at several independent points and searching in parallel, the algorithm avoids local minima and converging to sub optimal solutions.

A. Objective Function of the Genetic Algorithm

This is the most challenging part of creating a genetic algorithm is writing the the objective function. In this project, the objective function is required to evaluate the best PID controller for the system. An objective function could be created to find a PID controller that gives the smallest overshoot, fastest rise time or quickest settling time. However in order to combine all of these objectives it was decided to design an objective function that will minimize the performance indices of the controlled system instead [2]. Each chromosome in the population is passed into the objective function one at a time. The chromosome is then evaluated and assigned a number to represent its fitness, the bigger its number the better its fitness [3]. The genetic algorithm uses the chromosomes fitness value to create a new population consisting of the fittest members. Each chromosome consists of three separate strings constituting a *P*, *I* and *D* term, as defined by the 3-row bounds declaration when creating the population [3]. When the chromosome enters the evaluation function, it is split up into its three Terms. The newly formed PID controller is placed in a unity feedback loop with the system transfer function. This will result in a reduce of the compilation time of the program. The system transfer function is defined in another file and imported as a global variable. The controlled system is then given a step input and the error is assessed using an error performance criterion such as Integral square error or in short ISE.

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

The chromosome is assigned an overall fitness value according to the magnitude of the error, smaller the error larger the fitness value. Initializing the values of the parameters is as per Table 2. The flowchart of the GA control system is shown in Figure 2.

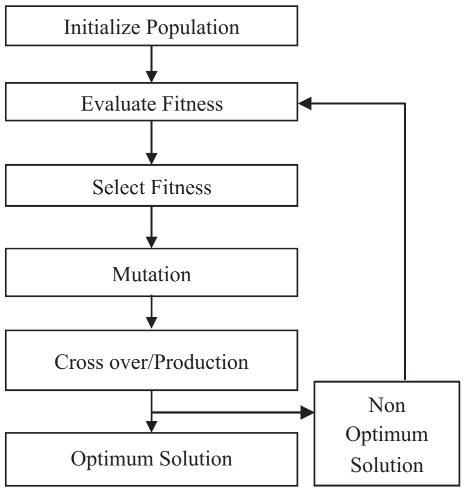

Fig. 2. Flowchart of GA.

3. EP based tuning of the controller

EP is generally used to optimize real-valued continuous functions. EP uses selection and mutation operators and does not use the crossover operator. The focus is on the observed characteristics of the population. The selection operator is used to determine chromosomes for mating in order to generate new chromosomes [22].

There are two important ways in which EP differs from GAs.

First, there is no constraint on the representation. The typical GA approach involves encoding the problem solutions as a string of representative tokens, the genome. In EP, the representation follows from the problem. A neural network can be represented in the same manner as it is implemented, for example, because the mutation operation does not demand a linear encoding [7].

Second, the mutation operation simply changes aspects of the solution according to a statistical distribution which weights minor variations in the behavior of the offspring as highly probable and substantial variations as increasingly unlikely.

The steps involved in creating and implementing evolutionary programming are as follows:

- Generate an initial, random population of individuals for a fixed size (according to conventional methods K_p, K_i, K_d ranges declared).
- Evaluate their fitness (to minimize integral square error).

$$\text{ISE} = \int_0^\infty e^2(t)dt \tag{7}$$

- Select the fittest members of the population.
- Execute mutation operation with low probability.
- Select the best chromosome using competition and selection.
- If the termination criteria reached (fitness function) then the process ends. If the termination criteria not reached search for another best chromosome.

Initializing the values of the parameters is as per Table 2. The flowchart of the EP control system is shown in Fig. 3.

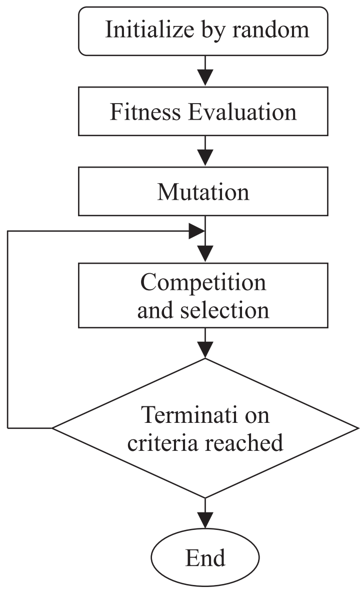

Fig. 3. Flow Chart of EP.

4. PSO based tuning of the controller

PSO is one of the optimization techniques and kind of evolutionary computation technique. The technique is derived from research on swarm such as bird flocking and fish schooling. In the PSO algorithm, instead of using evolutionary operators such as mutation and crossover to manipulate algorithms, for a d-variable optimization Problem, a flock of particles are put into the d-dimensional Search space with randomly chosen velocities and positions knowing their best values [8].

The algorithm proposed by Eberhart and kennedy (1995) uses a 1-D approach for searching within the solution space. For this study the PSO algorithm will be applied to a 2-D or 3-D solution space in search of optimal tuning parameters for PI, PD and PID control [21].

Consider position $X_{i,m}$ of the *I*-th particle as it traverses a *n*-dimensional search space: The previous best position for this *i*-th particle is recorded and represented as $Pbest_{i,n}$. The best performing particle among the swarm population is denoted as $gbest_{i,n}$ and the velocity of each particle within the *n*-dimension is represented as $V_{i,n}$. The new velocity and position for each particle can be calculated from its current velocity and distance, respectively [18].

So far (p best) and the position in the *d*-dimensional space [7]. The velocity of each particle, adjusted accordingly to its own flying experience and the other particles flying experience [7].

For example, the i th particle is represented, as

$$x_i = (X_{i,1}, X_{i,2},..., X_{i,d})$$

In the d-dimensional space. The best previous position of the i th particle is recorded as,

$$Pbest_i = (Pbest_{i,1}, Pbest_{i,2},..., Pbest_{i,d}) \tag{8}$$

The index of best particle among all of the particles in the group in $gbest_d$.The velocity for particle *i* is represented as

$$V_i = (V_{i,1}, V_{i,2},..., V_{i,d}) \tag{9}$$

The modified velocity and position of each particle can be calculated using the current velocity and distance from $Pbest_{i,d}$ to $gbest_d$ as shown in the following formulas

$$V_{i,m}^{(t+1)} = W \cdot V_{i,m}^{(t)} + c_1 * rand() * (Pbest_{i,m} - x_{i,m}^{(t)}) + \\ + c_2 * Rand() * (gbest_m - x_{i,m}^{(t)}) \tag{10}$$

$$x_{i,d}^{(t+1)} = x_{i,m}^{(t)} + v_{i,m}^{(t+1)} \qquad i = 1,2,...,n; \;\; m = 1,2,...,d \tag{11}$$

where

n	- number of particles in the group
d	- dimension
t	- pointer of iterations (generations)
$V_{i,m}^{(t+1)}$	- velocity of particle *I* at iteration *t*
W	- inertia weight factor
c_1, c_2	- acceleration constant
$rand(n)$	- random number between 0 and 1
$x_{i,d}^{(t)}$	- current position of particle i at iterations
$Pbest_{i,m}$	- best previous position of the ith particle
$gbest_m$	- best particle among all the particles in the population

In the proposed PSO method each particle contains three members *P*, *I* and *D*. It means that the search space has three dimension and particles must 'fly' in a three dimensional space. Initializing the values of the parameters is as per Table 2. The flowchart of the PSO – PID control system is shown in Fig. 4.

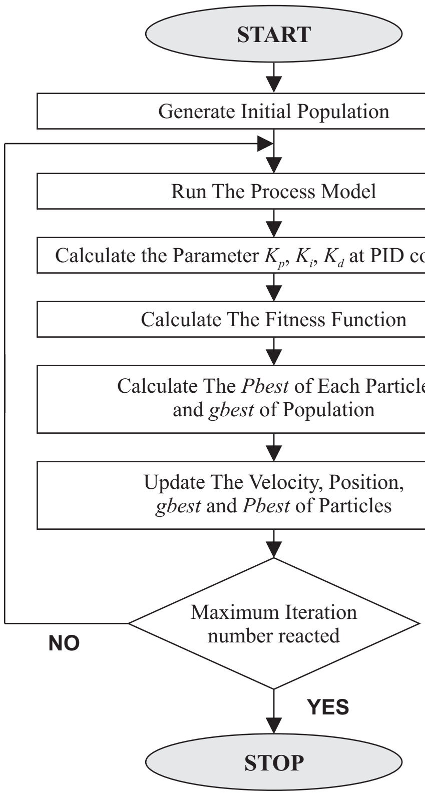
START
Generate Initial Population
Run The Process Model
Calculate the Parameter K_p, K_i, K_d at PID controller
Calculate The Fitness Function
Calculate The *Pbest* of Each Particle and *gbest* of Population
Update The Velocity, Position, *gbest* and *Pbest* of Particles
Maximum Iteration number reacted
NO
YES
STOP

Fig. 4. Flowchart of PSO.

5. ACO based tuning of the controller

ACO's are especially suited for finding solutions to different optimization problems. A colony of artificial ants cooperates to find good solutions, which are an emergent property of the ant's co-operative interaction. Based on their similarities with ant colonies in nature, ant algorithms are adaptive and robust and can be applied to different versions of the same problem as well as to different optimization problems [23]. The main traits of artificial ants are taken from their natural model. These main traits are (1) artificial ants exist in colonies of cooperating individuals, (2) they communicate indirectly by depositing pheromone (3) they use a sequence of local moves to find the shortest path from a starting position, to a destination point they apply a stochastic decision policy using local information only to find the best solution. If necessary in order to solve a particular optimization problem, artificial ants have been enriched with some additional capabilities not present in real ants [16].

An ant searches collectively foe a good solution to a given optimization problem. Each individual ant can find a solution or at least part of a solution to the optimization problem on its own but only when many ants work together they can find the optimal solution [4]. Since the optimal solution can only be found through the global cooperation of all the ants in a colony, it is an emergent result of such this cooperation. While searching for a solution the ants do not communicate directly but indirectly by adding pheromone to the environment. Based on the specific problem an ant is given a starting state and moves through a sequence of neighboring states trying to find the shortest path. It moves based on a stochastic local search policy directed by its internal state, the pheromone trails, and local information encoded in the environment. Ants use this private and public information inorder to decide when and where to deposit pheromone. In most application the amount of pheromone deposited is proportional to the quality of the move an ant has made. Thus the more pheromone, the better the solution found. After an ant has found a solution, it dies; i.e.it is deleted from the system [13].

ACO uses a pheromone matrix $\tau=\{\tau_{ij}\}$ for the construction of potential good solutions. The initial values of τ are

set $\tau_{ij} = \tau_0 \forall (i,j)$, where $\tau_0 > 0$

The probability $P_{ij}^A(t)$ of choosing a node *j* at node *I* is defined in the equation (12). At each generation of the algorithm, the ant constructs a complete solution using (12), starting at source node.

$$P_{ij}^A(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{i,j \in T^A} \tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta} \quad \text{if } i,j \in T^A \tag{12}$$

where $\eta_{ij} = \frac{1}{k_j}$, $j = [p,i,d]$:

representing heuristic functions.

α and β are constants that determine the relative influence of the pheromone values and the heuristic values on the decision of the ant.

T^A: is the path effectuated by the ant *A* at a given time.

The quantity of pheromone $\Delta\tau_{ij}^A$ on each path may be defined as

$$\Delta\tau_{ij}^A = \begin{cases} \frac{L^{min}}{L^A} & \text{if } i,j \in T^A \\ 0 & \end{cases} \tag{13}$$

else

where:

L^A - is the value of the objective function found by the ant *A*.

L^{min} - is the best solution carried out by the set of the ants until the current iteration.

The pheromone evaporation is a way to avoid unlimited increase of pheromone trails. Also it allows the forgetfulness of the bad choices.

$$\tau_{ij}(t) = p\tau_{ij}(t-1) + \sum_{A=1}^{NA} \Delta\tau_{ij}^A(t) \tag{14}$$

where:

NA - number of ants

P - the evaporation rate. $0 < p <= 1$.

A. Implementation algorithm

Step 1

Initialize randomly a potential solutions of the parameters (K_p, K_i, K_d) by using uniform distribution. Initialize the pheromone trail and the heuristic value.

Step 2

Place the A^{th} ant on the node. Compute the heuristic value associated in the objective (minimize the error).

Step 3

Use pheromone evaporation given by eqn (14) to avoid unlimited increase of pheromone trails and allow the forgetfulness of bad choices.

Step 4

Evaluate the obtained solutions according to the objectives.

Step 5

Display the optimum values of the optimization parameters.

Step 6

Globally update the pheromone, according to the optimum solutions calculated at step 5. Iterate from step 2 until the maximum of iterations is reached.

Initializing the values of the parameters is as per Table 2. The flowchart of the ACO – PID control system is shown in Fig. 5.

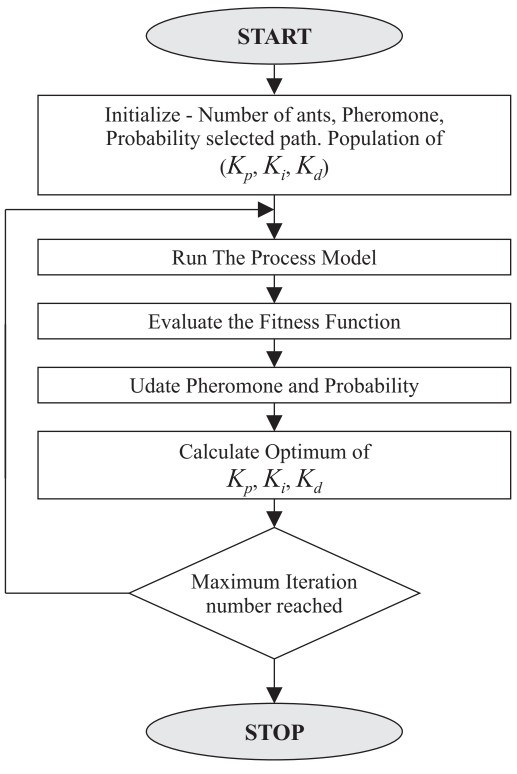
START
Initialize - Number of ants, Pheromone, Probability selected path. Population of (K_p, K_i, K_d)
Run The Process Model
Evaluate the Fitness Function
Udate Pheromone and Probability
Calculate Optimum of K_p, K_i, K_d
Maximum Iteration number reached
STOP

Fig 5. Flowchart of ACO.

6. Results and comparisons

The transfer function of DC motor has been taken to analyze the performance of various heuristic algorithms. Transfer function is given by,

$$G(S) = \frac{0.1433}{5.2e-007\, s^2 + 0.000217\, s + 2.265}$$

The initial values of PID gains are calculated using conventional Z–N method.

In this paper a time domain criterion is used for evaluating the PID controller. A set of good control parameters, *P*, *I* and *D* can yield a good step response that will result in performance criteria minimization in time domain [18]. These performance criteria in time domain include the overshoot, rise time and settling time.

To show the effectiveness of the heuristic method, a comparison is made with the conventional designed PID controller with GA, EP, and PSO & ACO method.

At first method, PID controller design using Z – N method & the values of designed PID controller are $K_p = 9.3883$, $K_i = 36.4170$, and $K_d = 0.6051$.

Initialize the values of the parameters EP, GA, PSO & ACO is as per table 2. The values of EP, GA, PSO and ACO designed PID controllers are tabulated in table 3. Performance characteristics of DC motor were indicated & compared with heuristic tuning methods as shown in Fig 6.

Simulation shows the performance characteristics of conventional method of controller tuning lead to a large settling time, overshoot, rise time & steady state error, GA, EP, PSO & ACO based tuning methods have proved their excellence in giving better result by improving the steady state characteristics and performance indices.

7. Conclusion

Research work has been carried out to get an optimal PID tuning by using GA, EP, PSO and ACO. Simulation results demonstrate the tuning methods that have a better control performance compared with the conventional ones. It is possible to consider several design criteria in a balanced and unified way. Approximations that are typical to classical tuning rules are not needed. Soft computing techniques are often criticized for two reasons: algorithms are computationally heavy and convergence to the optimal Solution cannot be guaranteed. PID controller tuning is a small-scale problem and thus computational complexity is not really an issue here. It took only a couple of seconds to solve the problem. Compared to conventionally tuned system, GA, EP , PSO and ACO tuned system has good steady state response and performance indices.

Table 2. PSO, GA, EP and ACO Parameters.

PSO PARAMETERS	GA PARAMETERS	EP PARAMETERS	ACO PARAMETERS
Population size:100	Population size:100	Population size:100	Population size:100
Wmax=0.6/ Wmin=0.1	Mutation rate:0.1	Normal distribution	No of Ants = 10
C1 = C2 = 1.5	Arithmetic Crossover	Mutation rate: 0.01	No. of Path = 15 C1 = C2 = 2
Iteration:100	Iteration:100	Iteration:100	Iteration :100
Fitnessfunction:ISE	Fitnessfunction:ISE	Fitnessfunction:ISE	Fitnessfunction: ISE

Table 3. Comparison result of Z-N and Heuristic methods.

Tuning Method	PID Parameters			Dynamic performance specifications			Performance Index
	K_p (Proportional gain)	K_i (Integral gain)	K_d (Derivative Gain)	T_r (Rise time)	T_s (Settling time)	M_p (%) (Peak overshoot)	ISE (Integral square error)
ZN	9.3883	36.4170	0.6051	1.27	2.235	1	2.2926
EP	10	100	0.1	0. 468	0.877	0. 0	1.334
GA	3	90	0.001	0. 444	0.781	0. 0	1.4406
PSO	1.5	500	0.02	0. 0761	0.13	0. 0	1.0024
ACO	10	200	0.21	0. 00105	0.429	0. 0	1.174

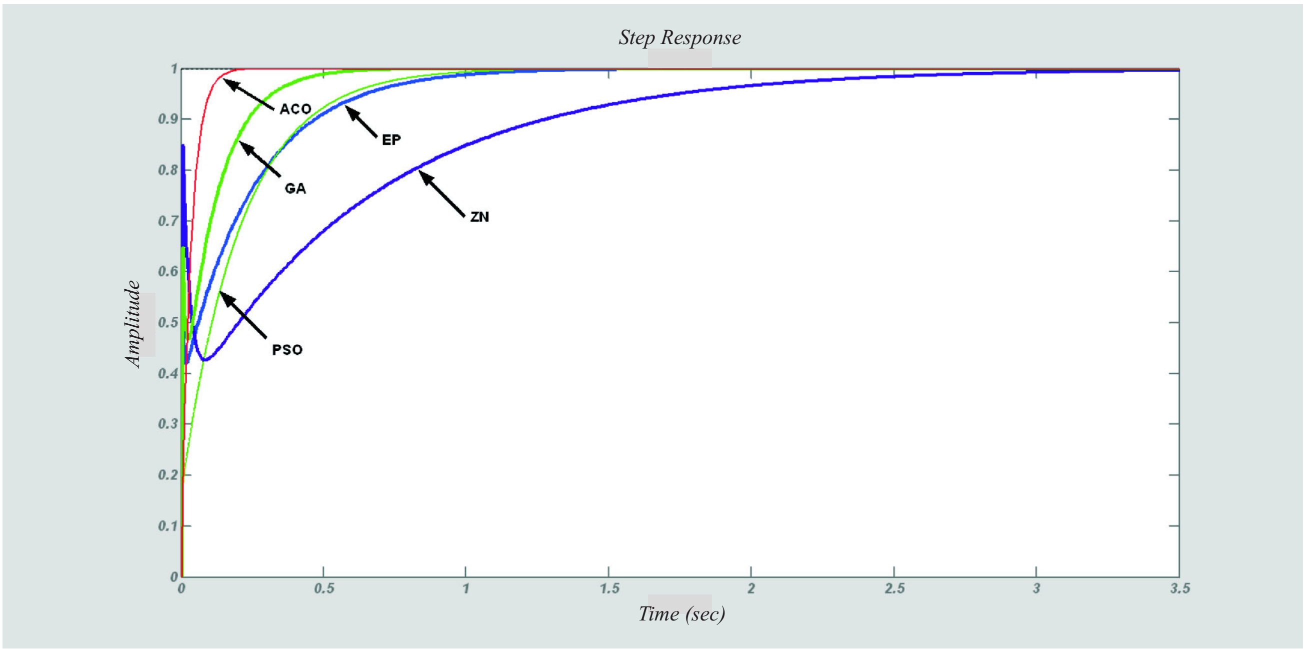

Fig. 6. Comparison result of Z-N and Heuristic methods.

AUTHORS

B. Nagaraj* - Tamilnadu Newsprint and Papers Ltd, Tamilnadu, India. Research Scholar Karpagam University. E-mail: nagarajice@gmail.com.
P. Vijayakumar - Karpagam College of Engineering, Tamilnadu, India.
* Corresponding author

References

[1] Ian Griffin, Jennifer Bruton "On-Line PID controller tuning using genetic algorithm". Available at: www.eeng.dcu.ie/~brutonj/Reports/IGriffin_MEng_03.pdf
[2] M.B.B. Sharifian, R. Rahnavard, H. Delavari "Velocity Control of DC Motor Based Intelligent methods and Optimal Integral State FeedbackController", *International Journal of Computer Theory and Engineering*, vol. 1, no. 1, April 2009.
[3] N. Thomas, P. Poongodi "Position Control of DC Motor Using Genetic Algorithm Based PID Controller". In: *Proceedings of the World Congress on Engineering 2009*, 1st-3rd July 2009, London, UK, vol. II.
[4] K.J. Astrom, T. Hagglund, " Automatic tuning of simple regulators with specification on phase and amplitude margins", *Automatica*, vol. 20, 1984, pp. 645- 651.
[5] A.A. Khan, N. Rapal "Fuzzy PID controller: design, tuning and comparison with conventional PID controller". In: *IEEE International Conference on Engineering of Intelligent Systems*, 2006, pp. 1-6, DOI 10.1109/ICEIS.2006.1703213.
[6] S. Saha, "Performance Comparison of Pid base Position control system with FLC based position control system", *TIG Research Journal*, vol. 1, no. 2, Sept. 2008.
[7] J. Lieslehto, "PID controller tuning using Evolutionary programming". In: *American Control Conference*, VA, USA, 25th-27th June 2001.
[8] M. Nasri, H. Nezamabadi-pour, M. Maghfoori, "A PSO-Based Optimum Design of PID Controller for a Linear Brushless DC Motor", *World Academy of Science, Engineering and Technology*, no. 26, 2007.
[9] B. Nagaraj, S. Subha, B. Rampriya, "Tuning Algorithms for PID Controller Using Soft Computing Techniques", *IJCSNS International Journal of Computer Science and Network Security*, vol. 8, no. 4, April 2008.
[10] H.S. Hwang, J.N. Choi, W.H. Lee, J.K. Kim,"A Tuning Algorithm for The PID Controller Utilizing Fuzzy Theory", *International Joint Conference on Neural Networks*, vol. 4, 1999, pp. 2210-2215.
[11] Jan Jantzen, "*Tuning of fuzzy PID controllers*". Den-

mark.Tech. Report no. 98-H 871(fpid), 30. Sept. 1998, pp. 1-22.

[12] Kiam Heong Ang, Gregory Chong, "PID Control System Analysis, Design, and Technology", *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, July 2005, pp. 559-576.

[13] I. Chiha, P. Borne, "Multi-Objective Ant Colony Optimization to tuning PID Controller". In: *Proceedings of the International Journal of Engineering*, vol. III, issue no. 2, March 2010.

[14] G. Dicaro, M.Dorigo, "Ant colonies for adaptive routing in packet switched communications network". In: A.E. Eiben, T. Back, M. Schoenauer, a H-P. Schwefel, ed., *Proceedings of PPSN-V 5th international conference on parallel problem solving from nature*, Lecture notes in csc, vol. 1498, Springer Verlag: Berlin, 1998, pp. 673-682.

[15] G. Zhou, J.D. Birdwell, "Fuzzy logic- based PID autotuner design using simulated annealing". In: *Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design*, 7th-9th March, 1994, pp. 67-72.

[16] H. Ying-Tung, C. Cheng-Long, C. Cheng-Chih, "Ant colony optimization for designing of PID controllers", *IEEE International Symposium on Computer Aided Control Systems Design,* Taipei,Taiwan, 24th September, 2004.

[17] B. Nagaraj, N. Murugananth, "A comparative approach approach of soft computing methodologies for industrial process tuning", *KYTO Journal Engineering Research*, vol. II, Dec 2009.

[18] N. Pillay, "A particle swarm optimization". Master Thesis Dept. of Electronics Engineering at DURBAN Univ. of Tech., 2009.

[19] E. Grassi, K. Taskatis, "PID controller tuning by frequency loop shaping: Application to diffusion furnace temp. control", *IEEE Transaction on Control System Tech.*, vol. VIII, no. 5, Sept. 2000.

[20] A. Karimi, D. Gracia, R. Longchamp, "PID Controller Tuning using Bode's Integrals", *IEEE transactions on Control System Tech*., vol. XI, no. 6, Nov. 2003.

[21] T.-H. Kim, I. Maruta, T. Sugia, "Particle Swarm Optimization based Robust PID Controller tuning", *IEEE Conference on Decision & Control*, 12th 14th Dec, 2007 New Orleans, LA, USA, pp. 200-205.

[22] N. Pillay, P. Govender, "A particle Swarm Optimization Approach for model independent tuning of PID control loop", *IEEE African 2007*, IEEE catalog: 04CH37590C, ISBN: 0-7803-8606.

[23] K. Ramkumar, S. Sharma, "Real Time Approach of Ant Colony Optimization", *International Journal of Computer Application*, vol. 3, no. 8, June 2010, pp. 34-46.