International Journal of Current Engineering and Technology
©2018 INPRESSCO®, All Rights Reserved

E-ISSN 2277 – 4106, P-ISSN 2347 – 5161
Available at http://inpressco.com/category/ijcet

Research Article

Robust Performance of PID Controller by using Artificial Intelligence

Shachi Tiwary[#*], Ashraf Jafri[#], Kushal Tiwari[^], Richa Tiwari[#] and Chaman Yadav[^]

#CSIT Durg, India
^CREDA Raipur, India

Received 10 March 2018, Accepted 15 May 2018, Available online 17 May 2018, **Vol.8, No.3 (May/June 2018)**

Abstract

This paper is meant to design method for determining the optimal proportional-integral-derivative (PID) controller parameters of plant system using the particle swarm optimization (PSO) algorithm and bacterial Foraging Optimization (BFO). There are several methods which are used to tune the controller parameters. They are categorized into two types known as classical methods and modern methods. In this paper the use of PSO method tuned the PID parameter to make them more general and to achieve the steady state error limit, also to improve the dynamic behaviour of the system. The performance and design criteria of automatic selection of controller constants are discussed below.

Keywords: *Plant system, PID controller, PSO and BFO Optimization.*

1. Introduction

During the past decades, the process control techniques in the industry have made great advances. Numerous control methods such as adaptive control, neural control, and fuzzy control have been studied. Among them, the best known is the proportional-integral-derivative (PID) controller, which has been widely used in the industry because of its simple structure and robust performance in a wide range of operating conditions. Unfortunately, it has been quite difficult to tune properly the gains of PID controllers because many industrial plants are often burdened with problems such as high order, time delays, and nonlinearities. For these reasons, it is highly desirable to increase the capabilities of PID controllers by adding new features. Many artificial intelligence (AI) techniques have been employed to improve the controller performances for a wide range of plants while retaining their basic characteristics. AI techniques such as neural network, fuzzy system, and neural-fuzzy logic have been widely applied to proper tuning of PID controller parameters PID controller consists of Proportional, Integral and Derivative gains. The PID feedback control system is illustrated in Fig. where r, e, y are respectively the reference, error and controlled variables (Dingyu Xuet *et al*, 1993). Where Kp is proportional gain, Ki is integral gain and Kd is derivative gain.

A PID controller is described by the following transfer function in the continuous s-domain:

$$Gc = P + I + D$$

$$Gc = K_p + \frac{ki}{s} + K_d S$$

$$Gc = Kp\,(1 + \frac{1}{Ti} S + T_d S)$$

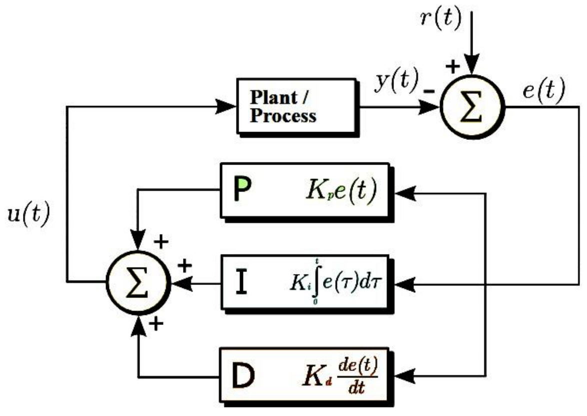

Fig.1 Block diagram of a PID controller in a close loop

The PID control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable (mv). The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining u(t) as the controller output, the final form of the PID algorithm is:

$$\mathrm{u}(t) = \mathrm{MV}(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{de(t)}{dt} \quad (1)$$

*Corresponding author's ORCID ID: 0000-0002-5069-9089
DOI: https://doi.org/10.14741/ijcet/v.8.3.10

Where,

K_p: Proportional gain, a tuning parameter
K_i: Integral gain, a tuning parameter
K_d: Derivative gain, a tuning parameter
e: Error SP - PV
SP: Set Point
PV: Process Variable
t: Time or instantaneous time (the present)
τ : Variable of integration; takes on values from time 0 to the present *t*.

Equivalently, the transfer function in the Laplace Domain of the PID controller is:

$L(s) = K_p + K_i/s + K_d s$

Where

s: Complex number frequency (Anitha Mary *et al*, 2012.

2. Swarm topology

Each particle i (total number of particles) has its neighbourhood N_i (a subset of P). The arrangement of the neighbourhoods is called the swarm topology, which can be represented by a graph. Usual topologies are: Fully connected topology, Circle topology and single sighted topology.

Characteristics of particle i at iteration t

$x_i^{(t)}$ - the position
$p_i^{(t)}$ -the individual best position
$l_i^{(t)}$- the local best position of the neighbouring particles
$v_i^{(t)}$ - the velocity of the particle
At the beginning of the algorithm, the particle positions are randomly initialized, and the velocities are set to 0, or to small random values.

Algorithm parameters

w(t) - Inertia weight; a damping factor, usually decreasing from around 0.9 to around 0.4 during the computation
$\emptyset_1$, $\emptyset_2$ - Acceleration coefficients; usually between 0 and 4

Manipulating its velocity

Many versions of the particle speed update exist, for example:

$$v_i^{(t+1)} = \omega^{(t)} v_i^{(t)} + \emptyset_1 c_1 (p_i^{(t)} - x_i^{(t)}) + \emptyset_2 c_2 (l_i^{(t)} - x_i^{(t)}) \quad (2)$$

The symbols u_1 and u_2 represent random variables with the C(0,1) distribution. The first part of the velocity formula is called inertia, the second one the cognitive (personal) component, the third one is the social (neighbourhood) component. Position of particle i changes according to

$$x_i^{(t+1)} = x_i + v_i^{(t+1)} \quad (3)$$

3. PSO Algorithm

PSO is optimization algorithm based on evolutionary computation technique. The basic PSO is developed from research on swarm such as fish schooling and bird flocking. After it was firstly introduced in 1995, a modified PSO was then introduced in 1998 to improve the performance of the original PSO.

Basic Fundamentals of PSO Algorithm

The basic fundamentals of the PSO algorithm technique are stated and defined as follows.

Particle X(i): A candidate solution represented by a k-dimensional real-valued vector, where k is the number of optimized parameters; at iteration i, the jth particle X(i,j) can be described as :

$$X_{j(i)} = [x_{j,1(i)}; x_{j,2(i)}; \ldots\ldots.; x_{j,k(i)} \ldots.; x_{j,d(i)}] \quad (4)$$

Where: x_s are the optimized parameters

$X_k(i,j)$ is the kth optimized parameter in the jth candidate solution d represents the number of control variables.

Population: This is the set or say population of n particles at iteration i.

$$Pop(i) = [X_{i(j)}, X_{2(i)}, \ldots\ldots.. X_{n(i)}]^T \quad (5)$$

Where n represent the number of individual solutions.

Swarm: As the name swarm defines the group of particles, it is a disorganized population of moving particles that tends to cluster together and each particle seems to moving in a random direction with their velocity.

Particle velocity V(i): The velocity of the moving particles represented by a d-dimensional real-valued vector; at iteration i, the jth particle $V_j(i)$ can be described as:

$$V_{j(i)} = [v_{j,1(i)}; v_{j,2(i)}; \ldots\ldots v_{j,k(i)}; \ldots\ldots; v_{j,d(i)}] \quad (6)$$

Weight factor w(i): This is a control parameter and sometimes it is called as inertia weight factor. the inertia factor decreases linearly from about 0.9 to 0.4 during a continuous run .

In general, this factor is set according to following equation**:**

$$W = w_{max} - (w_{max} - w_{min})/iter_{max} * iter \quad (7)$$

Individual best $X^*(i)$: The best position that is associated with the best fitness encountered thus far is called individual best $X^*(i)$. For each particle in the swarm, $X^*(i)$ can be determined and updated during the search. For the jth particle, individual best can be expressed as:

$$X_j^*(i) = [x_{j,1(i)}, x_{j,2*(i)}, \ldots\ldots x_{j,d*(i)}]^T \quad (8)$$

In a minimization problem with only one objective function f, the individual best of the jth particle $X_j^*(i)$ is updated. Otherwise individual best solution of the jth particle will be kept as in the previous iteration.

Global best $X^{**}(t)$: This is the best position among all of the individual best positions achieved so far.

Stopping Criteria: The search process will be terminated whenever one of the following criteria is satisfied (Yogendra Kumar Soni *et al*, 2013).

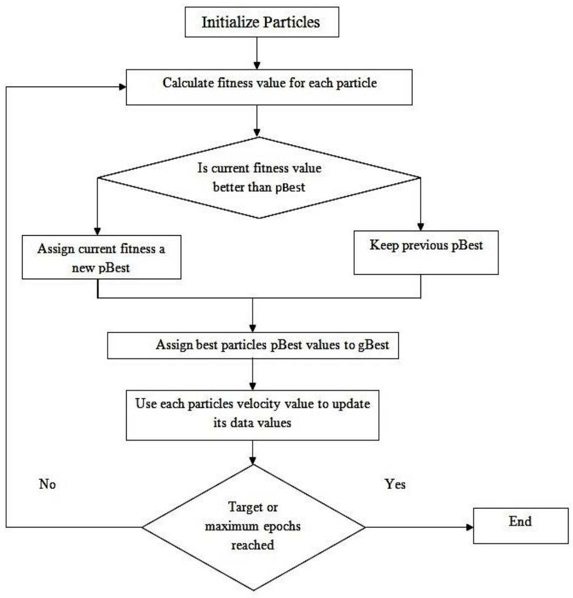
Initialize Particles
Calculate fitness value for each particle
Is current fitness value better than pBest
Assign current fitness a new pBest
Keep previous pBest
Assign best particles pBest values to gBest
Use each particles velocity value to update its data values
No
Yes
Target or maximum epochs reached
End

Fig.2 Flow diagram of PSO algorithm

4. Bacterial foraging optimization

Introduction Based on the research of foraging behaviour of E.coli bacteria Kevin M.Passino and Liu exploited a variety of bacterial foraging and swarming behaviour, discussing how to connect social foraging process with distributed non-gradient optimization.

Steps for BFO approach

The step involving for finding the best value or best position is given below including chemotactic step, reproduction step, and elimination step.

Step1 Initialize parameter $p, S, N_c, N_s, N_{re}, N_{ed}, P_{ed}$ C(i)(i=1,2....S) α^i

Step2 Elimination-dispersal loop: l= l+1

They will break into two parts due to the presence of high temperature. Due to the sudden change in the local environment where a bacterium population lives may occur due to various reasons like change in temperature may kill a group of bacteria that are currently in a region with a high concentration of nutrient gradients.

Step 3 Reproduction loop: k=k+1

The cycle of the bacteria formation will be continuing and they will try to make the swarm size constant up to a certain value [4]. The least healthy bacteria sooner or later dies but the healthier bacteria splits up to two bacteria two maintain the swarm size constant..

Step 4 Chemotaxis loop
Chemotaxis loop: *j=j*+1

[a] For *i* =1, 2…S take a chemotactic step for bacterium *i* as the iteration for the value of swarm.

[b] Compute fitness function, J (j, k, l).

Suppose (j, k, l) i_q represents ith bacterium at jth chemotactic, kth reproductive and lth elimination-dispersal step. C (i) is the size of the step taken in the random direction specified by the tumble (run length unit).

[c] Let J_{last} = Last value J (j, k, l) to save this value since we may find a better cost via a run.

[d] Tumble: generate a random vector

[e] Move: Let

$$\Theta_{i(j+1,k,l)} = \Theta_{i(j,k,l)} + C(i)\left[\frac{d(i)}{(dT(i)d(i))^{1/2}}\right] \quad (9)$$

These results in a step of size C(i) in the direction of the tumble for bacterium i.

[f] Compute J(j+1,k,l) for finding the new position of bacterium.

[g] Swim

I. Let n=0 (which is the counter for swim length)
II. While $n<N_s$ (not gone to the higher level)

- Let n=n+1
- If J(j+1,k,l) <J_{last} (if doing better)
- J_{last} = J(j+1,k,l) and let

$$\theta_{i(j+1,k,l)} = \theta_{i(j,k,l)} + C(i)\left[\frac{d(i)}{(dT(i)d(i))^{1/2}}\right] \quad (10)$$

And use this $\theta_{i(j+1,k,l)}$ to compute the new values of J which will be iterated by their respected nature.

[h] Else let $n=N_s$. This is the end of the while statement.

Step 5 If $j<N_c$ go to step 4. In this case continue chemotaxis since the life of the bacteria is not over.

Step 6 Reproduction: Reproduction step states that the bacteria energised by taking food and it will break into two parts equally, hence there is a new birth of bacteria.

For the given k and l and for each i=1,2,......,S,

let

$$J_{health} = \sum_{j=1}^{Nc+1} J\,(j,k,l) \qquad (11)$$

J_{health} denotes the health of the bacterium during each step and each position when they are moving in (j,k,l) coordinates.

According to above equation if the bacteria with highest J_{health} values die and the remaining bacteria with best value splits.

Step 7 If $k<N_{re}$ go to step 3. In this case, we have not reached the number of specified reproduction steps, so we start the next generation of the chemotactic loop.

Step 8 Elimination Dispersal: For i=1,2,3.... S, Eliminate and Disperse each bacterium (these keeps the number of bacteria in the population constant) (S. Morkos *et al*, 2012).

.

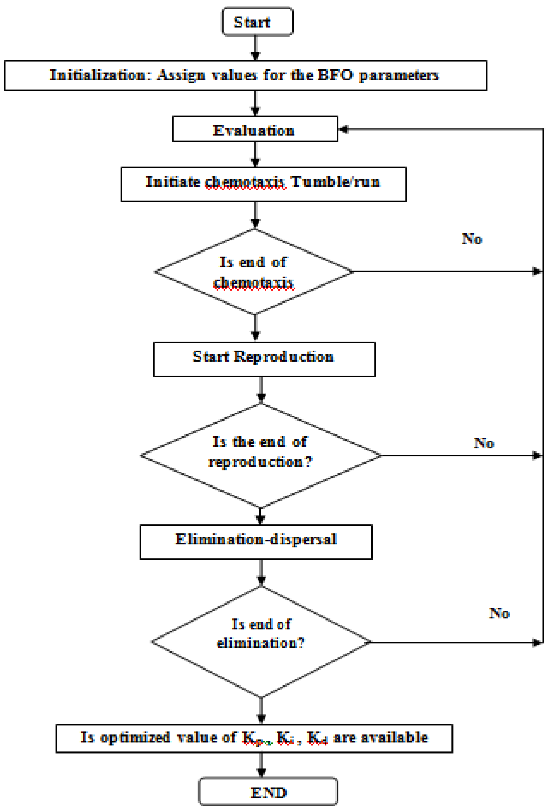
Start
Initialization: Assign values for the BFO parameters
Evaluation
Initiate chemotaxis Tumble/run
Is end of chemotaxis
No
Start Reproduction
Is the end of reproduction?
No
Elimination-dispersal
Is end of elimination?
No
Is optimized value of Kp, Ki, Kd are available
END

Fig.3 Flow diagram of BFO algorithm

5. Simulation and results

As the figure shows that manipulating the values of gain by PSO PID technique or PSO BFO technique it gives the required results. In both PSO as well as in BFO there are two important factor i.e. alpha and beta which is the base of the gain values. Alpha and Beta are the two constant terms which provides the system stability.

Matlab modelling of plant model using controller

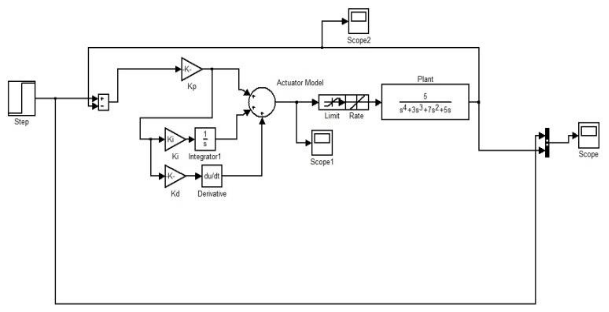

Fig.4 Matlab modelling for plant with PID controller

Functions are used for designing PID controller ISE, IAE, ITAE and ITSE .We set the following parameters

Dimension of search space =3;
The number of bacteria =10;
Number of chemotactic steps =10;
Limits the length of a swim =4;
The number of reproduction steps =4;
The number of elimination-dispersal events =2;
The number of bacteria reproductions per generation =s/2;
The probability that each bacteria will be eliminated/dispersed =0.25;
c(:,1)=0.5*ones(s,1); the run length.
We use the following PSO parameters
C1=1.2;
C2= 0.5;
W=0.9;

Output for the comparison of PSO and BFO with PID

This section describes about the comparison of PSO and BFO for tuning of PID controller.

Table 5.1 Comparison of PSO and BFO for 50 iterations with alpha = 10 and beta = 5.

S.No	Gen.	α	β	Kp	Ki	Kd	Mp%	Ts	Tr
1	50	10	5	0.96	0	0.85	22.5329	9.27	1.48
2	50	10	5	0.88	0	0.67	8.1781	8.31	1.69

The response in Fig.5.1 shows that the comparison of PSO and BFO when the number of iteration is 50 and alpha =10 and beta =5 in such a condition that the output response of the system gives the value of maximum overshoot, rise time, settling time, and also the value of the gain i.e. the proportional gain and the integral gain.

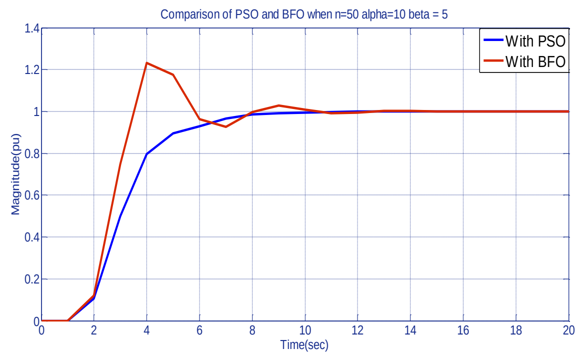

Fig. 5.1 Output response for comparison of PSO and BFO

When n=50 (i.e. number of iteration denoted by generation) in this case considering the value of Beta to be 5 and alpha to be 10, which will give the value of K_p = 0.9682 and K_d = 0.854 for PSO algorithm and similarly for BFO the value of Kp and Kd is 0.8809 and 0.6724 respectively. From the figure it is also clear that the maximum overshoot and settling time of PSO is greater than that of BFO, and the rise time is smaller than that of BFO.

Table 5.2 Comparison of PSO and BFO for 50 iterations with alpha = 10 and beta = 5

S. No	1	2
Gen.	50	50
α	10	10
β	10	10
Kp	0.7463	0.8403
Ki	0	0
Kd	0.8605	0.779
Mp%	0.663	2.3438
Ts	7.0386	8.1187
Tr	2.0542	1.8221

The response in Fig.5.2 shows that the comparison of PSO and BFO when the number of iteration is 50 and alpha =10 and beta =10 in such a condition the output response of the system gives the value of maximum overshoot, rise time, settling time, and also the value of the gain i.e. the proportional gain and the integral gain.

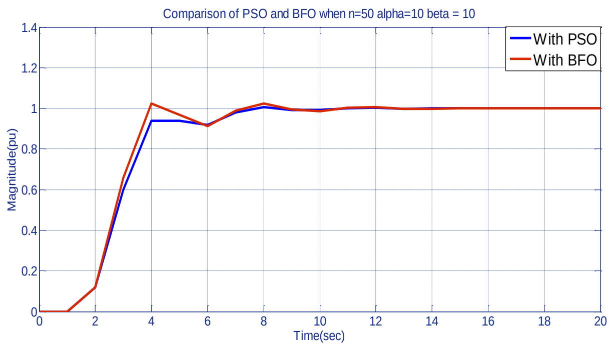

Fig. 5.2 Output response for comparison of PSO and BFO

When n=50 (i.e. number of iteration denoted by generation) in this case considering the value of Alpha to be 10 and Beta to be 10, which will give the value of Kp = 0.7463 and Kd = 0.8605 for PSO algorithm and similarly for BFO the value of Kp and Kd is 0.8403 and 0.7790 respectively. From the figure it is also clear that the maximum overshoot and settling time of BFO is greater than that of PSO, and the rise time is smaller than that of PSO.

Table 5.3 Comparison of PSO and BFO for 50 iterations with alpha = 10 and beta = 15

S.No	1	2
Gen	50	50
α	10	10
β	15	15
Kp	0.8188	0.8747
Ki	0	0
Kd	0.8333	0.8018
Mp%	1.9454	5.7273
Ts	6.89	8.31
Tr	1.88	1.74

The response in Fig.5.3 shows that the comparison of PSO and BFO when the number of iteration is 50 and alpha =10 and beta =15 in such a condition the output response of the system gives the value of maximum overshoot, rise time, settling time, and also the value of the gain i.e. the proportional gain and the integral gain.

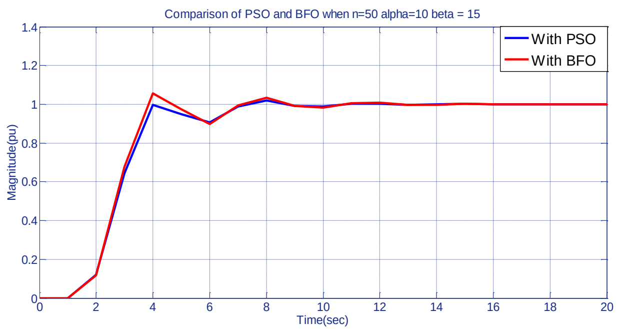

Fig. 5.3 Output response for comparison of PSO and BFO

When n=50 (i.e. number of iteration denoted by generation) in this case considering the value of Alpha to be 10 and Beta to be 15, which will give the value of K_p = 0.8188 and Kd = 0.8333 for PSO algorithm and similarly for BFO the value of K_p and K_d is 0.8747 and 0.8108 respectively.. From the figure it is also clear that the maximum overshoot and settling time of BFO is greater than that of PSO, and the rise time is smaller than that of PSO.

Table 5.4 Comparison of PSO and BFO for 50 iterations with alpha = 10 and beta = 20

S.No	1	2
Gen.	50	50
α	10	10
β	20	20
Kp	0.5631	0.9762
Ki	0	0
Kd	0.8155	0.4032
Mp%	0	23.08
Ts	7.7185	9.3541
Tr	3.2207	1.4851

The response in Fig.5.4 shows that the comparison of PSO and BFO when the number of iteration is 50 and alpha =10 and beta =20 in such a condition the output response of the system gives the value of maximum overshoot, rise time, settling time, and also the value of the gain i.e. the proportional gain and the integral gain.

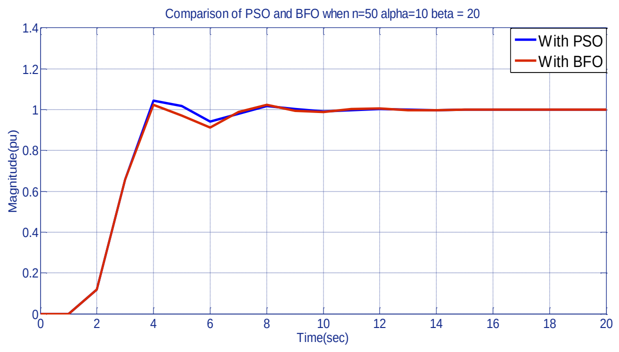

Fig. 5.4 Output response for comparison of PSO and BFO

When n=50 (i.e. number of iteration denoted by generation) in this case considering the value of Beta to be 20 and alpha to be 10, which will give the value of K_p = 0.5361 and K_d = 0.8155 for PSO algorithm and similarly for BFO the value of K_p and K_d is 0.9762 and 0.4032 respectively. From the figure it is also clear that the maximum overshoot and settling time of BFO is greater than that of PSO, and the rise time is smaller than that of PSO.

Conclusion

From the closed discussion it is seen that by applying PSO algorithm it provides optimal values for PID parameters for better system performance. Using PSO it can be seen that the best overshoot is achieved many times along with good rise time as well as settling time.

BFO algorithm is next optimization technique applied for optimization of PID parameters for stability enhancement of plant model. Overshoot, rise time and settling time are achieved in specified range but as compare to PSO it is not.

Acknowledgment

I would like to express my sincere gratitude to Mr. Mahesh Singh, Sr. Assistant Professor, SSTC (SSGI), Bhilai, Mr. Kushal Tiwari, Project Coordinator, CREDA, Raipur & Mr. Chaman Yadav, Project Officer, ECBC Cell, CREDA, Raipur who have contributed, directly or indirectly, in the accomplishment of this paper.

References

Dingyu Xuet and D.P. Atherton (Dec 1993), A reduction based pid controller design algorithm for a class of plant model,u.k. proceeding of 23 conference on decision and control.

Anitha Mary and l.Sivakumar (May 2012) A reduced order transfer function models for alstom gasifier using genetic algorithm international journal of computer applications (0975 – 8887) volume 46– no.5

Yogendra Kumar Soni and Rajesh Bhatt (March-2013) PSO optimized reduced order PID Controller designInternational Journal Of Advanced Resarch in Computer Engineering & Technology.Volume 2 Issue 3, ,pp 989-992 ISSN:2278-1323.

S. Morkos and H. Kamal (March 2012) Optimal Tuning of PID Controller using Bacteria Foraging based Particle Swarm Optimization Algorithm, Int. J. of Computers, Communications & Control, ISSN 1841 9836, Vol. VII, No. 1, pp. 101-114,

Vinay Gupta, AshisPatra (July 2011), —Design of Self-Tune PID Controller for Governor Control to Improve Dynamic Characteristics —International Conference (ICITM-2011) to be held on 3rd July2011 at Ranchi, India

S.R.Vaishnav, Z.J.Khan (2007), —Design and Performance of PID and Fuzzy Logic Controllers with Smaller Rule Set for Higher Order System‖, International Conference on Control & Automation (ICCA'07), Hong Kong, china, pp.1469-1472.

M. Zhuang and D.P. Athertong (1993), —Automatic Tuning of Optimum PID Controller‖, IEE proceeding part-D: control Theory and ApplicationVol.140, No.3, pp.216-224

K.J. Astrom and T. Haglund (1995), —PID Controller: Theory, Design and Tuning, 2nded.Research Tringle park, NC: Instrum.Soc.Amer

K.H. Ang, G. Chong and Y. Li (2005), PID Control System Analysis, Design and Technology, IEEE Transaction on Control System Technology, Vol.13, No.4, pp.559-576.

Ritu Gopal and Dr. Dharmendra Kumar Singh (June 2015) - Performance Analysis of Induction Motor For Sudden Load Disturbance Using PI And FUZZY Controller IJRITCC, Volume: 3 Issue: 6, ISSN: 2321-8169

Gaurav Singh Yadav, Amit Agrawal, Dr. Dharmendra Kumar Singh - Power Flow Problem Reduced Using Unified Power Flow Controller International Journal of Science and Research (IJSR)