Genetic Algorithm-Based Adaptive PID Controller

Shradhanand Verma and Rajani K. Mudi

Abstract Conventional PID controllers (CPID) usually fail to provide satisfactory performance for integrating and nonlinear systems due to large overshoots and oscillation. Nonlinear and adaptive PID controllers (APID) are being developed toward achieving the desired control performance for such systems. In this study, we make an attempt to develop a genetic algorithm-based adaptive PID controller (GA-APID) in order to attain adequate servo as well as regulatory performance. While designing our GA-APID, first we formulate the structure of the APID controller followed by its optimal parameter estimation for a given system using genetic algorithm. Performances of GA-APID for nonlinear and integrating systems are compared with those of CPID and APID reported in the leading literature. From detailed performance analysis, GA-APID is found to provide significantly improved performance over others.

Keywords Genetic algorithm · PID controller · Adaptive control

1 Introduction

PID controllers in its different forms are mainly used in process industries due to their simple structures and ease of implementation [1]. The parallel form of the conventional PID controller (CPID) with three adjustable gain parameters is extensively studied [2, 3]. Such controllers can provide reasonably acceptable performances for first-order and second-order linear self-regulating processes.

S. Verma (✉) · R.K. Mudi
Department of Instrumentation and Electronics Engineering, Jadavpur University, Sec-3, Block—LB/8, Salt-Lake, Kolkata 700098, India
e-mail: shradhanandei@gmail.com

R.K. Mudi
e-mail: rkmudi@yahoo.com

However, their performances for nonlinear and integrating systems are usually not satisfactory due to associated intolerably large overshoot [2, 4, 5]. Several attempts have been made to overcome this limitation [6–11] by developing adaptive PID controllers (APID) through nonlinear parameterization. However, most of the APID parameters are selected based on trial and error, or sometimes through heuristics [8–11]. Therefore, their performances may not be optimal. Keeping in mind this point, and the excellent optimization power of genetic algorithms (GA) [12], in this study, we attempt to develop GA-based adaptive PID (GA-APID) controller toward achieving optimum performance. Here, GA-APID design involves two steps—first we define the structure of the adaptive PID; then, GA is used to find its best set of parameters with respect to a given closed-loop performance index or objective function. Performances of the developed GA-APID for nonlinear and integrating systems with dead time are compared with other PID controllers. Considerably improved performance with respect to a large number of indices justifies the effectiveness of the developed GA-APID.

2 The Proposed PID Controller

2.1 The Conventional PID Controller

The discrete form of a conventional PID controller (CPID) can be expressed as

$$\begin{aligned} u^c(k) &= K_p\left[e(k) + \frac{\Delta t}{T_i}\sum_{i=0}^{k} \mathrm{e}(i) + \frac{T_d}{\Delta t}\Delta \mathrm{e}(k)\right] \\ \text{or} \quad u^c(k) &= K_p e(k) + K_i \sum_{i=0}^{k} e(i) + K_d \Delta e(k). \end{aligned} \tag{1}$$

In (1), $e(k) = r - y(k)$ is the process error, where r is the set point and y is process output, K_p is the proportional gain, $K_i = K_p\left(\frac{\Delta t}{T_i}\right)$ is the integral gain, $K_d = K_p\left(\frac{T_d}{\Delta t}\right)$ is the derivative gain, T_i is the integral time, T_d is the derivative time, and Δt is the sampling period. Proper selection of the three tuning parameters—K_p, T_i, and T_d—is a critical task to attain the desired closed-loop performance. Through decades, various methods have been developed for the tuning of PID parameters [3]. Among them, Ziegler-Nichols (ZN) continuous cycling method [13] is most widely used by practicing engineers for the initial settings of PID parameters [3]. In this study also, we have used ZN continuous cycling method [4] for the initial settings of the PID parameters, i.e., $K_p = 0.6K_u$, $T_i = 0.5t_u$, and $T_d = \frac{t_u}{8}$, where K_u is the ultimate gain and t_u is the ultimate period.

2.2 The Adaptive PID Controller

We have already mentioned that CPID usually fail to provide acceptable performance for nonlinear and integrating systems. In order to overcome such limitations, a number of attempts have been made in [5, 8–11] for online adjustments of various gain parameters of CPID, thereby making it an APID, so that an overall improved performance is achieved. In this study, we will concentrate on the APID presented in [8], where the three gain constants, i.e., $K_P, K_{i,}$ and K_d, are continuously modified by an online updating factor alpha (α) with the following simple heuristic relations:

$$K_p^m(k) = K_p(1 + k_1|\alpha(k)|) \tag{2}$$

$$K_i^m(k) = K_i(k_2 + k_3\alpha(k)) \tag{3}$$

$$K_d^m(k) = K_d(1 + k_4|\alpha(k)|). \tag{4}$$

Here,

$$\alpha(k) = e_N(k) \times \Delta e_N(k), \tag{5}$$

where $e_N(k) = \frac{e(k)}{|r|}$; and $\Delta e_N(k) = e_N(k) - e_N(k-1)$.

Now, the APID can be redefined as

$$u^m(k) = K_p^m(k)e(k) + K_i^m(k)\sum_{i=0}^{k} \mathrm{e}(i) + K_d^m(k)\Delta \mathrm{e}(k). \tag{6}$$

In Eq. (6), $K_p^m(k), K_i^m(k)$ and $K_d^m(k)$ are the modified proportional, integral, and derivative gains, respectively, at kth instant and $u^m(k)$ is the corresponding control action. $k_1, k_2, k_3,$ and k_4 are the four additional positive constants. The objective behind such online gain adjustments as described by relations (2)–(4) is that when the process is moving toward the set point, control action will be less aggressive to avoid possible large overshoots and/or undershoots, and when the process is moving away from the set point, control action will be more aggressive to make a rapid convergence of the system. Following this gain-adaptive technique, in [8], a significantly improved performance of APID is found for high-order and nonlinear systems both in set point and load disturbance responses. However, out of the *seven* parameters of [8], i.e., $K_P, K_i, K_d, k_1, k_2, k_3,$ and k_4,, the first *three* constants, i.e., $K_P, K_i,$ and K_d, are selected based on ZN ultimate cycle rule, whereas the remaining *four* constants, i.e., $k_1, k_2, k_3,$ and k_4, are chosen by trial. Therefore, there is further scope to achieve improved performance if we can find the most appropriate settings of these parameters. Keeping in mind this objective and the GA as a powerful optimization tool, we are motivated to develop the proposed GA-APID. In the present work, *all* the seven parameters (i.e., $K_P, K_i, K_d, k_1, k_2, k_3,$ and k_4) are selected through optimization using binary coded GA. In the next section, we will describe the optimization process.

2.3 GA-Optimized APID

2.3.1 Objective Function of the Genetic Algorithm

In the present optimization problem, *integral absolute error* (*IAE*) is defined as the objective function or fitness function. The *IAE* is calculated as $IAE = \int_0^\infty |e(t)|\mathrm{d}t$.

2.3.2 Different Operations Used in GA

Encoding—Here, the population size is 10. We use 4 bits for each of the 7 variables (i.e., $K_P, K_i, K_d, k_1, k_2, k_3$, and k_4) in a particular solution. So each chromosome has 28 bits. Here, we consider the following ranges of variables: K_P, K_i, K_d are $\pm$ 20 % of their respective CPID, k_1[05], k_2[01], k_3[05], and k_4[030].

Decoding—We convert encoded binary value of each variable to decimal value. Then, we bring this decimal value in its defined range to get the real value of the optimization variable. The following linear mapping rule is used for this purpose:

$X_i = X_i^L + \frac{X_i^U - X_i^L}{2^n - 1} \times$ (decimal value of the ith variable in the binary string). where $X_i^L \le X_i \le X_i^U$, X_i = real value of the ith optimization variable, X_i^U = upper *limitof* ith variable, X_i^L = lower limit of ith variable, and n = no. of bits used for each variable (here it is 4 bits). The decoded values thus obtained are used to simulate the closed-loop system response in MATLAB. This process is repeated for all the 10 solutions. From each closed-loop response, we calculate the value of the objective function, *IAE*.

Selection—We sort the values of *IAE* in ascending order and select the 50 % fittest roots (best solutions) from the top for the next stage, i.e., crossover.

Crossover—50 % chromosomes will go for crossover. These chromosomes are known as parent. The crossover operator produces two children for each parent pair. So here, we will get 50 % children after crossover. Therefore, after crossover, total population will again be 100 % = 50 % (parents) + 50 % (children).

Mutation—To avoid local optima, we mutate the strings. The mutation probability (percentage of bits in a population mutated in each iteration) is generally kept low for steady convergence (here, it is $\approx$1.78 %).

Convergence check—To ensure the convergence, we draw a curve between performance index (*IAE*) and number of iteration. Initially, the value of the objective function *IAE* goes on decreasing rapidly, but after some iteration, its value remains almost constant, which indicates the optimization is complete.

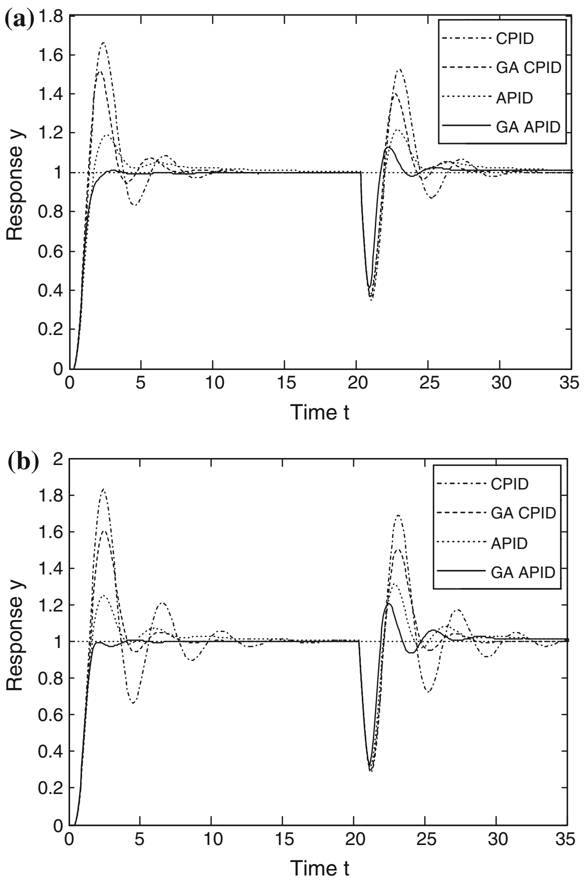

Fig. 1 **a** Responses of (7) with $L = 0.3$ s; **b** responses of (7) with $L = 0.4$ s

3 Results

For simulation study, we consider the following nonlinear (7) and integrating (8) systems with dead time (L):

$$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.2y^2 = u(t - L) \tag{7}$$

$$G_p(s) = \frac{e^{-Ls}}{s(s+1)} \tag{8}$$

Performance of our GA-APID is compared with conventional PID (CPID), GA-optimized conventional PID (GA-CPID), and the APID of [8]. For detailed comparison, in addition to the response characteristics, several performance indices, such as percentage overshoot ($\%OS$), rise time (t_r), settling time (t_s), integral

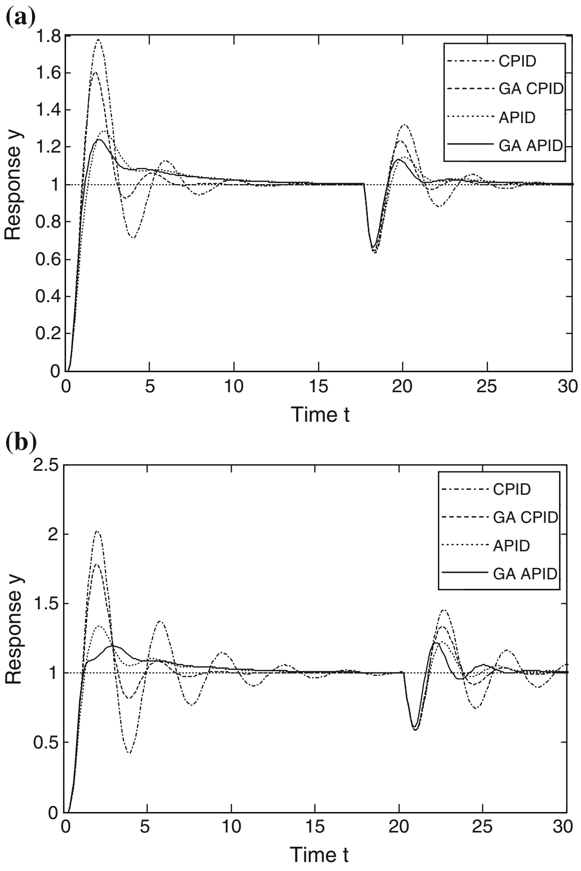

Fig. 2 **a** Responses of (8) with $L = 0.2$ s; **b** responses of (8) with $L = 0.3$ s

absolute error (*IAE*), and integral time absolute error (*ITAE*), are calculated for each setting. Closed-loop response curves (Figs. 1 and 2) for different PID controllers are presented as follows: CPID (– - –), GA-CPID (– –), APID (- - -), and GA-APID (—).

Table 1 and Fig. 1 present the performance comparison of different controllers for the nonlinear process of (7) with $L = 0.3$ and 0.4 s. Figure 1a and 1b shows remarkably improved performance of our proposed GA-APID during both set point change and load disturbance applied at $t = 20$ s, and this fact is clearly established from the various indices of Table 1. Responses of the integrating system of (8) with two different values of dead time ($L = 0.2$ and 0.3 s) are shown in Fig. 2. Table 2 provides the detailed performance comparison. In this case also, we find similar performance of GA-APID, though not to the same extent as that of the previous example.

From the above performance analysis, we observe that like CPID, GA-CPID also fails to provide acceptable performance due to excessively large overshoot,

Table 1 Performance analysis of (7)

L(s)	Controllers	*%OS*	t_r(s)	t_s(s)	*IAE*	*ITAE*
0.3	CPID	66.10	1.4	9.3	4.12	46.60
	GA-CPID	51.60	1.4	6.8	3.11	32.76
	APID	19.18	1.7	10.6	2.95	34.88
	GA-APID	0.97	2.6	2.1	1.93	20.10
0.4	CPID	83.11	1.5	13.3	5.78	72.40
	GA-CPID	60.77	1.6	7.4	3.81	41.39
	APID	25.15	1.7	10.8	3.27	39.45
	GA-APID	0.97	4.0	3.3	2.31	28.01

Table 2 Performance analysis of (8)

L(s)	Controllers	*%OS*	t_r(s)	t_s(s)	*IAE*	*ITAE*
0.2	CPID	77.50	1.1	10.2	3.45	27.19
	GA-CPID	60.21	1.1	6.5	2.22	15.28
	APID	28.75	1.4	11.0	2.46	19.44
	GA-APID	24.19	1.2	10.6	2.14	16.09
0.3	CPID	102.20	1.2	17.1	5.67	58.74
	GA-CPID	78.23	1.2	7.8	3.14	25.72
	APID	33.80	1.3	11.0	2.69	24.26
	GA-APID	19.54	1.2	11.8	2.42	21.27

possibly due to their linear control law, whereas our GA-optimized adaptive PID (GA-APID) can significantly improve the performance over APID, which justifies the usefulness of GA for further enhancement of APID [8].

4 Conclusion

In this work, we studied the performance of (GA-APID) controller for nonlinear and integrating systems with dead time under both set point change and load disturbance. Simulation results revealed that GA-APID is capable of providing remarkably improved servo as well as regulatory performance compared to even GA-CPID, and significantly overall improved performance in comparison with the recently reported APID.

References

1. Shinsky, F.G.: Process Control Systems—Application, Design, and Tuning. McGraw-Hill, New York (1998)
2. Astrom, K.J., Hang, C.C., Person, P., Ho, W.K.: Towards intelligent PID control. Automatica **28**(1), 1–9 (1992)
3. Ang, K.H., Chong, G.C.Y., Li, Y.: PID control system analysis, design, and technology. IEEE Trans. Control Sys. Technology **13**(4), 559–576 (2005)
4. Dey, C., Mudi, R.K., Simhachalam, D.: An auto-tuning PID controller for integrating plus dead-time processes. Adv. Mater. Res. **403–408**, 4934–4943 (2012)
5. Dey, C., Mudi, R.K., Simhachalam, D.: A simple nonlinear PD controller for integrating processes. ISA Trans. **53**(1), 162–172 (2014)
6. Seborg, D.E., Edgar, T.F.: Adaptive control strategies for process control: a survey. AICHE J. **32**(6), 881–913 (1986)
7. Kristiansson, B., Lennartson, B.: Robust and optimal tuning of PI and PID controllers. IEE Proc. Control Theory Appl. **149**(1), 17–25 (2002)
8. Dey, C., Mudi, R.K.: An improved auto-tuning scheme for PID controllers. ISA Trans. **48**(4), 396–409 (2009)
9. Mudi, R.K., Dey, C.: Performance improvement of PI controllers through dynamic set-point weighting. ISA Trans. **50**, 220–230 (2011)
10. Dey, C., Mudi, R.K., Lee, T.T.: Dynamic set-point weighted PID controller. Control Intell. Syst. **37**(4), 212–219 (2009)
11. Mudi, R.K., Dey, C., Lee, T.T.: An improved auto-tuning scheme for PI controllers. ISA Trans. **47**, 45–52 (2008)
12. Goldberg, D.E.: Genetic Algorithm in Search Optimization and Machine Learning. Addison-Wesley Publishing Co., Inc., MA (1989)
13. Ziegler, J.G., Nichols, N.B.: Optimum settings for automatic controllers. ASME Trans. **64**, 759–768 (1942)