Bio Inspired Algorithm Based Adaptive PID Controller

A thesis submitted in partial fulfillment of the requirement for the degree of Master of technology in Instrumentation & Electronics Engineering

Submitted by

Shuvam Roy

Examination Roll No.-M4IEE16-02

Reg. No.-129477 of 2014-2015

Class Roll No.-001411103003

Under the supervision of

Prof. Rajanikanta Mudi

Department of

Instrumentation & Electronics Engineering

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

KOLKATA-700032

2016

CERTIFICATE OF RECOMMENDATION

I hereby recommend that the thesis titled "***BIO-INSPIRED ALGORITHM BASED ADAPTIVE PID CONTROLLER***" carried out under my supervision by Mr. Shuvam Roy (Registration no.129477 of 2014-15) may be accepted in partial fulfillment of the requirement for the degree of "Master of Technology in Instrumentation & Electronics Engineering" of Jadavpur University.

.....

(*Prof. Rajanikanta Mudi*)

Thesis Supervisor

Department of

Instrumentation& Electronics Engineering

JADAVPUR UNIVERSITY

Salt Lake Campus

Kolkata (West Bengal)-700098

.....

(*Prof. Rajanikanta Mudi*)

Head of Department

Instrumentation & Electronics Engineering

JADAVPUR UNIVERSITY

Kolkata-700098

.....

(*Prof. Sivaji Bandyopadhyay*)

Dean

Faculty of Engineering and Technology

JADAVPUR UNIVERSITY

Kolkata-70032

*Certificate of Approval**

*(*Only in case the thesis is approved)*

The thesis at in stance is hereby approved as a creditable study of an Engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve this thesis for the purpose for which it is submitted.

---------- ----------

Signature of the examiner Signature of the Supervisor

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis contains literature survey and original research work by me, as a part of my Master of Technology in Instrumentation & Electronics engineering studies.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: SHUVAM ROY.
Roll Number: M4IEE16-02
Thesis Title: BIO-INSPIRED ALGORITHM BASED ADAPTIVE PID CONTROLLER
Signature with Date*:*

Acknowledgement

I would like to thank a lot of people who gave me unending support and inspiration from the beginning and without whose help this thesis and project would not have been completed.

First and foremost, I would like to thank my guide Prof. Rajani Kanta Mudi, whose suggestions, guidance and encouragement have helped me immensely in understanding the subject.

My sincere obligation goes to Prof. Bipan Tudu & Mr.Ujjwal Manikya Nath (Research scholar) for their constant support.

I would also like to thank all my classmates and the staffs of the department for the constant support and help they provided me all the time.

Regards

Shuvam Roy

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1

INTRODUCTION & SCOPE OF THE THESIS

1.1 Introduction

For industrial processes, Proportional, Integral, and Derivative (PID) controllers are mostly used by process engineers [1]. In spite of considerable advances in process control over the past half century, till today PID controllers / regulators are the backbone for the most industrial control systems [1]. Even if more sophisticated control techniques are developed, it is a common practice to have a hierarchical structure with PID control at the lowest level [3, 4]. According to a survey for process control systems in refinery, chemical, and paper industries, more than 95% of the control loops are found to be of PID type [8]. With its three term functionality covering control of both transient and steady-state responses, the PID controller offers the simplest and yet most efficient solution for many real world control problems [5, 6]. At present, the developments of PID controllers are mostly software based, so as to get the best out of PID control [5]. A number of software based techniques have also been realized in hardware modules while search still goes on to find the next key technology for PID tuning [7].

1.2 Close-loop Control with Three-term Controller

PID controllers are quite adequate for many control problems where there are modest performance requirements. Controllers used in the process industries are mainly concerned in maintaining the process variables-level, flow temperature, pressure, pH etc. at the desired operating value [8]. As these processes become large and / or more complex, the role of controller becomes more crucial [9]. In a close-loop feedback control, PID controller provides distinctive features for its three terms (P, I, and D). Depending on the instantaneous process error, necessary action is taken by the P term it provides an overall control action proportional to the error signal similar to all pass filters. I term has the ability to eliminate steady- state–error (offset) and it behaves like a low pass filter, where as anticipatory corrective measure is taken by D term and its behavior is identical to a high pass filter.

1.3 Conventional PID Controller

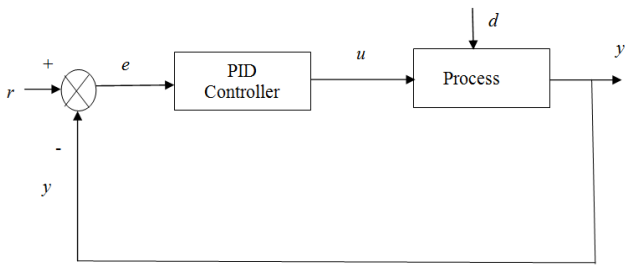


Fig. 1.1 *Block diagram of a close loop control with PID controller*

Standard nomenclature for different symbols used in Fig. 1.1

r =set point (the desired value of a controlled variable is referred to as its set point)

y=process output (controlled variables)

e=error signal

u=controller output

d=disturbance

Here, our objective is to make the controlled variable *y* equal to its set point *r* [10, 11].

Through decades, various methods have been developed for the tuning PID parameters. Among them Ziegler-Nichols (ZN) [6, 11] continuous cycling method is most widely used by practicing engineers for the initial settings of PID parameters.

If $T_i = \infty$ and $T_d = 0$ then controller turns out to be a proportional controller only. In proportional mode, controller fails to bring the process output to its desired value r, which results in an offset.

The introduction of integral action facilitates the achievement of equality between measured value and desired value, as a constant error produces an increasing controller output until the error becomes zero.

On the other hand, the introduction of derivative action facilitates that any change in the process value can be anticipated, and thus an appropriate correction may be added prior to the actual change. So the PID controller takes the corrective measure depending on the present, past, and future status of the error signal.

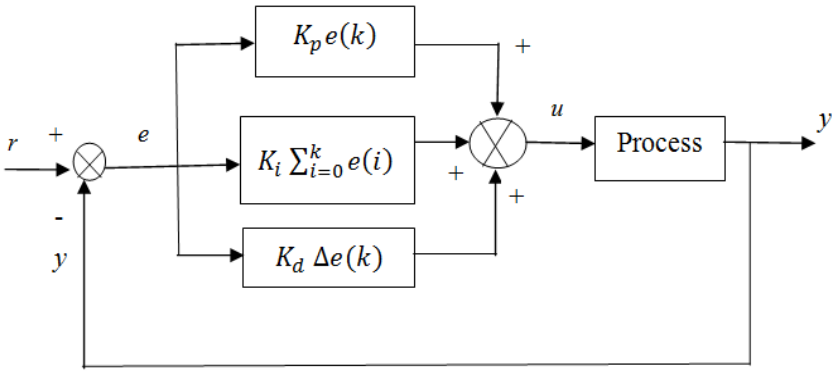


Fig. 1.2 *Parallel form of PID controller and process*

The conventional PID controller can be modeled by eq. 1.1

$$u_c = K_p[e(k) + \frac{\Delta t}{T_i}\sum_{i=0}^{k} e(i) + \frac{T_d}{\Delta t}\Delta e(k)] \quad (1.1a)$$

Or

$$u_c(k) = K_p\, e(k) + K_i \sum_{i=0}^{k} e(i) + K_d\, \Delta e(k)] \quad (1.1b)$$

Where K_p = proportional gain, $K_i = K_p(\Delta t / T_i)$ integral gain, $K_i = K_p(T_d / \Delta t)$ derivative gain, T_i = integral time, T_d = derivative time and Δt is the sampling time. Proper selection of these tuning parameters is a critical task to attain the desired close-loop performance. Though decades, various methods have been developed for the tuning of PID parameters [3]. Among them Ziegler-nichols method (ZN) continuous cycling method [11] is most widely used. In this study also, we have used ZN continuous cycling method [1] for the initial settings of the PID parameters, *i.e.*, $K_p = 0.6K_u$, $T_i = 0.5t_u$ and $T_d = 0.125t_u$ where K_u is the ultimate gain and t_u is the ultimate period.

1.4 Method Based on Performance Criteria

It is based on minimizing an appropriate performance criterion, either for optimum regulatory or for optimum servo performance. Based on the minimum *IAE, ITAE or IAE+ITAE* value, settings for PI and PID controllers are derived [11]. These settings are expected to provide desirable performance for time delay to time constant ratio from 0.1 to 1. Other tuning relations for PI controller for achieving minimum *IAE* value are suggested by Shinskey [8], Marlin [13], Edgar [10] etc. In 1993, Zhuang and Atherton suggested PI and PID settings based on the minimization of *ISE*, *ISTE*, and IST^1E [14]. For the FOPTD process model, repeated optimization is carried out for different values of time delay to time constant ratio. Using least square fit technique, simple relations for PID parameters are obtained from graphical results. For a given range of gain margin and phase margin values, the setting for a PID controller is given by Ho [15] with *ISE* minimization.

The *ISE* criterion penalizes large errors, while the *ITAE* criterion penalizes error that persists for longer periods of time. In general, the *ITAE* criterion is the preferred criterion in practice, because it usually results in the most conservative controller settings [15]. By contrast, the *ISE* criterion provides the most aggressive settings, while the *IAE* criterion tends to produce controller settings that are between those for the *ITAE* and *ISE* criteria. Based on *ITAE* minimization, different settings for PI and PID controllers are provided by Zhuang [16], Luyben [17] and many other researchers. PI and PID tuning relations based on *ITAE*

performance index are also developed for the FOPTD process model by Smith *et al*. [15].

In all the above tuning rules, the optimal controller settings are different for set-point changes in comparison to those for step load disturbances. In general, the controller settings for set-point changes are more conservative. Next, we provide a brief review on PID design and tuning methods proposed by various researchers.

1.5 PID Tuning Methods

Till today more than two hundred PID tuning rules have been proposed by the researchers [14], but none of them is suitable for all possible applications. Almost every tuning rule has some special feature for a specific class of processes hence before selecting the tuning rule we must know the nature of the process where it is to be applied.

An extensive list of tuning rules from the mid of twentieth century to the beginning of the twenty first century is given in [14]. Depending on the nature of these tuning relations they may be broadly classified into different categories, some of them are briefly described below.

1.5.1 Step Response / Process Reaction Curve Method [10]

The process reaction curve methods works by generating a process reaction curve (below) in response to a disturbance. Controller gain, integral time and derivative time can be calculated using this curve. The process reaction curve is identified by performing in an open loop step test of the process and finding model parameters for initial step disturbance P (%). These parameters are as follows: lag time L (min), change in PV in response to step disturbance KΔMV (%), reaction rate N (% min^{-1}), lag ratio R (dimensionless). A typical process reaction curve is generated using the following method:

1. Put the controller in manual mode
2. Wait until the process value reaches steady state or as close as possible (stable and not changing)
3. Introduce a small disturbance (step the output of the PID controller) - The step must be big enough to see a significant change in the process value. A rule of thumb is the signal to noise ratio should be greater than 5.
4. Collect data and plot
5. Repeat: making the step in the opposite direction.

K = the process gain $K = \dfrac{\Delta PV}{\Delta MV}$ (1.2)

\

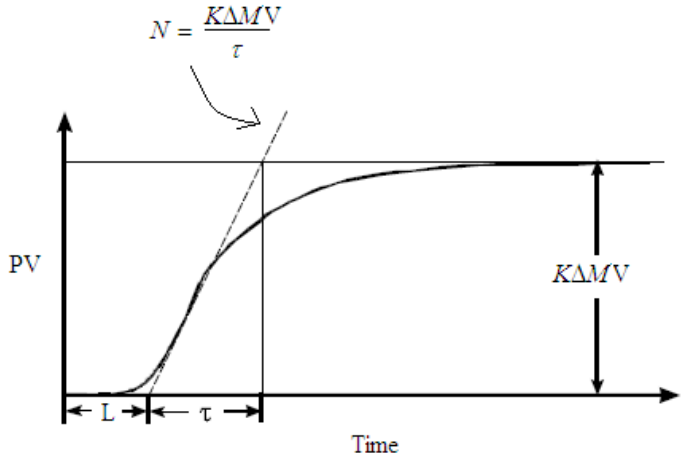


Fig. 1.3 *Process response Curve*

1.5.1 Ultimate Cycle Method [6]

Ziegler-Nichols method known as continuous cycling refers to sustained oscillation with constant amplitude. In this method, integration and derivative terms of the controller are disabled and the proportional gain is increased until a continuous oscillation occurs at gain (K_u) for the closed loop system. Considering gain and its related oscillating period (T_u), the PID parameters can be calculated from the following equations:

$$K_p = 0.6K_u \tag{1.3}$$

$$T_i = 0.5t_u \tag{1.4}$$

$$T_d = 0.125t_u \tag{1.5}$$

The main drawback of ultimate cycle based tuning method [11] is that the process is to be operated in continuous cycling situation and it may cause any type of mechanical failure of the actuator parts. But the most important feature of this tuning technique is its model free approach, *i.e.*, prior to the tuning no model is required for the process for which the controller is to be tuned. Procedure of ultimate cycle method is given below:

Step-1

After the process has reached steady state (at least approximately), eliminate the integral and derivative action by setting T_d =0 and T_i to the largest possible value.

Step –1

Set K_p equal to a small value (e.g., 0.5).

Step -3

Introduce a small, momentary set-point change so that the controlled variable moves away from the set point. Gradually increase K_p in small increments until continuous cycle occurs. The term continuous cycling refers to a sustained oscillation with constant amplitude. The numerical value of K_p that produces continuous cycling (for proportional-only control) is called the ultimate gain K_{cu}. The period of corresponding sustained oscillation is referred to as ultimate period T_u.

Step-4

Calculate the PID controller settings using Ziegler-Nichols (Z-N) tuning relations in Table 1.1

Table 1.1 *Controller setting based on the continuous cycling method*

Ziegler-Nichols (Z-N)	K_p	T_i	T_d
P	0.5 K_{cu}	-	-
PI	0.45 K_{cu}	$T_u/1.1$	-
PID	0.6 K_{cu}	$T_u/1$	$T_u/8$

1.6 Design of the Adaptive-PID Controller [10, 11]

Most of the conventional controllers (PID) are tuned based on Ziegler Nichols (ZN) [1, 1] tuning method for its simple tuning structure. But sometimes its performance is not satisfactory for higher order processes due to large non-linearity. In order to achieve satisfactory performances the parameters of conventional controllers (PID) are modified by an online gain updating factor (α), known as Adaptive PID controllers (APID) [10-11]. This online gain is updated by some heuristic relations. Each of such ZN [11, 13] tuned parameters of APID (i.e., proportional, integral and derivative gains) is updated online by the single modifying factor α through some simple relations. This online gain updating factor is function of $e(k)$ and $\Delta e(k)$.

Here $e(k)$ and $\Delta e(k)$ are expressed as

$$e(k) = r - y(k) \tag{1.6}$$

$$\Delta e(k) = e(k) - e(k-1) \tag{1.7}$$

Where *r (k)* is the set point and *y* (*k*) is the process output. The proposed gain updating factor α is defined by

$$\alpha(k) = e_N(k) \times \Delta e_N(k) \tag{1.8}$$

$$e_N(k) = e(k) \div |r| \tag{1.9}$$

$$\Delta e_N(k) = e_N(k) - e_N(k-1) \tag{1.10}$$

Variables in Eq. (1.9) and Eq. (1.10) are the normalized value of $e(k)$ and $\Delta e(k)$ respectively. From Eq. (1.11), without loss of generality it may be assumed that the possible variation of α will lie in the range [−1, 1] for all close-loop stable processes. In APID K_p, K_i, K_d will be continuously modified by the gain updating factor α with the following simple heuristic relations.

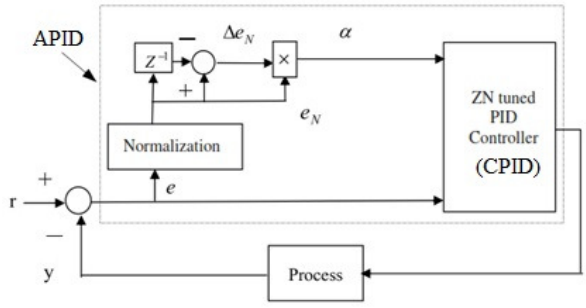


Fig.1.4 *Adaptive form of PID controller (APID)*

$$K_p^m(k) = K_p(1 + K_1 |\alpha(k)|) \tag{1.11}$$

$$K_i^m(k) = K_i(K_4 + K_2\alpha(k)) \tag{1.12}$$

$$K_d^m(k) = K_d(1 + K_3 |\alpha(k)|) \tag{1.13}$$

Thus from Eqs.(1) and (7)-(9), APID can be expressed as

$$u_m(k) = K_p^m(k)e(k) + K_i^m(k)\sum_{i=0}^{k} e(i) + K_d^m(k)\Delta e(k) \tag{1.14}$$

In Eqn. (1.14), $K_p^m(k)$, $K_i^m(k)$ and $K_d^m(k)$ are the modified proportional, integral and derivative gains respectively at k^{th} instant and $u_m(k)$ is the corresponding control action. K_1, K_2, K_3, K_4 are the four additional positive constants. However, out of seven parameters of [3], *i.e.,* $K_p, K_i, K_d, K_1, K_2, K_3, K_4$, the first three constants, *i.e.*, $K_p, K_i and K_d$ are selected based on ZN ultimate cycle method, where as the remaining four constants, *i.e.*, $K_1, K_2, K_3 and K_4$ are chosen by trial. The objective behind such online gain adjustments is

that, when the process is moving towards 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 a significantly improved performance of APID is found for high-order and nonlinear systems both in set-point and load disturbance responses.

1.7 Tuning Strategy of APID Controller

While designing APID [10, 11], the following major points are taken into consideration to provide the appropriate control action in different operating phases. For a better understanding, typical close-loop response of an under-damped second-order process is illustrated in Fig.1.5

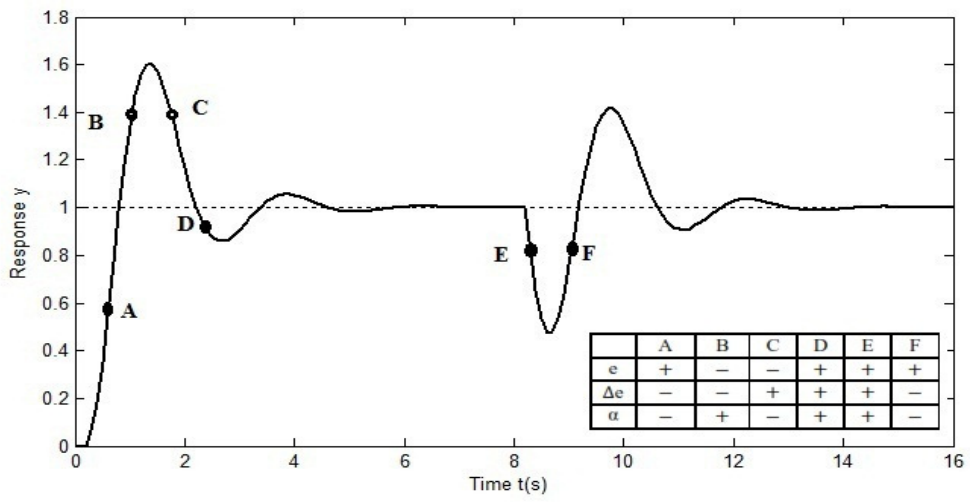


Fig. 1.5 *close loop response of second order process*

While the process is far from the set-point and moving fast towards it (e.g., points A, C, or F in Fig. 1.5), proportional gain should be reasonably large to reach the set-point quickly but the integral gain should be small enough to prevent the large accumulation of control action, which may result in a large overshoot or undershoot in future. At the same time, to reduce oscillations derivative gain should be increased for higher damping. Observe that, in such transient phase e and Δe are of opposite signs. Therefore, α becomes negative according to eqs (1.4) which will make both the proportional and derivatives gain higher, and integral gain lower than their corresponding initial values (i.e., $K_p^m > K_p, K_d^m > K_d, K_i^m < 0.3K_i$) as indicated by eqs. (1.11) - (1.13).Thus, the gain adaptive rules (eqs.(1.11)-(1.13)) try to adjust

the parameters of APID towards reducing the overshoot and/or undershoot, and oscillation in the process response.

When the process is moving further away from the set-point (e.g., points B, D, or E in Fig.1.5), increased proportional, and derivative as well as integral gains are expected to bring back the process variable to its desired value quickly. Under such situations both e and Δe will have the same sign, thereby making α positive (eqs (1.8)) which in turn makes all gain parameters of APID (i.e., K_p, K_i, K_d) larger than their respective initial values according to eqs (1.11) - (1.13). As a result the control action becomes more aggressive (i.e., $u^a > u^c$) which will try to restrict further deterioration of such situations. Therefore, APID satisfies the need for a relatively strong control action to improve process recovery.

From the above discussion it is evident that the proposed auto-tuning scheme always attempt to modify APID parameters (proportional, integral, and derivative gains) in the right directions to generate required control action in different transient phases for providing improved performance under both set-point change and load disturbance. Of course, depending on the type of response desired to achieve, suitable value of K_1, K_2, K_3 are to be selected by the designer either from the knowledge about the process to be controlled or through trial and error.

1.8 Literature Review

Automatic control is the application of control theory for regulation of processes without direct human intervention. In the simplest type of an automatic control loop, a controller compares a measured value of a process with a desired set value, and processes the resulting error signal to change some input to the process, in such a way that the process stays at its set-point despite of the disturbances [38]. Designing a system with features of automatic control generally requires the feeding of electrical or mechanical energy to enhance the dynamic features of an otherwise sluggish or variant, even errant system. This closed-loop control is an application of negative feedback to a system. An everyday example of a feedback control system is an automobile speed control, which uses the difference between the actual and the desired speed to vary the fuel rate. Since the system output is used to regulate its input, such a device is said to be a closed-loop control system. Practically every aspect of our day-to-day activities is affected by some type of control system. Control systems are found in abundance in all sectors of industry, such as quality control of manufactured products, automatic assembly lines, machine-tool control, space technology and weapon systems, computer control, transportation systems, power systems, robotics, *Micro Electro Mechanical Systems* (MEMS), nanotechnology, and many others [39].

There have been many developments in automatic control theory during recent years. It is difficult to provide an impartial analysis of an area while it is still developing; however, looking back on the progress of feedback control theory it is by now possible to distinguish some main trends. In about 170 BC the *Greek Ctesibius* invented a float regulator for a water clock, a device not unlike the ball and cock in a modern flush toilet. The invention of the mechanical clock in the 14th century made the water clock and its feedback control system obsolete. The float regulator does not appear again until its use in the *Industrial Revolution*. In [40] steam engine is invented in 1713, and this date marks the accepted beginning of the industrial revolution; however, its roots can be traced back into the 17th century.

The introduction of prime movers, or self-driven machines like advanced grain mills, furnaces, boilers, and the steam engine created a new requirement for automatic control systems including temperature regulators (1614), pressure regulators (1681), float regulators (1700) and speed control devices [41]. The design of feedback control systems up through the industrial revolution was by trial-and-error, together with a great deal of engineering intuition. Thus, it was more of an art than a science. In the mid-19th century, control theory began to acquire its written language, the language of mathematics. In 1868 the first rigorous mathematical analysis of a feedback control system was provided by J.C. Maxwell [41]. Thus, relative to this written language, we could call the period before about 1868 the prehistory of automatic control. The *First* and *Second World Wars* saw major advancements in the field of mass communication and signal processing. Other key advances in automatic controls include differential equations, stability theory and system theory (1938), frequency domain analysis (1940), and stochastic analysis (1941) [43]. With the advent of the space age in 1957, controls design, particularly in the *United States*, turned away from the frequency domain techniques of classical control theory and backed into the differential equation techniques of the late 19th century, which were contained in the time domain. The modern era saw time-domain design for navigation (1960), optimal control and estimation theory (1961), nonlinear control theory (1969), digital control and filtering theory (1974), and the personal computer (1983) [44]. As per Fridland in [45], we may call the period from 1868 to the early 1900's the *primitive period* of automatic control, the period from then until 1960 the *classical period*, and the period from 1960 through present times the *modern period*.

Today, a number of different controllers are used in industry and in many other fields. In quite general way those controllers can be divided into two main groups: unconventional controller [46] and conventional controller. Unconventional controllers utilize a new approach to the controller design in which knowledge of mathematical model of a process generally is not required. Examples of unconventional controllers are *fuzzy controller* and *neuro* or *neuro-*

fuzzy controller. As conventional controllers we can count *proportional-integral-derivative* (PID) controllers and other types such as optimal, adaptive, robust controller. It is a characteristic of all conventional controllers that one has to know a mathematical model of the process in order to design a controller. PID controller contains proportional (P), integral (I) and derivative (D) term. P term helps the process to reach its desired value quickly, but produces offset. To remove offset I term is included. I term eliminates offset but produce overshoot. Overshoot may not occur if D term is used because it provides damping and improves the dynamic characteristics of the process. The three terms can be used in various combinations to achieve good response in controlling purpose. The PID controller has been used successfully for regulating processes in industry for more than 60 years and used till today. A survey of Desborough and Miller in [47] indicates that more than 97% of regulatory controllers utilize the PID algorithm.

Although, a PID controller has only three adjustable parameters, finding appropriate settings is not simple which results in controller having poor tuning parameter and it becomes unable to provide satisfactory performance. Many researchers have attempted to develop the design methods for the PID controller in [48]. Ziegler-Nichols in [49] and Cohen-Coon in [50] provide tuning methods for PID controller in closed loop and open-loop respectively. Apart from open-loop and closed-loop tuning method, PID controller can be designed using gain and phase margin specification [51] and using optimization technique [51]. In ref [53, 54] PID controllers for integral with dead time process is discussed. However, both of the approaches required open-loop process model first. Process model can be obtained using step test in open-loop which is very time-consuming and may result in undesirable output changes. Then process parameters-*time delay* (θ), *process gain* (K), *time constant* (τ) are estimated from experimental data obtained using step test [55].

The main alternative is to use closed-loop experiment, one proposed by Ziegler-Nichols in [48]. This approach of classical method require very little information about the process i.e. *ultimate gain* (K_u) and *ultimate period of oscillation* (T_u) which can be obtained using single experiment. The recommended setting for a proportional-integral (PI) controller are $K_c = 0.45K_u$ and $\tau_I = 0.83P_u$. In general there are some disadvantages of closed-loop experimentation method. *First*, this is essentially trial and error method, since several values of gain must be tested before the ultimate gain (or the gain to give ¼ th decay ratio) is determined. *Second*, while one loop is being tested in this manner its output may affect several other loops, thus possibly upsetting an entire unit. Third, it can only be used on processes for which the phase lag exceeds -180 degrees at high frequencies. For example, it does not work on a simple second-order process. This problem can be circumvented by

introducing sustained oscillations with an on-off controller using the relay method in [56], but the system should be capable to withstand the oscillation resulting from on-off action. The work reported in [56] was further extended using proportional controller by Yuwana and Seborg [47]. Their work mainly concerned with first-order with delay model (FOPTD) where through *pade's approximation* the response seems to be second-order response.

Shamsuzzoha and Skogestad in [48] proposed guidelines for a PI/PID controller of an unidentified process models using closed-loop experiments. It requires one closed loop step set-point response experiment using a proportional only controller, and mainly uses information about the first peak (overshoot), which is easily identified. The set-point experiment is similar to that of Ziegler-Nichols (1941) but the controller gain is typically about one half, so the system is not at the verge of stability with sustained oscillations. The recommended controller in [48] suggests the value of gain change is a function of the height of the first peak (overshoot); the controller integral time is mainly a function of peak time.

At present a new set point weighting technique known as dynamic set point weighting [57] is proposed to improve the set point response and load rejection characteristics of Ziegler-Nichols tuned PID controllers when their responses of are not satisfactory. Instead of a fixed (single or multi-valued) set point weighting factor, here dynamic set point weighting is suggested based on the change of error (Δe) of the controlled variable and normalized dead time of the process under control.

Finally when the controller response tuned by ZN method is found to be unsatisfactory in some cases for higher order & non-linear processes, then an improved auto-tuning scheme is proposed [58, 59] to overcome the disadvantages of ZN tuned PID controllers (ZNPID). The ZNPIDs are upgraded by some easily interpretable heuristic rules through an online gain modifying factor defined on the instantaneous process states is known as adaptive PID controller (APID). This study thereby making the scheme suitable for a wide range of processes and more generalized too.

It might be possible that their performances are not optimal for some cases. To achieve optimal performances now a day optimization techniques are used severely to find the optimal settings of the APID controller. To design our controller, initially the parameters of the conventional PID controller are obtained using ZN method and after that an adaptation scheme is incorporated to get enhanced performance. Finally, we optimize various tuning parameters by the optimization algorithm for various processes. These optimization algorithms are based on the minimization of objective functions such as Integral-absolute-error [IAE] [39], Integral time & absolute error [ITAE] or integral square error [ISE] *etc*.

Performances of the optimized PID are not found to be satisfactory. Experimental results exhibit remarkably improved performance of the optimal APID when compared with ZN tuned PID, optimized PID and APID. This fact justifies our present study, *i.e.*, incorporation of optimization techniques in APID rather than PID.

In this leading literature we attempt to develop the optimized APID controllers with respect to an objective function based on different optimization algorithms such as Genetic algorithm, [60] Artificial bee colony algorithm (ABC) [61], Bacterial foraging optimization algorithm [62] and Particle swarm optimization algorithm [63] to develop GA-APID,BFO-APID,PSO-APID and ABC-APID respectively. Finally, we compare the performances of optimized APID based on different algorithms stated above and compare with other controllers with other controllers. Experimental results exhibit remarkably improved performance of the optimal APID (GA-APID, BFO-APID, PSO-APID or BFO-APID) when compared with PID, optimized PID (GA-APID, BFO-APID, PSO-APID or BFO-APID), and APID.

1.9 Scope of the Thesis

Our literature survey reveals that a lot of works has been done towards improving the performance of PID controllers with increased robustness. In a broad sense, such development works on controller tuning are mostly dependent on the process model. However, for a practical process it is very difficult to find its exact model, as a result, most of the theoretical developments have limitation from practical implementation point of view. Along with mathematical complexity in finding out the appropriate process model, there is always a certain amount of uncertainty in model parameters. Model parameters are also changing with time due to natural phenomena like aging, scaling, erosion etc. So obtaining the desired performance from a PID controller is not only goal. Additionally it has to be robust enough to withstand the model uncertainties as well as process nonlinearities. At the same time, it is found that an optimally tuned controller is more prone to fragile. So depending on the area of application, there should be a compromise between optimality and robustness of selected parameters.

Soft-computing tools like fuzzy logics, neural networks and different optimization techniques are also used by the researchers to obtain optimal settings of PID parameters. In such cases the engineers have tried to incorporate the human intelligence in the controller behavior. Certain improvements are found in the controller performances on making them more intelligent but at the higher computational complexity. A controller designed to reduce the initial overshoot during set-point change usually fails to offer good load rejection

behavior. On the other hand, a controller with better load capability cannot restrict the overshoot in the set-point response. Although in some cases improvements in the process behavior are observed during both set-point and load disturbance responses. In case of APID controller the gain is online updated continuously by an online updating factor through some heuristic relations. It might possible that their performances are not optimal in some cases. We can solve this type of problems by using optimization algorithms. In our case, we have decided to explore the optimal power of different optimization algorithms to design our proposed Adaptive PID (APID) controllers. Here, all the above said designs including two steps- first, we define the structure of the adaptive PID controller and then the algorithms are used to find their best set of parameters with respect to an objective function. In our experimental purpose we have studied the performances of the developed adaptive controllers based on different optimization algorithms over PID & APID controllers for various types of process with dead time.

For the optimization purpose we have chosen *IAE* and *IAE+ITAE* as objective functions, because they provide the overall improved performance, as we know, lower values of *IAE* & *ITAE* indicates improved set point response and good load rejection respectively.

In chapter-2, we have presented the detailed description of the Genetic algorithm based APID controllers with respect to two objective functions (*IAE+ITAE, IAE*). Initially we have optimized the conventional PID controllers & compare it with PID controller. In this case we found that the result is not very encouraging. So, we followed the same procedure on APID controllers and found that it provides the best performance when compared with others.

In chapter-3, we have presented the detailed description of the Bacterial foraging optimization (BFO) based APID controllers with respect to two objective functions (*IAE+ITAE, IAE*). Initially we have optimized the conventional PID controllers and compared it with PID controller. In this case also we found that the result is not quite satisfactory. Like previous case, then we applied BFO on APID controllers. The resulting optimal APID (BFO-APID) compared to PID, APID and BFO-PID controllers.

In chapter-4, we have explored the overall performance of the Particle swarm optimization based APID controllers with respect to two objective functions (*IAE+ITAE, IAE*). Initially the PID and APID controllers are tuned by ZN method. In case of APID controller the value of the four variables are taken as constant based on trial and error method. After that the optimal

PID (PSO-PID) is designed and found that the overall performance is not quite satisfactory. This fact motivated us to apply the same technique on APID controller with seven tunable parameters ($K_p, K_i, K_{d,} k_1, k_2, k_3, k_4$), which provides an improved performance over other controllers. The same impressive performance is obtained with increased dead time also.

In chapter-5, we have observed the overall performance of the Artificial bee colony algorithm based APID controllers with respect to two objective functions (*IAE+ITAE, IAE*). Initially the PID and APID controllers are tuned by ZN method. In case of APID controller the value of the four variables are chosen by trial and error method. After that the optimal PID (ABC-PID) is designed and found that the overall performance is not quite satisfactory. This fact motivated us to apply the same technique on APID controller with seven tunable parameters ($K_p, K_i, K_{d,} k_1, k_2, k_3, k_4$). This simulation study reveals that ABC-APID provides significantly improved performance, even with increased dead time.

In Chapter-6, first we have provided a brief summary of the present study. Then we have discussed the implementation issues of the four optimization techniques, genetic algorithm, particle swarm optimization, bacterial foraging and artificial bee colony based optimization while designing optimal PID controllers in Chapters 2, 3, 4, 5. Lastly, we also try to point out future scopes for further improvement.

References

[1] K. J. Astrom and T. Hagglund, Automatic tuning of PID controllers, Instrument Society of America, Research Triangle Park, 1988.

[2] K. J. Astrom and T. Hagglund, Revisiting the Ziegler-Nichols step response method for PID control, Journal of Process Control, Vol. 14, No. 6, pp. 635-650, 1004.

[3] L. Desbourough and R. Miller, Increasing customer value of industrial control performance monitoring - Honeywell's experience, Proc. Sixth International Conference on Chemical Process Control, AICHE Symposium Series Number 316, Vol. 98, 1001.

[4] K. J. Astrom and T. Hagglund, The future of PID control, Control Engineering Practice, Vol. 9, No. 11, pp. 1163-1175, 1001.

[5] K. H. Ang, G. Chong, and Y. Li, PID control system analysis, design, and technology, IEEE Transaction on Control System Technology, Vol. 13, No. 4, pp. 559-576, 1005.

[6] J. G. Ziegler and N. B. Nichols, Optimum setting for automatic controllers, ASME Transaction, Vol. 64, No. 11, pp. 759-768, 1941.

[7] P. Marsh, Turn on, tune in, New Electronics, Vol. 31, No. 4, pp. 31-31, 1998.

[8] F. G. Shinsky, Process Control Systems - Application, Design and Tuning, McGraw-Hill, New York, 1998.

[9] B. W. Bequette, Process Control Modeling, Design, and Simulation, Pearson Education, New Jersey, 1003.

[10] [10] A. O'Dwyer, Hand book of PI and PID Controller Tuning Rules, Imperial College Press, London, 1003.

[11] D. E. Seborg, T. F. Edgar, and D. A. Mellichamp, Process Dynamics and Control, John Wiley & Sons, Singapore, 1004.

[12] T. E. Marlin, Process Control, McGraw-Hill, New York, 1995.

[13] T. F. Edgar *et al*., Perry's Chemical Engineers' Handbook, McGraw-Hill, New York, 1997.

[14] M. Zhuang and D. P. Atherton, Automatic tuning of optimum PID controllers, IEE Proc. - Control Theory Applications, Vol.140, No. 3, pp. 116-114, 1993.

[15] C. A. Smith and A. B. Corripio, Principles and Practice of Automatic Control, John Wiley, New York, 1997.

[16] H. P. Huang, M. L. Roan, and J. C. Jeng, On-line adaptive tuning for PID controllers, IEE Proc. - Control Theory Applications, Vol. 149, No. 1, pp. 60-67, 1001.

[17] B. D. Tyreus and W. L. Luyben, Tuning PI controllers for integrator / dead-time process, Industrial & Engineering Chemistry Research, Vol. 31, pp. 1615-1618, 1991.

[18] D. E. Seborg and T. F. Edgar, Adaptive control strategies for process control: A survey, AICHE Journal, Vol. 31, No. 6, pp. 881-913, 1986.

[19] K. J. Astrom and B. Wittenmark, A Survey of Adaptive Control Applications, Proc. IEEE 34th International Conference on Decision & Control, pp. 649-654, 1995.

[20] R. K. Mudi, C. Dey, and T. T. Lee, An improved auto-tuning scheme for PI controllers, ISA Transaction, Vol. 47, No. 1, pp. 45-51, 1008.

[21] C. Dey and R. K. Mudi, An improved auto-tuning scheme for PID controllers, ISA Transaction, Vol. 48, No. 4, pp. 396-409, 1009.

[22] I. K. Kookos, A. I. Lygros, and K. G. Arvanitis, On-line PI controller tuning for integrator / dead-time processes, European Journal of Control, Vol. 5, pp.19-31, 1999.

[23] E. Poulin, and A. Pomerleau, PI settings for integrating processes based on ultimate cycle information, IEEE Transaction on Control System Technology, Vol. 7, No. 4, pp. 509-511, 1999.

[24] C. Dey and R. K. Mudi, A simple auto-tuning PID controller for integrating plus dead-time processes, Control and Intelligent Systems.

[25] M. Chidambaram and R. Padma Sree, A simple method of tuning PID controllers for integrator / dead-time processes, Computers and Chemical Engineering, Vol. 17, No. 1, pp. 111-115, 1003.

[26] R. Padma Sree, M. N. Srinivas, and M. Chidamabaram, A simple method of tuning PID controllers for stable and unstable FOPTD systems, Computers and Chemical Engineering, Vol. 18, No. 11, pp. 1101-1118, 1004.

[27] A. R. Benaskeur and A. Desbiens, Backstepping-based adaptive PID control, IEE Proc. - Control Theory Applications, Vol. 149, No. 1, pp. 54-59, 1001.

[28] H. Panagopoulos, K. J. Astrom, and T. Hagglund, Design of PID controllers based on constrained optimization, IEE Proc. - Control Theory Applications, Vol. 149, No. 1, pp. 31-40, 1001.

[29] F. Lin, R. D. Brandt, and G. Saikalis, Self-tuning of PID controllers by adaptive interaction, Proc. American Control Conference, pp. 3676-3681, 1000.

[30] I. Kaya and D. P. Atherton, A PI-PD controller design for integrating processes, Proc. American Control Conference, pp. 158-161, 1999.

[31] R. C. Panda, C. C. Yu, and H. P. Huang, PID tuning rules for SOPDT systems: Review and some new results, ISA Transaction, Vol. 43, No. 1, pp. 183-195, 1004.

[32] R. R. Pecharroman and F. L. Pagola, Improved identification for PID controllers auto-tuning, Proc. European Control Conference -ECC-1999, in CD, Germany, 1999.

[33] R. Toscana, A simple robust PI / PID controller design via numerical optimization approach, Journal of Process Control, Vol. 15, No. 1, pp. 81-88, 1005.

[34] K. J. Astrom and T. Hagglund, Automatic tuning of simple regulators with specifications on phase and amplitude margins, Automatica, Vol. 10, No. 5, pp. 645-651, 1984.

[35] C. C. Hang, K. J. Astrom, and Q. G. Wang, Relay feedback auto-tuning process controllers - a tutorial review, Journal of Process Control, Vol. 11, No. 1, pp. 143-161, 1001.

[36] C. Dey, R.K. Mudi, and D. Simhachalam, An Auto-tuning PID Controller for Integrating Plus Dead-time Processes. Advanced Materials Research. 403-408, 4934-4943(1011).

[37] C. Dey, R.K. Mudi, and D.Simhachalam,; A Simple Nonlinear PD Controller for Integrating Processes. ISA Trans. 53(1), 161-171(1014).

[38] G. Stephanopoulos, 'Chemical Process Control: An Introduction to Theory and Practice.' PHI, New Delhi, 1006.

[39] F. Golnaraghi and B. C. Kuo, 'Automatic Control System.' John Wiley & Sons, 9th ed., New York, U.S.A., 1010.

[40] http://en.wikipedia.org/wiki/Thomas_NewcomenWikipedia.

[41] M. Bokharaie, '*A summary of the History of Control Theory*.' Internal Rept., School of Elect. Eng., Ga. Inst. of Technology, Atlanta, GA 30331, 1973.

[42] http://en.wikipedia.org/wiki/James_Clerk_MaxwellWikipedia.

[43] http://en.wikipedia.org/wiki/Automatic_control.

[44] K. Ogata, 'Modern Control Engineering.' Prentice Hall, 5th ed., New Jersey, U.S.A., 1010.

[45] B. Friedland, '*Control System Design: An Introduction to State-Space Methods.'* New York: McGraw-Hill, 1986.

[46] Z. Vukich and O. Kuljaca, 'Lecture on PID Controllers.' Faculty of Electrical Engineering and Computing, 1001.

[47] L. D. Desborough and R. M. Miller, 'Increasing customer value of industrial control performance monitoring-Honeywell's experience.' Chemical Process Control–VI (Tuscon, Arizona, Jan. 1001), AIChE Symposium Series No. 316, 98, U.S.A, 1001.

[48] K. J. Astrom and T. Hagglund, 'PID Controllers : Theory, Design and Tuning.' Instrument Society of Amarica, 1995.

[49] J. G. Ziegler, N. B. Nichols and Y. B. Rochester, 'Optimum Settings for Automatic Controllers.', Transaction of ASME, pp. 759-765, 1941.

[50] G. H. Cohen and G. A. Coon, 'Theoretical Consideration of Retarded Control.', Transactions ASME, vol. 75, pp. 817-834, 1953.

[51] W. K. Ho, C. C. Hang and L. S. Cao, 'Tuning of PID Controllers Based on Gain and Phase Margin Specification.' Automatica, vol. 31(03), pp. 497-501, 1995.

[52] H. Panagopoulus, K. J. Astrom and T. Hagglund, 'Design of PID Controllers Based on Constrained Optimisation.' Process Inst. Elect. Engineering, vol.149, pp. 31-40, 1001.

[53] A. Visioli, 'Optimal tuning of PID controllers for integral and unstable processes.' IEE Proc.-Control Theory Appl., vol. 148, pp. 180-184, 1001.

[54] M. Chidambaram and R. P. Sree, 'A Simple Method of Tuning of PID Controller for Integrating/Dead Time Processes.' Computers and Chemical Engineering, vol. 17, pp. 111-115, 1003.

[55] Faculty of Engineering Technology Mechanical Automation and Mechatronics, University Twente, 'System Identification and Parameter Estimation.' Copyright R.G.K.M. Aarts, Enschede, 1011/1011 ed., 1011.

[56] K. J. Astrom and T. Hagglund, 'Automatic tuning of simple regulators with specifications on phase and amplitude margins.' Automatica, vol. 10, pp. 645–651, 1984.

[57] C. Dey, R. K. Mudi, and T. T. Lee, "Dynamic set-point weighted PID controller," Control and Intelligent Systems, vol. 37, 1009, no. 4, pp. 111-119.

[58] R. K. Mudi, C. Dey, and T. T. Lee, An improved auto-tuning scheme for PI controllers, ISA Transaction, Vol. 47, No. 1, pp. 45-51, 1008.

[59] C. Dey and R. K. Mudi, An improved auto-tuning scheme for PID controllers, ISA Transaction, Vol. 48, No. 4, pp. 396-409, 1009.

[60] David.E.Goldberg, john H.Holland, on genetic algorithms and machine learning, machine learning, N0.3, 95-99, (1988).

[61] Karaboga. D., Basturk. B., Artificial bee colony (ABC) optimization algorithm for solving constrained optimization Problems, LNCS: Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing, pp.789-798 (1007).

[62] Passino, K.M., Bacterial foraging optimization, International Journal of Swarm Intelligence Research, 1(1), 1-16, (1010).

[63] James McCaffrey, http://msdn.microsoft.com/en-us/maPSOzine/hh335067.aspx.

CHAPTER-2

REAL CODED GENETIC ALGORITHM BASED ADAPTIVE PID CONTROLLER

2.1 Introduction

God is the creator of the whole universe. Ever since its creation evolution has been a part and parcel of its functioning. New organisms have evolved from their ancestors; and this evolution is governed by a simple law which Charles Darwin named as –"Survival of the Fittest".

Genetic Algorithms [1] are search algorithms based on natural selection and natural genetics. They combine survival of fittest among structures with structured yet randomized information exchange to form a search algorithm. Genetic Algorithm has been developed by John Holland [1, 2] and his co-workers in the University of Michigan in the early $60's$. Genetic algorithms are theoretically and empirically proved to provide robust search in complex spaces. Its validity in–Function Optimization and Control Applications is well established.

Genetic Algorithms (GA) provide a general approach for searching for global minima or maxima within a bounded, quantized search space. Since GA only requires a way to evaluate the performance of its solution guesses without any prior information, they can be applied generally to nearly any optimization problem. GA does not guarantee convergence nor that the optimal solution will be found, but do provide, on average, a "good" solution. GA is usually extensively modified to suit a particular application. As a result, it is hard to classify a "generic" or "traditional" GA, since there are so many variants. However, by studying the original ideas involved with the early GA and studying other variants, one can isolate the main operations and compose a "traditional" GA. An improvement to the "traditional" GA to provide faster and more efficient searches for GAS that does not rely on average chromosome convergence (i.e. applications which are only interested in the best solution).

The "traditional" GA is composed of a fitness function, a selection technique, and crossover and mutation operators which are governed by fixed probabilities. These operations form a genetic loop as shown in Figure. Since the probabilities are constant, the average number of local and global searches in each generation is fixed. In this sense, the GA exhibits a fixed convergence rate and therefore will be referred to as the fixed-rate

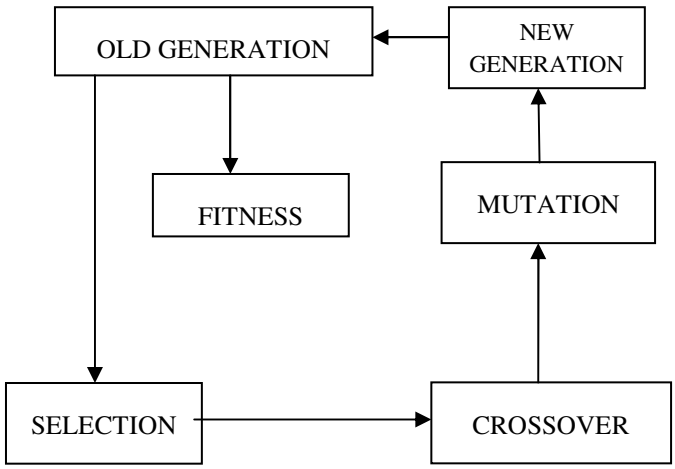


Fig. 2.1 *Genetic loop*

2.2 Real Coded Genetic Algorithm [2]

The concept of the genetic algorithm was first formalized by Holland and extended to functional optimization by DeJong [4] .It imitates the mechanism of the natural selection and evolution and aims to solve an optimization problem with object function f(x) where x=[x1 x2 ……. xN] is the N-dimensional vector of optimization parameters. It has proved to be an effective and powerful global optimization algorithm forms any combinatorial optimization problems, especially for those problems with discrete optimization parameters, no differentiable and/or discontinuous object function. Genes and chromosomes are the basic building blocks of the real coded GA. The real coded binary GA encodes the optimization parameters into decimal number.

The binary GA [5] does not operate directly on the optimization parameters but on a discretisized representation of them. Discretization error will inevitably be introduced when encoding a real number. The encoding and decoding operations also make the algorithm more computationally expensive for problems with real optimization parameters. It is therefore worth developing a novel GA which works directly on the real optimization parameters. The real-coded GA is consequently developed. Both theoretical proof and practical experiences show that RGA usually works better than binary GA, especially for problems with real optimization parameters. The RCGA operates on a population of chromosomes (or individuals, creatures, etc) simultaneously. It starts from an initial population, generated randomly within the search space.

Once the initialization is completed, the population enters the main RCGA loop and performs a global optimization for searching the optimum solution of the problem. In a RCGA loop, Preprocessing, three genetic operations, and post processing are carried out in turn. The RCGA loop continues until the termination conditions are fulfilled.

In our case we have decided to use the optimal power of Genetic algorithm (GA) to design our proposed Adaptive PID (GA-APID) [7, 8] controllers. Here, all the above said designs including two steps- first, we define the structure of the adaptive PID controller and then the algorithms are used to find their best set of parameters with respect to an objective function. In our experimental purpose we have studied the performances of the developed adaptive controllers based on GA algorithms over PID & APID controllers for different processes with dead time.

2.3 Objective Function of the Genetic Algorithm

The function to be optimized is known as objective function. Here, minimization of the *integral-absolute-error* (*IAE*) [9] or *integral-time-absolute-error* (*ITAE*) or combination of both i.e., (*IAE+ITAE*) is defined as the objective function (performance index or fitness function). The *IAE* and *ITAE* are calculated as:

$$IAE = \int_0^t |e(t)|\,dt \tag{2.1}$$

$$IAE + ITAE = \int_0^t |e(t)|\,dt + \int_0^t t\,|e(t)|\,dt \tag{2.2}$$

2.4. Genetic Algorithm Parameters

Table 2.1 *Genetic algorithm parameters*

Population size	10
Selection probability	50%(single point crossover)
Crossover probability	50%
Mutation probability	≈ 1.78
Variables	$K_p, K_i, K_d, k_1, k_2, k_3, k_4$
Range of variables	K_p, K_i, K_d , are ±20% of their respective Conventional PID, K_1, [0.5],K_2[0,5],,K_3[0,30], K_4[0,1]

2.5 Different Operations Used in Genetic Algorithm

Encoding – In order to use GA to solve the maximization or minimization problem, unknown variables are first coded in some string structures. It is important to mention that coding of variables is not necessary. There exist some studies where GAs are directly used on the variables themselves, Binary- coded strings having 1s and 0s are mostly used. The length of the string is usually determined according to the desired solution accuracy. In Real Coded Genetic Algorithm does not work on the encoding of a parameter set as real valued individuals are directly used.

Selection – It is the usually the first operator applied on population. Chromosomes are selected from the population to be parents to be crossover and produce offspring. According to Darwin's evolution theory of survival of the fittest, the best one should survive and create new offspring. It is also known as reproduction operator. There exist a number of reproduction operators in GA literature but the essential idea in all of them is that the above average strings are picked from the current population and their multiple copies are inserted in the mating pool in a probabilistic manner.

Various methods of selecting chromosomes for parents to crossover are [2]:

i. Roulette-wheel selection
ii. Tournament selection
iii. Boltzmann selection
iv. Rank selection

In spite of various selection methods as given above, We have used simple MATLAB command for the selection purpose because of its simplicity. We sort the values of ***IAE*** **or** ***IAE+ITAE*** in ascending order and select the 50% fittest roots (best solutions) from the top for the next stage, i.e., Crossover.

Crossover – After the selection phase is over, the population is enriched with better individuals. Selection makes clones of good strings, but does not create new ones. Cross over operator is applied to the mating pool with a hope that it would create a better string.

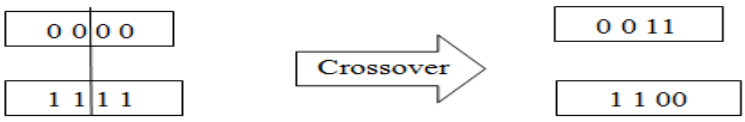
The aim of crossover operator is to search the parameter space. In addition, search is to be made in a way that the information stored in the present string is maximally preserved because these parent strings are instances of good string selected during reproduction (selection).It is a recombination operator. This is mainly responsible for search of new strings.

Since selection rate is 50%, therefore 50% chromosomes will go for crossover. These chromosomes are known as parent. The crossover operator produces two children for each parent pair. Therefore, after crossover total population will again be 100% = 50% (parents) + 50% (children).

Note that because I have 10 chromosomes. So selection gives the 5 chromosomes. These 5 chromosomes known as parents they go for crossover to produce children. Here we will get ${}^{5}C_{2}$, i.e., 10 no of children but we consider only 5 children. Thus total population will again be 10 (i.e., 5 as parents and 5 as children).

Let binary value of each variable is 0000 and 1111. I want to retain 50% information for each variable then new binary value of the variables will be 0011 and 1100 (here LSB is changing). We can also understand this idea by observing the picture form of crossover which is given below.

Picture form of crossover -



Thus by changing the cross over point we can change the % of information exchange. In my MATLAB program I have taken cross over point at the middle so that 50% information will exchange.

In GA literature, the term crossover rate is usually denoted as Pc, the probability of crossover. The probability varies from 0 to 1. After the crossover, with cross over probability Pc, Pc percentage of information will exchange and (1-Pc) percentage of information will remain same for each variable which will go through this operation. Cross over operator is mainly responsible for the search of new string.

There exist many types of cross over operations in the genetic algorithm which are given below:

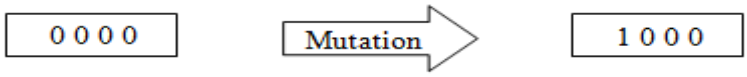
Single - point cross over.

Two - point crossover.

Multi - point cross over.

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 ≈ 1.78%). It is considered a background operator in GA. Once a string is selected for mutation, a randomly chosen element of the string is changed. For example If GA chooses bit position 1^{th} (from MSB side) for mutation in the binary string, the resulting string is 1000 (picture form is also given below).

Picture form of mutation -



.

2.6 Steps of Genetic Algorithm Based Optimization

1. Generate the initial population.
2. Fitness evaluation.
2. Selection.
4. New population generation by crossover and mutation.
5. Fitness evaluation.
6. Repeat step 2-5 until stopping criteria is reached.

2.7 Results

For simulation study, we consider the following systems with dead-time (*L*)

$G_p(s) = e^{-Ls}/(s+1)^2$, L=0.2s, and 0.3s (2.3)

$G_p(s) = e^{-Ls}/s(s+1)$, L=0.2s, and 0.3s (2.4)

$G_p(s) = (1-\beta s)/(1+s)^3$, β = 0.1s, and 0.2s (2.5)

$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.2y^2 = u(t-L)$, L=0.3s, and 0.4s (2.6)

PH model- $G_p(s) = e^{-Ls}/(s+1)(0.1s+1)^2$, L=0.01s, and 0.02s (2.7)

For each process model we have used four different types of controller.

(a) **PID**

(b) **GA-PID** – K_p, K_i, K_d are ±20% of their respective Conventional PID and these are calculated by genetic algorithm.

(c) **APID.**

(d) **GA-APID** - In this all seven parameters are varying within the defined range and these are calculated by genetic algorithm.

We have calculated the close loop response characteristics for above process model by using different controllers. 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 absolute error (IAE) and integral time absolute error (ITAE) are calculated for each controller. Performance of our GA-APID is compared with PID, GA-PID, and APID. Fourth-order Range - Kutta method is used for numeric integration. The detailed performance analysis for various types of process is discussed below.

2.7.1 Second Order Linear Process

Transfer function of the process is given by

$$G_p(s) = e^{-Ls}/(s+1)^2 \tag{2.8}$$

Response of second order linear process in (2.8) with L=0.2s, and L=0.3s under PID, GA-PID, APID, and GA-APID is shown in Fig. 2.2. Performance indices of the process in (2.8) for different controllers are given in Table 2.2. (a) and Table 2.2. (b). Though the controller are tuned for L=0.2s, a higher value i.e., L=0.3s is also tested without changing controllers settings (for PID and APID). Performance analysis reveals that unlike PID, GA-PID, and APID, GA-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance.

Table 2.2 (a) -*Performance analysis of second order linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE+ITAE	**PID**	60.30	0.90	4.40	2.08	9.29
		GA-PID	44.07	0.80	2.20	1.58	6.14
		APID	5.37	1.40	4.20	1.36	6.26
		GA-APID	0.85	2.10	1.90	1.36	4.64
L=0.3s		**PID**	93.95	0.90	7.90	4.08	23.11
		GA-PID	78.68	0.90	7.20	2.77	14.00
		APID	15.58	1.00	5.70	1.81	9.63
		GA-APID	1.54	2.50	6.10	1.79	7.59

Table 2.2 (b) -*Performance analysis of second order linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE	**PID**	60.30	0.90	4.40	2.08	9.29
		GA-PID	34.36	0.90	2.80	1.58	6.35
		APID	5.37	1.40	4.20	1.36	6.26
		GA-APID	0.00	9.10	7.90	1.63	5.99
L=0.3s		**PID**	93.95	0.90	7.90	4.08	23.11
		GA-PID	65.81	0.90	6.90	2.88	12.34
		APID	15.58	1.00	5.70	1.81	9.63
		GA-APID	0.00	9.20	7.90	2.03	9.69

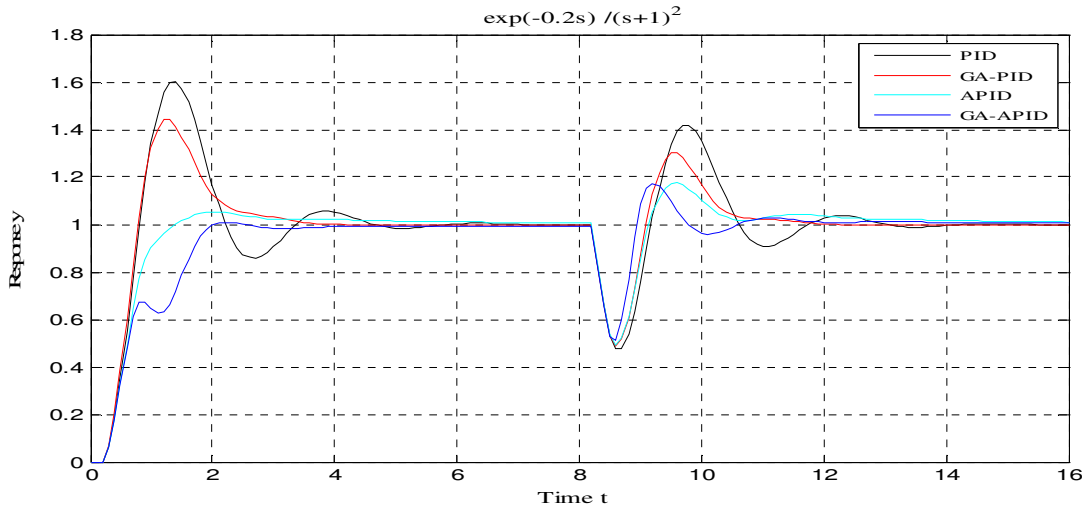


Fig. 2.2 (a) *Response of second order linear process for L=0.2s, minimization of IAE+ITAE*

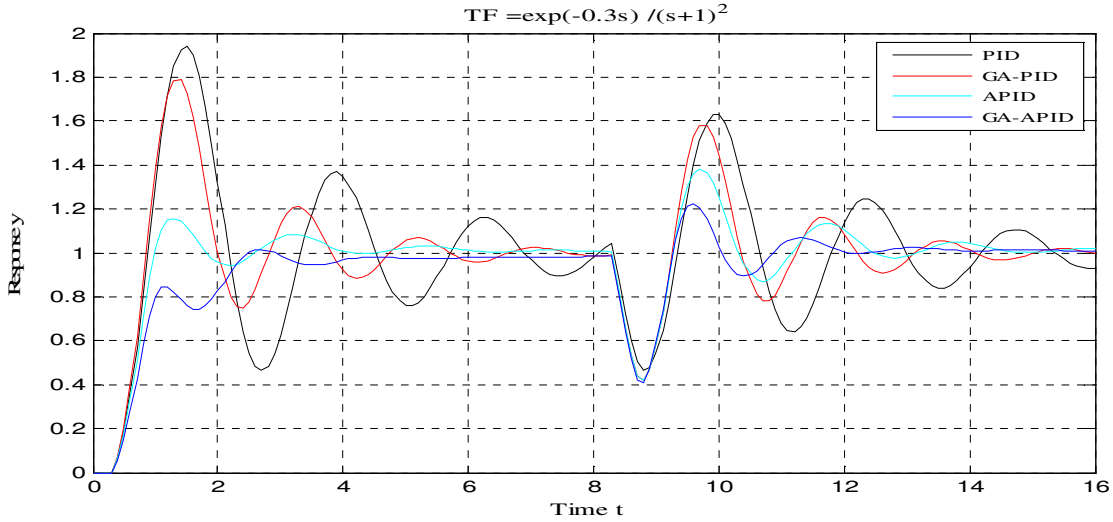


Fig. 2.2(b) *Response of second order linear process for L=0.3s, minimization of IAE+ITAE*

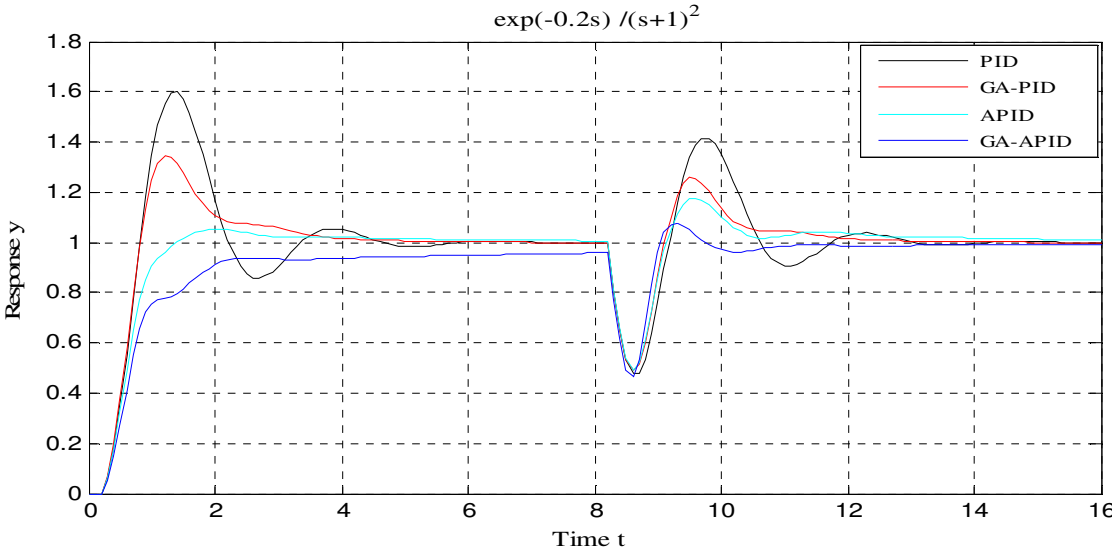


Fig. 2.2(c) *Response of second order linear process for L=0.2s, minimization of IAE*

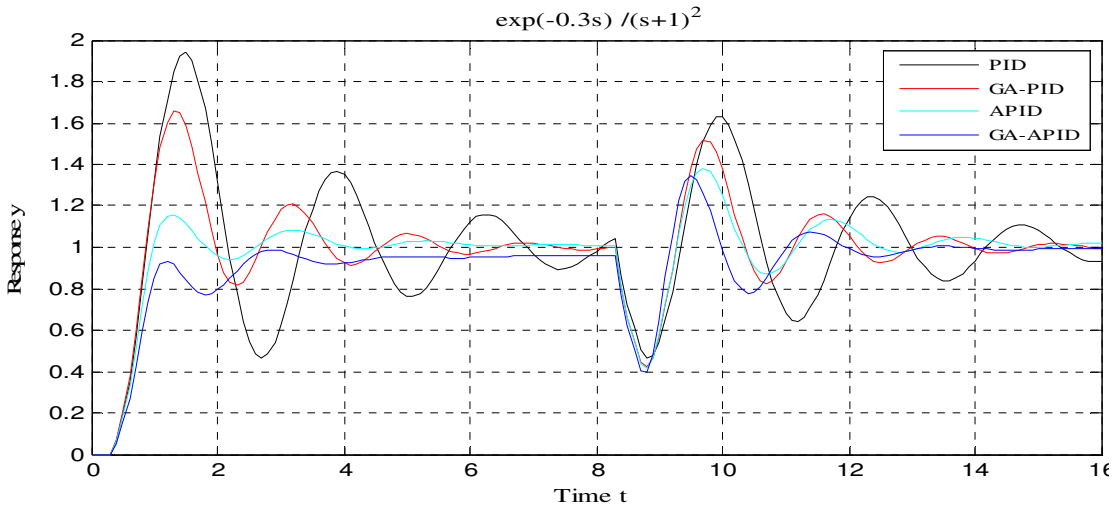


Fig. 2.2(d) *Response of second order linear process for L=0.3s, minimization of IAE*

2.7.2 First Order with Integrating Process

Transfer function of the process is given by

$$G_p(s) = e^{-Ls} / s(s+1) \tag{2.9}$$

Response of first order integrating process in (2.9) with L=0.2s and L=0.3s under PID, GA-PID, APID, and GA-APID is shown in Fig. 2.3 Performance indices of the process in (2.9) for different controllers are shown in Table 2.2(a) and Table 2.2(b). Though the controller are tuned for L=0.2s a higher value i.e., L=0.3s is also tested without changing controllers settings (for PID and APID).

Unlike PID, GA-PID, and APID, GA-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance but not to the same extent as that of the previous case.

Table 2.3 (a) -*Performance analysis of first order integrating process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE+ITAE	**PID**	77.50	1.10	10.20	2.44	27.19
		GA-PID	54.41	1.00	5.40	1.92	12.09
		APID	28.75	1.40	11.00	2.46	19.44
		GA-APID	8.35	1.10	17.40	1.79	17.99
L=0.3s		**PID**	102.2	1.20	17.10	5.70	54.79
		GA-PID	83.02	1.00	9.70	2.15	24.13
		APID	33.80	1.30	11.00	2.68	22.02
		GA-APID	29.43	1.10	17.40	2.47	21.56

Table 2.3(b) -*Performance analysis of first order integrating process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE	**PID**	77.50	1.10	10.20	2.44	27.19
		GA-PID	51.47	1.00	5.60	1.92	12.27
		APID	28.75	1.40	11.00	2.46	19.44
		GA-APID	6.31	1.20	17.40	1.81	18.42
L=0.3s		**PID**	102.2	1.20	17.10	5.70	54.79
		GA-PID	79.48	1.00	9.70	2.02	22.88
		APID	33.80	1.30	11.00	2.68	22.02
		GA-APID	26.02	1.10	17.40	2.49	25.81

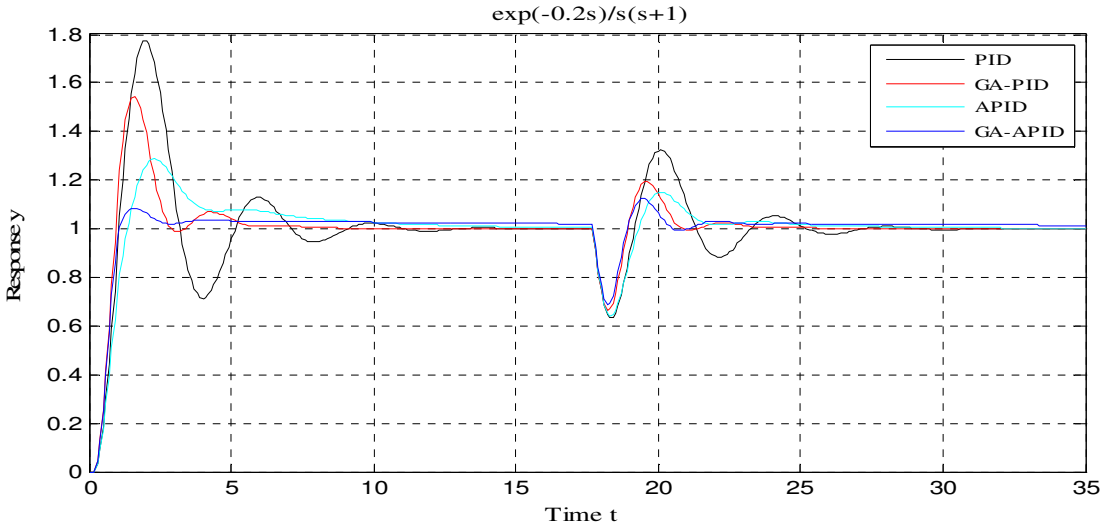


Fig. 2.3 (a) *Response of first order integrating process for L=0.2s, minimization of IAE+ITAE*

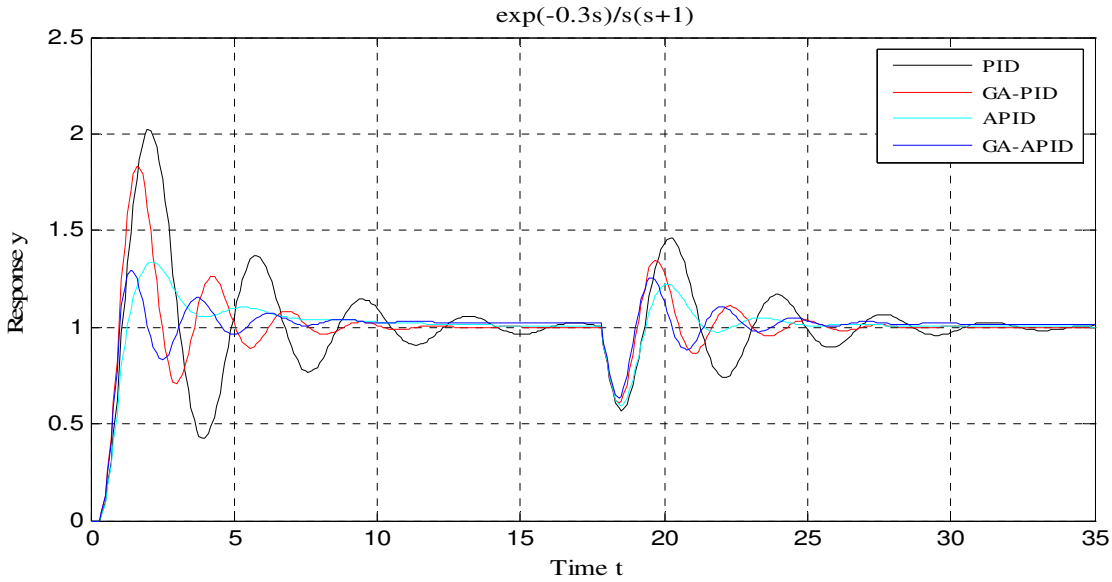


Fig. 2.3 (b) *Response of first order integrating process for L=0.3s, minimization of IAE+ITAE*

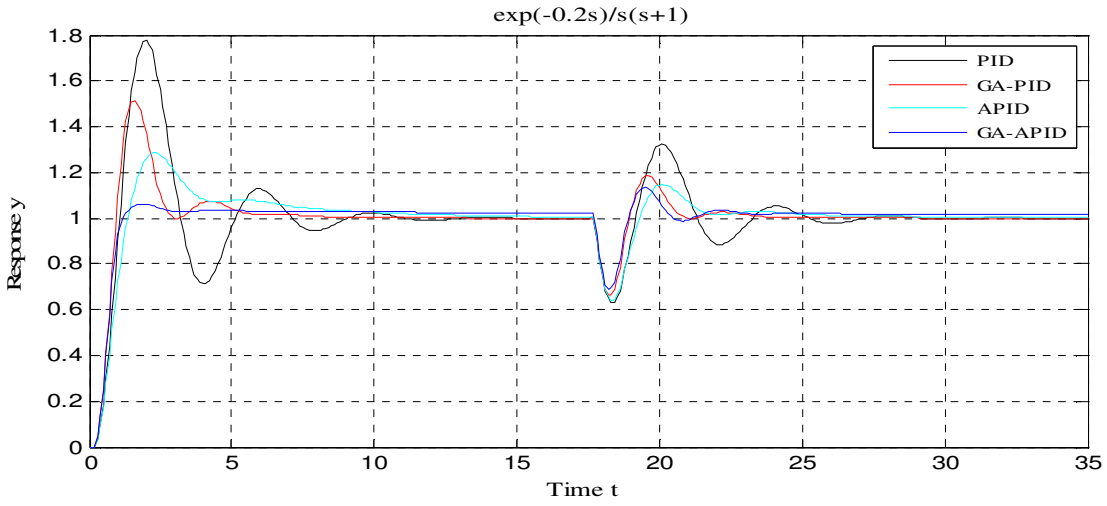


Fig. 2.3 (c) *Response of firstorder integrating process for L=0.2s, minimization of IAE*

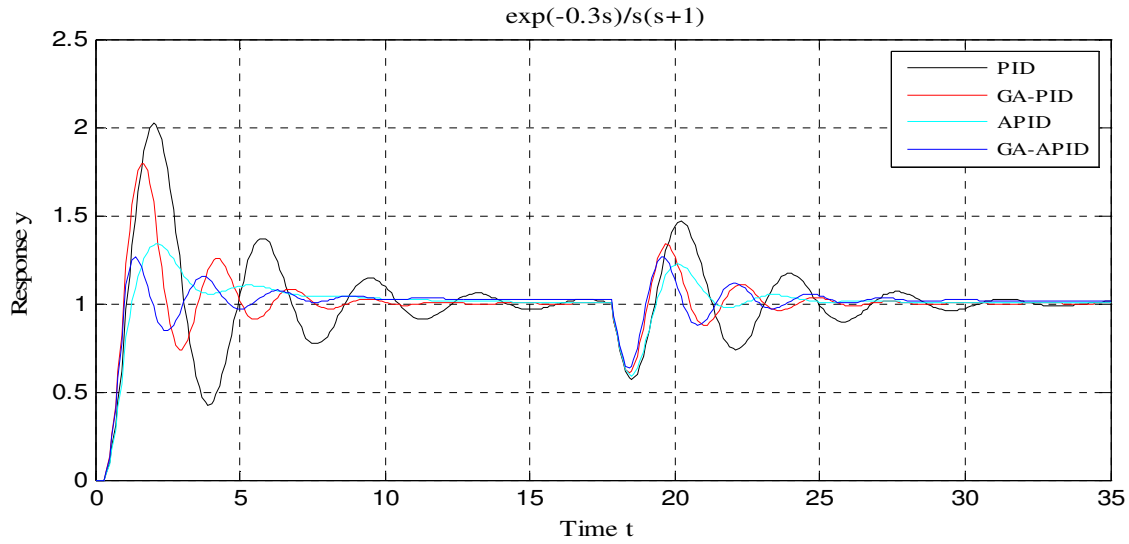


Fig. 2.3 (d) *Response of first order integrating process for L=0.3s, minimization of IAE*

2.7.2 Third Order Linear Process

Transfer function of the process is given by

$$G_p(s) = (1-\beta s)/(1+s)^3 \qquad (2.10)$$

Response of third order linear process in (2.10) with L=0.1s and L=0.2s under PID, GA-PID, APID, and GA-APID is shown in Fig. 2.4 Performance indices of the process in (2.10) for

different controllers are recorded in Table 2.4(a) and Table 2.4(b). Though the controller are tuned for L=0.1s, a higher value i.e., L=0.2s is also tested without changing controllers settings (for PID and APID). Performance analysis reveals that unlike PID, GA-PID, and APID, GA-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance. For all conditions (i.e., different *L* and different objective) response of GA-APID is very much improved..

Table 2.4 (a) -*Performance analysis of third order linear process*

Dead time	**Objective function**	**Controllers**	**%OS**	**T_r(s)**	**T_s(s)**	**IAE**	**ITAE**
L=0.1s	**IAE+ITAE**	**PID**	46.47	1.80	9.60	2.29	30.63
		GA-PID	29.85	1.70	7.00	2.42	20.09
		APID	2.44	2.40	10.30	2.67	26.14
		GA-APID	1.48	2.30	4.40	2.03	18.70
L=0.2s		**PID**	58.30	1.80	12.30	4.29	44.44
		GA-PID	40.73	1.70	10.00	2.00	27.26
		APID	7.91	2.20	9.70	2.90	29.62
		GA-APID	6.42	2.00	7.40	2.39	23.25

Table 2.4 (b) -*Performance analysis of third order linear process*

Dead time	**Objective function**	**Controllers**	**%OS**	**T_r(s)**	**T_s(s)**	**IAE**	**ITAE**
L=0.1s	**IAE**	**PID**	46.47	1.80	9.60	2.29	30.63
		GA-PID	29.58	1.70	7.00	2.42	20.09
		APID	2.44	2.40	10.30	2.67	26.14
		GA-APID	1.48	2.30	4.60	2.13	20.14
L=0.2s		**PID**	58.30	1.80	12.30	4.29	44.44
		GA-PID	40.73	1.70	10.10	2.14	29.53
		APID	7.91	2.20	9.70	2.90	29.62
		GA-APID	4.01	2.20	8.10	2.51	24.21

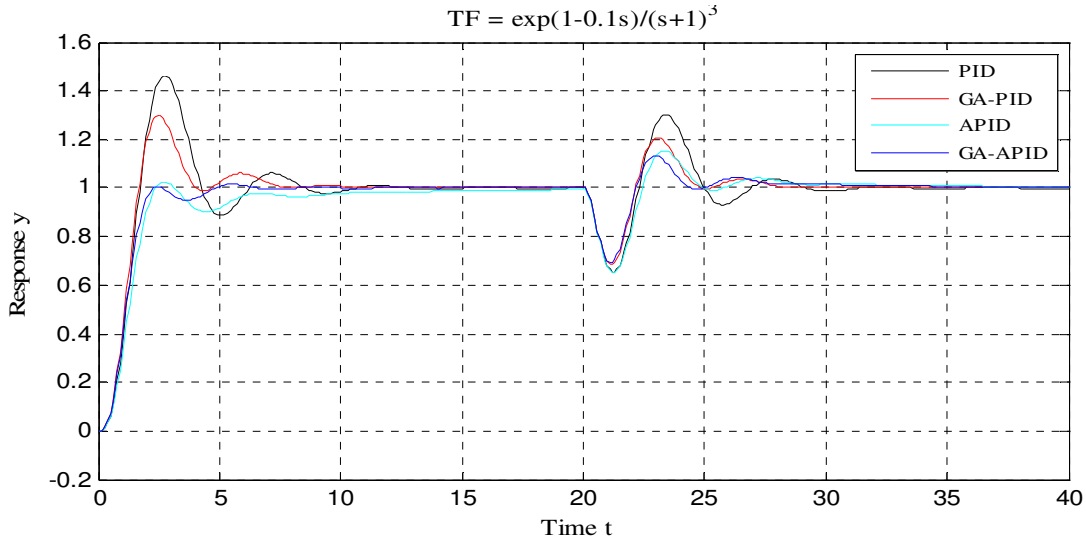


Fig. 2.4 (a) *Response of third order linear process for L=0.1s, minimization of IAE+ITAE*

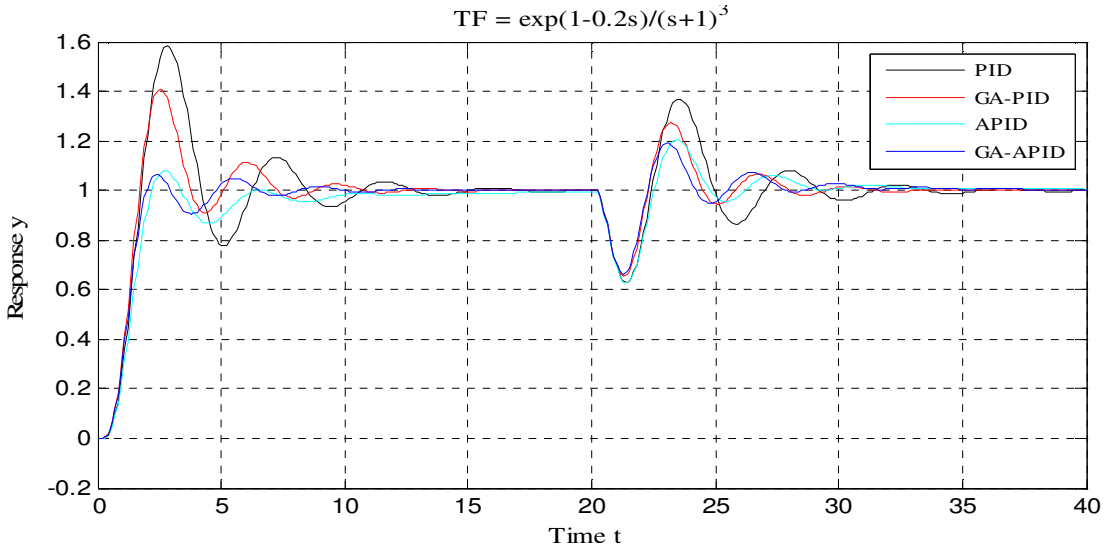


Fig. 2.4 (b) *Response of third order linear process for L=0.2s, minimization of IAE+ITAE*

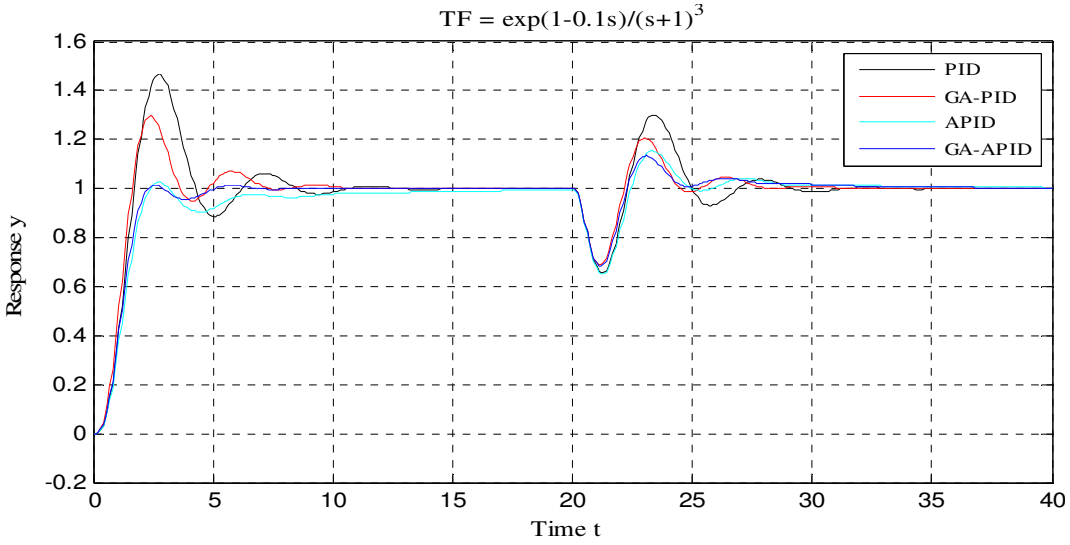


Fig. 2.4(c) *Response of third order linear process for L=0.1s, minimization of IAE*

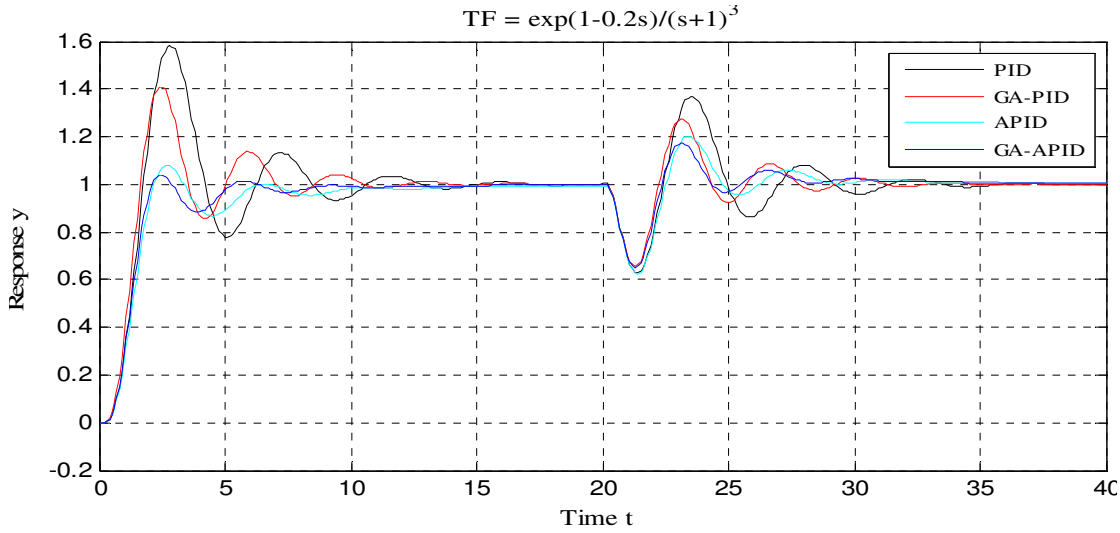


Fig. 2.4(d) *Response of third order linear process for L=0.2s, minimization of IAE*

2.7.4 Second Order Nonlinear Process

$$\frac{d^2y}{dt^2}+\frac{dy}{dt}+0.2y^2=u(t-L) \tag{2.11}$$

Response of second order non linear process in (2.11) with L=0.3s and L=0.4s under PID, GA-PID, APID, and GA-APID is shown in Fig. 2.5. Performance indices of the process in (2.11) for different controllers are given in Tables 2.5(a) and 2.5(b). Though the controller are tuned for L=0.3s, a higher value i.e., L=0.4s is also tested without changing controllers

settings (for PID and APID). We can conclude that unlike PID, GA-PID, and APID, GA-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance.

Table 2.5 (a) -*Performance analysis of second order non linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.3s	IAE+ITAE	**PID**	66.10	1.40	9.30	4.13	46.60
		GA-PID	50.28	1.30	6.20	2.80	28.25
		APID	19.18	1.70	10.60	2.95	34.88
		GA-APID	0.78	2.20	2.90	2.12	16.17
L=0.4s		**PID**	83.11	1.50	13.30	5.78	72.4
		GA-PID	70.77	1.30	9.80	4.07	46.03
		APID	25.15	1.70	10.80	2.26	39.45
		GA-APID	5.54	2.30	5.70	2.74	27.89

Table 2.5 (b) -*Performance analysis of second order non linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.3s	IAE	**PID**	66.10	1.40	9.30	4.13	46.60
		GA-PID	45.75	1.40	6.40	2.99	29.87
		APID	19.18	1.70	10.60	2.95	34.88
		GA-APID	0.78	2.20	2.90	2.12	16.17
L=0.4s		**PID**	83.11	1.50	13.30	5.78	72.4
		GA-PID	61.20	1.40	6.50	2.41	35.42
		APID	25.15	1.70	10.80	2.26	39.45
		GA-APID	5.54	2.30	5.70	2.74	27.89

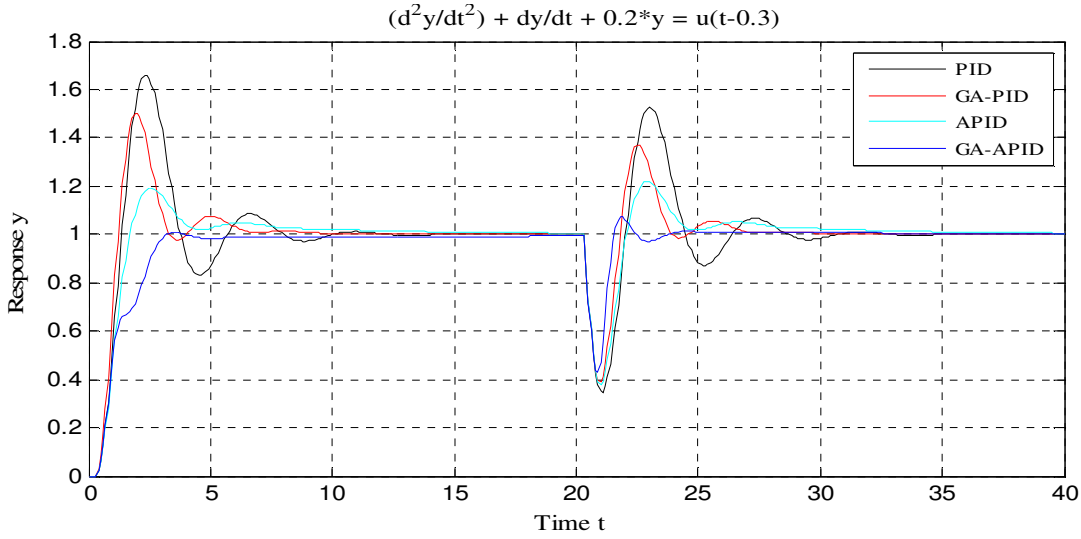


Fig. 2.5 (a) *Response of second order non linear process for L=0.3s, minimization of IAE+ITAE*

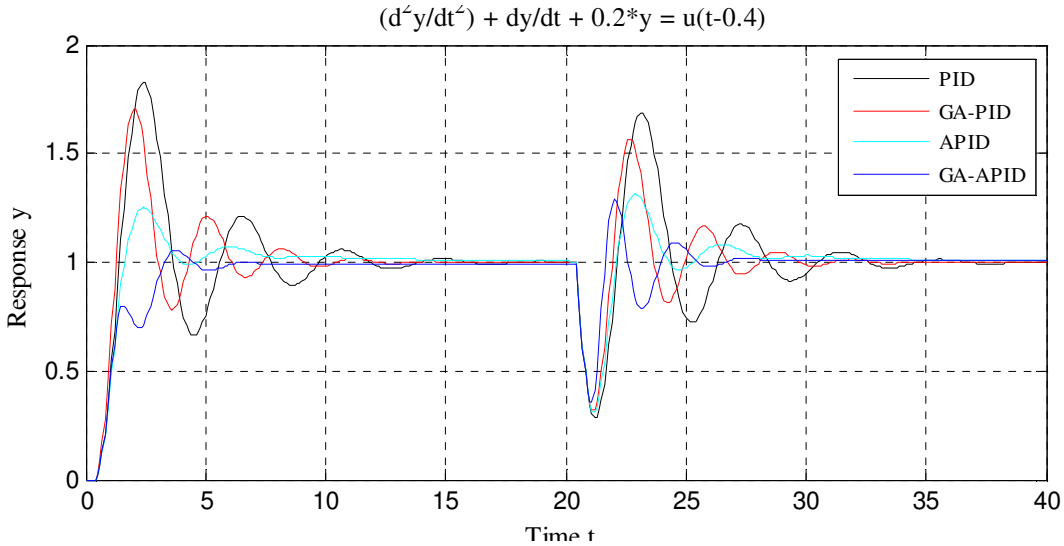


Fig. 2.5 (b) *Response of second order non linear process for L=0.4s, minimization of IAE+ITAE*

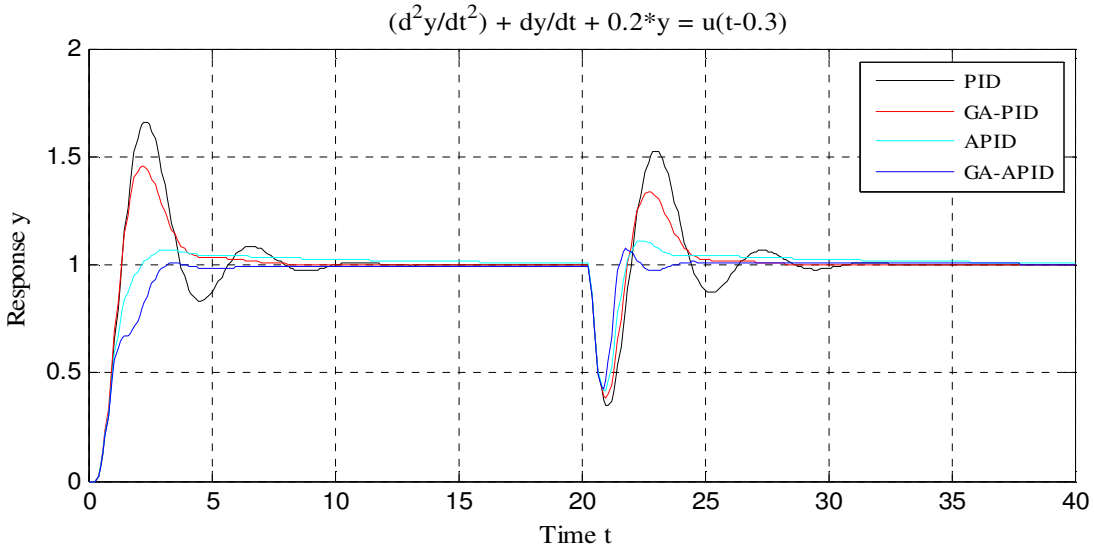


Fig. 2.5 (c) *Response of second order non linear process for L=0.3s, minimization of IAE*

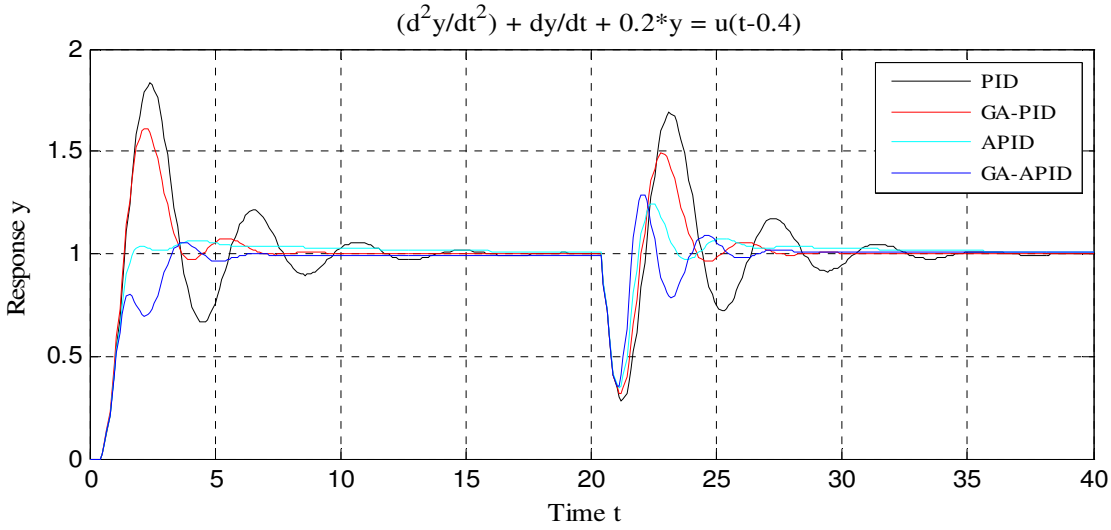


Fig. 2.5 (d) *Response of second order non linear process for L=0.4s, minimization of IAE*

2.7.5 pH-Neutralization Process

For pH-neutralization process we consider following linear model

$$G_p(s) = e^{-Ls}/(s+1)(0.1s+1)^2 \qquad (2.12)$$

Response of pH-neutralization process in (2.12) with L=0.01s and L=0.02s under PID, GA-PID, APID, and GA-APID is shown in Fig. 2.6. Performance indices of the process in (2.12) for different controllers are provided in Tables 2.6(a) and 2.6(b). Though the controller are

tuned for L=0.01s, a higher value i.e., L=0.02s is also tested without changing controllers settings (for PID and APID). We can conclude that unlike PID, GA-PID, and APID, GA-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance

Table 2.6 (a) *-Performance analysis of for pH neutralization process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.01s	IAE+ITAE	**PID**	64.21	0.26	1.69	0.72	1.40
		GA-PID	44.48	0.23	1.14	0.47	0.62
		APID	23.94	0.30	1.77	0.52	0.84
		GA-APID	0.49	0.36	1.19	0.33	0.41
L=0.02s		**PID**	72.34	0.27	1.78	0.84	1.33
		GA-PID	52.86	0.24	1.59	0.54	0.75
		APID	28.49	0.2	1.84	0.56	0.88
		GA-APID	2.17	0.32	1.22	0.37	0.48

Table 2.6 (b) *-Performance analysis of for pH neutralization process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.01s	IAE	**PID**	64.21	0.26	1.69	0.72	1.09
		GA-PID	43.79	0.23	1.49	0.46	0.62
		APID	23.94	0.30	1.77	0.52	0.84
		GA-APID	0.31	0.38	0.75	0.33	0.41
L=0.02s		**PID**	72.34	0.27	1.78	0.84	1.33
		GA-PID	52.35	0.23	1.59	0.54	0.76
		APID	28.49	0.2	1.84	0.56	0.88
		GA-APID	1.72	0.33	1.20	0.37	0.48

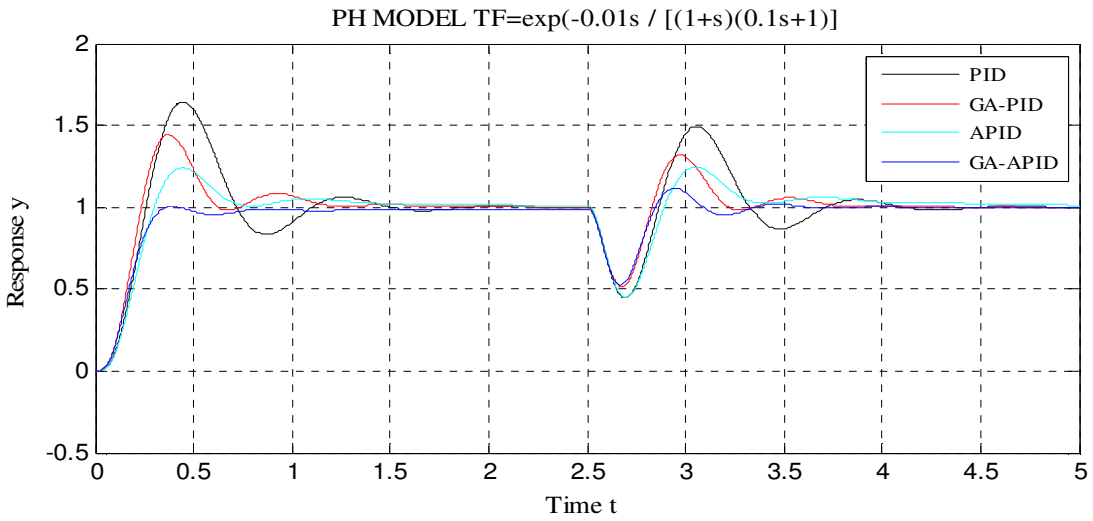


Fig. 2.6 (a) *Response of pH neutralization process for L=0.01s, minimization of IAE+ITAE*

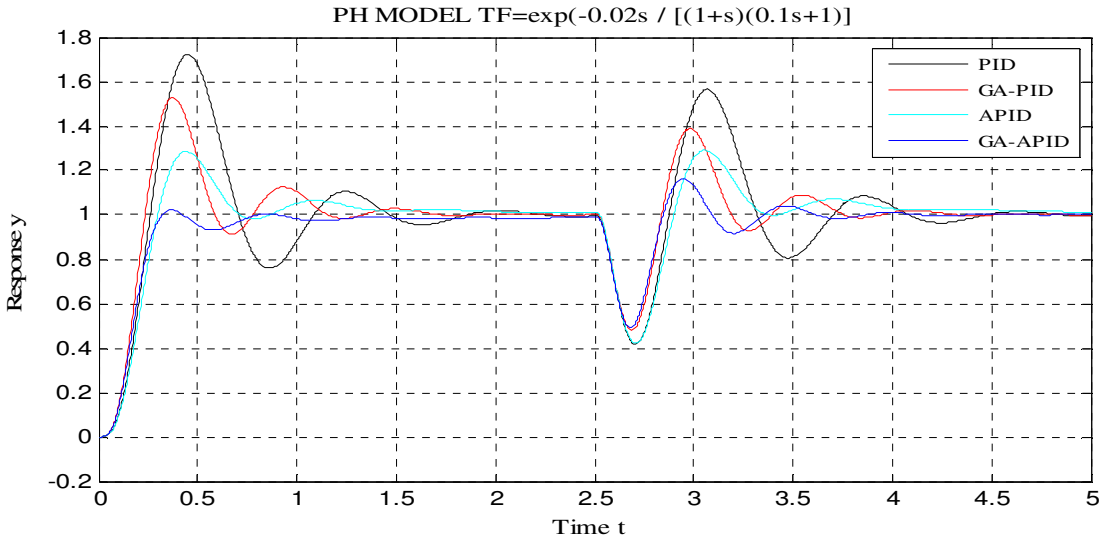


Fig. 2.6(b) *Response of pH neutralization process for L=0.02s, minimization of IAE+ITAE*

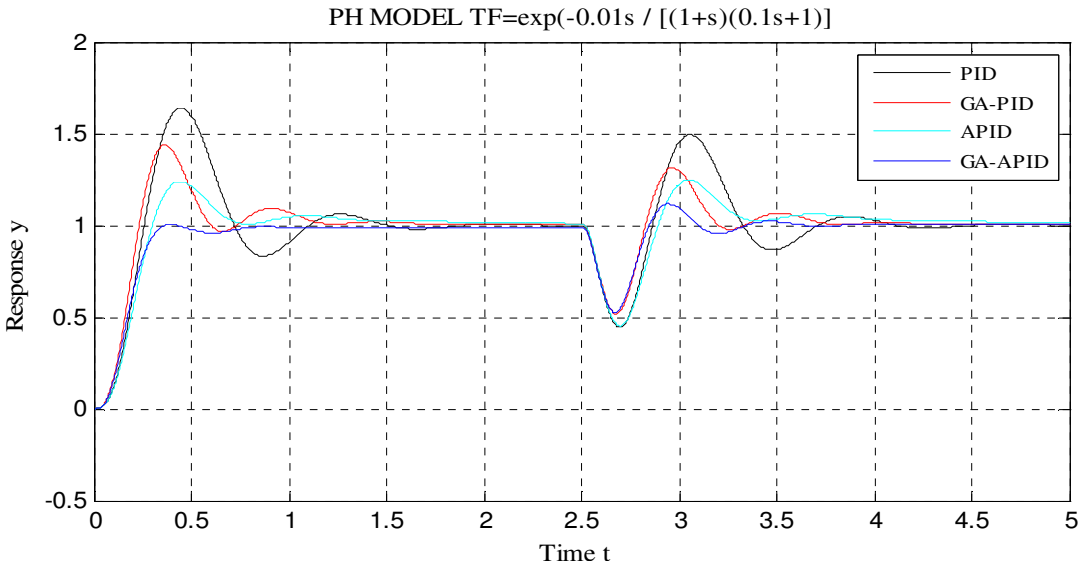


Fig. 2.6 (c) *Response of pH neutralization process for L=0.01s, minimization of IAE*

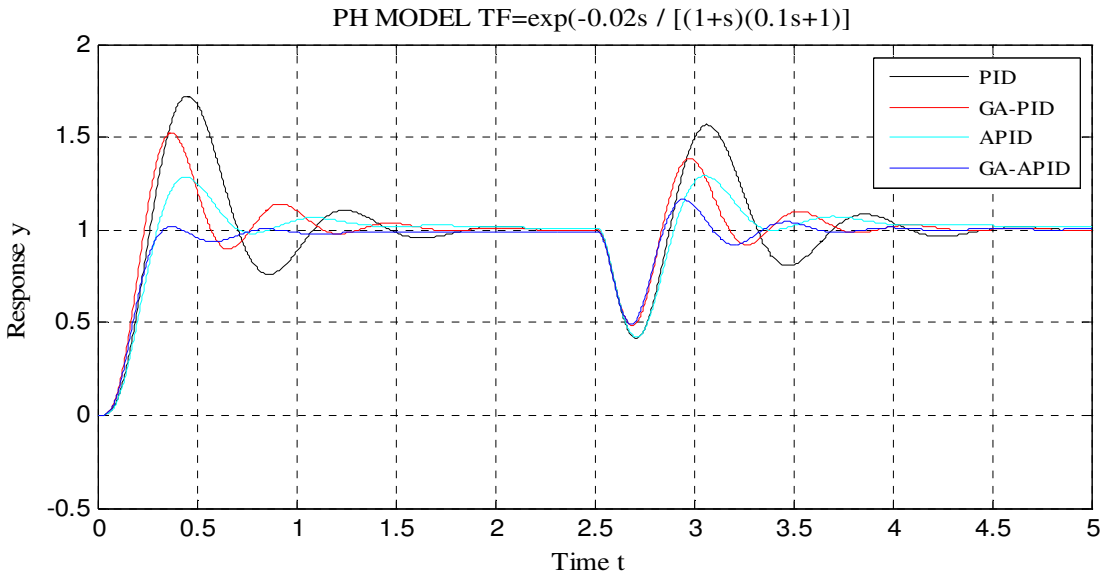


Fig. 2.6 (d) *Response of pH neutralization process for L=0.02s, minimization of IAE*

2.8 Conclusion

In this study we have explored the possibility of performance enhancement of Adaptive PID controller (APID) through real coded Genetic algorithm optimization technique. From the experimental results we observed that both GA-APID provided remarkably enhanced performance over Adaptive PID controller (APID) due to set point changes as well as load

disturbances. We also observed that genetic algorithm based Adaptive PID controller (GA-APID) exhibited comparatively better performance over GA optimized PID controller (GA-PID) for all the systems taken above. GA-APID provided minimum overshoot as well as minimum IAE & ITAE. Initially we have used IAE+ITAE as objective function in order to obtain both load disturbances and set point tracking satisfactorily i.e. overall improved performance with a dead time. After that we increased the dead time in order to see the robustness of the controller and we still found that the proposed GA-APID maintained overall improved performances. We also observed the similar enhanced performance with the objective function IAE.

References

[1] D.E. Goldberg, Genetic algorithm in search optimization and machine learning. Addison-Wesley Publishing Co., Inc.(1989).

[2] David. E. Goldberg, john H .Holland, on genetic algorithms and machine learning, machine learning, N0.2, 95-99, (1988).

[3] Randy L. Haupt, and Sue Ellen Haupt, Practical Genetic Algorithms, Wiley publishing Co. Second edition (2004).

[4] J. Bala, J. Huang, H. Vafaie, K. DeJong and H. Wechsler, on Hybrid learning using Genetic algorithms and decision trees for pattern classification, IJCAI conference, Montreal, August 19-25, (1995).

[5] S.Rajasekaran, and G.A.Vijaylaxmi Pai, Neutral Networks, Fuzzy logic, and Genetic Algorithms, PHI publishing co., sixth edition (2003).

[6] James McCaffrey, http://msdn.microsoft.com/en-us/magazine/hh335067.aspx

[7] C.Dey and R.K. Mudi, An improved auto-tuning scheme for PID controllers. ISA Trans. 48(4), 396–409(2009).

[8] S. Verma and R.K. Mudi, Genetic algorithm based adaptive PID controller, Proc. International Conference on Intelligent Computing, Communication and Devices (ICCD-2014), ASIC 2014, April 18 -19, 2014.

[9] Seborg, D.E., Edgar. T.F.,; Adaptive control strategies for process control: A survey. AICHE J. 32(6), 881–913 (1986).

CHAPTER-3

BACTERIAL FORAGING OPTIMIZATION ALGORITHM BASED ADAPTIVE PID CONTROLLER

3.1 Introduction

The bacterial foraging optimization (BFO) [1] algorithm mimics how bacteria forage over a landscape of nutrients to perform parallel non-gradient optimization which has been widely accepted as a global optimization algorithm of current interest for distributed optimization and control. Since its inception, BFOA has drawn the attention of researchers from diverse fields of knowledge especially due to its biological motivation and graceful structure. It has already been applied to many real world problems and proved its effectiveness over many variants of BFO and PSO.[2, 3] Applications to fuzzy controller construction/tuning, neural network training, job-shop scheduling, electromagnetic, stock market predication, optimal power flow, motor control, temperature control, system identification, and others have not received as much attention to-date. On the other hand, additional applications and studies of the method still holds potential. There is still a wide variety of domains in which BFO could be useful for. This fact motivated us to design this adaptive PID controller (APID) [4, 5] by using BFO algorithm.

3.2 E. Coli. Bacteria

Escherichia coli [2, 3] are single-celled bacteria that live in our gut. The E. coli cell only weights about 1 picogram, and is composed of about 70% water. It is equipped with a set of rotary motors only 45 nm in diameter. Each motor drives a long, thin, helical filament that extends several cell body lengths out into the external medium. The assemblage of motor and filament is called a flagellum. [6] The concerted motion of several flagella - decision-making sensors- enables a cell to swim. A cell can move toward regions that it deems more favorable by measuring changes in the concentrations of certain chemicals in its environment (mostly nutrients), deciding whether life is getting better or worse, and then modulating the direction of rotation of its flagella, i.e., control system of E. coli bacterium enables it to search for food and try to avoid noxious substances. Such motions of E. coli are called taxes where motile behavior depends on the flagella (the actuator).

3.3 Bacterial Foraging Optimization Algorithm

The bacterial foraging [7, 8] system consists of four principal mechanisms, namely chemo taxis, swarming, reproduction, and elimination dispersal; and it works on the assumption that animal search for and obtain nutrients in a way that maximizes their energy intake E per unit time T spent foraging, i.e., animal search for and obtain nutrients in a way that maximizes the ratio E/Tor maximizes the long-term average rate of energy intake. Clearly, evolution optimizes the foraging strategies, since animals that have poor foraging performance do not survive.

As nutrients are distributed in patches, sequence of foraging strategy consists of finding a patch of food (e.g., group of bushes with barriers), deciding whether to enter it and search for food and when to leave the patch. During foraging there can be risks due to predators, the prey may be mobile so it must be chased, and the physiological characteristics of the forager constrain its capabilities and ultimately success. Generally, patches are encountered sequentially, and sometime great effort and risk are needed to travel from one patch to another. If an animal encounters a nutrient-poor patch but based on past experience it expects that there should be a better patch elsewhere, then it will consider risks and efforts to find another patch. For an animal there is optimal time to leave the patch and venture out to try to find a richer one since it does not wants to waste resources that are readily available as well as it also does not want to waste time in the face of diminishing energy returns.

Basically optimal foraging theory formulates the foraging scenarios as an optimization problem where via computation or analytical methods are provided as an optimal foraging policy that specifies how foraging decisions are made, for instance, dynamic programming. Search and optimal foraging decision-making of animals can be broken into three basic types: cruise (e.g. tuna fish, hawks), salutatory (e.g. birds, fish, lizards and insects), and ambush (e.g. snakes, lions), and since no animal can make optimal decisions this is why we can say that the optimal foraging formulation is only meant to be a model that explains what optimal behavior would be like.

3.4 Optimal Foraging Formulation

Suppose that we want to find the minimum of (J) θ, $\in R^P$, where we do not have measurements or an analytical description of the gradient ∇ (J) θ. To solve this non gradient optimization problem, here, we use bacterial foraging concept. If θ is the position of a bacterium and (J) θ represents the combined effects of attractants and repellants from the environment, then (J) $\theta < 0$, (J) $\theta = 0$, and (J) $\theta > 0$ representing that the bacterium at location θ is in nutrient-rich, neutral, and noxious environments, respectively. Basically, chemotaxis is a one of the principal mechanisms of bacterial foraging system which is implemented to a type of optimization where bacteria try to climb up the nutrient concentration (find lower and lower values of (J) θ), avoid noxious substances, and search for ways out of neutral media (avoid being at positions θ where $J(\theta) \geq 0$). Actually the concept based on biased random walk.

Chemotaxis- It is the tendency of a bacterium to move toward distant sources of nutrients. Biologically an E .coli bacterium can move in two different ways. In this process, the bacterium alternates between tumbling (changing direction) and swimming behaviors. Let j

be the index for the chemo tactic step. Let k be the index for the reproduction step. Let l be the index of the elimination-dispersal event. Let P(j ,k ,l)= { θ^i (j ,k ,l) |i= 1,2,.....,S}.represent the position of each member in the population of the S bacteria at the j^{th} chemotactic step, k^{th} reproduction step, and l^{th} elimination-dispersal event. Here, a tumble is represented by a unit walk with random direction φ (j) ∈ R^p. The unit length random direction may be represented as φ (j) = $\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$, where Δ(i) is a vector with each element a random number on [-1,1] A swim is indicated as movement in the same direction as the previous tumble. Here, let (i, j, k, l) denote the cost at the location of the i^{th} bacterium.

Suppose θ^i (j, k, l) represents the bacterium at j^{th} chemo tactic, k^{th} reproductive and l^{th} elimination-dispersal step. C (i) is the size of the step taken in the random direction specified by the tumble (run length unit). Then in computational chemotaxis the movement of the bacterium may be represented by

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\varphi(j) \tag{3.1}$$

If $\theta^i(j+1,k,l)$ the cost of the i^{th} bacterium J(I, j+1, k, l) is better (lower) than at $\theta^i(j,k,l)$, then another step size C(i) in this same direction will be taken. If that step resulted in a position with a better cost value than at the previous step, another step is taken. This swim is continued as long as it continues to reduce the cost, but only up to a maximum number of steps, N_s, where N_s number of swimming length. This represents that the cell will tend to keep moving if it is headed in the direction of increasingly favorable environments. With the activity of run or tumble taken at each step of the chemotaxis process, a step fitness, denoted as J (i, j, k, l), will be evaluated. In order to construct algorithm we introduce another chemotaxis related parameter which is the length of the lifetime of the bacteria as measured by the number of chemotactic steps they take during their life. We must note that in computer simulations, we will use much smaller population sizes and will keep the population size fixed. We will all allow, p>3 so we can apply the method to higher dimensional optimization problems.

Swarming- During foraging an interesting group behavior of E. coli bacteria as well as of other several motile species has been observed in order to examine the intricate and stable spatio-temporal patterns (swarms) formation in semisolid nutrient medium. If a group of E. coli cells is placed in the center of a chemotactic media with a single nutrient chemo effecter (sensor), they move out from the center in a traveling ring of cells by moving up the nutrient gradient to consume nutrient. Moreover the cells release attractant aspertate and congregate into groups with high density if high levels of the nutrient called succinate are used as the

nutrient. The cells provide an attraction signal to each other as a result they swarm together. Mathematically this swarming effect can be represented with

$$J_{cc}(\theta,(p(j,k,l)) = \sum_{i=1}^{S} J_{cc}^{i}(\theta,\theta^{i}(j,k,l))$$

$$= \sum_{i=1}^{S}[-d_{attract}\exp(-w_{attract}\sum_{m=1}^{P}(\theta_m - \theta_m^i)^2)] + \sum_{i=1}^{S}[-h_{repellent}\exp(-w_{repellent}\sum_{i=1}^{P}(\theta_m - \theta_m^i)^2)]$$

Where, J_{cc}(P (j, k, l)) is the objective valued function to be added to the actual objective function (to be minimized) to present a time varying objective function, S is the total number of bacteria, p is the number of variables to be optimized, which are present in each bacterium and $\theta = [\theta_1, \theta_2, \ldots . \theta_p]$ is a point in the p-dimensional search domain. The different coefficients $d_{attractant}$, $w_{attractant}$, $h_{repellent}$, $w_{repellent,}$ should be chosen properly. The values for these parameters are simply chosen to illustrate general bacterial behaviors, not to represent a particular bacterial chemical signaling scheme. The particular values of the parameters are chosen with the nutrient profile in mind.

Reproduction- The health status of each bacterium is calculated as the sum of the step fitness during its life, i.e., $\sum_{j=1}^{N_c} J(i,j,k,l)$ where N_c is the maximum step in a chemotaxis process. All bacteria are sorted in reverse order according to health status. The least healthy bacteria eventually die while each of the healthier bacteria (those yielding lower value of the objective function) asexually split into two bacteria, which are then placed in the same location, which are then placed in the same locations. Thus, the population of bacteria keeps constant. Basically after

N_c is the number of chemotactic steps, a reproduction step is taken. Let N_{re} be the number of reproduction steps to be taken and assume that S is a positive even integer and also let S_r=S/2 be the number of population members who have had sufficient nutrients so that they will reproduce (split in two) with no mutations. For reproduction, the population is sorted in order of ascending accumulated cost and then the S_r least healthy bacteria die and the other S_r healthy bacteria each split into two bacteria, which are placed at the same location. This keeps swarm size constant.

Elimination and Dispersal- The chemotaxis provides a basis for local search, and the reproduction process speeds up the convergence which has been simulated by the BFOA. While to a large extent, only chemotaxis and reproduction are not enough for global optima searching.

Since bacteria may get stuck around the initial positions or local optima, it is possible for the diversity of BFO algorithm to change either gradually or suddenly to eliminate the accidents of being trapped into the local optima. Gradual or sudden changes in the local environment where a bacterium population lives may occur due to various reasons e.g. a significant local rise of temperature may kill a group of bacteria that are currently in a region with a high concentration of nutrient gradients. In BFO algorithm, the dispersion event happens after a certain number of reproduction processes. Then some bacteria are chosen, according to a preset probability P, to be killed and moved to another position within the environment.

In our case we have decided to use the optimal power of Bacterial foraging optimization algorithm (BFO) to design our proposed Adaptive PID (BFO-APID) [3] controllers. Here, all the above said designs including two steps- first, we define the structure of the adaptive PID controller and then the algorithms are used to find their best set of parameters with respect to an objective function. In our experimental purpose we have studied the performances of the developed adaptive controllers based on BFO algorithms over PID & APID controllers for different processes with dead time.

3.5 Simple steps of Bacteria foraging Optimization Algorithm

1. Initialize the population size.
2. Elimination-dispersal loop: $l = l+1$
3. Reproduction loop: $k = k+1$
4. Chemotaxis loop: $j = j+1$
    a. For $i = 1, 2...S$ take a chemotactic step for bacterium i as follows.
    b. Compute fitness function $J(i, j, k, l)$.
    c. Save the best fitness function.
    d. Tumble: generate a random vector between [-1, 1].
    e. Compute the fitness function again and save the best one.
5. If $j < N_c$ (no. of bacteria in the population) go to step 3. Since the life of the bacteria is not over.
6. Reproduction.
7. If $k < N_{re}$ (no. of reproduction steps) 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 chemotaxis step.

Elimination-dispersal loop with probability P_{ed} (elimination-dispersal probability). To do this, if a bacterium is eliminated, simply disperse another one to a random location on the

quantization domain. If $l < N_{ed}$ (no. of elimination-dispersal events), then go to step 2; otherwise end.

3.6 Objective Function of the Bacteria foraging Optimization Algorithm

The function to be optimized is known as objective function. Here, minimization of the *integral-absolute-error* (*IAE*) [9] or *integral-time-absolute-error* (*ITAE*) or combination of both i.e., (*IAE+ITAE*) is defined as the objective function (performance index or fitness function). The *IAE* and *ITAE* are calculated as:

$$IAE = \int_0^t |e(t)|\, dt \tag{3.2}$$

$$IAE + ITAE = \int_0^t |e(t)|\, dt + \int_0^t t\,|e(t)|\, dt \tag{3.3}$$

3.7 Results

For simulation study, we consider the following systems with dead-time (L)

$G_p(s) = e^{-Ls}/(s+1)^2$, L=0.2s, and 0.3s (3.4)

$G_p(s) = e^{-Ls}/s(s+1)$, L=0.2s, and 0.3s. (3.5)

$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.2y^2 = u(t-L)$, L=0.3s, and 0.4s. (3.6)

PH model- $G_p(s) = e^{-Ls}/(s+1)(0.1s+1)^2$, L=0.01s, and 0.02s. (3.7)

For each process model we have used four different types of controller.

(a) **PID**

(b) **BFO-PID** $-K_p, K_i, K_d$ are ±20% of their respective Conventional PID and these are calculated by BFO algorithm.

(c) **APID.**

(d) **BFO-APID** - In this all seven parameters are varying within the defined range and these are calculated by BFO algorithm.

We have calculated the close loop response characteristics for above process model by using different controllers. 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 absolute error (IAE) and integral time absolute error (ITAE) are calculated for each controller. Performance of our BFO-APID is compared with Conventional PID, BFO-

PID, and APID. Fourth-order Range-Kutta method is used for numeric integration. The detailed performance analysis for various types of process is discussed below.

3.7.1 Second Order Linear Process

Transfer function of the process is given by

$$G_p(s) = e^{-Ls}/(s+1)^2 \tag{3.8}$$

Response of second order linear process in (3.8) with L=0.2s, and L=0.3s under PID, BFO-PID, APID, and BFO-APID is shown in Fig. 3.1. Performance indices of the process in (3.8) for different controllers are given in Table 3.1(a) and Table 3.1(b). Though the controller are tuned for L=0.2s, a higher value i.e., L=0.3s is also tested without changing controllers settings (for PID and APID). Performance analysis reveals that unlike PID, BFO-PID, and APID, BFO-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance.

Table 3.1 (a) -*Performance analysis of second order linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE+ITAE	**PID**	60.30	0.90	3.40	2.08	9.29
		BFO-PID	24.84	0.90	2.80	1.51	7.03
		APID	5.37	1.40	3.20	1.36	6.26
		BFO -APID	1.04	2.10	3.30	1.30	3.68
L=0.3s		**PID**	93.95	0.90	7.90	3.08	23.11
		BFO -PID	48.64	1.00	7.00	2.55	13.37
		APID	15.58	1.00	5.70	1.81	9.63
		BFO -APID	3.92	2.40	3.30	1.30	3.68

Table 3.1 (b) -*Performance analysis of second order linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE	**PID**	60.30	0.90	3.40	2.08	9.29
		BFO-PID	14.47	1.00	3.40	1.46	7.07
		APID	5.37	1.40	3.20	1.36	6.26
		BFO -APID	0.30	2.20	1.90	1.38	3.56
L=0.3s		**PID**	93.95	0.90	7.90	3.08	23.11
		BFO -PID	34.89	1.00	5.50	1.95	9.49
		APID	15.58	1.00	5.70	1.81	9.63
		BFO -APID	14.45	2.20	5.40	2.14	10.73

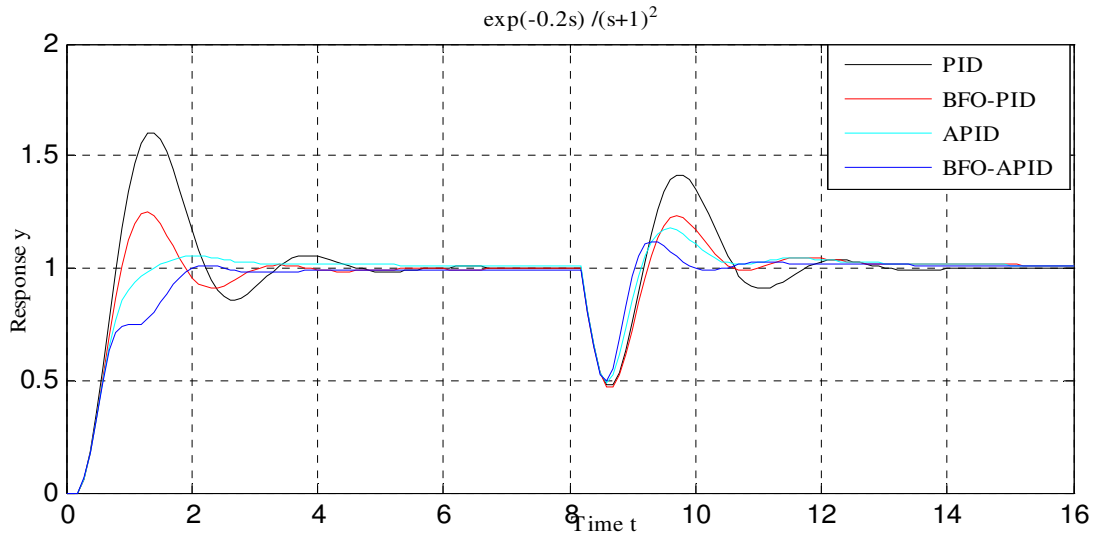


Fig. 3.1 (a) *Response of second order linear process for L=0.2s, minimization of IAE+ITAE*

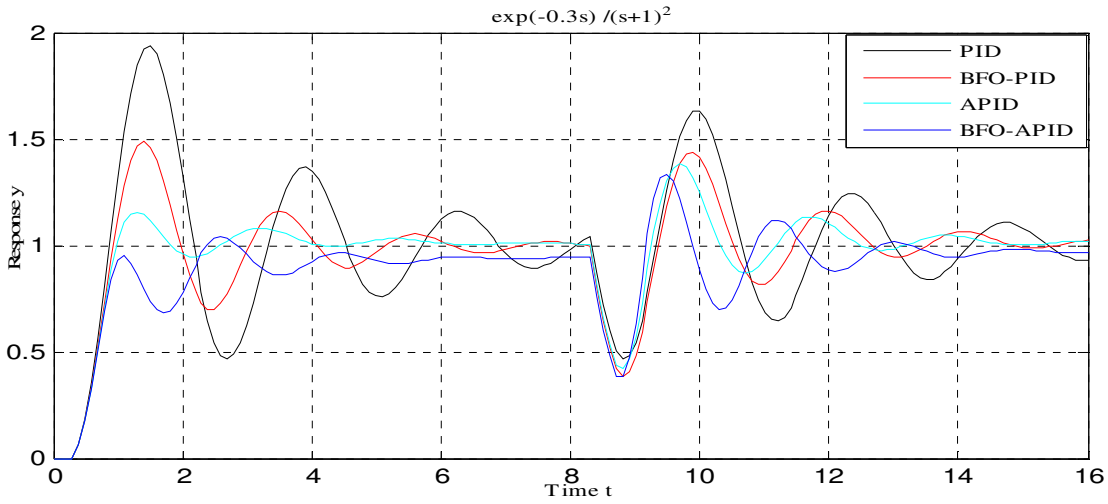


Fig. 3.1 (b) *Response of second order linear process for L=0.3s, minimization of IAE+ITAE*

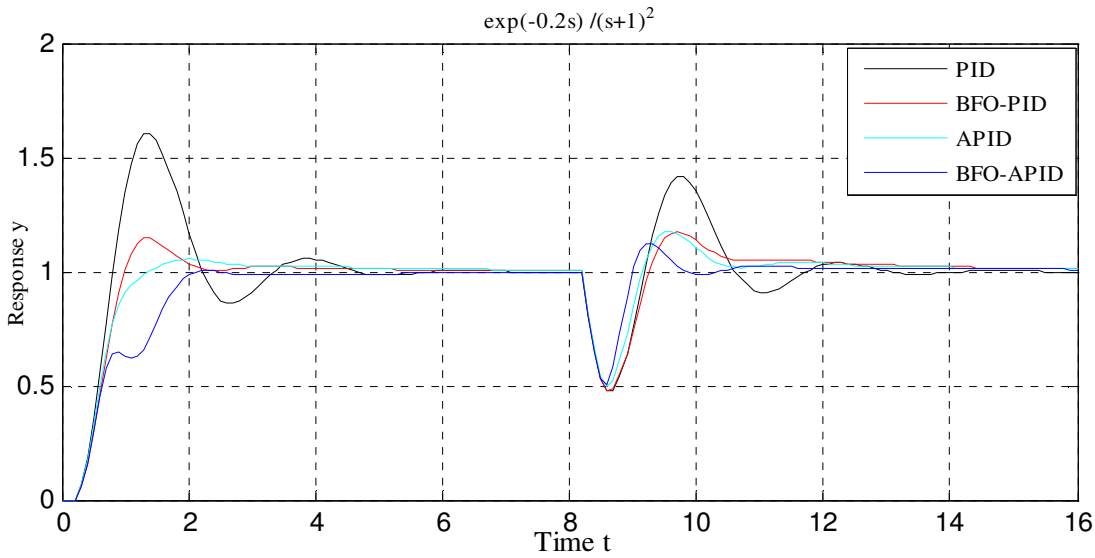


Fig. 3.1 (c) *Response of second order linear process for L=0.2s, minimization of IAE*

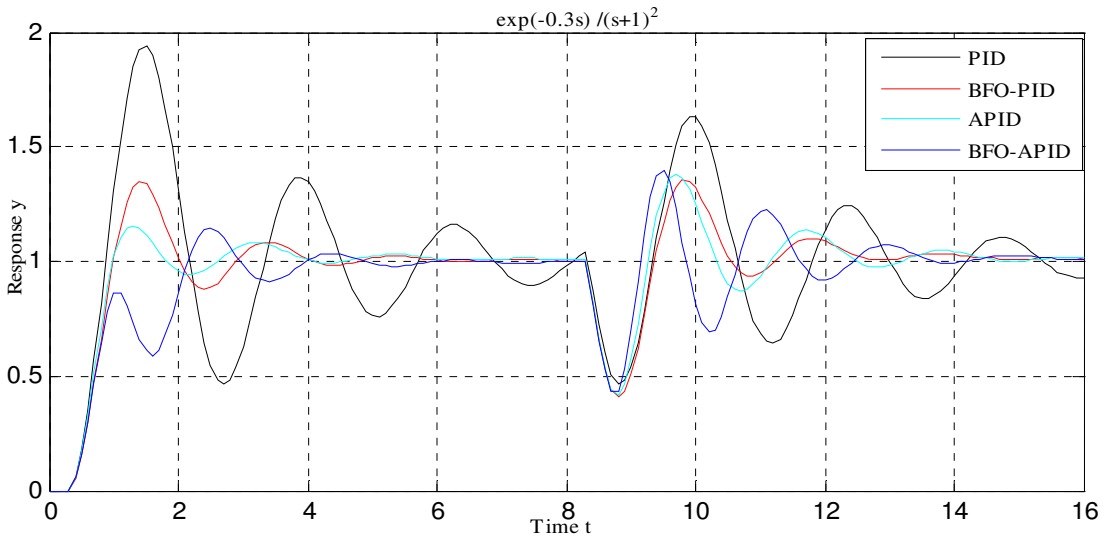


Fig. 3.1 (d) *Response of second order linear process for L=0.3s, minimization of IAE*

3.7.2 First Order with Integrating Process

Transfer function of the process is given by

$$G_p(s) = e^{-Ls} / s(s+1) \tag{3.9}$$

Response of first order integrating process in (3.9) with L=0.2s and L=0.3s under PID, BFO-PID, APID, and BFO-APID is shown in Fig. 3.2 Performance indices of the process in (3.9) for different controllers are shown in Table 3.2(a) and Table 3.2(b). Though the controller are tuned for L=0.2s a higher value i.e., L=0.3s is also tested without changing controllers settings (for PID and APID).

Unlike PID, BFO-PID, and APID, BFO-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance but not to the same extent as that of the previous case.

Table 3.2 (a) *-Performance analysis of first order integrating process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE+ITAE	**PID**	77.50	1.10	10.2	3.44	27.19
		BFO-PID	65.91	1.10	6.40	2.63	18.86
		APID	28.75	1.40	11.00	2.46	19.44
		BFO -APID	25.67	1.90	12.80	2.66	19.87
L=0.3s		**PID**	102.20	1.20	17.10	5.70	54.79
		BFO -PID	89.31	1.20	12.30	3.07	34.52
		APID	33.80	1.30	11.00	2.68	22.02
		BFO -APID	29.55	2.10	13.10	2.80	21.73

Table 3.2 (b) *-Performance analysis of first order integrating process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE	**PID**	77.50	1.10	10.20	3.44	27.19
		BFO-PID	65.91	1.10	6.40	2.63	18.86
		APID	28.75	1.40	11.00	2.46	19.44
		BFO -APID	25.67	1.90	12.80	2.66	19.87
L=0.3s		**PID**	102.20	1.20	17.10	5.70	54.79
		BFO -PID	89.31	1.20	12.30	3.07	34.52
		APID	33.80	1.30	11.00	2.68	22.02
		BFO -APID	29.55	2.10	13.10	2.80	21.73

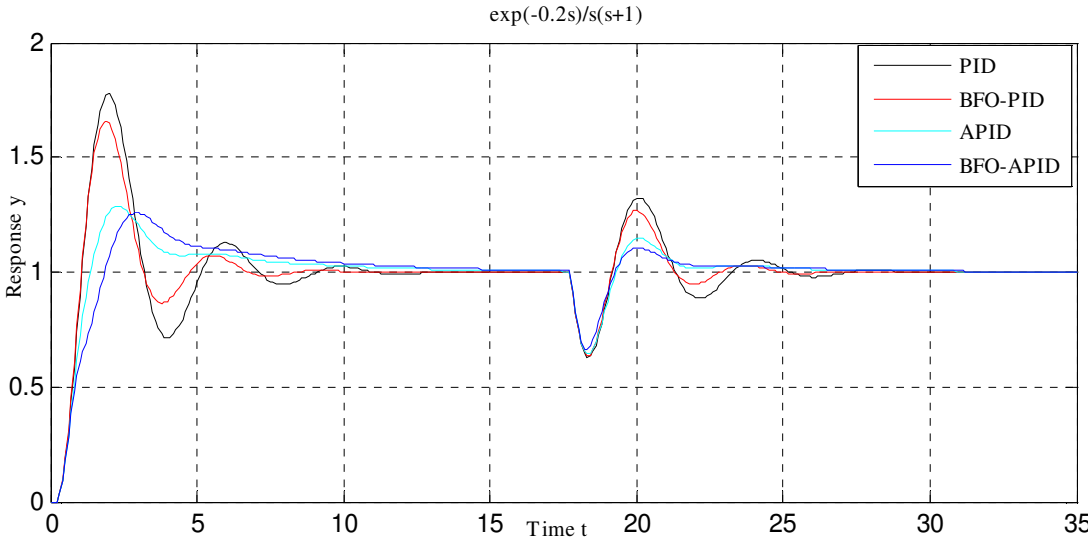


Fig. 3.2 (a) *Response of first order integrating process for L=0.2s, minimization of IAE+ITAE*

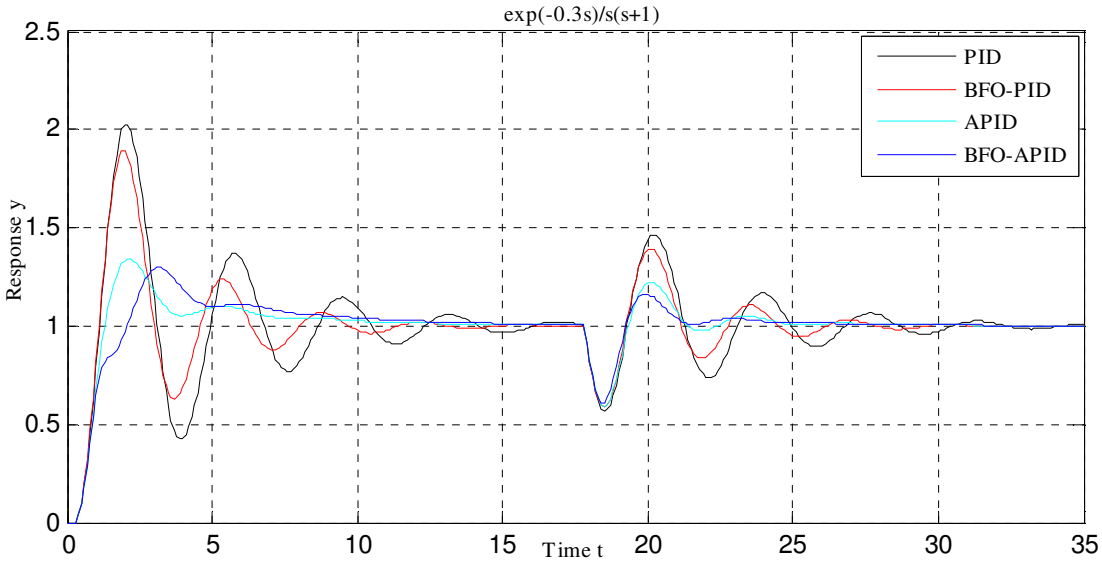


Fig. 3.2 (b) *Response of first order integrating process for L=0.3s, minimization of IAE+ITAE*

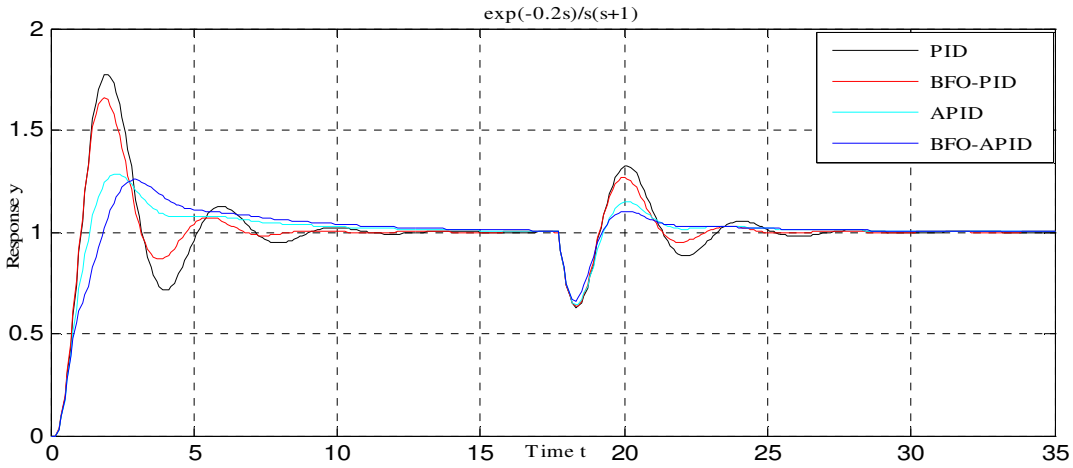


Fig. 3.2 (c) *Response of first order integrating process for L=0.2s, minimization of IAE*

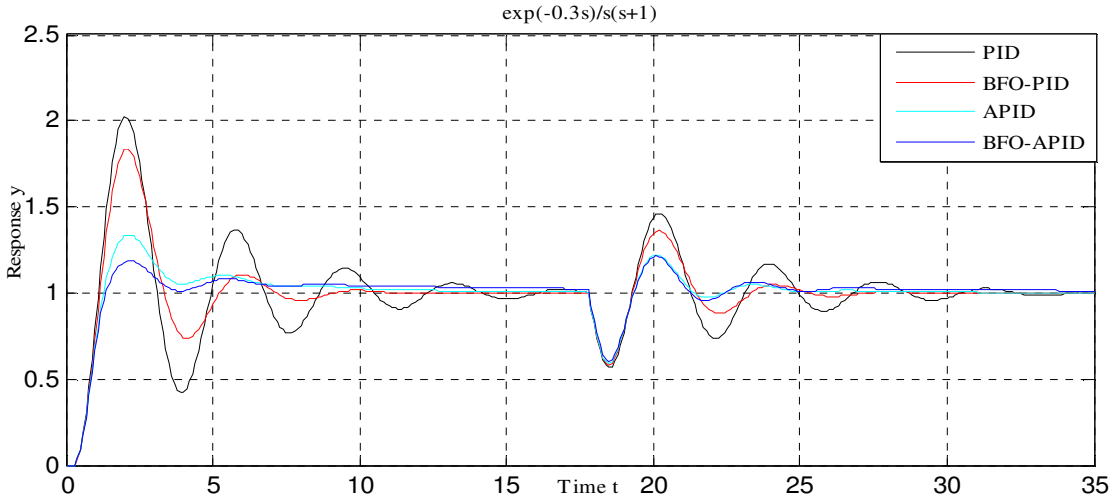


Fig. 3.2 (d) *Response of first order integrating process for L=0.2s, minimization of IAE*

3.7.3 Second Order Nonlinear Process

$$\frac{d^2y}{dt^2}+\frac{dy}{dt}+0.2y^2=u(t-L) \tag{3.10}$$

Response of second order non linear process in (3.10) with L=0.3s and L=0.4s under PID, BFO-PID, APID, and BFO-APID is shown in Fig. 3.3. Performance indices of the process in (3.10) for different controllers are given in Tables 3.3(a) and 3.3(b). Though the controller are tuned for L=0.3s, a higher value i.e., L=0.4s is also tested without changing controllers settings (for PID and APID). We can conclude that unlike PID, BFO-PID, and APID, BFO-

APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance.

Table 3.3 (a) *-Performance analysis of second order non-linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.3s	IAE+ITAE	**PID**	66.10	1.40	9.30	3.13	46.60
		BFO-PID	54.2	1.50	6.80	3.54	36.48
		APID	19.18	1.70	10.60	2.95	34.88
		BFO -APID	9.74	1.90	8.90	2.71	30.98
L=0.4s		**PID**	83.11	1.50	13.30	5.78	72.40
		BFO -PID	67.15	1.50	6.10	3.92	41.28
		APID	25.15	1.70	10.80	3.26	39.45
		BFO -APID	16.46	1.90	8.90	3.04	35.48

Table 3.3 (b) *-Performance analysis of second order non-linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.3s	IAE	**PID**	66.10	1.40	9.30	3.13	46.60
		BFO-PID	52.48	1.40	6.90	3.16	33.31
		APID	19.18	1.70	10.60	2.95	34.88
		BFO -APID	11.37	2.00	11.30	2.81	32.51
L=0.4s		**PID**	83.11	1.50	13.30	5.78	72.40
		BFO -PID	69.67	1.40	9.90	3.32	49.81
		APID	25.15	1.70	10.80	3.26	39.45
		BFO -APID	12.02	1.8	11.80	3.03	35.83

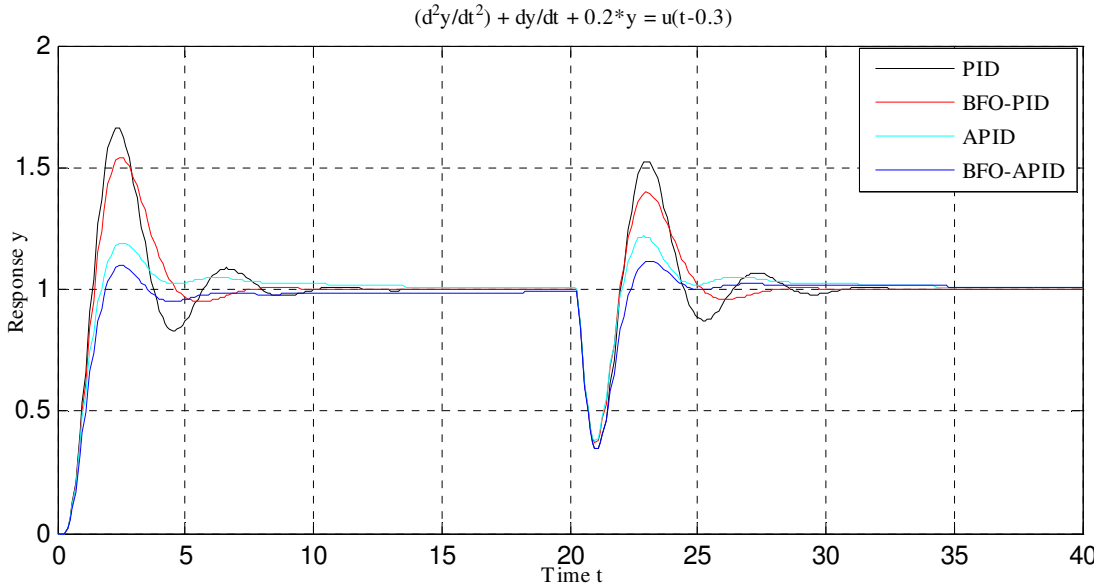


Fig. 3.3 (a) *Response of second order integrating process for L=0.3s, minimization of IAE+ITAE*

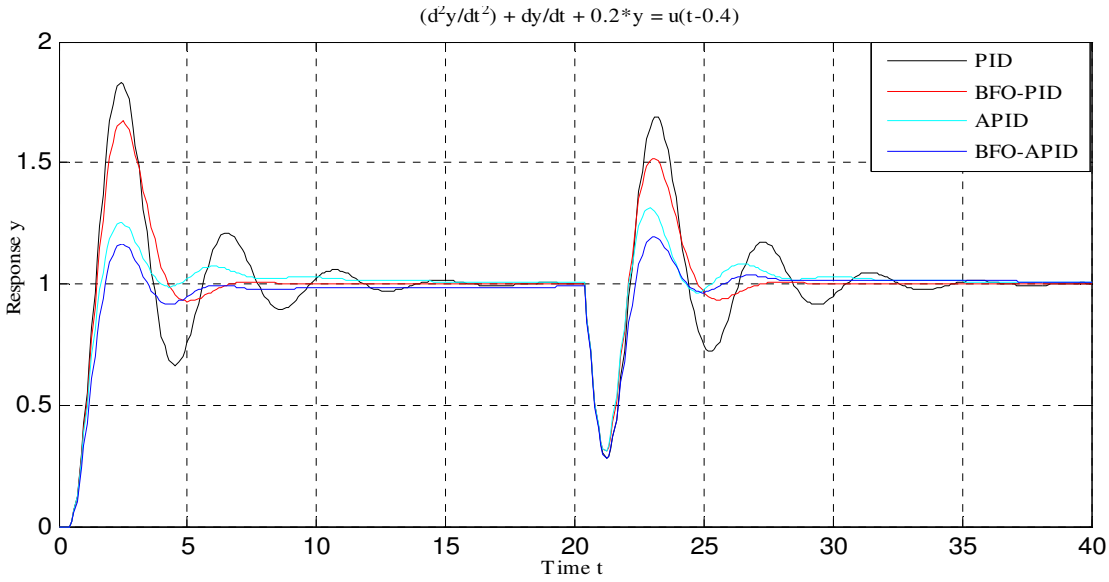


Fig. 3.3 (b) *Response of second order integrating process for L=0.4s, minimization of IAE+ITAE*

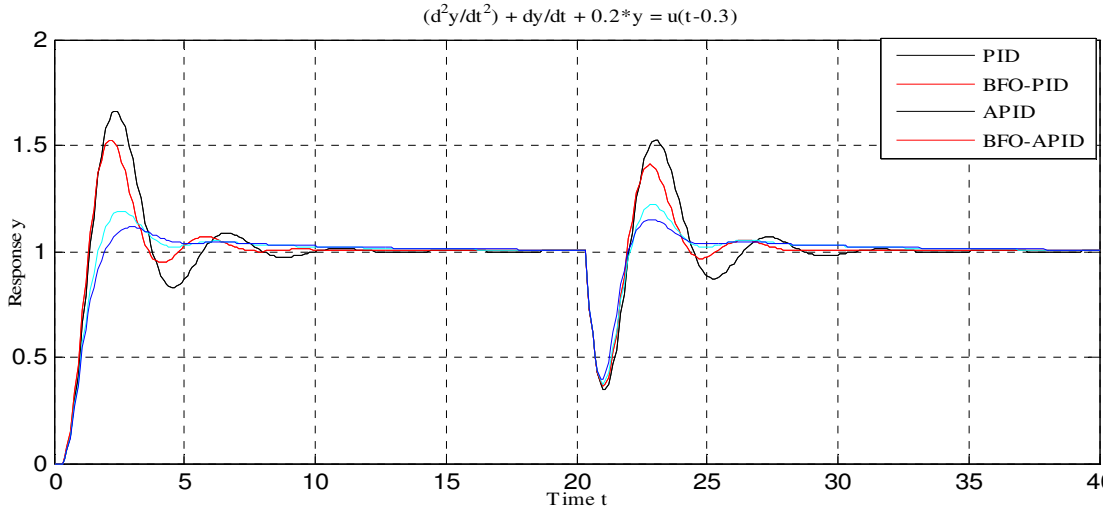


Fig. 3.3 (c) *Response of second order integrating process for L=0.3s, minimization of IAE*

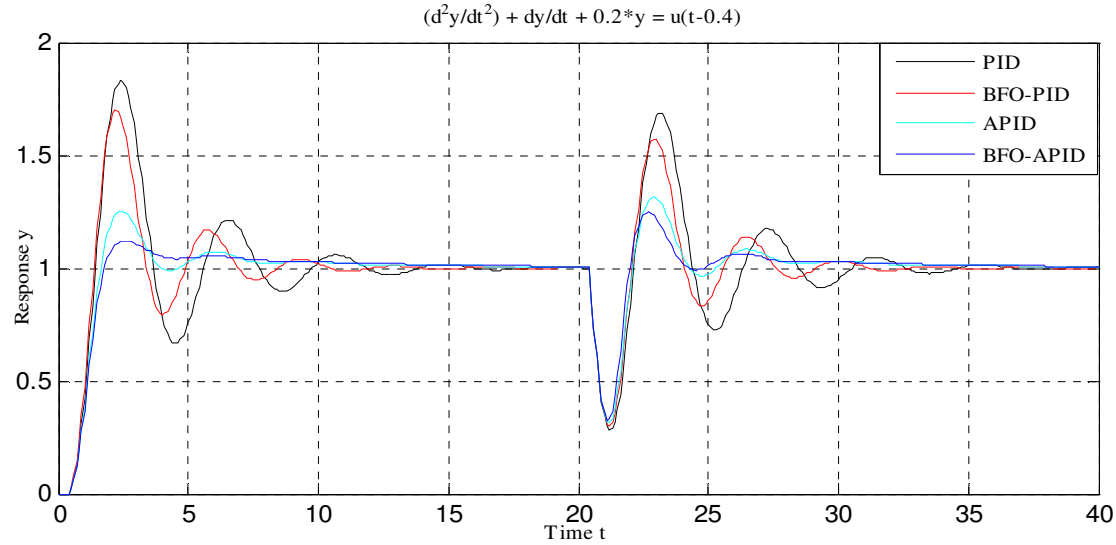


Fig. 3.3 (d) *Response of second order integrating process for L=0.4s, minimization of IAE*

3.7.4 pH-Neutralization Process

For pH-neutralization process we consider following linear model

$$G_p(s) = e^{-Ls}/(s+1)(0.1s+1)^2 \tag{3.11}$$

Response of pH neutarlization process in (3.11) with L=0.01s and L=0.02s under PID, BFO-PID, APID, and BFO-APID is shown in Fig. 3.3. Performance indices of the process in (3.11)

for different controllers are given in Tables 3.3(a) and 3.3(b). Though the controller are tuned for L=0.01s, a higher value i.e., L=0.02s is also tested without changing controllers settings (for PID and APID). We can conclude that unlike PID, BFO-PID, and APID, BFO-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance.

Table 3.4 (a) -*Performance analysis of pH-neutralization process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.01s	IAE+ITAE	**PID**	64.21	0.26	1.69	0.72	1.09
		BFO-PID	52.56	0.26	1.29	0.65	0.92
		APID	23.94	0.30	1.77	0.52	0.84
		BFO -APID	4.02	0.38	0.86	0.41	0.60
L=0.02s		**PID**	72.34	0.27	1.78	0.84	1.33
		BFO -PID	58.46	0.28	1.24	0.69	0.99
		APID	28.49	0.30	1.84	0.56	0.88
		BFO -APID	6.40	0.36	0.88	0.43	0.63

Table 3.4 (b) -*Performance analysis of pH-neutralization process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.01s	IAE	**PID**	64.21	0.26	1.69	0.72	1.09
		BFO-PID	52.56	0.26	1.29	0.65	0.92
		APID	23.94	0.30	1.77	0.52	0.84
		BFO -APID	3.02	0.38	0.86	0.41	0.60
L=0.02s		**PID**	72.34	0.27	1.78	0.84	1.33
		BFO -PID	58.46	0.28	1.24	0.69	0.99
		APID	28.49	0.30	1.84	0.56	0.88
		BFO -APID	6.40	0.36	0.88	0.43	0.63

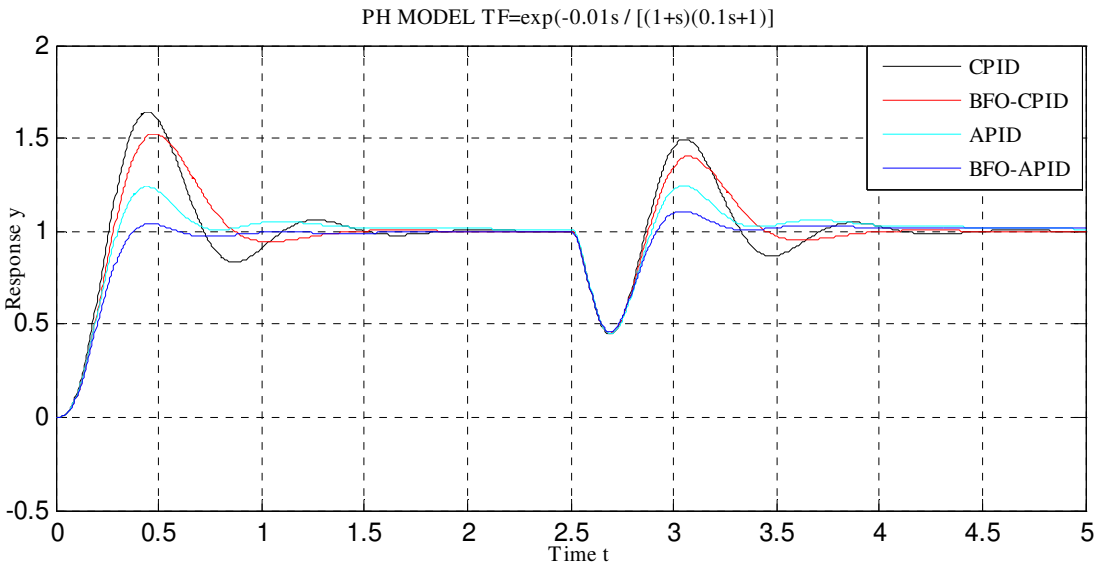


Fig. 3.4 (a) *Response of pH neutralization process for L=0.01s, minimization of IAE+ITAE*

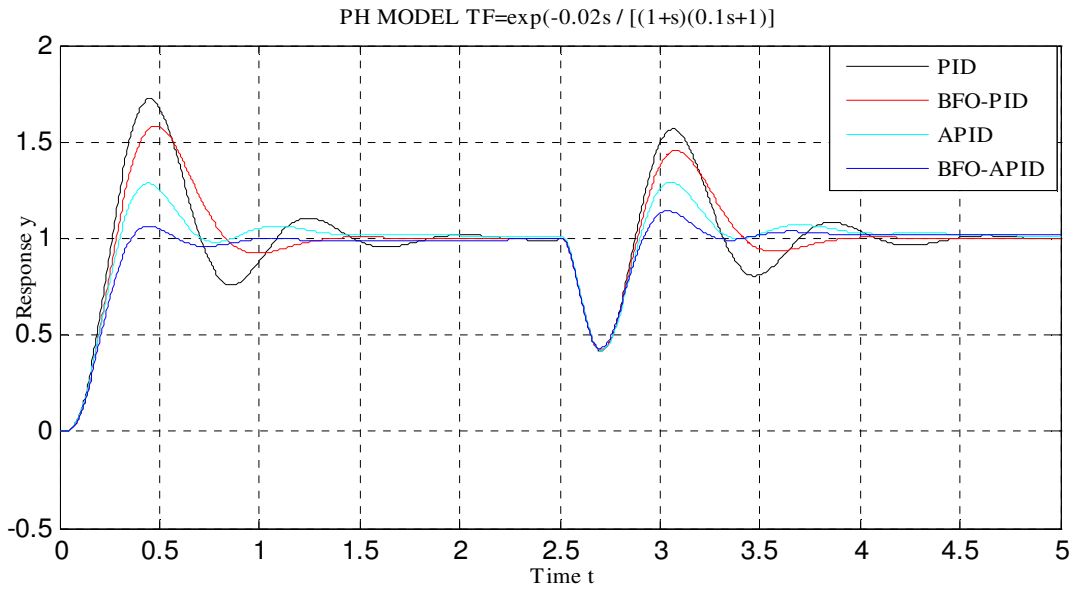


Fig. 3.4 (b) *Response of pH neutralization process for L=0.02s, minimization of IAE+ITAE*

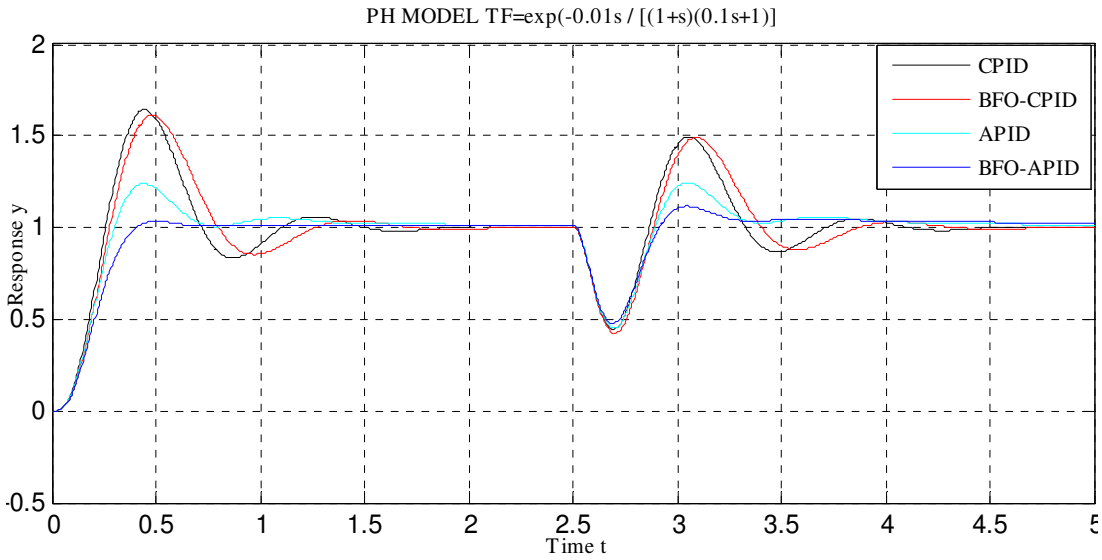


Fig. 3.4 (c) *Response of pH neutralization process for L=0.01s, minimization of IAE*

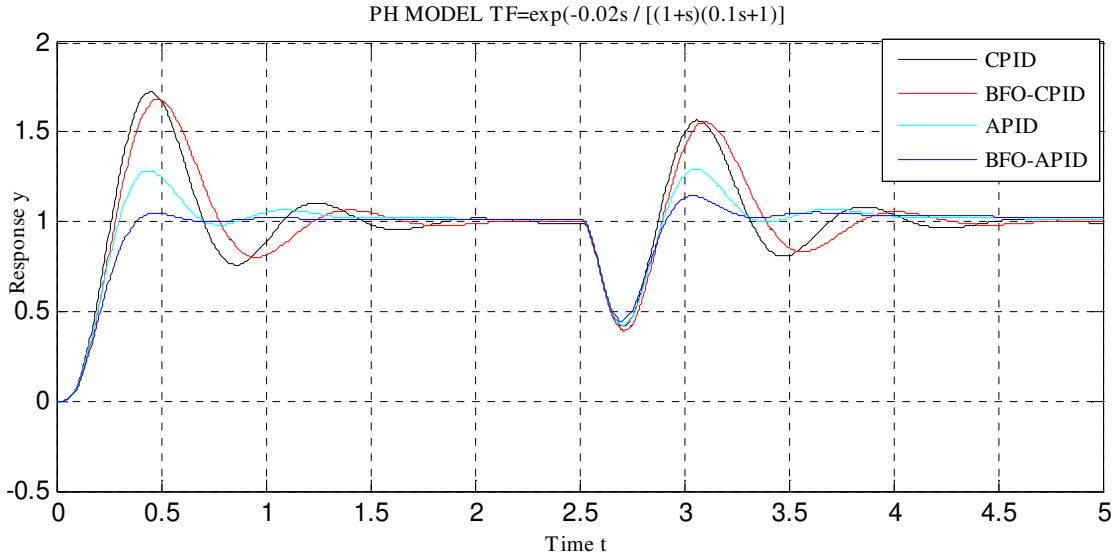


Fig. 3.4 (d) *Response of pH neutralization process for L=0.02s, minimization of IAE*

3.8 Conclusion

In this study we have exhibited how the performance of adaptive PID (APID) controllers is enhanced when it is used with the optimal power of bio-inspired algorithms. Here, BFO are said to be bio-inspired because of the fact that they depicts how E. coli bacteria are sustained

biologically. When the conventional controller with the optimal power of BFO are not able to provide the satisfactory performances over APID controller, we decided to develop the BFO-APID in order to get overall improved performance for both set point change as well as load disturbances. We also observed that BFO-APID provides improved performance as it provides minimum overshoot & improved rise time & settling time. It also provides lower value of IAE & ITAE which indicates that it can track set point and reject load disturbance properly. Initially we have used IAE+ITAE as objective function with a dead time. After that the dead time is improved in order to see the robustness of the controller & we found the performance is still satisfactory for BFO-APID. We also get the similar enhanced performance when we use only IAE as an objective function.

References

[1] Passino, K.M., Biomimicry of bacterial foraging for distributed optimization and control, IEEEControl System, vol. 22, no.3, pp.55-67, (2002).

[2] Passino, K.M., Bacterial foraging optimization, International Journal of Swarm Intelligence Research, 1(1), 1-16, (2010).

[3] Alavandar, S., Jain, T, NIBFOM, M. J., Bacterial foraging of optimized hybrid pre compensated PD control of two link rigid-flexible manipulator, Personal E- collection.

[4] Dey, C., Mudi, R.K., ; An improved auto-tuning scheme for PID controllers. ISA Trans. 48(3), 396–409 (2009).

[5] Dey, C., Mudi, R.K., Lee. T.T., ; An improved auto-tuning scheme for PI controllers. ISA Transactions. 47, 45- 52 (2008).

[6] Berg, H.C., Biological and Medical Physics Biomedical Engineering , Springer, Netherlands, 2005

[7] Chen, H., Zhu, Y., Hu, K., Cooperative Bacterial Foraging Optimization, Hindawi Publishing Corporation Discrete Dynamics in Nature and Society,10.1155,815247,2009

[8] Das, S., Biswas, A., Dasgupta, S, Abraham, A., and Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications, Personal E-collection.

[9] Seborg, D.E., Edgar. T.F.,; Adaptive control strategies for process control: A survey. AICHE J. 32(6), 881–913 (1986).

CHAPTER-4

PARTICLE SWARM OPTIMIZATION ALGORITHM BASED ADAPTIVE PID CONTROLLER

4.1 Introduction

In this chapter we will vary all the seven parameters, i.e., $K_p, K_i, K_d, k_1, k_2, k_3 and k_4$ of APID [1] within a defined range. Particle swarm optimization (PSO) is used to find out best set of solution which will optimize the given objective function. In 1995, Edward and Kennedy [2-4] first introduced the PSO method motivated by social behavior of organisms such as fish schooling and bird flocking. It is also a population based search technique. PSO can be easily implemented and usually results in faster convergence rates than other techniques. Unlike the PSO, PSO has no evolution operators such as crossover and mutation.

4.2 Particle Swarm Optimization (PSO)

Particle swarm optimization [2-4] is an artificial intelligence (AI) technique which can be is used to find approximate solutions to extremely difficult or impossible numeric maximization and minimization problems. The idea behind the algorithm was inspired by social behavior of animals such as bird flocking and fish schooling. This theory can be understood by the concept of techniques used by birds or fishes for searching the food in wide area. Suppose the following scenario, a group of birds or fishes are randomly searching for food in a wide area. There is only one piece of food in the area being searched. All the birds/fishes do not know the exact location where the food is. In that condition, they travel in search space according to the own experience as well as neighbour's experience. That mean, in each iteration, they compare the distance between its own location and the target with respect to its previous experience as well as the best position of neighbor which is closest to the target. After that they modify its own speed for the best strategy to find the food. This is the basic principle of Particle Swarm Optimization (PSO).

In technical term, each bird or fish is called "Particle" and its flock is called "Particle Population". All the particles have own fitness or objective value which is calculated by the objective function. For the optimization of objective function, particles positions are updated by velocity vector which depends on its personal influence as well as social influence.

Technically in other words PSO starts with random set of solutions that is called the particles. Each particle has positions (value of the variables) and velocities. The particles update their velocities and positions based on the local best solution (best solution associated with current population) and global best solution (best solution associated with population found so far).

To avoid confusion I am giving some nomenclature which is used in PSO.

Table 4.1 *Parameters of PSO algorithm*

Term	Explanation
Particle	One set of solution, i.e., one set of value of variables.
Position	Value of individual variable.
velocity	Corresponding to each variable there is a velocity. It is a vector quantity.
Population	It is the no of particle in a swarm, i.e., set of solutions
fitness	f (particle)
Variables	Define all the independent variables
Range of variables	Define the range of all independent variables
Velocity range	[-|(span of the variable)| , |(span of the variable)|] Where span is the difference between largest and smallest value of that variable.
Local _best _particle	It is the best solution associated with minimum fitness of the current population.
Global _best _particle	It is the best solution associated with minimum fitness of the population found so far.

4.3 Objective Function of Particle Swarm Optimization

The function to be optimized is known as objective function. Here, minimization of the *integral-absolute-error* (*IAE*) or *integral-time-absolute-error* (*ITAE*) or combination of both i.e., (*IAE+ITAE*) is defined as the objective function (performance index or fitness function). The *IAE* and *ITAE* are calculated as:

$$IAE = \int_0^t |e(t)|\,dt \tag{4.1}$$

$$IAE + ITAE = \int_0^t |e(t)|dt + \int_0^t t\,|e(t)|\,dt \tag{4.2}$$

4.4 Initial Settings of Particle Swarm Optimization -

Table 4.2 *Initial settings of the algorithm*

Population	10
Variables	$K_p, K_i, K_d, k_1, k_2, k_3 and k_4$
Range of variables	K_p, K_i, K_d are ±20% of their respective PID, k_1 [0, 5], k_2 [0 ,5], k_3[0 ,30], and k_4[0,1].
Velocity range	[-|(span of the variable)| , |(span of the variable)|] Where span is the difference between largest and smallest value of that variable.

4.5 Different Operations Used in Particle Swarm Optimization

Population

Here, the population size is 10. We have 7 variables $K_p, K_i, K_d, k_1, k_2, k_3 and k_4$. We generate random set of solution by MATLAB-command random (population size, variables), i.e., random (10, 7). All the values are lying between 0 and 1 and matrix size is 10*7.

Bring the particle's positions (variables) in range

Since value of the variables is not in the defined range therefore we bring it in a defined range to get the real value of the optimization variable. The following linear mapping is used for this purpose.

Let i denotes the particles so i=1, 2, 3...10

And let j denotes the variables so j=1, 2, 3…7

$$particle(i, j) = X_{\min}(j) + \frac{X_{\max}(j) - X_{\min}(j)}{span} \times \{decimalvalue(i, j) - 0.0001\} \quad (4.3)$$

Where, particle (i, j) = real value of variable

$X_{\max}(j)$ = Maximum value of jth variable of any set of solutions

$X_{\min}(j)$ = Minimum value of jth variable of any set of solutions

Span of decimal no = 0.9999 – 0.0001

Velocity and velocity range

Each variable has velocity which is a vector quantity. To generate velocity the same MATLAB command is used as we have used to generate population, i.e., random (10, 7). Now we bring the decimal value of velocity in a defined range to get the real value of velocity. The following linear mapping is used for this purpose.

$$velocity(i,j) = V_{\min}(j) + \frac{X_{\max}(j) - X_{\min}(j)}{span} \times \{decimalvalue(i,j) - 0.0001\} \tag{4.4}$$

Where, velocity (i,j) is the real value of velocity.

Velocity range is defined as:

$V_{\max}(j) = +|(X_{\max}(j) - X_{\min}(j)|$ is the maximum velocity of j^{th} variable of any set of solution.

$V_{\max}(j) = +|(X_{\max}(j) - X_{\min}(j)|$ is the minimum velocity of j^{th} variable of any set of solution.

Other unknowns are same as previous case.

Particle's velocity update, particle's position update and inertia weight (w)

We update particle's velocity first then we update particle's position by the following method. Update particle's velocity by

$$New\_velocity(i,j) = w * velocity(i,j) + c_1 * r_1(local\_best\_particle(i,j) - particle(i,j)) + c_2 * r_2(global\_best\_particle(i,j) - p(i,j)) \tag{4.5}$$

Update particle's position by

$$partcle\ (i,j) = particle\ (i,j) + new\_velocity\ (i,j) \tag{4.6}$$

Where i=1, 2, 3…10 is the particles and j=1, 2, 3…7 is the variables. i and j is used to indicate the coordinate of the particle. Here w is the inertia weight. It is used to control the search. At the starting phase of search inertia weight is almost equal to 0.99 and search is in exploration mode. At the end phase of search inertia weight is very low (almost equal to 0.01) and search is in exploitation mode. Thus at the starting phase of search change in particle's velocity and position are very high compared to end phase of search where change in particle's velocity and position are very small.

Note- if the exploration mode is very high (w >1) then you jump from one solution to another solution with much gap (span) because of the high velocity of the particle. You may over jump best solution. And if exploitation mode is very high (w $\approx$0.01) then program will

take too much time to converge. Because w affects our results drastically therefore you cannot give importance to anyone of the modes. I have made w as a time varying quantity (dynamic nature) to control the both modes, i.e., w gradually decreases from 0.99 to 0.01 as the no of iteration increases.

Local weight and global weight

c_1 and c_2 are constant. c_1 is called the cognitive or personal or local weight. c_2 is called the social or global weight. c_1 =2 and c_2 =2. r_1 and r_2 are the random number and both are in range of (0,1).

Surety that you are at global optima

From the second term in the velocity update equation (4.4), we can say that particles always try to move towards Local_best_particle, i.e., local minimum and from the third term we can say particles always try to move towards Global_best_particle, i.e., global minimum.

If the Global_best_particle is too far from the Local_best_particle, then Global_best_particle has huge impact on the velocities and positions of the particles which are nearer to Local_best_particle. It means that for those particles high change in velocities and positions occur. Therefore particles finally move towards Global_best_particle. Also the random variables r_1 and r_2 add a random component to the particles movement and help to prevent particles from getting stuck at a non-optimal local minimum solution.

4.6 Steps of Particle Swarm Optimization Algorithm

(1) Generate population with uniform random number and bring it within the define range.

(2) Generate velocity with uniform random number and bring it within the define range.

(3) For each particles, i.e., i=1, 2, 3…10

Evaluate the fitness, i.e., objective function, i.e., f (particle (i))

(4) Find out the particle associated with minimum fitness. Let for ith particle we are getting minimum fitness then

Local_best_particle = $particle(i^{th})$

Global_best_particle = $particle(i^{th})$

Start of PSO loop: repeat until slope of objective is almost zero or until maximum no of iteration is reached.

(i) For each particles, i.e., i=1, 2, 3...10

For each variables, i.e., j=1, 2, 3…7

{Generate the random number r_1, r_2.

Update the velocity by

$$New\_velocity(i,j) = w * velocity(i,j) + c_1 * r_1 (local\_best\_particle(i,j) - particle(i,j)) + c_2 * r_2 (global\_best\_particle(i,j) - p(i,j)) \quad (4.7)$$

Check the velocity limit and if it is out of range, bring it in range.

If $new\_velocity(i,j) > V_{max}(j)$ then $new\_velocity(i,j) = V_{max}(j)$

Else if $new\_velocity(i,j) < V_{min}(j)$ then $new\_velocity(i,j) = V_{min}(j)$

(ii) For each particles, i.e., i=1, 2, 3...10

For each variables, i.e., j=1, 2, 3…7

{Update the position by $partcle(i,j) = particle(i,j) + new\_velocity(i,j)$ (4.8)

Check the particle's position limit and if it is out of range bring it in range.

If $particle(i,j) > X_{max}(j)$ then $particle(i,j) > X_{max}(j)$

Else if $particle(i,j) < X_{max}(j)$ then particle $particle(i,j) < X_{max}(j)$

(iii) For each particles, i.e., i=1, 2, 3…10

Evaluate the fitness, i.e., objective function i.e. f (particle (i))

(iv) Find out the particle associated with minimum fitness in the current population. Let for i^{th} particle we are getting minimum fitness.

Local_best_particle=particle (i^{th})

If $f(local\_best\_particle < f(global\_best\_particle)$

$(local\_best\_particle = (global\_best\_particle)$

End

Hence Global_best_particle is our solution for which fitness is minimum.

4.7 RESULTS

For simulation study, we consider the following systems with dead-time (*L*)

$G_p(s) = e^{-Ls}/(s+1)^2$, L=0.2s, and 0.3s (4.9)

$G_p(s) = e^{-Ls}/s(s+1)$, L=0.2s, and 0.3s. (4.10)

$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.2y^2 = u(t-L)$, L=0.3s, and 0.4s. (4.11)

PH model- $G_p(s) = e^{-Ls}/(s+1)(0.1s+1)^2$, L=0.01s, and 0.02s. (4.12)

For each process model we have used four different types of controller.

(a) **PID.**

(b) **PSO-PID** $-K_p, K_i, K_d$ are ±20% of their respective PID and these are calculated by genetic algorithm.

(c) **APID.**

(d) **PSO-APID** - In this all seven parameters are varying within the defined range and these are calculated by PSO algorithm.

We have calculated the close loop response characteristics for different controllers. 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 absolute error (IAE) and integral time absolute error (ITAE) are calculated for each controller. Performance of our PSO-APID is compared with PID, PSO-PID, and APID. Fourth-order Range-Kutta method is used for numeric integration. The detailed performance analysis for various types of process is discussed below.

4.7.1 Second Order Linear Process

Transfer function of the process is given by

$$G_p(s) = e^{-Ls}/(s+1)^2 \tag{4.13}$$

Response of second order linear process in (4.13) with L=0.2s, and L=0.3s under PID, PSO-PID, APID, and PSO-APID is shown in Fig. 4.1. Performance indices of the process in (4.13) for different controllers are given in Table 4.3 (a) and Table 4.3 (b). Though the controller are tuned for L=0.2s, a higher value i.e., L=0.3s is also tested without changing controllers settings (for PID and APID). Performance analysis reveals that unlike PID, PSO-PID, and APID, PSO-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance.

Table 4.3 (a) *-Performance analysis second order linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE+ITAE	**PID**	60.30	0.90	4.40	2.08	9.29
		PSO-PID	44.49	0.80	3.80	1.60	6.65
		APID	5.37	1.40	4.20	1.36	6.26
		PSO -APID	0.57	1.00	6.30	1.23	4.94
L=0.3s		**PID**	93.93	0.90	12.20	4.41	35.28
		PSO -PID	61.81	1.00	3.50	2.27	10.04
		APID	15.58	1.00	4.70	1.85	14.10
		PSO -APID	0.40	2.60	7.90	1.98	8.75

Table 4.3 (b) *-Performance analysis second order linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE	**PID**	60.30	0.90	4.40	2.08	9.29
		PSO-PID	44.49	0.80	3.80	1.60	6.65
		APID	5.37	1.40	4.20	1.36	6.26
		PSO -APID	2.95	2.10	3.10	1.47	4.49
L=0.3s		**PID**	93.93	0.90	12.20	4.41	35.28
		PSO -PID	61.81	1.00	3.50	2.27	13.98
		APID	15.58	1.00	4.70	1.85	14.10
		PSO -APID	4.24	2.30	4.80	1.94	8.69

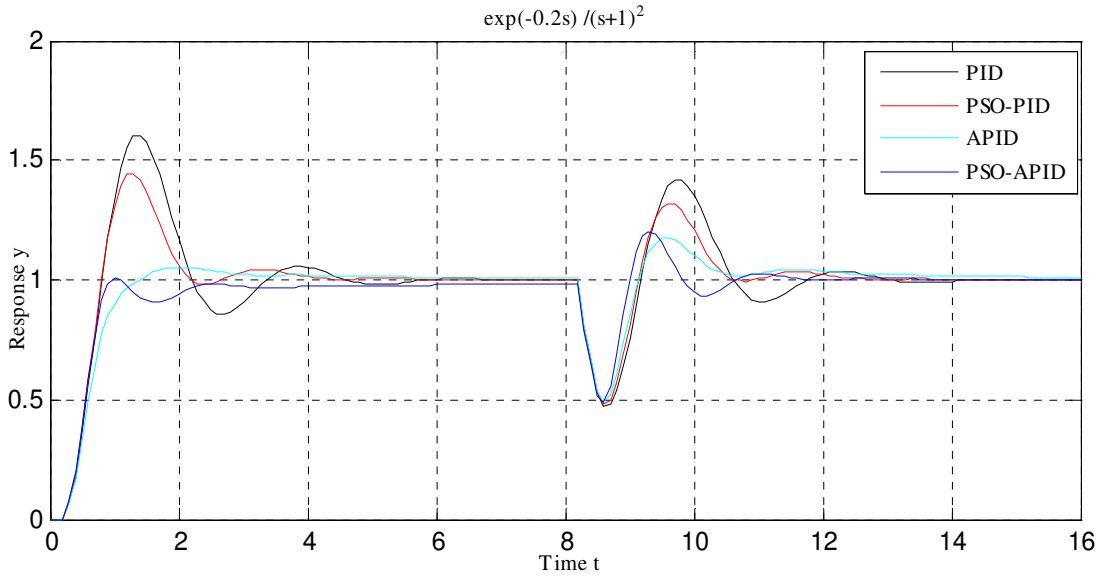


Fig. 4.1 (a) *Response of second order linear process for L=0.2s, minimization of IAE+ITAE*

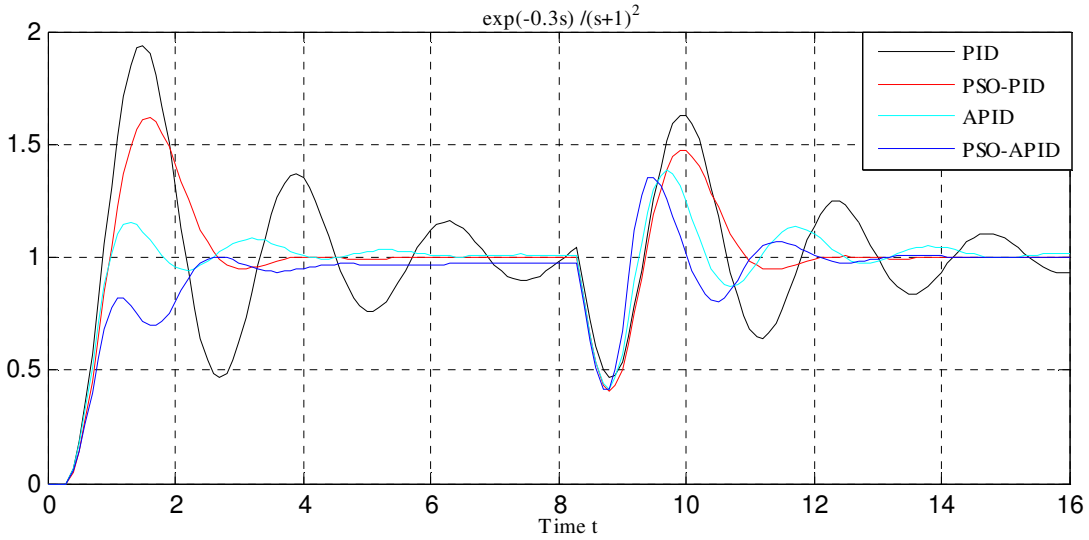


Fig. 4.1 (b) *Response of second order linear process for L=0.3s, minimization of IAE+ITAE*

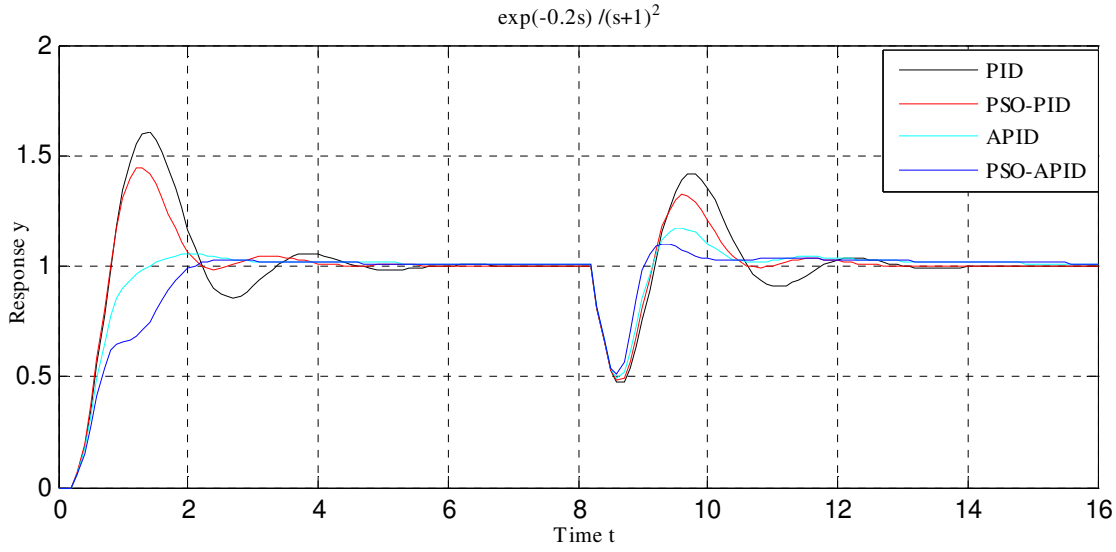


Fig. 4.1 (c) *Response of second order linear process for L=0.2s, minimization of IAE*

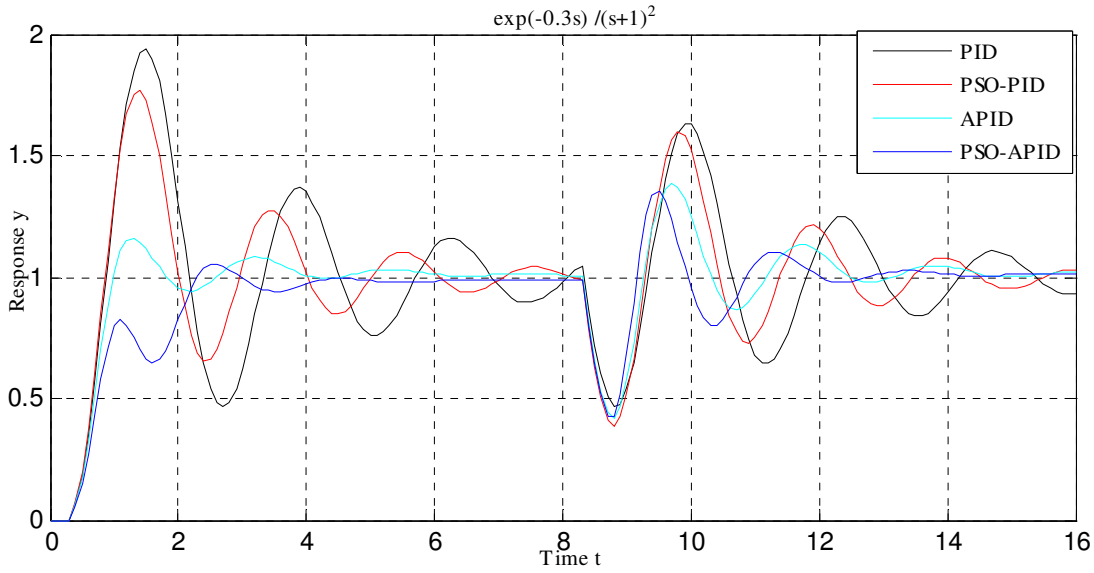


Fig. 4.1 (d) *Response of second order linear process for L=0.3s, minimization of IAE*

4.7.2 First Order with Integrating Process

Transfer function of the process is given by

$$G_p(s) = e^{-Ls} / s(s+1) \tag{4.14}$$

Response of first order integrating process in (4.14) with L=0.2s, and L=0.3s under PID, PSO-PID, APID, and PSO-APID is shown in Fig. 4.2. Performance indices of the process in (4.14) for different controllers are given in Table 4.4 (a) and Table 4.4 (b). Though the controller are tuned for L=0.2s, a higher value i.e., L=0.3s is also tested without changing controllers settings (for PID and APID). Performance analysis reveals that unlike PID, PSO-PID, and APID, PSO-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance.

Table 4.4 (a) *-Performance analysis first order integrating process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE+ITAE	**PID**	77.50	1.10	10.20	3.44	27.19
		PSO-PID	59.02	1.10	6.00	2.22	14.88
		APID	28.75	1.40	11.00	2.46	19.44
		PSO -APID	24.56	1.70	11.70	2.30	16.52
L=0.3s		**PID**	93.93	0.90	12.20	4.41	35.28
		PSO -PID	58.59	1.20	6.40	2.45	16.79
		APID	15.58	1.00	4.70	1.85	14.10
		PSO -APID	19.79	1.50	13.10	2.42	19.72

Table 4.4 (b) *-Performance analysis first order integrating process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE	**PID**	77.50	1.10	10.20	3.44	27.19
		PSO-PID	59.02	1.10	6.00	2.22	14.88
		APID	28.75	1.40	11.00	2.46	19.44
		PSO -APID	13.88	1.50	17.40	2.15	21.76
L=0.3s		**PID**	93.93	0.90	12.20	4.41	35.28
		PSO -PID	83.06	1.10	9.60	3.35	26.13
		APID	15.58	1.00	4.70	1.85	14.10
		PSO -APID	12.65	2.50	17.40	2.47	25.45

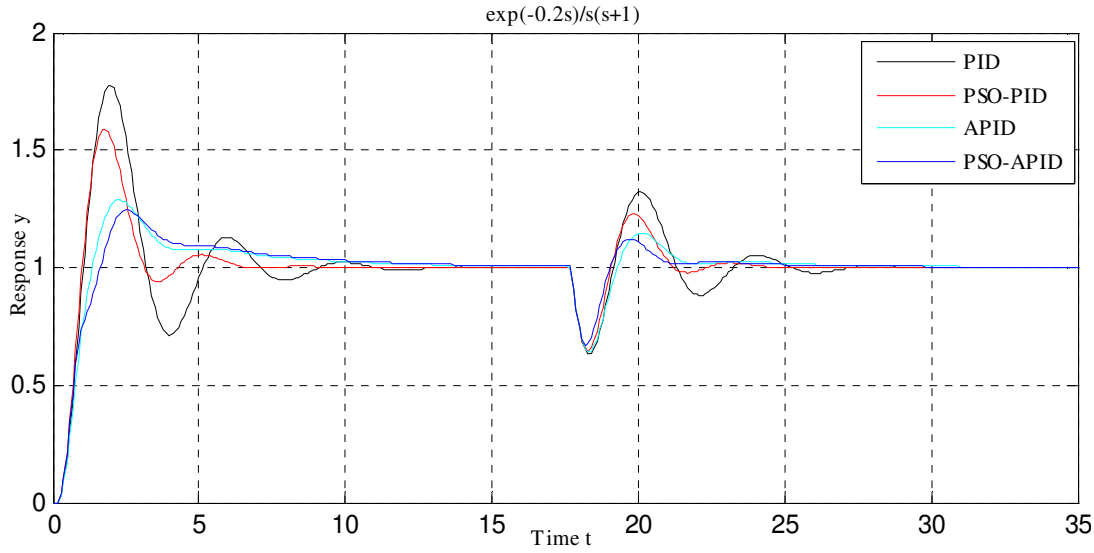


Fig.4.2 (a) *Response of first order integrating process for L = 0.2s, minimization of IAE+ITAE*

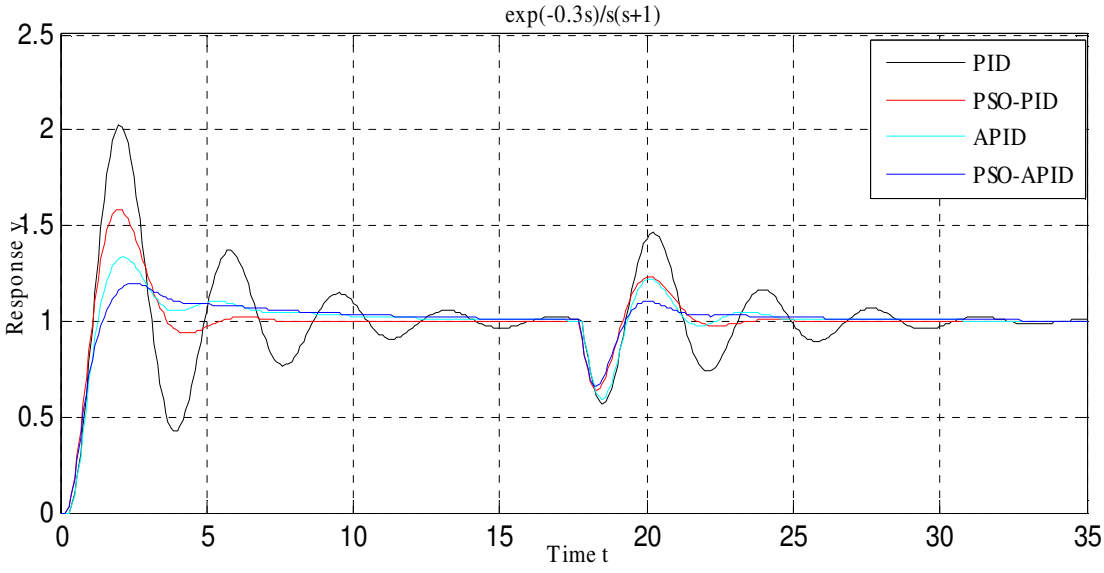


Fig.4.2 (b) *Response of first order integrating process for L = 0.3s, minimization of IAE+ITAE*

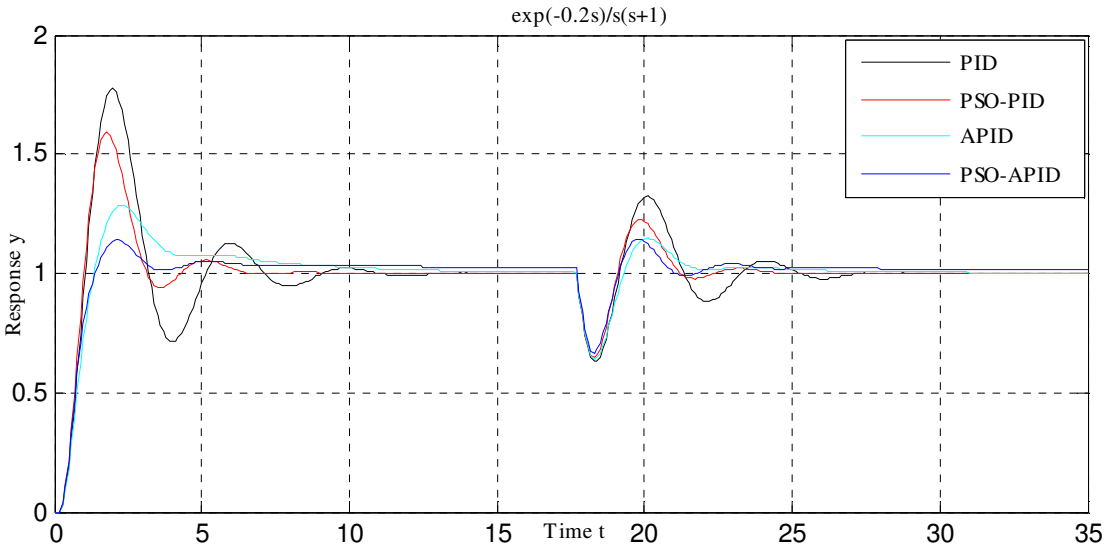


Fig.4.2 (c) *Response of first order integrating process for L = 0.3s, minimization of IAE*

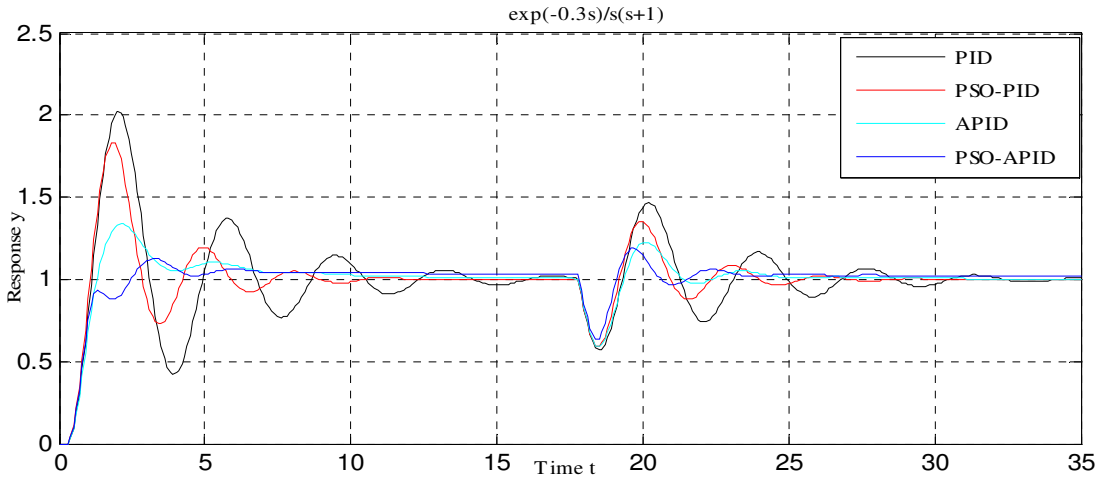


Fig.4.2 (d) *Response of first order integrating process for L = 0.3s, minimization of IAE*

4.7.3 Second Order Nonlinear Process

$$\frac{d^2y}{dt^2}+\frac{dy}{dt}+0.2y^2=u(t-L) \tag{4.15}$$

Response of second order non linear process in (4.15) with L=0.3s and L=0.4s under PID, PSO-PID, APID, and PSO-APID is shown in Fig. 4.3. Performance indices of the process in (4.15) for different controllers are given in Tables 4.4(a) and 4.4(b). Though the controller are

tuned for L=0.3s, a higher value i.e., L=0.4s is also tested without changing controllers settings (for PID and APID). We can conclude that unlike PID, PSO-PID, and APID, PSO-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance.

Table 4.5 (a) -*Performance analysis second order non-linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.3s	IAE+ITAE	**PID**	66.10	1.40	9.30	4.13	46.60
		PSO-PID	51.59	1.40	6.80	3.10	32.76
		APID	19.18	1.70	10.60	2.95	34.88
		PSO -APID	4.34	2.80	4.00	2.01	18.84
L=0.4s		**PID**	83.11	1.50	13.30	4.78	72.40
		PSO -PID	60.12	1.60	7.40	3.82	41.28
		APID	25.15	1.70	10.80	3.26	39.45
		PSO -APID	3.93	3.60	4.80	2.47	29.98

Table 4.5 (b) -*Performance analysis second order non-linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.3s	IAE	**PID**	66.10	1.40	9.30	4.13	46.60
		PSO-PID	51.59	1.40	6.80	3.10	32.76
		APID	19.18	1.70	10.60	2.95	34.88
		PSO -APID	2.75	2.70	3.60	1.97	19.75
L=0.4s		**PID**	83.11	1.50	13.30	4.78	72.40
		PSO -PID	69.18	1.40	9.80	4.30	49.67
		APID	25.15	1.70	10.80	3.26	39.45
		PSO -APID	3.93	3.60	4.80	2.47	29.98

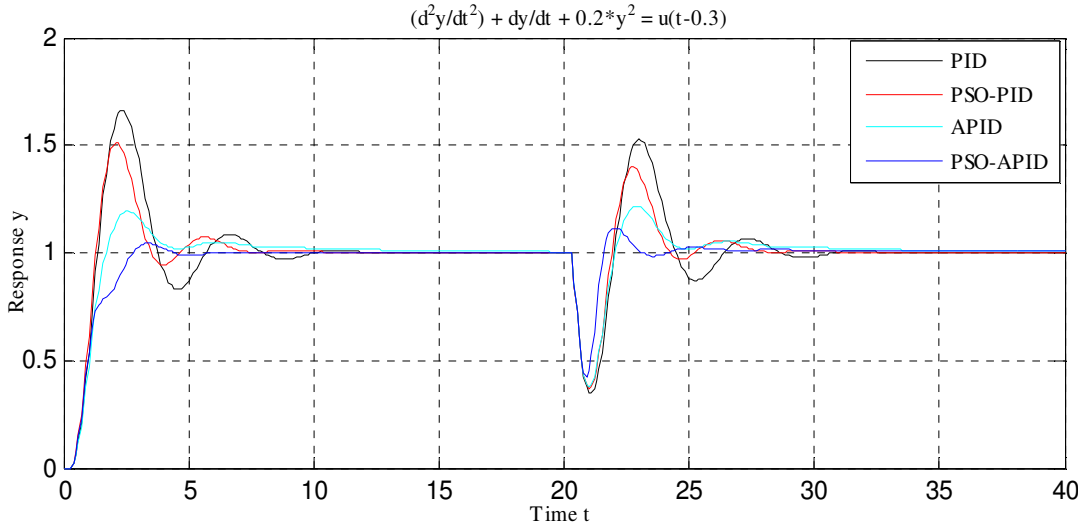


Fig.4.3 (a) *Response of second order nonlinear process for L = 0.3s, minimization of IAE+ITAE*

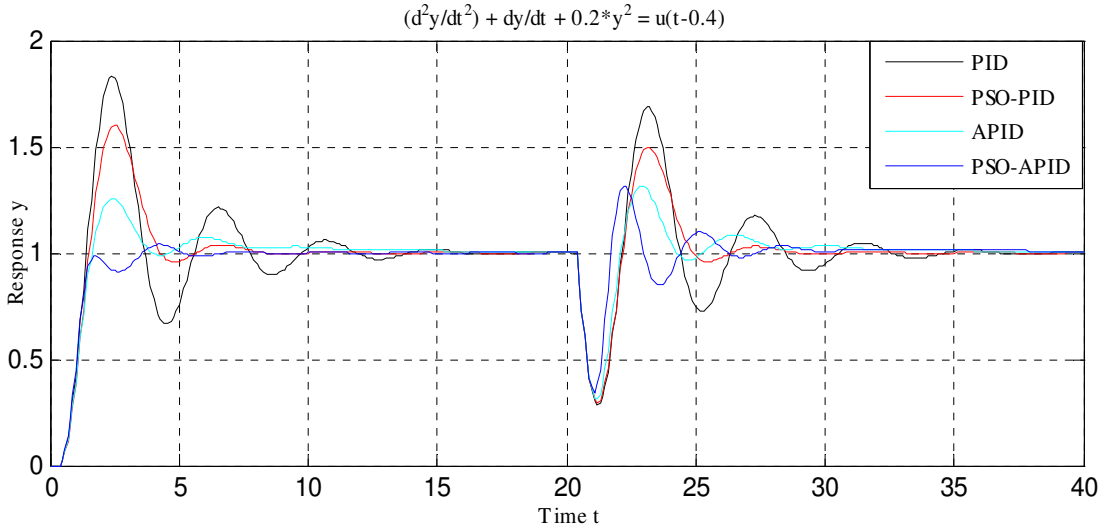


Fig.4.3 (b) *Response of second order nonlinear process for L = 0.4s, minimization of IAE+ITAE*

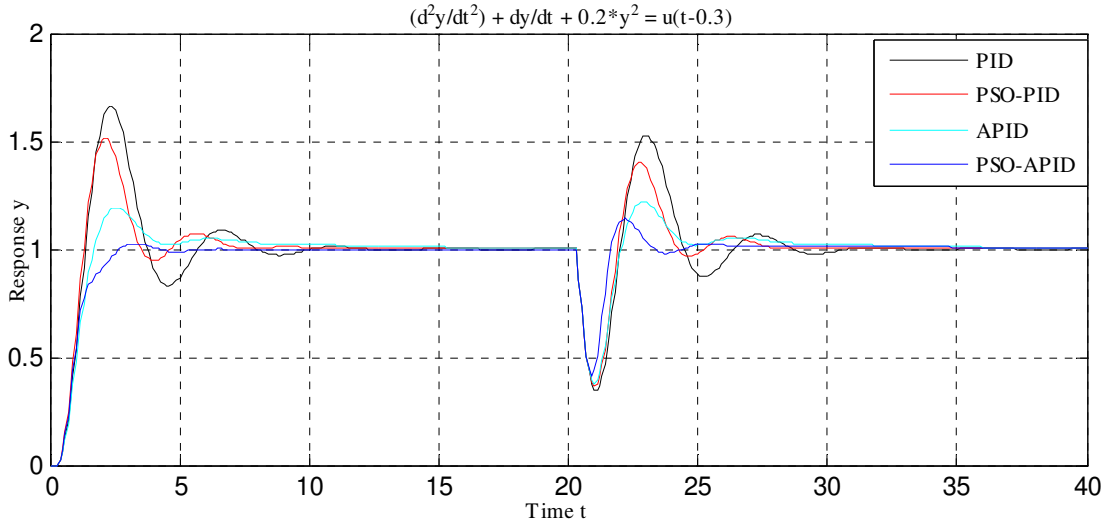


Fig.4.3 (c) *Response of second order nonlinear process for L = 0.3s, minimization of IAE*

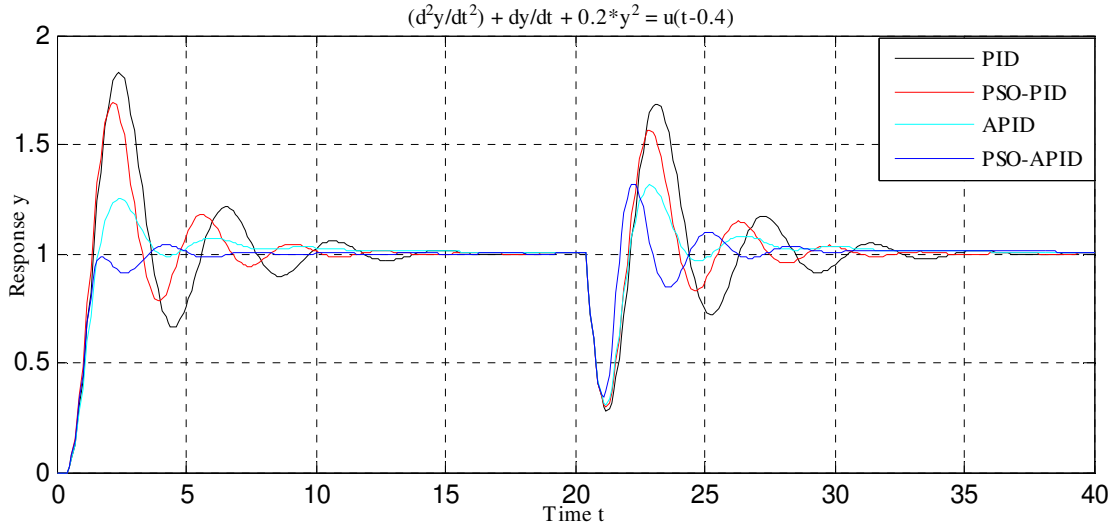


Fig.4.3 (d) *Response of second order nonlinear process for L = 0.4s, minimization of IAE*

4.7.4 pH-Neutralization Process

For pH-neutralization process we consider following linear model

$$G_p(s) = e^{-Ls} /(s+1)(0.1s+1)^2 \tag{4.16}$$

Response of pH neutralization process in (4.16) with L=0.01s and L=0.02s under PID, PSO-PID, APID, and PSO-APID is shown in Fig. 4.4. Performance indices of the process in (4.16) for different controllers are given in Tables 4.6(a) and 4.6(b). Though the controller are tuned for L=0.01s, a higher value i.e., L=0.02s is also tested without changing controllers settings

(for PID and APID). We can conclude that unlike PID, PSO-PID, and APID, PSO-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance.

Table 4.6 (a) -*Performance analysis pH-neutralization process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.01s	**IAE+ITAE**	**PID**	64.21	0.26	1.69	0.72	1.40
		PSO-PID	48.43	0.25	1.23	0.52	0.94
		APID	23.94	0.3	1.77	0.52	0.84
		PSO -APID	9.75	0.31	0.81	0.39	0.79
L=0.02s		**PID**	72.34	0.27	1.78	0.84	1.33
		PSO -PID	53.55	0.27	1.27	0.59	1.07
		APID	28.49	0.30	1.84	0.56	0.88
		PSO -APID	6.28	0.33	1.49	0.4	0.73

Table 4.6 (b) -*Performance analysis pH-neutralization process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.01s	**IAE**	**PID**	64.21	0.26	1.69	0.72	1.40
		PSO -PID	48.43	0.25	1.23	0.52	0.94
		APID	23.94	0.3	1.77	0.52	0.84
		PSO -APID	10.49	0.31	0.79	0.39	0.82
L=0.02s		**PID**	72.34	0.27	1.78	0.84	1.33
		PSO -PID	56.14	0.25	1.23	0.60	1.12
		APID	28.49	0.30	1.84	0.56	0.88
		PSO -APID	14.27	0.30	1.03	0.43	0.90

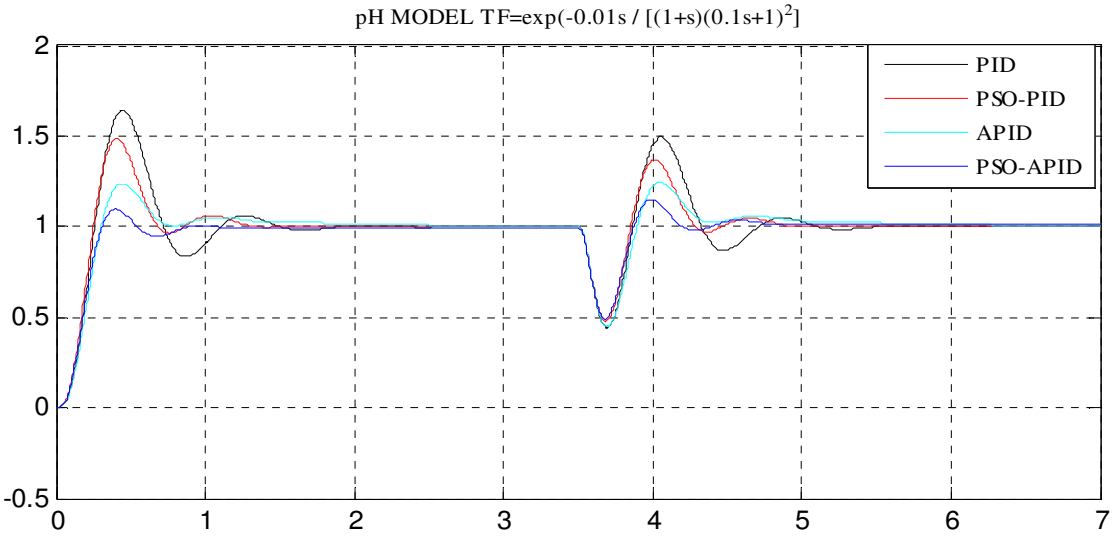


Fig.4.4(a) *Response of pH neutralization process for L = 0.01s, minimization of IAE+ITAE*

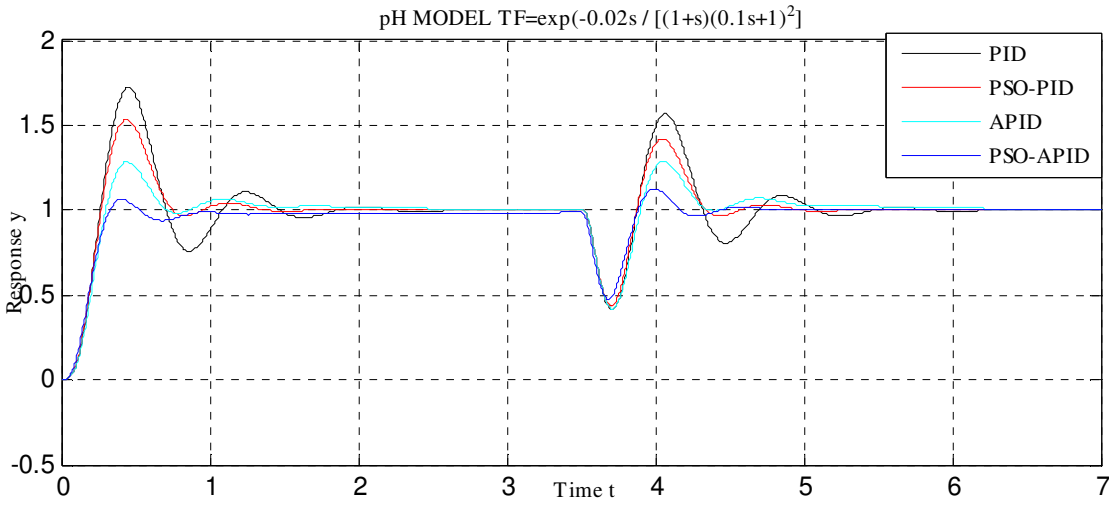


Fig.4.4(b) *Response of pH neutralization process for L = 0.02s, minimization of IAE+ITAE*

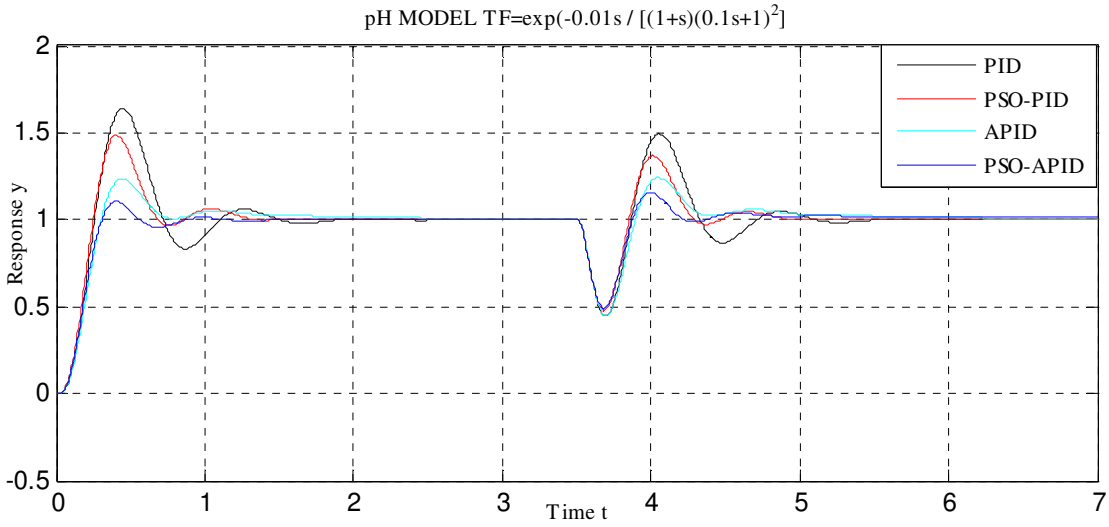


Fig.4.4(c) *Response of pH neutralization process for L = 0.01s, minimization of IAE*

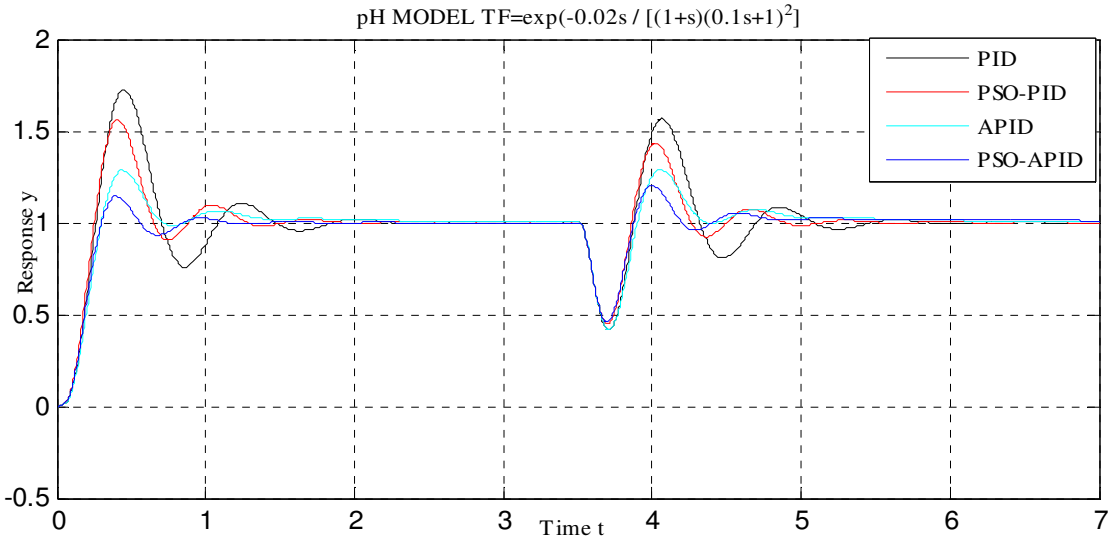


Fig.4.4(d) *Response of pH neutralization process for L = 0.02s, minimization of IAE*

4.8 Conclusion

In this study we have explored the possibility of performance enhancement of Adaptive PID controller (APID) through Particle Swarm algorithm optimization technique. From the experimental results we observed that both PSO-APID provided remarkably enhanced performance over Adaptive PID controller (APID) due to set point changes as well as load disturbances. We also observed that genetic algorithm based Adaptive PID controller (PSO-

APID) exhibited comparatively better performance over PSO optimized conventional PID controller (PSO-CPID) for all the systems taken above. PSO-APID provided minimum overshoot as well as minimum IAE & ITAE. Initially we have used IAE+ITAE as objective function in order to obtain both load disturbances and set point tracking satisfactorily i.e. overall improved performance with a dead time. After that we increased the dead time in order to see the robustness of the controller and we still found that the proposed PSO-APID maintained overall improved performances. We also observed the similar enhanced performance when *IAE* is used an objective function.

References

[1] C. Dey, and R.K. Mudi, An improved auto-tuning scheme for PID controllers. ISA Trans. 48(4), 396–409(2009).

[2] James McCaffrey, http://msdn.microsoft.com/en-us/maPSOzine/hh335067.aspx

[3] Microsoft, Msdn maPSOzine (august-2011), 86-91.

[4] Randy L. Haupt, and Sue Ellen Haupt, Practical Genetic Algorithms, Wiley publishing Co. Second edition (2004).

CHAPTER-5

ARTIFICIAL BEE COLONY ALGORITHM BASED ADAPTIVE PID CONTROLLER

5.1 Introduction

Artificial Bee Colony (ABC) algorithm [1] which is one of the most recently introduced optimization algorithms simulates the intelligent foraging behavior of a honey bee swarm. Artificial Bee Colony (ABC) algorithm was proposed by Karaboga [2, 3] for optimizing numerical problems. The algorithm simulates the intelligent foraging behavior of honey bee swarms. It is a very simple, robust and population based stochastic optimization algorithm. In ABC algorithm, the colony of artificial bees contains three groups of bees: employed bees, onlookers and scouts. A bee waiting on the dance area for making a decision to choose a food source is called onlooker and one going to the food source visited by it before is named employed bee. The other kind of bee is scout bee that carries out random search for discovering new sources. The employed bee whose food source has been exhausted by the bees becomes a scout. The position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. In ABC system, artificial bees fly around in a multidimensional search space and some (employed and onlooker bees) choose food sources depending on the experience of themselves and their nest mates, and adjust their positions. Some (scouts) fly and choose the food sources randomly without using experience. If the nectar amount of a new source is higher than that of the previous one in their memory, they memorize the new position and forget the previous one. Thus, ABC system combines local search methods, carried out by employed and onlooker bees, with global search methods, managed by onlookers and scouts, attempting to balance exploration and exploitation process.

5.2 Behavior of Honey Bee swarm [3]

The minimal model of forage selection that leads to the emergence of collective intelligence of honey bee swarms [4] consists of three essential components: food sources, employed foragers and unemployed foragers and the model defines two leading modes of the behavior: the recruitment to a nectar source and the abandonment of a source.

Food source: The value of a food source depends on many factors such as its proximity to the nest, its richness or concentration of its energy, and the ease of extracting this energy. For the sake of simplicity, the “profitability” of a food source can be represented with a single quantity.

Employed foragers: They are associated with a particular food source which they are currently exploiting or are “employed” at. They carry with them information about this particular source, its distance and direction from the nest, the profitability of the source and share this information with a certain probability.

Unemployed foragers: They are continually at look out for a food source to exploit. There are two types of unemployed foragers: scouts, searching the environment surrounding the nest for new food sources and onlookers waiting in the nest and establishing a food source through the information shared by employed foragers. The mean number of scouts averaged over conditions is about 5-10%.

The exchange of information among bees is the most important occurrence in the formation of the collective knowledge. While examining the entire hive it is possible to distinguish between some parts that commonly exist in all hives. The most important part of the hive with respect to exchanging information is the dancing area. Communication among bees related to the quality of food sources takes place in the dancing area. This dance is called a waggle dance.

Since information about all the current rich sources is available to an onlooker on the dance floor, probably she can watch numerous dances and decides to employ herself at the most profitable source. There is a greater probability of onlookers choosing more profitable sources since more information is circulated about the more profitable sources. Employed foragers share their information with a probability proportional to the profitability of the food source, and the sharing of this information through waggle dancing is longer in duration. Hence, the recruitment is proportional to the profitability of the food source.

5.3 Proposed Approach

In this work, a particular intelligent behavior of a honey bee swarm, foraging behavior, is considered and a new artificial bee colony (ABC) algorithm simulating this behavior of real honey bees is described for solving multidimensional and multimodal optimization problems.

The main steps of the algorithm are given below:

1 .Initialize the solution population

2 .Evaluate populations

3 .Generate new solutions for the employed bees and keep the best solution

4. Select the visited solution for onlooker bees by their fitness

5 .Generate new solutions for the onlooker bees and keep the best solution

5. Determine if exist an abandoned food source and replace it using a scout bee

7. Save in memory the best solution so far

8. Repeat steps 2-7 until stopping criteria is reached.

Each cycle of the search consists of three steps: moving the employed and onlooker bees onto the food sources and calculating their nectar amounts; and determining the scout

bees and directing them onto possible food sources. A food source position represents a possible solution to the problem to be optimized. The amount of nectar of a food source corresponds to the quality of the solution represented by that food source. Onlookers are placed on the food sources by using a probability based selection process. As the nectar amount of a food source increases, the probability value with which the food source is preferred by onlookers increases, too. Every bee colony has scouts that are the colony's explorers. The explorers do not have any guidance while looking for food. They are primarily concerned with finding any kind of food source. As a result of such behavior, the scouts are characterized by low search costs and a low average in food source quality. Occasionally, the scouts can accidentally discover rich, entirely unknown food sources. In the case of artificial bees, the artificial scouts could have the fast discovery of the group of feasible solutions as a task. In this work, one of the employed bees is selected and classified as the scout bee. The selection is controlled by a control parameter called "limit". If a solution representing a food source is not improved by a predetermined number of trials, then that food source is abandoned by its employed bee and the employed bee is converted to a scout. The number of trials for releasing a food source is equal to the value of "limit" which is an important control parameter of ABC.

In a robust search process exploration and exploitation processes must be carried out together. In the ABC algorithm, while onlookers and employed bees carry out the exploitation process in the search space, the scouts control the exploration process. In the case of real honey bees, the recruitment rate represents a "measure" of how quickly the bee swarm locates and exploits the newly discovered food source. Artificial recruiting process could similarly represent the "measurement" of the speed with which the feasible solutions or the optimal solutions of the difficult optimization problems can be discovered. The survival and progress of the real bee swarm depended upon the rapid discovery and efficient utilization of the best food resources. Similarly the optimal solution of difficult engineering problems is connected to the relatively fast discovery of "good solutions" especially for the problems that need to be solved in real time.

In our case we have decided to use the optimal power of artificial bee colony algorithm (ABC) to design our proposed Adaptive PID (ABC-APID) [5, 5] controllers. Here, all the above said designs including two steps- first, we define the structure of the adaptive PID controller and then the algorithms are used to find their best set of parameters with respect to an objective function. In our experimental purpose we have studied the performances of the developed adaptive controllers based on ABC algorithms over PID & APID controllers for different processes with dead time.

5.4 Objective function of ABC algorithm

The function to be optimized is known as objective function. Here, minimization of the *integral-absolute-error* (*IAE*) [7] or *integral-time-absolute-error* (*ITAE*) or combination of both i.e., (*IAE+ITAE*) is defined as the objective function (performance index or fitness function). The *IAE* and *ITAE* are calculated as:

$$IAE = \int_0^t | e(t) | dt \tag{5.1}$$

$$IAE + ITAE = \int_0^t | e(t) | dt + \int_0^t t | e(t) | dt \tag{5.2}$$

5.5 Initial Settings of Artificial Bee Colony Algorithm

Table 5.1 *Initial settings of the algorithm*

Population	10
Variables	$K_p, K_i, K_d, k_1, k_2, k_3, k_4$
Range of variables	K_p, K_i, K_d , are ±20% of their respective Conventional PID, K_1, [0.5],K_2[0,5],,K_3[0,30], K_4[0,1]
Limit	(population*dimension)/2

5.6 Results

For simulation study, we consider the following systems with dead-time (*L*)

$G_p(s) = e^{-Ls}/(s+1)^2$, L=0.2s, and 0.3s (5.3)

$G_p(s) = e^{-Ls}/s(s+1)$, L=0.2s, and 0.3s. (5.4)

$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.2y^2 = u(t-L)$, L=0.3s, and 0.4s. (5.5)

PH model- $G_p(s) = e^{-Ls}/(s+1)(0.1s+1)^2$, L=0.01s, and 0.02s. (5.6)

For each process model we have used four different types of controller.

(a) **PID.**

(b) **ABC-PID** – K_p, K_i, K_d are ±20% of their respective Conventional PID and these are calculated by ABC algorithm.

(c) **APID.**

(d) **ABC-APID** - In this all seven parameters are varying within the defined range and these are calculated by ABC algorithm.

We have calculated the close loop response characteristics for above process model by using different controllers. 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 absolute error (IAE) and integral time absolute error (ITAE) are calculated for each controller. Performance of our ABC-APID is compared with PID, ABC-PID, and APID. Fourth-order Range-Kutta method is used for numeric integration. The detailed performance analysis for various types of process is discussed below.

5.6.1 Second Order Linear Process

Transfer function of the process is given by

$$G_p(s) = e^{-Ls}/(s+1)^2 \tag{5.7}$$

Response of second order linear process in (5.7) with L=0.2s, and L=0.3s under PID, ABC-PID, APID, and ABC-APID is shown in Fig. 5.1. Performance indices of the process in (5.7) for different controllers are given in Table 5.2 (a) and Table 5.2 (b). Though the controller are tuned for L=0.2s, a higher value i.e., L=0.3s is also tested without changing controllers settings (for PID and APID). Performance analysis reveals that unlike PID, ABC-PID, and APID, ABC-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance.

Table 5.2 (a) -*Performance analysis second order linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE+ITAE	**PID**	60.30	0.90	4.40	2.08	9.29
		ABC-PID	27.90	0.90	3.40	1.44	5.56
		APID	5.37	1.40	4.20	1.36	5.26
		ABC -APID	2.08	2.10	2.30	1.31	4.78
L=0.3s		**PID**	93.93	0.90	12.20	4.41	35.28
		ABC -PID	55.25	0.90	7.50	2.78	15.12
		APID	15.58	1.00	5.70	1.85	14.10
		ABC -APID	3.67	2.10	2.80	1.45	5.22

Table 5.2 (b) *-Performance analysis second order linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE	**PID**	60.30	0.90	4.40	2.08	9.29
		ABC -PID	21.12	1.00	3.50	1.45	5.84
		APID	5.37	1.40	4.20	1.36	5.26
		ABC -APID	0.00	9.10	2.10	1.43	4.81
L=0.3s		**PID**	93.93	0.90	12.20	4.41	35.28
		ABC -PID	44.87	1.00	7.50	2.78	15.12
		APID	15.58	1.00	5.70	1.85	14.10
		ABC -APID	0.17	2.50	2.10	1.47	5.22

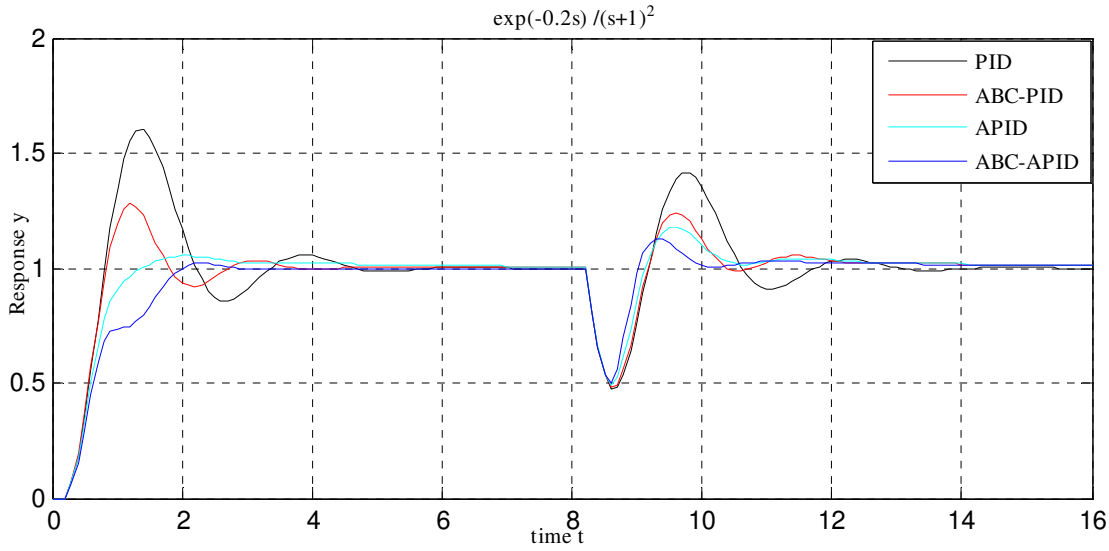


Fig. 5.1 (a) *Response of second order linear process for L=0.2s, minimization of IAE+ITAE*

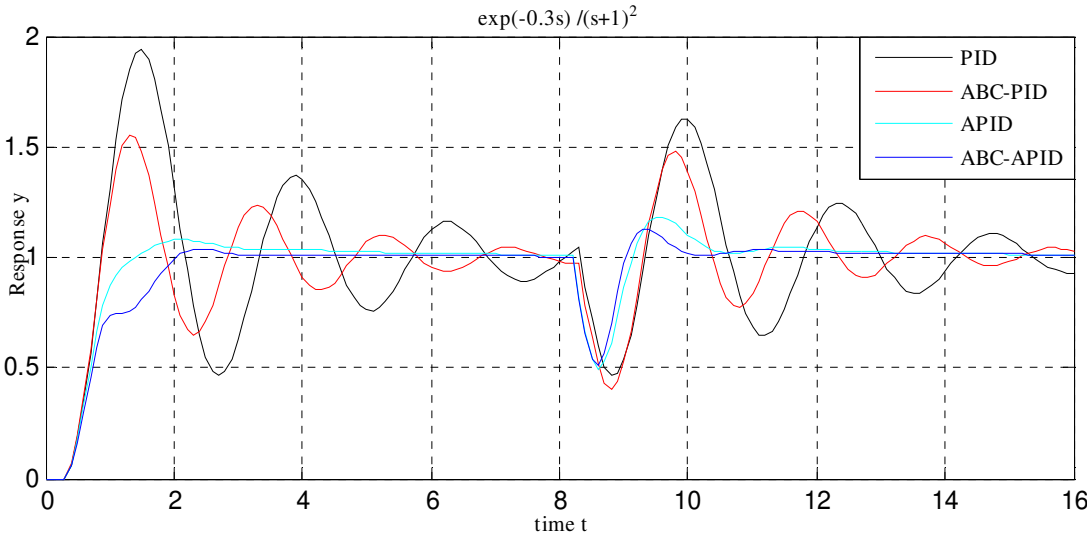


Fig. 5.1 (b) *Response of second order linear process for L=0.3s, minimization of IAE+ITAE*

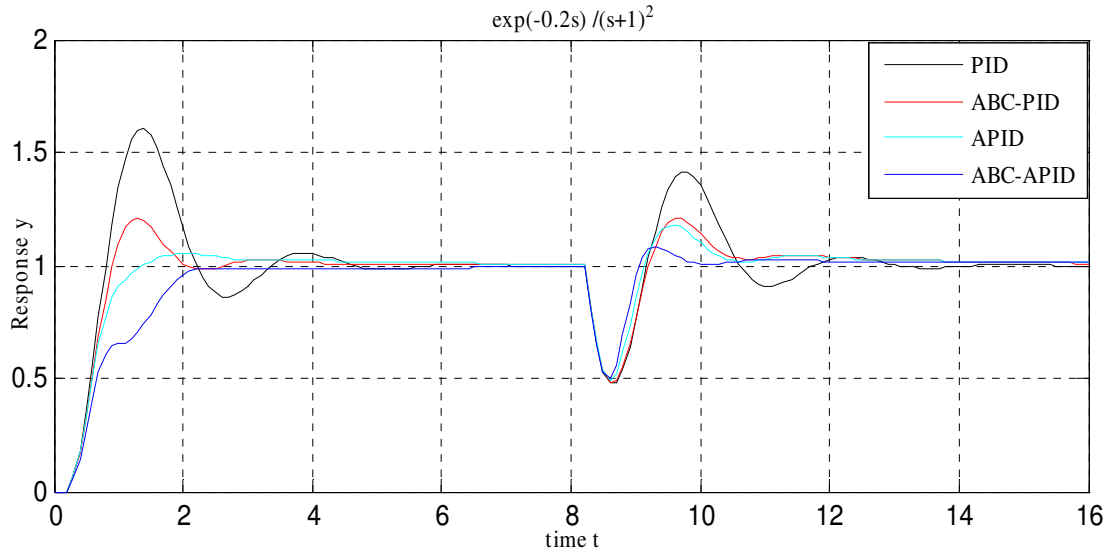


Fig. 5.1 (c) *Response of second order linear process for L=0.2s, minimization of IAE*

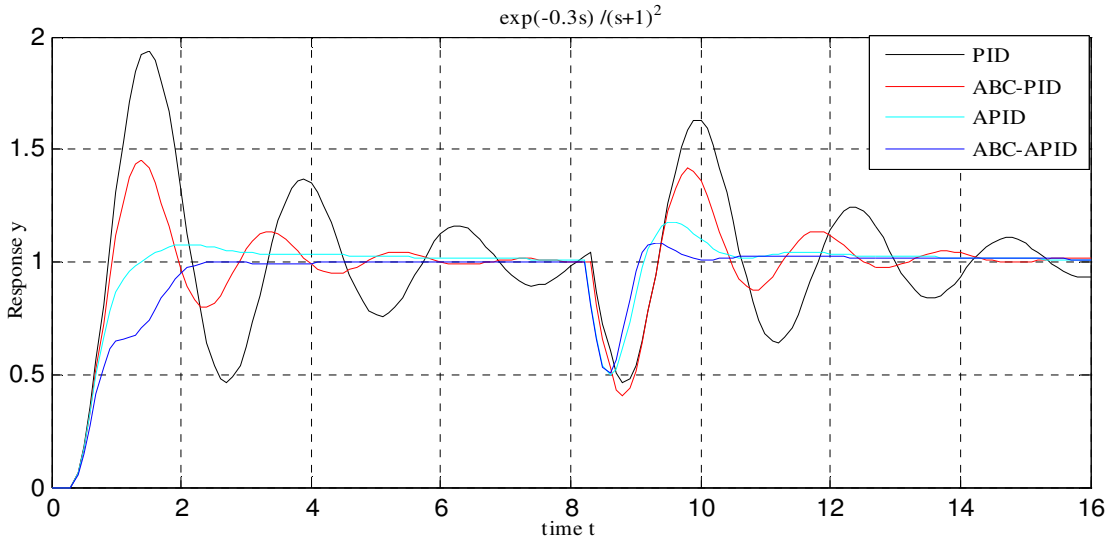


Fig. 5.1 (d) *Response of second order linear process for L=0.3s, minimization of IAE*

5.6.2 First Order with Integrating Process

Transfer function of the process is given by

$$G_p(s) = e^{-Ls} / s(s+1) \tag{5.8}$$

Response of first order integrating process in (5.8) with L=0.2s, and L=0.3s under PID, ABC-PID, APID, and ABC-APID is shown in Fig. 5.2. Performance indices of the process in (5.8) for different controllers are given in Table 5.3 (a) and Table 5.3 (b). Though the controller are tuned for L=0.2s, a higher value i.e., L=0.3s is also tested without changing controllers settings (for PID and APID). Performance analysis reveals that unlike PID, ABC-PID, and APID, ABC-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance.

Table 5.3 (a) *-Performance analysis first order integrating process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE+ITAE	**PID**	77.50	1.10	10.20	3.44	27.19
		ABC-PID	60.01	1.10	5.10	2.30	15.44
		APID	28.75	1.40	11.00	2.46	19.44
		ABC -APID	20.83	1.30	13.10	2.25	18.51
L=0.3s		**PID**	102.2	1.20	17.10	5.70	54.79
		ABC -PID	83.44	1.10	8.80	3.35	25.90
		APID	33.80	1.30	11.00	2.68	22.02
		ABC-APID	27.55	1.20	13.40	2.51	21.67

Table 5.3(b) *-Performance analysis first order integrating process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.2s	IAE	**PID**	77.50	1.10	10.20	3.44	27.19
		ABC -PID	59.74	1.10	5.00	2.26	15.44
		APID	28.75	1.40	11.00	2.46	19.44
		ABC-APID	18.84	1.30	17.40	2.25	21.82
L=0.3s		**PID**	102.2	1.20	17.10	5.70	54.79
		ABC -PID	83.44	1.10	9.80	3.35	26.58
		APID	33.80	1.30	11.00	2.68	22.02
		ABC -APID	27.55	1.20	13.40	2.51	21.67

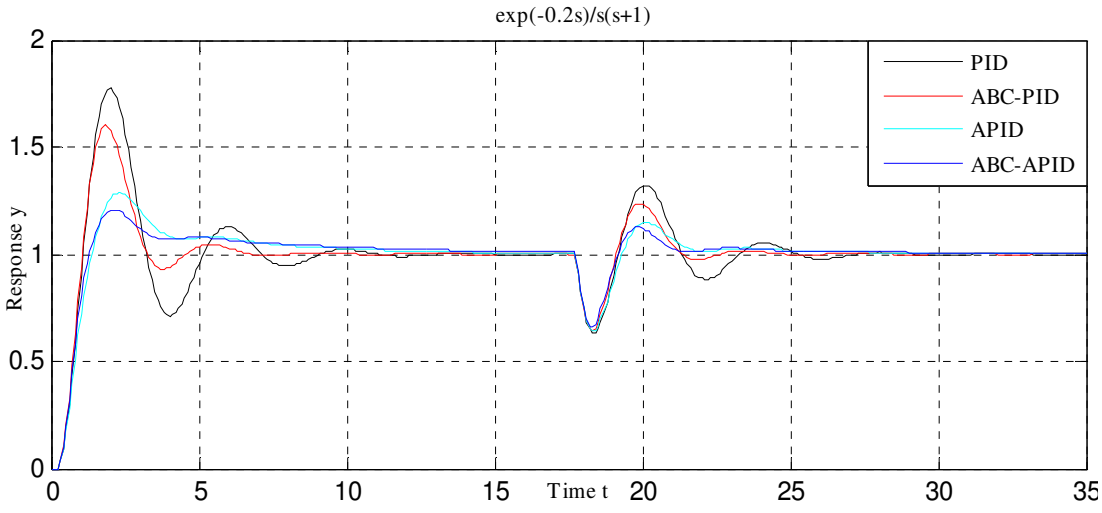


Fig. 5.2 (a) *Response of first order integrating process for L=0.2s, minimization of IAE+ITAE*

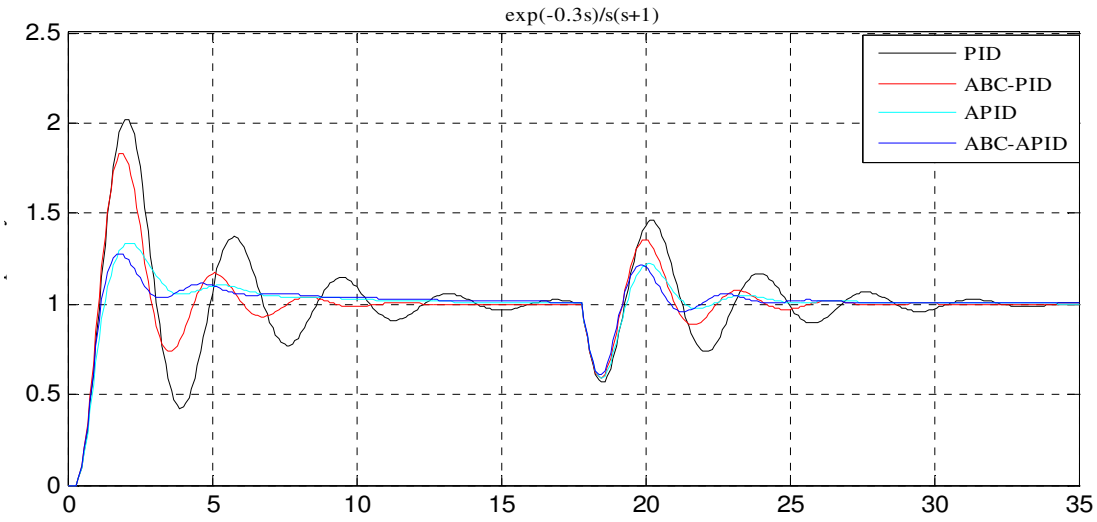


Fig. 5.2 (b) *Response of first order integrating process for L=0.3s, minimization of IAE+ITAE*

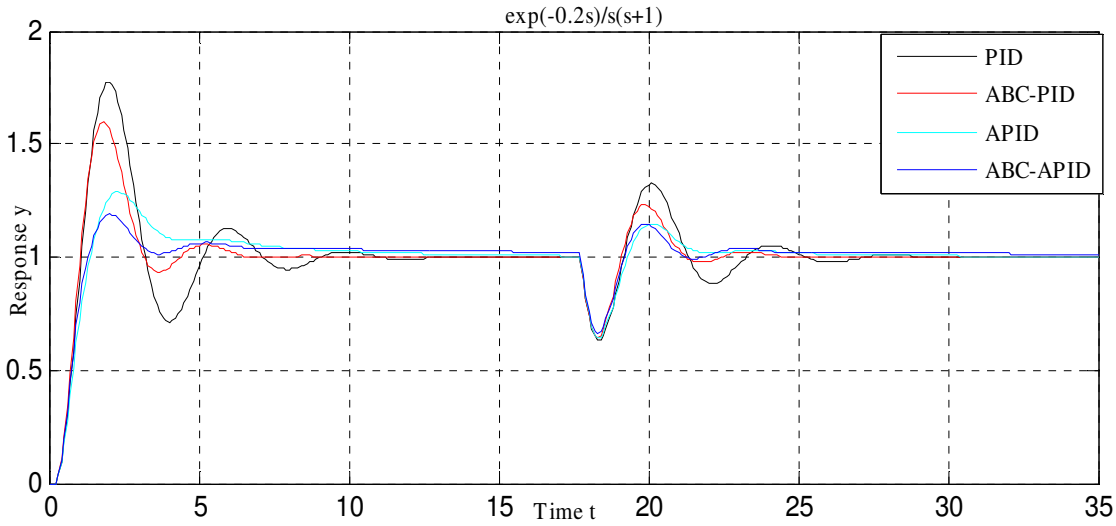


Fig. 5.2 (c) *Response of first order integrating process for L=0.2s, minimization of IAE*

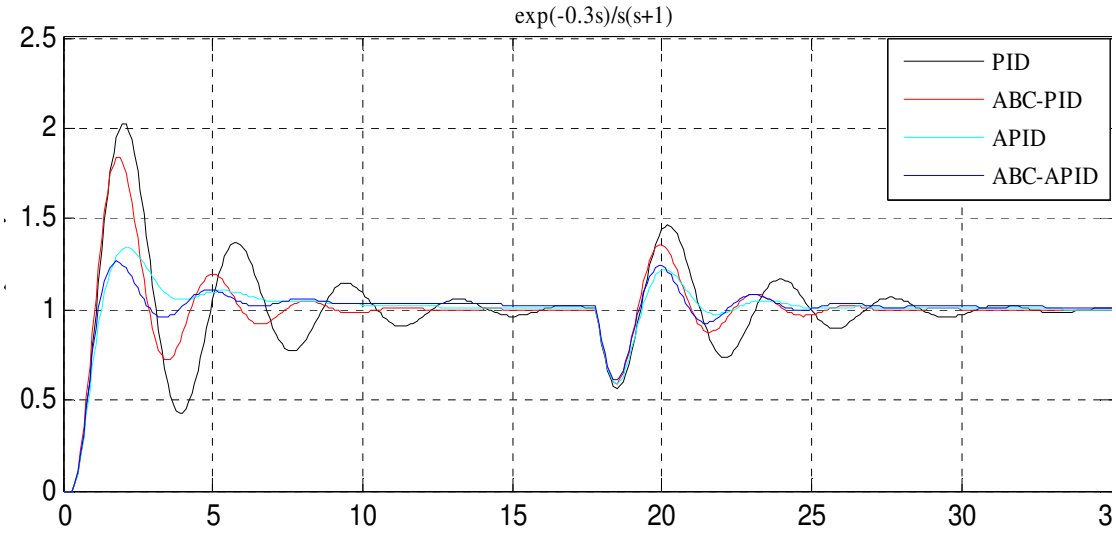


Fig. 5.2 (d) *Response of first order integrating process for L=0.3s, minimization of IAE*

5.6.3 Second Order Nonlinear Process

$$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 0.2y^2 = u(t-L) \tag{5.9}$$

Response of second order non linear process in (5.9) with L=0.3s and L=0.4s under PID, ABC-PID, APID, and ABC-APID is shown in Fig. 5.3. Performance indices of the process in (5.9) for different controllers are given in Tables 5.4(a) and 5.4(b). Though the controller are tuned for L=0.3s, a higher value i.e., L=0.4s is also tested without changing controllers settings (for PID and APID). We can conclude that unlike PID, ABC-PID, and APID, ABC-APID is capable of providing acceptable and remarkably improved performance during both set point change and load disturbance.

Table 5.4 (a) -*Performance analysis of second order non-linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.3s	IAE+ITAE	**PID**	66.10	1.40	9.30	4.13	46.60
		ABC-PID	51.96	1.40	5.90	3.14	32.98
		APID	19.18	1.70	10.60	2.95	34.88
		ABC -APID	9.82	1.80	10.40	2.54	29.18
L=0.4s		**PID**	83.11	1.50	13.30	5.78	72.4
		ABC - PID	68.82	1.40	9.90	4.22	48.25
		APID	25.15	1.70	10.80	3.26	39.45
		ABC-APID	1.20	3.10	2.40	2.50	30.01

Table 5.4 (b) -*Performance analysis of second order non-linear process*

Dead time	Objective function	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.3s	IAE	**PID**	66.10	1.40	9.30	4.13	46.60
		ABC-PID	51.93	1.50	7.50	3.39	35.93
		APID	19.18	1.70	10.60	2.95	34.88
		ABC-APID	2.69	2.90	5.60	2.37	27.31
L=0.4s		**PID**	83.11	1.50	13.30	5.78	72.4
		ABC -PID	65.59	1.50	7.50	4.13	46.24
		APID	25.15	1.70	10.80	3.26	39.45
		ABC-APID	4.29	3.40	5.20	2.54	26.07

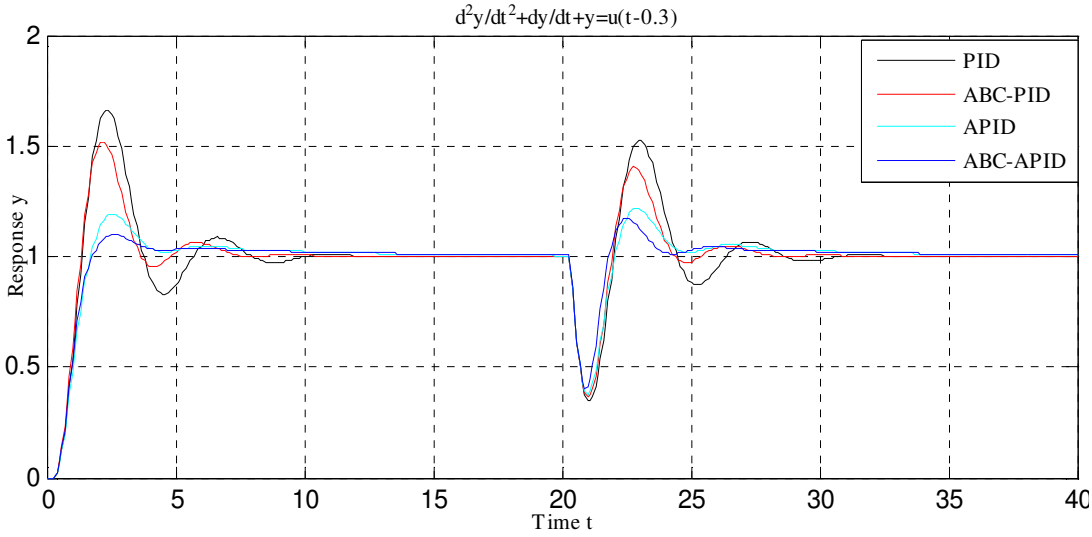


Fig. 5.3 (a) *Response of second order non-linear process for L=0.3s, minimization of IAE+ITAE*

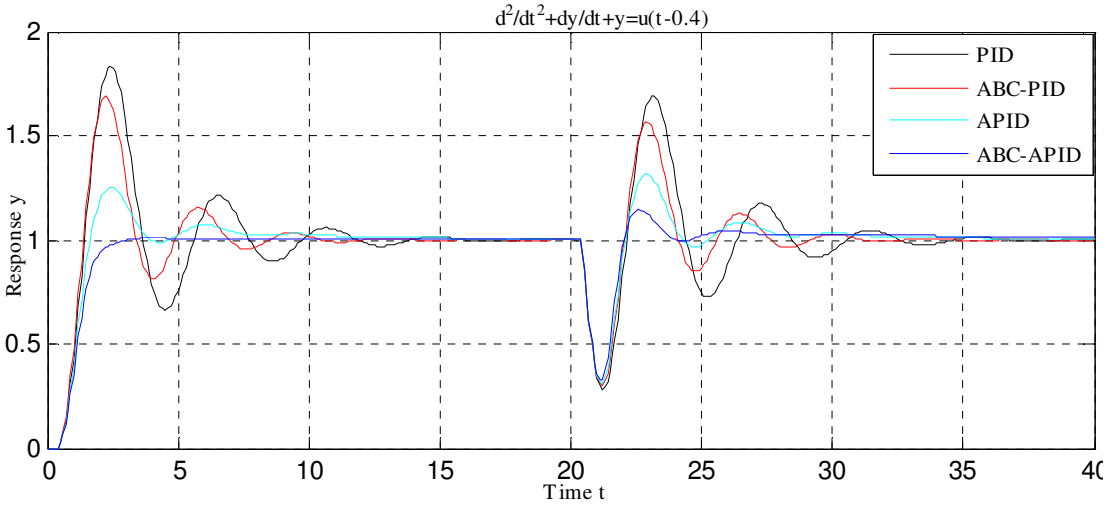


Fig. 5.3 (b) *Response of second order non-linear process for L=0.4s, minimization of IAE+ITAE*

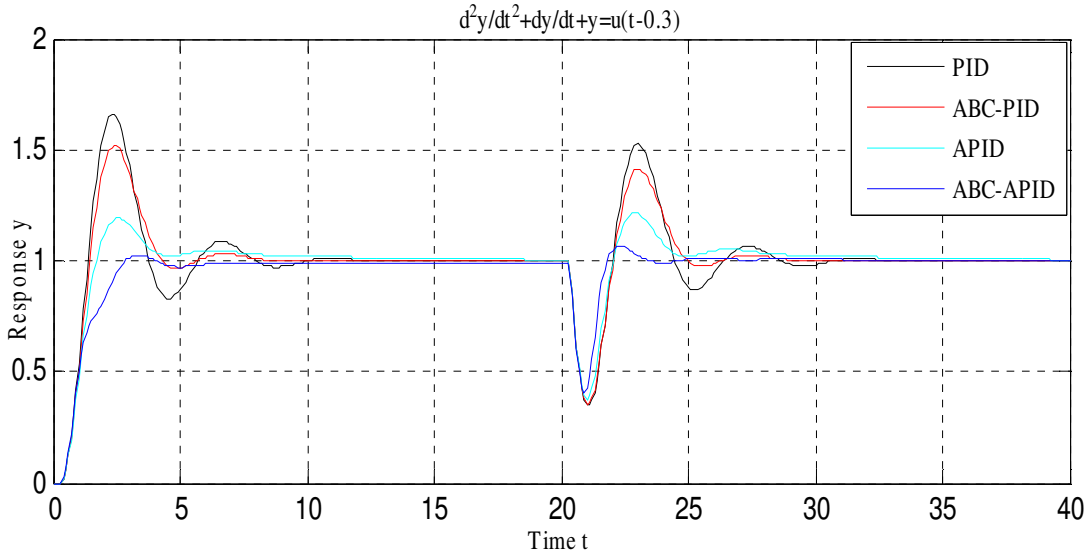


Fig. 5.3 (c) *Response of second order non-linear process for L=0.3s, minimization of IAE*

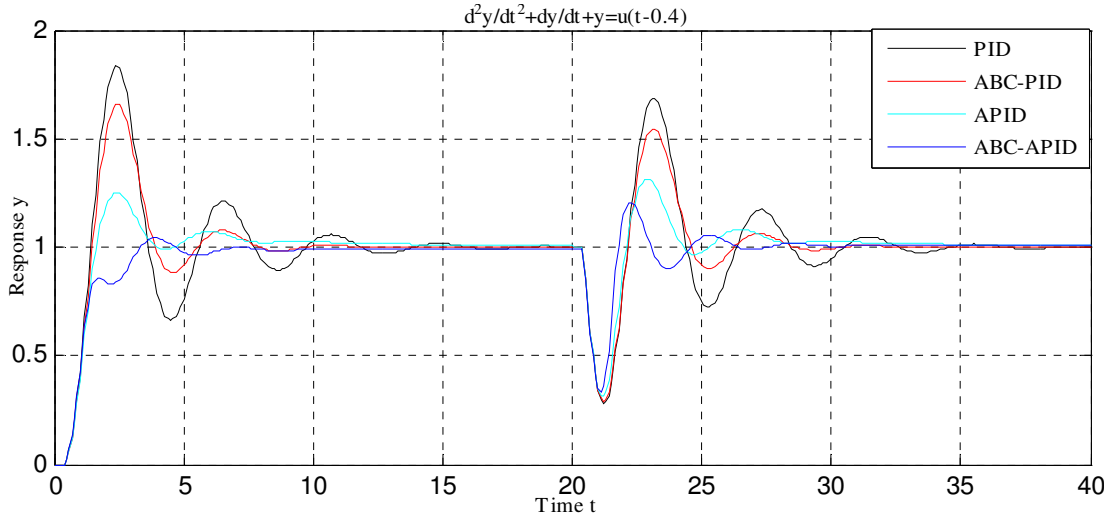


Fig. 5.3 (d) *Response of second order non-linear process for L=0.4s, minimization of IAE*

5.6.4 pH-Neutralization Process

For pH-neutralization process we consider following linear model

$$G_p(s) = e^{-Ls}/(s+1)(0.1s+1)^2 \tag{5.10}$$

Response of pH neutralization process in (5.10) with L=0.01s and L=0.02s under PID, ABC-PID, APID, and ABC-APID is shown in Fig. 5.4. Performance indices of the process in (5.10) for different controllers are given in Tables 5.5(a) and 5.5(b). Though the controller are tuned for L=0.01s, a higher value i.e., L=0.02s is also tested without changing controllers settings (for PID and APID). We can conclude that unlike PID, ABC-PID, and APID, ABC-APID is

capable of providing acceptable and remarkably improved performance during both set point change and load disturbance.

Table 5.5(a)- *Performance analysis for pH-neutralization process*

Dead time(s)	Objective	Controllers	%OS	T_r(s)	T_s(s)	IAE	ITAE
L=0.001	IAE+ITAE	**PID**	64.21	0.26	1.69	0.72	1.40
		ABC-PID	49.54	0.26	1.25	0.54	0.97
		APID	23.94	0.30	1.77	0.53	1.13
		ABC-APID	8.59	0.33	0.59	0.41	0.89
L=0.002		**PID**	72.33	0.26	1.78	0.84	1.70
		ABC-PID	56.70	0.26	1.27	0.61	1.12
		APID	28.49	0.30	1.84	0.56	1.18
		ABC-APID	11.74	0.32	1.06	0.44	0.95

Table 5.5(b) *-Performance analysis for pH-neutralization process*

Dead time(s)	Objective function	Controllers	%OS	T_r (s)	T_s(s)	IAE	ITAE
L=0.01	IAE	**PID**	64.21	0.26	1.69	0.72	1.40
		ABC-PID	49.53	0.25	1.25	0.54	0.97
		APID	23.94	0.30	1.77	0.53	1.13
		ABC-APID	8.59	0.33	0.59	0.41	0.89
L=0.02		**PID**	72.33	0.26	1.78	0.84	1.70
		ABC-PID	56.7	0.26	1.27	0.61	1.12
		APID	28.49	0.30	1.84	0.56	1.18
		ABC-APID	11.74	0.32	1.06	0.44	0.95

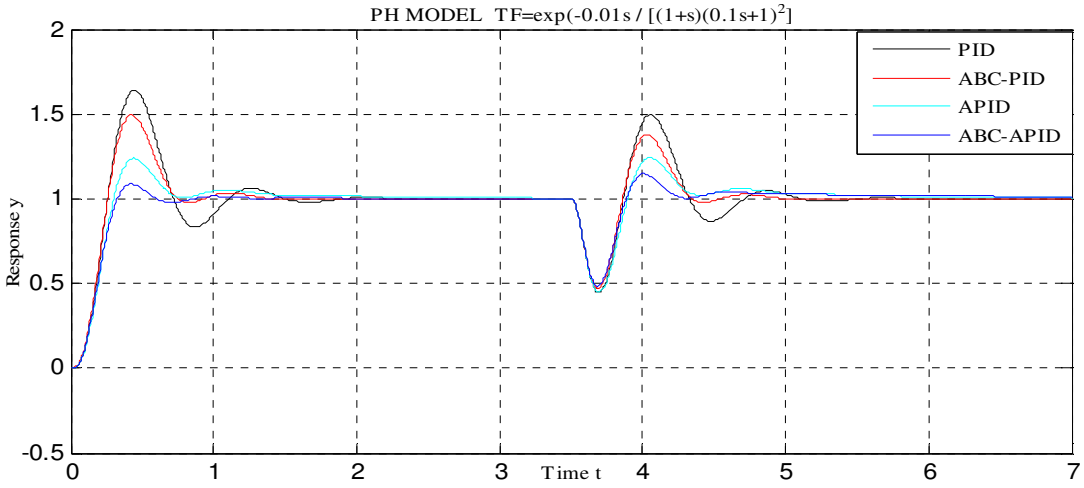


Fig. 5.4 (a) *Response of pH neutralization process for L=0.01s, minimization of IAE+ITAE*

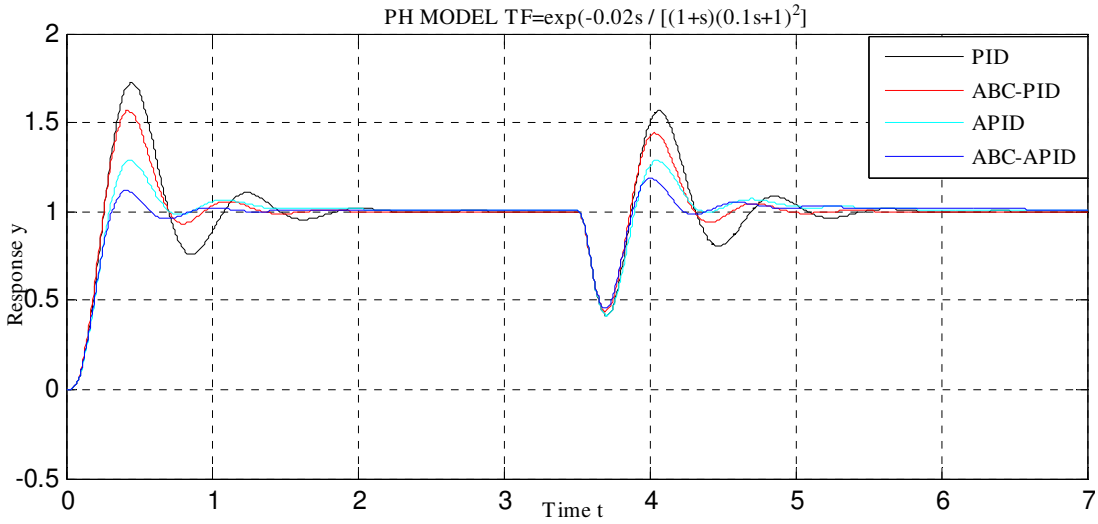


Fig. 5.4 (b) *Response of pH neutralization process for L=0.02s, minimization of IAE+ITAE*

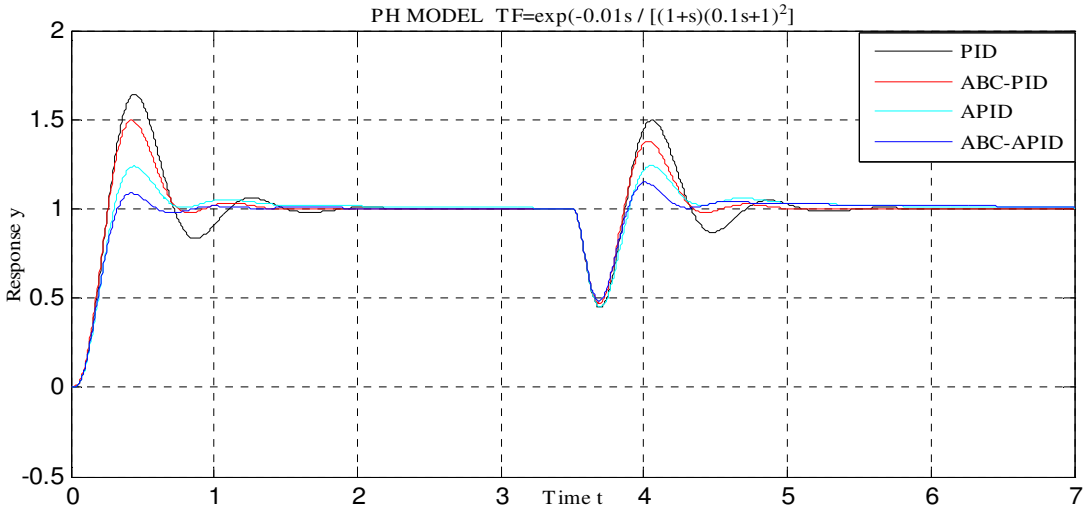


Fig. 5.4 (c) *Response of pH neutralization process for L=0.01s, minimization of IAE*

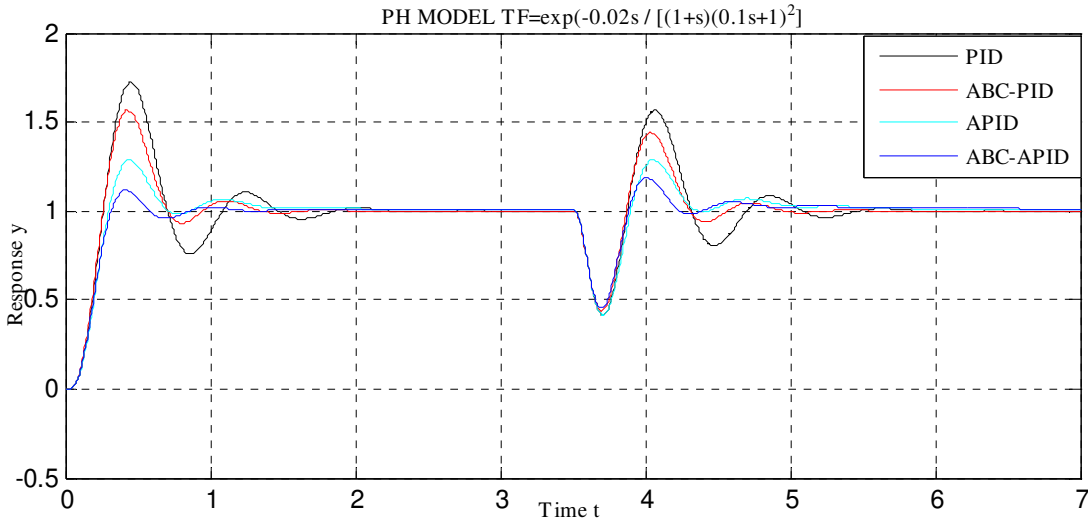


Fig. 5.4 (d) *Response of pH neutralization process for L=0.02s, minimization of IAE*

5.7 Conclusion

In this study we have exhibited how the performance of adaptive PID (APID) controllers is enhanced when it is used with the optimal power of bio-inspired algorithm. Here, ABC is said to be bio-inspired because of the fact that they depicts how the honey bees are sustained biologically. When the conventional controller with the optimal power of ABC are not able to provide the satisfactory performances over APID controller, we decided to develop the ABC-APID respectively in order to get overall improved performance for both set point change as

well as load disturbances. We also observed that ABC-APID provides improved performance as it provides minimum overshoot & improved rise time & settling time. It also provides lower value of IAE & ITAE which indicates that it can track set point and reject load disturbance properly. Initially we have used IAE+ITAE as objective function with a dead time. After that the dead time is improved in order to see the robustness of the controller & we found the performance is still satisfactory for ABC-APID. We also get the similar enhanced performance when we use only IAE as an objective function.

References

[1] Karaboga. D., Basturk. B.,; Artificial bee colony (ABC) optimization algorithm for solving constrained optimization Problems, LNCS: Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing, pp.789-798 (2007).

[2] Akay. B., Karaboga. D.,; Parameter Tuning for the Artificial Bee Colony Algorithm, Lecture Notes in Artificial Intelligence, pp.608-619 (2007).

[3] Karaboga, D., An idea based on honey bee swarm for numerical optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[4] B. Basturk, Dervis Karaboga, An Artificial Bee Colony (ABC) Algorithm for Numeric function optimization, IEEE Swarm Intelligence Symposium 2006, May 12-14, 2006, Indianapolis, Indiana, USA.

[5] Dey, C., Mudi, R.K., ; An improved auto-tuning scheme for PID controllers. ISA Trans. 48(4), 396–409 (2009).

[6] Dey, C., Mudi, R.K., Lee. T.T.,; An improved auto-tuning scheme for PI controllers. ISA Transactions. 47, 45- 52 (2008).

[7] Seborg, D.E., Edgar. T.F.,; Adaptive control strategies for process control: A survey. AICHE J. 32(5), 881–913 (1986).

CHAPTER-6

CONCLUSION & FUTURE SCOPE

6.1 Conclusion

In this study, we have incorporated four optimization techniques - Genetic algorithm (GA), Bacterial foraging optimization (BFO) Particle swarm optimization algorithm (PSO) and Artificial bee colony algorithm (ABC) on an already developed adaptive PID (APID) controller with a view to (i) overcome its empirical and trial method of choosing appropriate tunable parameters, and (ii) achieving its optimal performance. Here, all the seven tunable parameters of the APID controller have been optimized by GA, BFO, PSO or ABC algorithm for various types of processes. The derived optimal controllers, GA-APID, BFO-APID, PSO-APID and ABC-APID are tested through extensive simulation experiments, even with increased dead-time for checking robustness. Performances of the optimal controllers (GA-APID, BFO-APID, PSO-APID and ABC-APID) have been compared with PID, optimized PID (GA-PID, BFO-PID, PSO-PID and ABC-PID), and APID. Detailed performance analysis revealed that all the GA-APID, BFO-APID, PSO-APID and BFO-APID provide significantly improved performances over others, justifying the usefulness of this study.

6.2 Future Scope

In this study we have used four different optimization algorithms to optimize the parameters of the both PID & adaptive PID (APID) controllers for different transfer functions. These algorithms are taken on the basis of their convergence rate & runtime. Initially we used Ziegler-Nichols (ZN) method to tune the parameters i.e., K_p, K_i, K_d of the PID controller. Now, this tuning can be done by using relay feedback method or Cohen-Coon method instead of ZN method which may give improved performance of the controller parameters.

While developing GA-APID, PSO-APID, BFO-APID & ABC-APID respectively, we have considered range of variables i.e., K_p, K_i, K_d are ±20% of their respective Conventional PID, and for the four constants are $k_1[0,5], k_2[0,5], k_3[0,30]$ and $k_4[0,1]$. Therefore, by increasing the range of the variables, we may further improve the performance.

Here, we consider only IAE & IAE+ITAE as objective functions with single weightage. The performance of the controller may be improved if we increase the weight of objective functions. Moreover, when a multivariable process comes into consideration the behavior of the controller should changes & then how the algorithms optimize the parameter of the controllers is an important phenomenon to observe. Hence, we can modify this study by using multivariable process instead of single variable process.

For future aspect we may study the stability of the process when the controller is optimized by the algorithm used. This stability analysis can be done by using Bode plot, Nyquist criteria etc.

We have studied the robustness of the controller by increasing the dead time for each transfer function. Now, this robustness can be observed by using Kharitonov polynomials which can specific the dead time up to which the controller can show its robustness specifically. Finally the performance can be improved by using different optimization algorithms.