PID Controller Tuning Techniques: A Review

Article · November 2012

CITATIONS
109

READS
23,800

3 authors, including:

Hari om Bansal
Birla Institute of Technology and Science Pilani
90 PUBLICATIONS **1,557** CITATIONS
SEE PROFILE

Shreeraman Ponpathirkoottam
Deutsches Forschungszentrum für Künstliche Intelligenz
8 PUBLICATIONS **182** CITATIONS
SEE PROFILE

PID Controller Tuning Techniques: A Review

Hari Om Bansal, Rajamayyoor Sharma, P. R. Shreeraman

Electrical and Electronics Engineering Department, Birla Institute of Technology and Science

Pilani, India

hobansal@gmail.com

***Abstract*- This paper presents a review of the current as well as classical techniques used for PID tuning. PID controllers have been used for industrial processes for long, and PID tuning has been a field of active research for a long time. The techniques reviewed are classified into classical techniques developed for PID tuning and optimization techniques applied for tuning purposes. A comparison between some of the techniques has also been provided. The main goal of this paper is to provide a comprehensive reference source for people working in PID controllers.**

Keywords- PID Controllers; Tuning; Classical Techniques; Intelligent Computational Techniques

I. INTRODUCTION

Proportional Integral and Derivative (PID) controllers have been used in industrial control applications for a long time. PID controllers date to 1890s governor design [1]-[2]. Despite having been around for a long time, majority of industrial applications still use PID controllers. According to a survey in 1989, 90% of process industries use them [3]. This widespread use of PID in industry can be attributed to their simplicity and ease of re-tuning on-line [4].

The PID controller is so named because its output sum of three terms, proportional, integral and derivative term. Each of these terms is dependent on the error value e between the input and the output,

$$\text{output} = K_p \times e(t) + K_i \times \int_0^t e(t)\,dt + K_d \times \frac{de}{dt} \quad (1)$$

where K_p, K_i and K_d are the P, I and D parameters respectively. K_i and K_d can also be written as,

$$K_i = K_p \times T_i, K_d = K_p \times \frac{1}{T_d} \quad (2)$$

where T_i and T_d are reset time and derivative time respectively. These terms determine the type of system response. The properties of P, I and D are discussed briefly here.

Proportional term: This term speeds up the response as the closed loop time constant decreases with the proportional term but does not change the order of the system as the output is just proportional to the input. The proportional term minimizes but does not eliminate the steady state error, or offset.

Integral term: This term eliminates the offset as it increases the type and order of the system by 1. This term also increases the system response speed but at the cost of sustained oscillations.

Derivative term: This term primarily reduces the oscillatory response of the system. It neither changes the type and order of the system nor affects the offset.

A change in the proportionality constants of these terms changes the type of response of the system. That is why PID tuning, which is the variation of the PID proportionality constants, is of utmost importance. This paper talks about the different types of PID tuning techniques implemented and the comparison between some of them.

There have been various types of techniques applied for PID tuning, one of the earliest being the Ziegler Nichols technique. These techniques can be broadly classified as classical and computational or optimization techniques.

A. Classical Techniques

Classical techniques make certain assumptions about the plant and the desired output and try to obtain analytically, or graphically some feature of the process that is then used to decide the controller settings. These techniques are computationally very fast and simple to implement, and are good as a first iteration. But due to the assumptions made, the controller settings usually do not give the desired results directly and further tuning is required. A few classical techniques have been reviewed in this paper.

B. Computational or Optimization Techniques

These are techniques which are usually used for data modeling and optimization of a cost function, and have been used in PID tuning. Few examples are neural networks (computational models to simulate complex systems), genetic algorithm and differential evolution. The optimization techniques require a cost function they try to minimize. There are four types of cost functions used commonly,

- Integral Absolute Error

$$\text{IAE} = \int_0^T |e(t)| \quad (3)$$

- Integral Square error

$$\text{ISE} = \int_0^T |e(t)|^2 \quad (4)$$

- Integral Time Absolute Error

$$\text{ITAE} = \int_0^T t|e(t)| \quad (5)$$

- Integral Time square Error

$$\text{ITSE} = \int_0^T t|e(t)|^2 \quad (6)$$

Computational models are used for self tuning or auto tuning of PID controllers. Self tuning of PID controllers essentially sets the PID parameters and also models the

process by using some computational model and compares the outputs to see if there are any process variations, in which case the PID parameters are reset to give the desired response.

The existent types of adaptive techniques are classified based on the fact that if the process dynamics are varying [5], then the controller should compensate these variations by adapting its parameters. There are two types of process dynamics variations, predictable and unpredictable. The predictable ones are typically caused by nonlinearities and can be handled using a gain schedule, which means that the controller parameters are found for different operating conditions with an auto-tuning procedure that is employed thereafter to build a schedule. Different techniques have been used to replace the gain schedule mentioned above. In the discussion of various techniques its usage in self tuning is also mentioned.

II. CLASSICAL TECHNIQUES

Most classical techniques make assumptions of the plant model and try to derive the controller settings for these general models. To determine the dynamics of these systems, the step response of the systems are obtained. This response is characterized by different equations, using which different classical methods have been developed. Ziegler and Nichols [6] proposes that many industrial processes have step response as given in the Fig. 1. Where K being the static gain, θ the dead time and τ_1 the time constant. The parameter a is determined by the intercept of the line (tangent to the graph) with the y-axis and θ the x intercept.

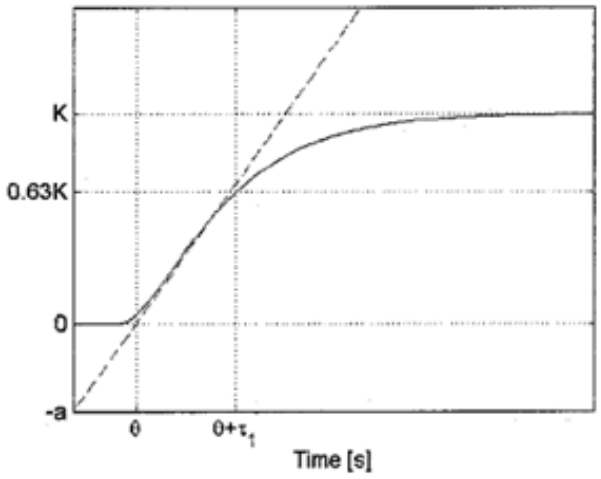

Fig. 1 Step response of first order system[6]

Various methods used for PID tuning are discussed in the following sections.

A. Ziegler Nichols Method

This is by far the most popular tuning method in use. It was proposed by John Ziegler and Nathaniel Nichols [6] in 1942 and is still a simple, fairly effective PID tuning method. There are two methods proposed by Ziegler and Nichols. The proposed Ziegler Nichols setting is given in Table I. This method was used to tune PID controllers for spindle motor systems [8].

TABLE I DETERMINATION OF PARAMETERS

Controller	K_p	T_i	T_D
P	1/a	-	-
PI	0.9/a	3.33 θ	-
PID	1.2/a	2 θ	0.5 θ

The second method is based on knowledge of the response to specific frequencies. The idea is that the controller settings can be based on the most critical frequency points for stability. This method is based on experimentally determining the point of marginal stability. This frequency can be found by increasing the proportional gain of the controller, until the process becomes marginally stable. These two parameters define one point in the Nyquist plot. The gain is called ultimate gain K_u and the time period T_p. The PID parameter setting is given in [6].

The Ziegler and Nichols method is the first PID tuning techniques made and they are made based on certain controller assumptions. Hence, there is always a requirement of further tuning; because the controller settings derived are rather aggressive and thus result in excessive overshoot and oscillatory response. Also for the first method the parameters are rather difficult to estimate in noisy environment. In the second method, as the system is driven towards instability for determining the parameters, practically this can be quite detrimental to the system.

B. Cohen Coon Method

Cohen and Coon [9] design a method with the PID controller parameters decided based on a FOLPD model. The main design requirement is the rejection of load disturbances. The controller parameter settings are given in [9].

Despite a better model, the results of the Cohen Coon method are not much better than the Ziegler Nichols method.

III. COMPUTATIONAL AND INTELLIGENT OPTIMIZATION TECHNIQUES

The various intelligent optimization techniques are discussed below.

A. Immune Algorithm

Artificial Immune Systems (AIS) are computational systems inspired by the principles and processes of the vertebrate immune system, which learns about the foreign substances to defend the body against them.

The immune system has two types of responses, primary and secondary. The former is the response when it first encounters the antigen. In this period, the system learns about the antigen, creating a memory of it. The later occurs when the antigen is encountered for the second time, which is a more rapid and larger response. The cells primarily involved in this system are B cells. Against the antigen, the level to which a B cell is stimulated relates partly to how well its antibody binds the antigen.

In [10] the algorithm is specified in detail. The input data or reference level is taken as the antigen invading the

system, and then the system responds and learns till the required solution is generated. In [10], the author has used an immune algorithm to tune a PID controller based on gain and phase margins. In [11], Kim and Chu used the algorithm for disturbance function based tuning. The immune algorithm is also used for auto tuning of PID controllers in [12].

B. Ant colony Optimization

Ant Colony Optimization (ACO) [13-14] is a recently developed meta-heuristic approach to solving optimization problems based on working of an ant colony. More precisely, it is based on the ant colony finding the shortest path to the food. Each ant tries to find the food through some random path leaving behind a trail of pheromones. The pheromone trail weakens with reducing no. of ants passing through that path and strengthens with increasing no. of ants passing through it. So basically it is a search algorithm which depends on a number of ants acting together moving towards the optimal solution.

According to Ibtissem Chiha et. al [15] the ants are driven by a probability rule to choose their solution to the problem. The probability rule between two nodes i and j, depends on two factors,

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{h\in S}[\tau_{ih}]^{\alpha}[\eta_{ih}]^{\beta}} \quad (7)$$

The factor η_{ij} is the inverse of the cost function. This factor does not change during algorithm execution; instead the factor τ_{ij} (related to pheromone which has an initial value τ_0) is updated after each iteration. The parameters α and β enable the user to direct the algorithm search in favour of the heuristic or the pheromone factor. The change in pheromone quantity in each path is given by,

$$\Delta\tau_{ij}^{A} = \begin{cases} \frac{L^{min}}{L^{A}}, & if\ i,j \in T^{A} \\ 0, & else \end{cases} \quad (8)$$

L^A is the solution of the ant A and L^{min} is the best solution found so far. The pheromone for the next iteration is decided as,

$$\tau_{ij}(t) = \rho\tau_{ij}(t-1) + \sum_{A=1}^{NA}\Delta\tau_{ij}{}^{A}(t) \quad (9)$$

NA being the number of ants, ρ being the evaporation rate, designed to allow elimination of bad choices.

Ant colony optimization was used for PID tuning in [15]. It was used to minimize a multi-objective function and its results were found to be better than genetic algorithm and Ziegler Nichols method. In [16] authors have demonstrated the use of bees algorithm to tune a PID controller and solving complex systems. The results of ACO, PSO and bees algorithm are compared and presented in [17].

C. Bacteria Forage Technique

Since the selection behavior of bacteria tends to eliminate entities with poor foraging strategies and favor the propagation of genes of those that have successful foraging strategies, they are applied to find an optimal solution through methods for locating, capturing, and ingesting food [18]. Foraging theory is discussed in [19].

All papers on PID tuning with bacteria foraging technique [18], [20], [21] study the foraging behavior of E. Coli, a common bacteria [22]-[23]. The behavior of E.Coli is described in [18] as,

- If in a neutral medium, it alternates between tumbles and runs and searches the environment.
- If swimming up a nutrient gradient (or out of noxious substances) or swimming longer (up a nutrient gradient or down a noxious gradient) it seeks an increasingly favorable environment.
- If swimming down a nutrient gradient (or up a noxious substance gradient), it searches to avoid unfavorable environments.

E. coli occasionally engages in a conjugation that affects the characteristics of a population of bacteria. There are attractants that bacteria like, attraction to oxygen (aerotaxis), light (phototaxis), temperature (thermotaxis), and magnetotaxis (it is affected by magnetic lines of flux). Some bacteria change their shape and number of flagella based on the medium to reconfigure and ensure efficient foraging in a variety of media. The main goal based on bacterial foraging is to find the best position of the bacteria with respect to the attractant and repellent profile. A hybrid approach consisting of genetic algorithm and bacteria forage for tuning of PID controller for AVR system is proposed in [21].

D. Genetic Algorithm

Genetic algorithm (GA) is a search algorithm that explores the search space in a manner analogous to evolution in nature [24]. It uses probabilistic rules to search for and change the potential solutions in the search space, using a cost function to analyze the fitness of solutions. GA requires the solution to be represented in a way that is analogous to genes so that the processes that bring about a change in the genes (like mutation) can be used. Usually this is done by representing the solutions in a binary format.

The standard genetic algorithm is given below and flowchart of the algorithm is shown in Fig. 2.

- Initialization, firstly initial solutions are randomly selected from the search space.
- Selection, during each iteration, a proportion of solutions is selected, based on the fitness function (fitter solutions are more likely to get selected), for breeding the next generation of solutions. The selection is done in a probabilistic manner.
- Reproduction, selected solutions are paired up and crossover and mutation operation are performed to get the next generation of solutions.
- Termination, the iterations are terminated when the termination condition (time or accuracy) is reached.

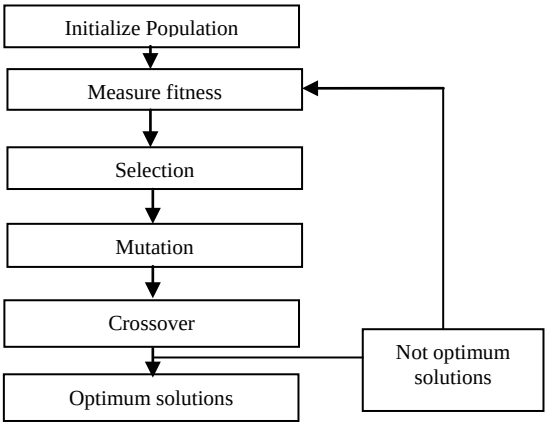

Fig. 2 Flowchart of genetic algorithm based tuning

GA is very popular in PID tuning, and has gained wide applications in control systems [25]. Girishraj *et. al* [25] used GA for improving performance of a PID controller used in bioreactor and compared the performance with Ziegler Nichols, Skogestad modification [26] and IMC rule [27] and found that GA outperformed both in terms of overshoot, disturbance rejection, gain margin and phase margin. The limitations of GA in tuning a multivariable system were explored in [28]. GA has been used in position and speed control of a DC motor [29]-[30]. GA has been used for PID of reverse osmosis and cascade control systems tuning in [31]-[33].

Lot of work has been done in using GA along with other computational techniques. In [21], Kim *et al*. use bacteria forage along with GA for PID controller tuning of AVR systems. GA was used with NN in [34] and with fuzzy logic in [35] for developing self tuning methods.

E. Differential Evolution

Differential Evolution (DE) is a method for doing numerical optimization without explicit knowledge of the gradient of the problem to be optimized. The DE method is originally due to Storn and Price and works on multidimensional real-valued functions which are not necessarily continuous or differentiable. DE optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its simple formulae of vector-crossover and -mutation, and then keeping whichever candidate solution has the best score or fitness on the optimization problem at hand. In this way the optimization problem is treated as a black box that merely provides a measure of quality given a candidate solution and the gradient is therefore not needed. Differential evolution is used for online PID tuning in [36].

F. Evolutionary Programming

Generally, the EP algorithm for global optimization contains four parts, initialization, mutation, competition, and reproduction. Mutation is based on the current values and a Gaussian random variable. Furthermore, a quasi-random sequence (QRS) is used to generate an initial population for EP [37] to avoid causing clustering around an arbitrary local optimum [37]. Evolutionary programming was used in [38] for PID tuning using IAE and compared with results of [39], [40], [41], and [42] which were fuzzy logic based and the results were superior for evolutionary programming and the same as results with a genetic algorithm.

G. Artificial Neural Networks

An Artificial Neural Network (ANN), usually called 'Neural Network' (NN), is a mathematical model or computational model that tries to simulate the structure and/or functional aspects of biological neural networks. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase.

Though ANN can model even highly non linear systems, it is not used in control due to limited applicability in PID controllers [34], partially because the neural network control design has some drawbacks because of some intrinsic shortcomings of ANN theory, e.g., the number of layers and the numbers of neurons per layer are often hard to be determined. According to [43] SVM based self tuning controller is simpler to implement.

Almost all work done using neural networks is for self tuning PID controllers. The flow chart used in [34] for self-tuning PID Control using GA and ANN is shown in Fig. 3.

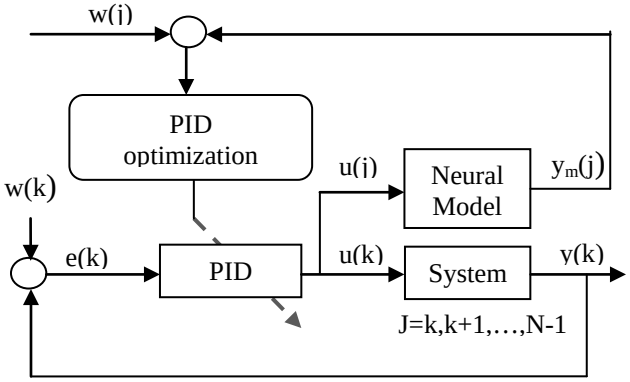

Fig. 3 Flowchart of neural network based controller

Similarly, [44] uses neural networks for self-tuning discrete PID controller, and compares to a relay method proposed by Astrom and Hagglund [4] and obtains much better results. In [43], Iplikci compared the results of self tuning PID controller using neural networks and SVM, and found that neural network based controller gave better results in noiseless environments and SVM performed better in noisy conditions. A NN-like self-tuning PID control scheme applied in the motion control of a Two Wheeled Vehicle (TWV) is presented in [45].

H. Simulated Annealing

Simulated Annealing (SA) simulates the process of heating a metal above its recrystallisation temperature, and cooling it to change the metal's properties. The optimization

technique involves perturbation of the design variables and then observing the change in the objective function. If the solution is better than the current solution, the design variables are updated and the new solution is accepted according to the Metropolis algorithm [46] based on Boltzmann probability. The perturbations keep reducing according to some reduction constant. The algorithm ends when the desired solution to the objective function is reached or the perturbations are too small for significant change in the objective function. A flow chart of the algorithm is shown in Fig. 4.

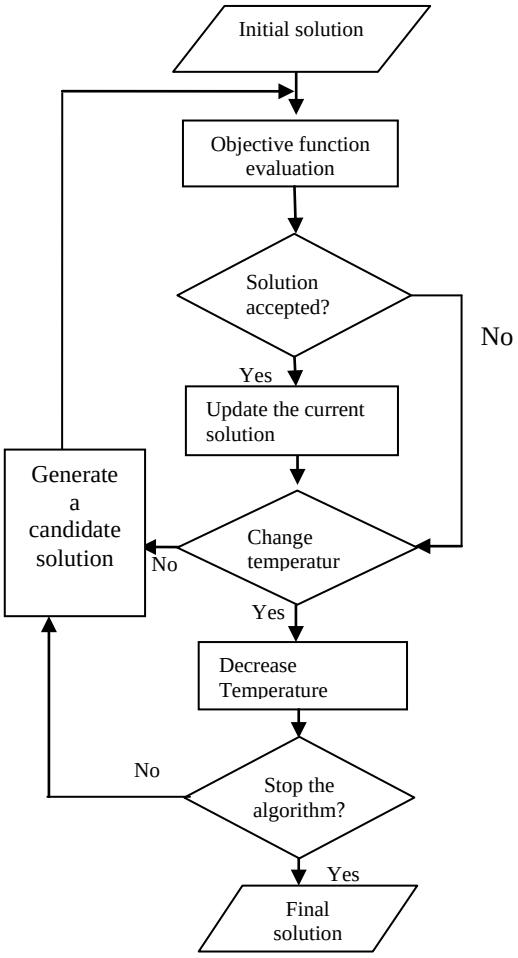

Fig. 4 Flowchart of standard simulated annealing

Simulated annealing has been used in PID tuning in [47] for controllers for time-delay Systems for a high-performance drilling process. The standard algorithm for SA is computationally very intensive due to an extremely large and nonlinear, multimodal search space [48]. So a modified version of SA named Orthogonal Simulated Annealing (OSA) is used in [49] for simultaneous optimization of multiple fuzzy neural networks using in PID tuning of various controllers.

I. Support Vector Machine

Support Vector Machine (SVM) is a set of related supervised learning methods used for classification and regression. In simple words, given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. More formally, a SVM constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, which can be used for classification, regression or other tasks.

As put in [50], for the training sample $\mathbf{x}_i \in R$ belonging to the class $y_i \in \{-1, 1\}$, each class can be linearly separated and the discrimination function is described as follows,

$$f(x_i) = w^T x_i + b, \tag{10}$$

where **w** is an adjustable weight vector and *b* is the bias. The separating hyperplane is the plane with $f(\mathbf{x}) = 0$, and needs to satisfy the following constraints such that the hyperplane may be uniquely determined for all training data.

$$y_i.\left((w^T x_i) + b\right) \geq 1 \quad (i = 1,2,\ldots,l) \tag{11}$$

The distance between the hyperplane and the nearest training points (called margin) has to be maximized to obtain the best separation.

SVM was used for PID tuning in [50] by dividing the range of uncertainty of the parameters and then using multiple SVM (decision tree structure with SVMs). Iplikci [43] uses SVM in obtaining the nonlinear autoregressive with exogenous inputs (NARX) model of the plant for both PID parameter tuning and also correction of the PID output during control, which was compared with Neural Networks used for the same. In [51] Least Squares SVM (LS-SVM) is used to design a self tuning PID. It is an extension of the standard SVM, which involves equality instead of inequality constraints in SVM and works with a least squares cost function [52]. Zhao *et al.* [51] also states that this design does not have the drawbacks which are present in the neural network design, present due to some intrinsic shortcomings of ANN theory, e. g., the number of layers and the numbers of neurons per layer often being hard to determine.

J. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a popular optimization technique developed by Eberhart and Kennedy in 1995. In the technique, there is a population of particles which move through the solution space to find the optima. In PSO technique the system keeps a track of the best solution obtained till now and each individual particle keeps a track of its own individual best solution. Based on these two, each particle moves to a new position decided by a velocity and its current position. The velocity is dependent on the global and particle's best solution. As put in [53],

If the *i*-th particle of the swarm is represented by the *D*–dimensional vector *Xi* = (*xi*1, *xi*2,..., *xiD*) and the best particle in the swarm, i.e. the particle with the smallest function value, is denoted by the index *g*. The best previous position (the position giving the best function value) of the *i*-th particle is recorded and represented as *Pi* =(*pi*1, *pi*2,..., *piD*), and the position change (velocity) of the *i*-th particle is *Vi* =(*vi*1, *vi*2 ,..., *viD*). The particles are manipulated according to the equations,

$$v_{id} = w.v_{id} + c_1.r_1.(p_{id\_} - x_{id}) + c_2.r_2.(p_{gd\_} - x_{id}) \tag{12}$$

$$x_{id} = x_{id} + v_{id} \tag{13}$$

where d = 1, 2, . , D; i = 1, 2,., N and N is the size of population; w is the inertia weight; $c1$ and $c2$ are two positive constants and $r1$ and $r2$ are random values in the range [0, 1].

This is one of the ways in which the PSO algorithm can be developed. PSO has been used in PID tuning in [53] for a ball and hoop system with ISE as the cost function, where its performance was compared to Genetic algorithm and Ziegler & Nichols and it was found to outperform both of them in terms of settling time, overshoot, and equaled genetic algorithm based tuning in rise time. In [17], PSO was compared with ant colony optimization and bee's algorithm for all four cost functions, and PSO obtained the best fitness value (better rise time and settling time) for all cost functions, though the best overshoot was in case of ant colony optimization in all cases. PSO also was the fastest among the three. Giriraj Kumar *et al.* [54] also uses PSO based tuning for high performance drilling systems.

Maolong [55] consider social behavior too complex to be described by PSO. Quantum-behaved particle swarm optimization (QPSO) is a stochastic optimization algorithm that was originally motivated by the thinking model of an individual of the social organism [55]. QPSO gave even better results than PSO, GA and Ziegler Nichols on accounts of rise time, overshoot and settling time [55]. PSO has also been used for self tuning PID in [56] and when compared with real coded genetic algorithm (RGA), it produced better results.

K. Fuzzy Logic

Fuzzy logic control is one of the interfaces between control engineering and artificial intelligence. The Fuzzy logic controller (FLC) adds to the conventional PID controller to adjust the parameters of the PID controller on-line according to the change of the signals error and change of the error. The design specifications of the FLC vary with the plant being used and the PID controller parameter ranges in combination with which it is to be used. The basic building block of the controller remains similar. Fig. 5 shows the commonly used FLC and its role in the PID tuning as described in [57].

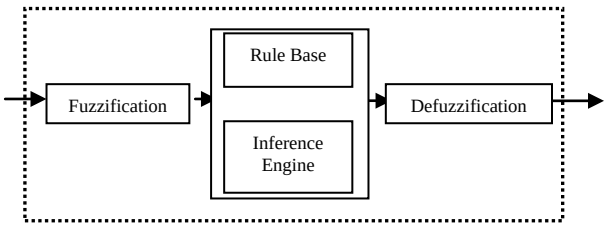

Fig. 5 Basic block diagram for fuzzy control

As shown in Fig. 6, the error and derivative of the error are inputs to the fuzzy interface. The model most commonly employed in the fuzzy interface is the Mamdani model, as defined in [58]. The operation of the Mamdani rule base can be broken down into four parts,

1) Mapping each of the crisp inputs into a fuzzy variable (fuzzification);
2) Determining the output of each rule given its fuzzy antecedents;
3) Determining the aggregate output(s) of all fuzzy rules;
4) Mapping the fuzzy output(s) to crisp output(s) (defuzzification).

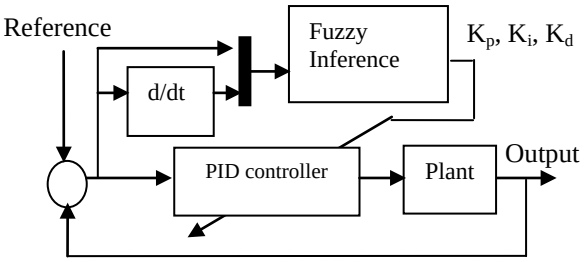

Fig. 6 Flow chart of a FLC based controller

The fuzzy rules depend on the plant to be controlled and the type of the controller and from practical experience [59]. Jantzen [60] states that integral rule bases might be difficult to design. Hence, PD fuzzy based system along with integral error control is an ideal design. There exist also designs where a PI-like fuzzy controller and PD –like fuzzy controllers are used in cohesion to achieve a PID-like controller.

L. Response Surface Method

The response surface methodology (RSM) is a collection of mathematical and statistical techniques that are useful for modeling and analysis in applications, where a response interest is influenced by several variables and the objective is to optimize this response. RSM has been developed by Box and Wilson (1951) to explore the potential of statistical design in industrial experiments. This methodology has gained extensive application in a wide variety of industrial processes. RSM constructs polynomial approximations to build functional relationships between design variables and performances. The input variables are sometimes called independent variables or factors, and the performance measures or quality characteristics are called as responses. For the most response surface, the relationship between the response variable of interest (y) and the factors (x_1, x_2, . . . ,x_k) may be described in the following second-order equation.

$$y = f(x) = \beta_o + \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1}^{n} \beta_{ii} x^2{}_i + \sum_{i=1,\, j>1}^{n} \beta_{ij} x_i x_j + \varepsilon \qquad (14)$$

where ε represents the noise or error observed in the response y, β is polynomial coefficient, and n is the number of factors. Eq. (14) can also be expressed in a matrix form as

$$Y = XB + \varepsilon \qquad (15)$$

The method of least squares is used to estimate the polynomial coefficients in approximating polynomials such that the sum of squares of the errors is minimized. Then matrix B of polynomial coefficients can be obtained from the formula

$$B = \left(X^T X\right)^{-1} X^T Y \qquad (16)$$

The response surface analysis is done in terms of the fitted surface. Once a response surface model is obtained, statistical analysis technique such as analysis of variance (ANOVA) can be used to check the fitness of the model. The tuning of a PID controller using RSM is explained in [61-64].

IV. CONCLUSIONS

A large number of techniques used for PID tuning were reviewed in this paper. A brief description of the technique was followed by discussion of the work done in tuning and self tuning of PID controller using the technique. If some comparative analysis was carried out, it was mentioned and the results of the comparison were also elucidated. A comprehensive comparative study of all the techniques, tested simultaneously under different conditions, still needs to be conducted to gauge the comparative performance of different techniques on a common platform.

REFERENCES

[1] S. Bennett, A History of Control Engineering 1930-1955, IEE Control Engineering Series 47, 1993.

[2] S. Bennett, "Nicholas Minorsky and the Automatic Steering of Ships," IEEE Control Systems Magazine, vol. 4, no .4, pp. 10–15, 1984.

[3] M. Araki, PID Control in Control systems, Robotics and Automation, vol II, edited by Heinz Unbehauen, Encyclopedia of Life Support Systems (EOLSS), Developed under the Auspices of the UNESCO, Eolss Publishers, Oxford ,UK.

[4] K. J. Astrom and T. Hagglund, PID controllers, Theory, design and tuning, 2nd edition, Instrument Society of America, 1995.

[5] T. Hagglund and K. J Astrom, "Industrial Adaptive Controllers Based on Frequency Response Techniques," Automatica, vol. 27, no. 4, pp. 599-609, 1991.

[6] J. G. Ziegler and N. B Nichols, "Optimum Settings for Automatic Controller," Transaction of ASME, vol. 64, pp. 759-768, 1942.

[7] J. C. Cool, F. J. Schijff, T. J. Viersma, and Regeltechniek, Control Engineering, Delta Press, Amerongen, 1991.

[8] C. S. Soh, C. Bi and K. C. Chua, "Direct PID Tuning For Spindle Motor Systems," Asia-Pacific Magnetic Recording Conference, Singapore, pp. 1-2, Nov-Dec.2006.

[9] G. H. Cohen, and G. A. Coon, "Theoretical considerations of retarded control," Transactions of ASME, vol. 75, pp. 827-834, 1953.

[10] Dong Hwa Kim, Tuning of PID Controller Using Gain/Phase Margin and Immune Algorithm, IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications, Helsinki University of Technology, Espoo, Finland, pp. 6-74, 2005.

[11] Dong Hwa Kim and Jae Hoon Cho, "Intelligent Tuning of PID Controller with Disturbance Function Using Immune Algorithm," International Conference on Computational Intelligence for Measurements and Applications, vol. 1, pp. 286-291, 2004.

[12] Shi Zhongzhi, Shi Zhongzhi, Shi Zhongzhi, "Auto-tuning of reference model based PID controller using immune algorithm," Proceedings of the World on Congress on Computational Intelligence, pp. 483-488, 2002.

[13] M. Dorigo and G. Dicaro, The Ant Colony Optimization Meta-heuristic, In New Ideas in Optimization, McGraw Hill, London, UK, pp. 11–32, 1999.

[14] M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant Algorithms for Discrete Optimization," Artificial Life, vol. 5, no. 2, pp. 137–172, 1999.

[15] Ibtissem Chiha, Noureddine Liouane and Pierre Borne, "Multi-Objective Ant Colony Optimization to tuning PID controller," International Journal of Engineering, vol. 3, no. 2, pp. 11-16, 2009.

[16] D. T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, and M. Zaidi, "The Bees Algorithm, A Novel Tool for Complex Optimisation Problems," Proc. 2nd Virtual International Conference on Intelligent Production Machines and Systems, Elsevier (Oxford), pp. 454-459, 2006.

[17] Karl O. Jones, andré Bouffe, Comparison of bees algorithm, ant colony optimization and particle swarm optimization for PID controller tuning, Proceedings of the 9th international Conference on Computer Systems and Technologies and Workshop For PhD Students in Computing, CompSysTech 08, 374, ACM, New York, IIIA. 9-1 – IIIA. 9-6.

[18] Dong Hwa Kim, and Jae Hoon Cho, "Robust Tuning of PID Controller Using Bacterial-Foraging- Based Optimization," Journal of Advanced Computational Intelligence and Intelligent Informatics, vol. 9, no. 6, pp. 669-676, 2005.

[19] Stephens, D.W. and Krebs J. R., Foraging Theory, Princeton University Press, Princeton, New Jersey, 1986.

[20] Dong Hwa Kim1 and Jae Hoon Cho, "Adaptive Tuning of PID Controller for Multivariable System Using Bacterial Foraging Based Optimization," Atlantic Web Intelligence Conference, pp. 231-235, 2005.

[21] Dong Hwa Kim, Jae Hoon Cho, "A Biologically Inspired Intelligent PID Controller Tuning for AVR Systems," International Journal of Control, Automation, and Systems, vol. 4, no. 5, pp. 624-636, 2006.

[22] K. M. Passino, Biomimicry of Bacterial Foraging for Distributed Optimization, University Press, Princeton, New Jersey, 2001.

[23] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," IEEE Control Systems Magazine, vol. 22, no. 3, pp. 52-67, 2002.

[24] M. Salami and G. Cain, "An Adaptive PID Controller Based on Genetic Algorithm Processor," IEE Genetic Algorithms in Engineering Systems, Innovations and Applications Conference, Publication No. 414, 12-14 September, 1995.

[25] S. M. G. Kumar, R. Jain, and N. Anantharaman, "Genetic Algorithm Based PID Controller Tuning for a Model Bioreactor," Indian Chemical Engineer, vol. 50, no. 3, pp. 214-226, 2008.

[26] S. Skogestad, "Simple Analytic Rules for Model Reduction and PID Controller tuning," Journal of Process Control, vol.13, pp. 291-309, 2003.

[27] B. W. Bequette, Process Control, Modeling, Design, and Simulation, Prentice Hall, Inc., New Jersey, 2003.

[28] Karl O. Jones and Wilfried Hengue, Limitations of multivariable controller tuning using genetic algorithms, International Conference on Computer Systems and Technologies – CompSysTech 2009, IIIA.20-1 – IIIa.20.5.

[29] Neenu Thomas, P. Poongodi, "Position Control of DC Motor Using Genetic Algorithm Based PID Controller," Proceedings of the World Congress on Engineering, London, U.K., vol. 2 , pp. 1618-1622, 2009.

[30] M. B. B. Sharifian, R. Rahnavard and H. Delavari, "Velocity Control of DC Motor Based Intelligent methods and Optimal

Integral State Feedback Controller," International Journal of Computer Theory and Engineering, vol. 1, no. 1, pp. 1793-1801, 2009.

[31] Cătălin Nicolae CALISTRU, "PID Robust Control via Symbolic Calculus and Global Optimization Techniques," 7th International Conference on Development and Application Systems, pp. 313-318, 2004.

[32] M. V. Sadasivarao and M. Chidambaram, "PID Controller tuning of cascade control systems using genetic algorithm," Journal of Indian Inst. Sci., vol. 86, pp. 343–354, 2006.

[33] Jin-Sung Kim, Jin-Hwan Kim, Ji-Mo Park, Sung-Man Park, Won-Yong Choe and Hoon Heo, "Auto Tuning PID Controller based on Improved Genetic Algorithm for Reverse Osmosis Plant," International Journal of Computer Systems Science and Engineering, vol. 3, no. 4, pp. 232 – 237, 2008.

[34] DOLEŽEL, Petr, MAREŠ, Jan, "Self-tuning PID Control using Genetic Algorithm and Artificial Neural Networks," ASR 2009 Instruments and Control, pp. 33- 39, 2009.

[35] Abdel Badie Sharkawy, "Genetic fuzzy self-tuning PID controllers for Antilock Braking Systems," Alexandria Engineering Journal, vol. 45, no. 6, pp.657-673, 2006.

[36] Fei Gao, Hengqing Tong, "Differential Evolution, An Efficient Method in Optimal PID Tuning and Online Tuning," Proc. of the International Conference on Complex Systems and Applications, pp. 785-789, 2006.

[37] Y. J. Cao, "Eigenvalue Optimisation Problems via Evolutionary Programming," Electronics Letters, vol. 33, no.7, pp. 642-643, 1997.

[38] Wei-Der Chang and Jun-Juh Yan, "Optimum setting of PID controllers based on using evolutionary programming algorithm," Journal of the Chinese Institute of Engineers, vol. 27, no. 3, pp. 439-442, 2004 .

[39] S. Z. He, S. Tan and F. L. Xu, "Fuzzy Self- Tuning of PID Controllers," Fuzzy Sets and Systems, vol. 56, no. 1, pp. 37-46, 1993.

[40] S. G. Tzafestas and N. P. Papanikolopoulos, "Incremental Fuzzy Expert PID Control," IEEE Transactions on Industrial Electronics, vol. 37, no. 5, pp. 365-371, 1990.

[41] A.Visioli, "Fuzzy Logic Based Set-Point Weighting for PID Controllers," IEEE Transactions on Systems, Man, and Cybernetics-Part A, Systems and Humans, vol. 29, no. 6, pp. 587- 592, 1999.

[42] C. Liu, J. X Xu, and C. C. Hang, "Comparison between a Fuzzy PID Controller and a Kind of Nonlinear PID Controller," Proceedings of 36th IEEE International Conference on Decision and Control, vol. 3, pp. 2736-2741, 1997.

[43] S. Iplikci, "A comparative study on a novel model-based PID tuning and control mechanism for nonlinear systems," International Journal of Robust and Nonlinear Control, vol. 20, no. 13, pp. 1483-1501, 2010.

[44] Corneliu Lazar, Sorin Carari, Draguna Vrabie, Marius Kloetzer, "Neuro-predictive control based self-tuning of PID controllers," Proceedings of European Symposium on Artificial Neural Networks, pp. 391-396, 2004.

[45] Tsai-Juin Ren, Tien-Chi Chen and Chun-Jung Chen, "Motion control for a two-wheeled vehicle using a self-tuning PID controller," Control Engineering Practice, vol. 16, no. 3, pp. 365-375, 2008.

[46] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A. H. Teller and E.Teller, "Equation of State Calculation using Fast Computing Machines," Journal of Chemical Physics, vol. 21, pp. 1087–1092, 1953.

[47] Rodolfo E. Haber, Rodolfo Haber-Haber, Raúl M. del Toro and José R. Alique, Using Simulated Annealing for Optimal Tuning of a PID Controller for Time-Delay Systems: An Application to a High-Performance Drilling Process, Computational and Ambient Intelligence, Springer, vol. 4507/2007, pp. 1155-116 2, 2007.

[48] X. Tang, R. Tian and D. F. Wong, "Fast evaluation of sequence pair in block placement by longest common subsequence computation," IEEE Transactions on Computer Aided Design Integrated Circuits Systems, vol. 20, no.12, pp. 1406–1413, 2001.

[49] Shinn-Jang Ho,Li-Sun Shu, and Shinn-Ying Ho, "Optimizing Fuzzy Neural Networks for Tuning PID Controllers Using an Orthogonal Simulated Annealing Algorithm OSA," IEEE Transaction on Fuzzy Systems, vol. 14 , no. 3, pp. 421-434, 2006.

[50] K. Takao, Y. Toru and H. Takao, "A Design of PID Controllers with a Switching Structure by a Support Vector Machine," International Joint Conference on Neural Networks, pp. 4684 – 4689, 2006.

[51] Jun Zhao, Ping Li, Xue-song Wang, "Intelligent PID Controller Design with Adaptive Criterion Adjustment via Least Squares Support Vector Machine," Control and Decision Conference, pp. 7-9, 2009.

[52] J. A. K. Suyken and J. Vandewalle, "Least Squares Support Vector Machine Classifiers," Neural Processing Letters, vol. 9, no. 3, pp. 293-300, 1999.

[53] Mohammed El-Said El-Telbany, "Employing Particle Swarm Optimizer and Genetic Algorithms for Optimal Tuning of PID Controllers, A Comparative Study," ICGST-ACSE Journal, vol. 7, no. 2, pp. 49-54, 2007.

[54] S. M. G. Kumar, J. Deepak and R. K. Anoop, "PSO based tuning of a PID controller for a High performance drilling machine," International Journal of Computer Applications, vol. 1, no. 19, pp. 12-18, 2010.

[55] Maolong Xi, Jun Sun and Wenbo Xu, "Parameter Optimization of PID Controller Based on Quantum-behaved Particle Swarm Optimization Algorithm," Proceedings of the International Conference on Complex Systems and Applications, pp. 603-607, 2007.

[56] Chih-Cheng Kao, Chin-Wen Chuang, Rong-Fong Fung, "The self-tuning PID control in a slider–crank mechanism system by applying Particle Swarm Optimization Approach," Mechatronics, vol. 16, no. 8, pp. 513-522, 2006.

[57] Zulfatman and M. F. Rahmat, "Application of self-tuning Fuzzy PID controller on industrial hydraulic actuator using system identification approach," International Journal on Smart Sensing and Intelligent Systems, vol. 2, no. 2, pp. 246-261, 2009.

[58] Bryan Davis, System Modeling using a Mamdani Rule Base, Project Report, University of Florida.

[59] Leonid Reznik, Fuzzy Controllers, Newnes, Oxford, 1997.

[60] Jan Jantzen, Tuning Of Fuzzy PID Controllers, Technical Report No. 98-H 871 (fpid), Department of Automation,Technical University of Denmark, Denmark, 1998.

[61] Ethel Nakano and Arthur Jutan, "Application of Response Surface Methodology in Controller Fine-Tuning," ISA Transactions, vol. 4, pp. 353-366, 1994.

[62] Masood Sahraian and Srinivas Kodiyalam, "Tuning PID Controllers Using Error-Integral Criteria and Response Surfaces Based Optimization," Engineering Optimization, vol. 33, no. 2, pp. 135 – 152, 2000.

[63] Aiping Jiang and Arthur Jutan, "Response Surface Tuning Methods in Dynamic Matrix Control of a Pressure Tank System," Industrial and Engineering Chemistry Research, vol. 39, no. 10, pp. 3835–3843, 2000,

[64] J. D. Faucher and P. Maussion, "Response Surface Methodology For The Tuning of Fuzzy Controller Dedicated to Boost Rectifier with Power Factor Correction," IEEE, ISIE, Montreal, Quebec, Canada, pp. 199-204, 2006.

Hari O. Bansal obtained his Bachelor's degree in Electrical Engineering from University of Rajasthan in 1998. He received his Post-graduate from Malviya Regional Engineering College (now MNIT), Jaipur and completed his PhD in Electrical Engineering from BITS, Pilani in 2000 & 2005 respectively. Presently, he is an Assistant Professor in the Electrical and Electronics Engineering Department, BITS, Pilani. His fields of interest include application of soft computing techniques in control systems, power systems, power electronics, and power quality.

Rajamayyoor Sharma has recently completed his bachelor's degree in Electrical and Electronics Engineering from BITS, Pilani. His area of interest includes control systems, power systems and power quality.

P.R.Shreeraman has recently completed his bachelor's degree in Electrical and Electronics Engineering from BITS, Pilani. His area of interest includes automation, powers systems and power quality.