ISSN 1996-3343

Asian Journal of Applied Sciences

http://knowledgiascientific.com

Knowledgia SCIENTIFIC
A Place to Publish Outstanding Research

Asian Journal of Applied Sciences 2 (3): 275-295, 2009
ISSN 1996-3343

A Novel Fast and Efficient Evolutionary Method for Optimal Design of Proportional Integral Derivative Controllers for Automatic Voltage Regulator Systems

[1,2]S.M.A. Mohammadi, [1]A.A. Gharaveisi and [2]M. Mashinchi
[1]Department of Electrical Engineering, Shahid Bahonar University of Kerman, Iran
[2]Department of Mathematics, Shahid Bahonar University of Kerman, Iran

Abstract: An efficient and powerful design method for calculating optimal Proportional-Integral-Derivative (PID) controllers for AVR systems is proposed. The method is an improved version of the Discrete Action Reinforcement Learning Automata (DARLA) while discrete probability functions (DPF) of the design variables are not considered independent. The results of the proposed method called Extended Discrete Action Reinforcement Learning Automata (EDARLA) are compared to the results obtained by the well known Ziegler-Nichols (ZN), conventional DARLA and Genetic Algorithms (GA) and conventional CARLA approaches. The extensive simulation results prove superiority of the proposed design method in terms of optimality, efficiency, computation burden and being less sensitive to the ranges considered for the design variables that is the search space. Besides being successful in providing globally optimal results, due to high efficiency and lower computation time, the proposed approach can be considered an interesting candidate for designing and tuning optimal adaptive PID controllers for many practical systems.

Key words: DARLA, CARLA, GA, EDARLA, PID controller, reinforced learning, AVR system

INTRODUCTION

Reinforcement learning (RL) approaches are new but quite promising approaches giving a new scientific insight into the intelligent systems area with immense practical applications (Oh *et al*., 2000; Kamal and Murata, 2008; Duan *et al*., 2007; Charvillat and Grigoras, 2007; Howell *et al*., 1997a, 2000b, 2001c). Reinforcement learning is different to supervised learning, which is a kind of learning being widely studied in current research in machine learning, statistical pattern recognition and artificial neural networks. Supervised learning basically means learning from examples provided by a knowledgeable external supervisor. This is an important kind of learning, but it alone is not adequate without learning from interactions. Interactive problems are often off-line. An agent with its goal embedded in an environment learns how to transform one environmental state into another. The agents with the ability of performing this task with minimal human supervision are called autonomous. Learning from an environment is more robust because agents are directly affected by the dynamics of the environment (the system under control).

Generally, adapting and tuning of the process parameters can be performed by either Continuous Action Reinforcement Learning Automata (CARLA), or Discrete Action Reinforcement Learning Automata (DARLA) (Howell *et al*., 1997). DARLA uses discrete action space making it more appropriate for the discrete engineering applications. CARLA was actually developed as an extension of the discrete stochastic learning automata methodology. Both DARLA and CARLA operate through

Corresponding Author: Ali Akbar Gharaveisi, Department of Electrical Engineering, Shahid Bahonar University of Kerman, Iran

interactions with a random or unknown environment by selecting actions in a stochastic trial and error process. CARLA replaces the discrete action space with a continuous one, using continuous probability distributions and hence making it more appropriate for engineering applications with continuous time variables. The only interconnection mechanism between DARLA is provided through the environment and via a shared performance or evaluation function. In each iteration, every action has an associated discrete probability functions (DPF) being used as a basis for its evaluation. The calculation is done separately for n disjoint DPFs where n is the number of parameters must be tuned so that an index function is optimized.

In this study we consider an n-dimensional search space for the state of the environment (system) with a joint multi-variable discrete probability function for each cell in the search space which is updated at discrete samples. As will be shown, DARLA and CARLA methods significantly suffer from being too sensitive to the ranges considered for the design variables. If the ranges are too small, then there will be low chance to reach globally optimal results. On the other side, having large sets for the variables makes the method time consuming. The proposed approach (EDARLA) is potentially promising to achieve better results and can more easily be adjusted so that the speed of convergence is significantly improved.

Numerous control methods such as fuzzy control, adaptive control and neuro-fuzzy control have been studied by Kim and Han (2006), Ying (2000) and Seng *et al*. (1999). Among them, the best known is the Proportional-Integral-Derivative (PID) controller, which has been widely used in the industry because of its simple structure and robust performance in a wide range of operating conditions. Unfortunately, it is still rather difficult to tune properly the parameters of PID controllers, because many industrial plants are often burdened with problems such as being of high order and existence of nonlinearities (Ho *et al*., 2006). One of the first methods used as a classical tuning rule was proposed by Ziegler and Nichols (Guillermo *et al*., 2005). In general, it is often hard to determine optimal or near optimal PID parameters with the ZN method in many industrial plants For these reasons, it is highly desirable to increase the capabilities of PID controllers by adding new researches.

Genetic Algorithm (GA) has recently received much interest for achieving high efficiency and searching globally optimal solution in search space (Chou, 2006; Lin and Xu, 2006). Due to its high potential for global optimization, GA has received great attention in control systems such as the search of optimal PID controller parameters. Although GA has widely been applied to many control systems, its natural genetic operations would still result in enormous computational efforts (Chou, 2006). Though the GA methods have been employed successfully to solve complex optimization problems, recent research has identified some deficiencies in GA performance. This degradation in efficiency is more apparent in applications with the parameters being optimized are highly correlated. So, the crossover and mutation operations cannot ensure better fitness of offspring, because chromosomes of the population have similar structures and their average fitness is high toward the end of the evolutionary process (Liu, 2008). Moreover, the poor premature convergence of GA degrades its performance and reduces its search capability (Liu, 2008).

To explore the superiority of the proposed optimization approach, the EDARLA method has been applied to design a PID controller for an Automatic Voltage Regulator (AVR) system for power generation system as an important industrial plant. The generator excitation system regulates the generator's voltage and controls the reactive power flow using an AVR system. The role of the AVR is to hold the terminal voltage magnitude of a synchronous generator at a pre-specified level. Hence, the performance and stability of the AVR system seriously affect the security of the whole power system. In this paper, besides demonstrating how to employ the classic CARLA and DARLA as well as the proposed EDARLA methods to obtain the optimal PID controller parameters, it is shown that the proposed method has better performance compared to the conventional reinforcement learning methods.

CARLA AND DARLA TECHNIQUES

Here, both CARLA and DARLA techniques are briefly reviewed (Howell *et al*., 1997).

CARLA

In order to practically implement CARLA, the probability distributions $f_i(x_i)$ are stored and updated at successive sample points. The sampled vector x_i must be updated after each iteration k according to its updated probability distribution $f_i(x_i, k + 1)$. Every action set producing some improvements in the system performance achieves a higher performance score denoted by $\beta(k)$ and their probability of reselection is increased through the corresponding learning sub-system. It is achieved by modifying $f_i(x)$ by Gaussian neighborhood function centered at the successful action value. The neighborhood function incorporates in increasing the probability of the original successful action as well as the probability of all actions close to it. The assumption is that the performance surface over a range for each action is continuous and stationary or with slow variation. As the system learns, each probability distribution usually converges to a single Gaussian. Referring to the ith action (parameter), each x_i is defined within a pre-specified range $[x_i(min), x_i(x_i, k)]$. In each of iterations, each new action is randomly chosen based on its probability distribution function $f_i(x_i, k)$, which is initially a uniform function:

$$f_i(x_i,1) = \begin{cases} 1/\left[x_i(max) - x_i(min)\right] \text{where} \quad x_i \in \left\{x_i(max) - x_i(min)\right\} \\ 0 \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{1}$$

The new action i is selected by:

$$z_i(k) = \int_0^{x_i(k)} f_i(x_i,k)\,dx_i \tag{2}$$

where, z(k) takes random values uniformly within the range [0,1]. When the set of all updated actions are available, then the set is again evaluated in the environment for an appropriate timeframe and the scalar function J_{cal} (k) is calculated. Where, $J_{cal}(k)$ is the cost function at kth iteration and calculated based on the Performance Index (PI) to be optimized. This PI function can generally be defined based on the desired characteristics of the system under control such as steady state error, amplitude of the control signal, overshoot and settling time. Then the multiplier $\beta(k)$ is calculated as follows:

$$\beta(k) = \min\left\{\max\left\{0, \frac{J_{med} - J(k)}{J_{med} - J_{min}}\right\}, 1\right\} \tag{3}$$

As seen, the cost J_{cal} (k) is compared with both average and minimum costs J_{med}, J_{min} calculated based on a memory set of R previous values. The algorithm uses a reward/inaction rule. When an action set generates a cost below the current median level has no effect ($\beta(k) = 0$) and the maximum reinforcement (reward) is also capped at $\beta(k) = 1$.

After performance evaluation, each probability density function is updated according to the following rule:

$$f_i(x_i,k+1) = \begin{cases} \alpha(k)\left[f_i(x_i,k) + \beta(k)H(x_i,r)\right] & \text{if} \quad x_i \in \{x_i(min), x_i(max)\} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where, H(x,r) is a symmetric Gaussian neighborhood function centered at the action choice r = x(k):

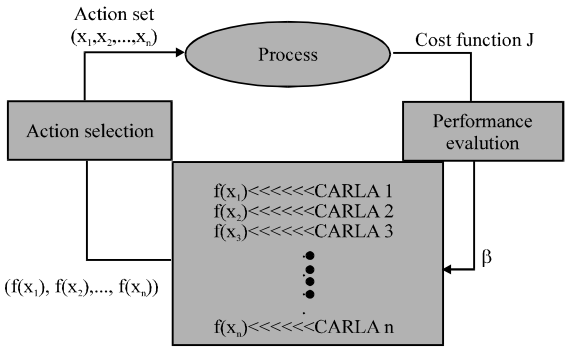

Fig. 1: Learning system by CARLA

$$H(x_i, r) = \left(\frac{g_h}{(x_{max} - x_{min})}\right) \exp\left(-\frac{(x_i - r)^2}{2(g_w (x_{max} - x_{min}))^2}\right) \quad (5)$$

And g_h and g_w are adjustable constants that influence the speed and resolution of the learning process by adjusting the normalized 'height' and width of H. The parameter $\alpha(k)$ is chosen according to Eq. 6 to renormalize the distribution functions fort k+1 iteration:

$$\alpha(k) = \frac{1}{\int_{x_i(min)}^{x_i(max)} \left[f_i(x_i, k) + \beta(k) H(x_i, r)\right] dx_i} \quad (6)$$

For practical implementation, the distribution functions are each stored at discrete points with equal inter-sample probability and linear interpolation is usually used to determine the values at intermediate positions. A typical layout of the method is shown in Fig. 1.

DARLA

In order to implement DARLA, each DPF $f_i(d)$ must be stored and updated at discrete sample points. The most efficient data storage can be achieved using equal inter-sample probability rather than equal sampling at d = 1,2,3,...,N, but this imposes some more computational burden. Like CARLA, any action set that produces an improvement in the system performance receives a higher-performance score $\beta(k)$ and thus its probability of reselection is increased. This is achieved by modifying $f_i(d)$ through the use of an Inverse Exponential Neighborhood function centered at the recent successful action. The neighborhood function increases the probability of the original action and the probability of all actions 'close ' to that selected as well. The assumption is that the performance surface over a range in each action is discrete and slowly varying. Within each iteration k of the algorithm, each action is chosen based on the corresponding probability distribution function $f_i(d, k)$ which is initially chosen to be a uniform function:

$$f_i(d,1) = \begin{cases} 1/N_i & \text{where} \quad d = 1,2,....N_i, \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

The action d is selected by solving:

$$\sum_d f_i(d) = \text{Cumulative} \tag{8}$$

where, the constant cumulative is uniformly selected at random within the range [0,1]. When all n actions are selected, the set is evaluated in the environment for a suitable time and the scalar cost $J_{cal}(k)$ is calculated. Performance evaluation is then carried out using Eq. 9 and 10:

$$\beta(k) = \min\left\{\max\left\{0, \frac{J_{med} - J(k)}{J_{med} - J_{min}}\right\}, 1\right\} \tag{9}$$

Again, the algorithm uses a reward/inaction rule. The action sets generating a cost not better than the current average level receive no reward ($\beta(k) = 0$) and the maximum reinforcement (reward) is also capped at $\beta(k) = 1$. After performance evaluation, each discrete probability function is updated according to the following rule:

$$f_i(d, k+1) = \begin{cases} \alpha(k)\left[f_i(d_i, k) + \beta(k)Q(d_i, r)\right] & \text{if } d_i = 1, 2, \ldots N;\ i = 1, 2, \ldots n \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where, Q is a symmetric Inverse Exponential neighborhood function centered on the action choice, $r = d_i(k)$:

$$Q(d_i, r) = \lambda 2^{-(d_i - r)^2} \tag{11}$$

And λ is an adjustable parameter that influences speed and resolution of the learning. The parameter $\alpha(k)$ is also calculated by Eq. 12 to renormalize the distribution functions in k+1-th iteration:

$$\alpha(k) = \frac{1}{\sum_{i=1}^{N_i} f_i(d_i, k) + \beta(k)Q(d_i, r)} \tag{12}$$

A typical layout of the learning system by DARLA is shown in Fig. 2.

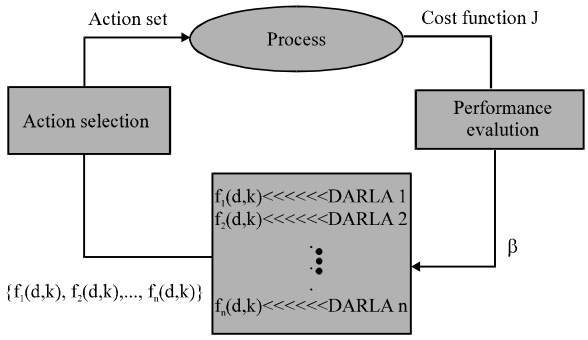

Fig. 2: Learning system by DARLA

The Proposed Extended DARLA METHOD (EDARLA)

Let n be the number of parameters must be adjusted so that the PI function reaches its minimum value. We can search for the controller's parameters in a for each cell dimensional space using a common discrete probability function (CDPF) $f_{X1,X2,...,Xn}(x_1, x_2, ...,x_n)$ for each cell rather than n separate DPFs. The idea behind this strategy is that a CDPF has by far much more information than n separate DPFs. Each cell in the n-dimensional space must be stored and updated at discrete sample points which are here updated to either a new value or zero for simplicity. A typical layout for the proposed method is shown in Fig. 3.

In this method, instead of n times calculating DPFs, we calculate one matrix function so that the speed of convergence increases by efficient matrix calculation algorithms. More importantly, it also potentially improves chance of reaching some better results closer to the globally optimal results. Suppose n = 3 then each three-dimensional probability function forms a cubic space as shown in Fig. 4. The number of distinct values at each dimension is denoted as Ndiv1, Ndiv2 and Ndiv3, respectively.

Within each iteration, each action has an associated probability density function $f_X(x)$ being used as the basis for its selection. The action sets improving the system performance, receive a higher-performance score, thus their probability of reselection increases through the learning sub-system. This is achieved by modifying $f_X(x)$ through the use of a three-dimensional neighborhood function centered at the successful action. The neighborhood function increases the probability of the original action, as well as the probability of the actions close to the point selected. With all *n* actions selected, the set is now evaluated in the environment for a suitable time and a scalar cost value $J_{cal}(k)$ is calculated and compared with a memory set of previous values as described before.

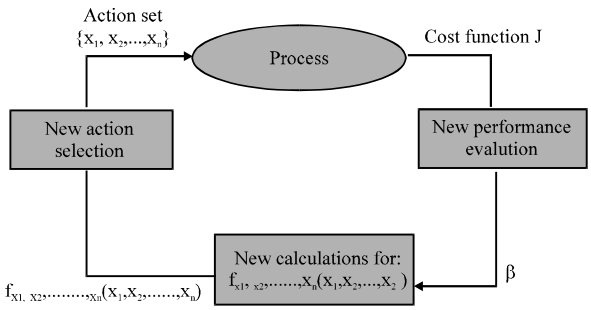

Fig. 3: Learning system by Proposed Algorithm

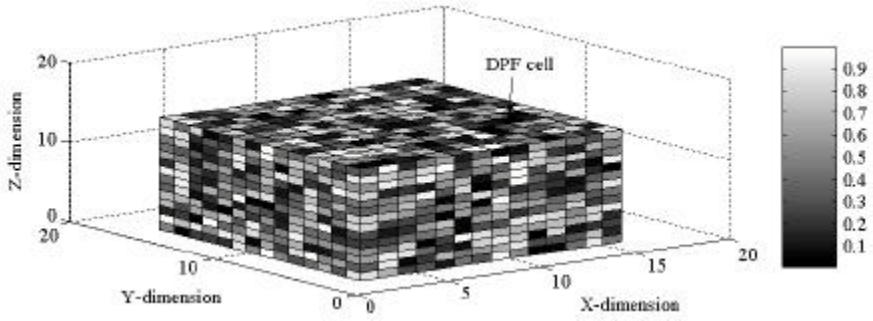

Fig. 4: A typical three- dimensional probability functional cube

After performing the performance evaluation process, each probability density function is updated according to a specified rule. Furthermore, EDARLA can consider several loops for searching. For example, if the number of loops is two, then the optimization is done through two stages (loops). In the first stage, the best subsection of the search space is found, then in the second stage, the algorithm proceeds searching for the best results within the determined subsection. This way, the algorithm would be much faster and less sensitive to the size of the initial search space defined as will be proved by various simulation results. The proposed method can be summarized as follows:

Step 1

Initialize the number of design variables (Npar) depending on the system under study, the number of divisions for each parameter (Ndiv) which, the number of iterations (Niter) and finally the number of loops (Nloop) which determines the number of subsections.

Step 2

Start the loop.

Step 3

Generate a Npar-dimensional space and then set a Npar-dimensional uniform CDPF matrix which has $(Ndiv)^{Npar}$ cells and each cell has its own CDPF which is initially the same for all cells, but it must be updated in each iteration. The calculation of CDPF is very important, as the best cell has the highest CDPF.

Step 4

Start the iterations for the current loop which of course depends on its number of iterations specified and the number of loops.

Step 5

Select at random a cell with respect to its CDPF from the cubic structure shown so that the selection probability is proportional to its CDPF. It must be noted that there are different ways for this selection such as roulette wheel selection method used in this study (Liu, 2008). This step is performed according to the following three sub-steps:

- Suppose a circle that has $(N_{div})^{Npar}$ sectors. Which each sector of the circle is respect to a cell. Angle of each sector calculates by:

$$\theta_i = 360 \times CDPF(i), \quad i = 1, 2, \ldots, (N_{div})^{Npar} \tag{13}$$

That CDPF (i) is the ith CDPF for: $i = 1,2,\ldots,(N_{div})^{Npar}$

- Now, select sector θ_{rand} at random (According to the CDPFs) within [0°, 360°]:

$$\theta rand = rand * 360 \tag{14}$$

- If θ_i, $\theta_{rand} \leq \theta_{i+1}$ for some i, then sector i in the original search space is selected which is recognized by its index, $(i_1, i_2, \ldots, i_{Npar})$, in the cubic structure.

Step 6

The selected index I s applied to Eq. 15 and N_{par} parameters (N_{par} actions) are obtained, the set is evaluated in the environment (control system) and the scalar cost $J_{cal}(k)$ is calculated.

$$P(i) = E(i) * l(i) + G(i), \quad i = 1, 2, \ldots, N_{par} \tag{15}$$

where, P(i), l(i) and G(i) are the value of the ith parameter or action, length of ith interval and beginning of ith interval, respectively.

Step 7

Evaluate the cost function according to the performance index of the system under control such as time and rate of error, control input, overshoot, steady-state error, etc. There are some coefficients which are the weighting elements. After the transient response, using a record of the system response, calculate the cost function ($J_{cal}(k)$) which is minimum for the best cell. The procedure compares the calculated cost function of the selected cell with the minimum of the cost function. It must be noted that in the first iteration, the minimum performance index is the same recently calculated performance index. The parameter $\beta(k)$ is updated according to the following rule in which k is the number of iteration:

$$\beta(k) = \begin{cases} 1 & \text{if} \quad Jcal < Jmin \\ 0 & \text{if} \quad Jcal > Jmin \end{cases} \tag{16}$$

If calculated cost function is improved ($J_{cal}(k)<J_{min}$), then go to step 8, else set the CDPF of the selected cell to zero and then go to step 9. This modification has improved the efficiency of the EDARLA.

Step 8

Take the calculated cost function ($J_{cal}(k)$) as the minimum cost function (J_{min}) , selected cell as the New-Group and update the CDPF matrix as described below:

$$\begin{aligned} &f_{X_1, X_2, \ldots, X_{Npar}}(x_1, x_2, \ldots, x_{Npar}; k+1) = f_{X_1, X_2, \ldots, X_{Npar}}(x_1, x_2, \ldots, x_{Npar}; k) + \beta(k) H(k) \\ &H(k) = 2 \frac{\| Indx\_Cell - New\_Group(k) \|^2}{N_{div}} \end{aligned} \tag{17}$$

where, Indx_Cell is the index of each cell and New_Group (k) is the current group's center. One can compare the procedure with the original DARLA (11). The same normalization task is performed similar to (12).

Step 9

If the iteration is finished, go to the step 2 for another loop and repeat the process to the Final-Group, else go to the step 5.

The flowchart of the proposed reinforcement learning method is given by Fig. 5.

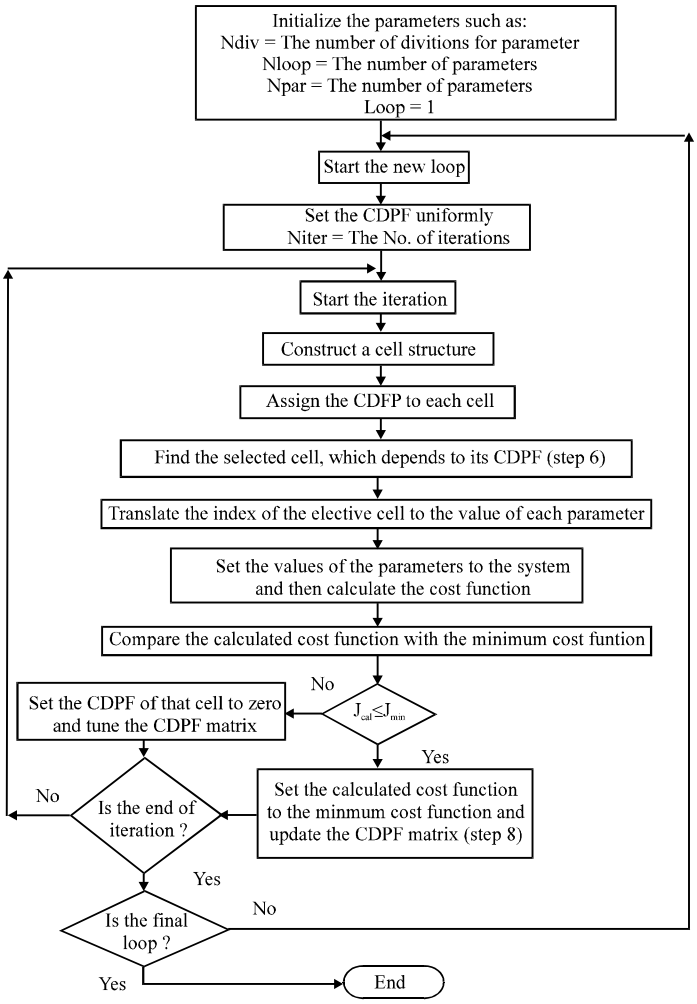

Fig. 5: The flowchart of the proposed reinforcement learning method (EDARLA)

THE PROPOSED OPTIMAL PID CONTROLLER FOR AVR SYSTEM

So, far, PID controllers have widely been used in process control. With simple structure, they yet can effectively control various large industrial processes. There are many tuning approaches for these controllers, but each has own disadvantages or limitations. As a result, the design of PID controllers still remains a remarkable challenge for researchers. In simple words, the PID controller is used to improve the dynamic response as well as reduce or eliminate the steady-state error. The derivative term normally adds a finite zero to the open loop plant transfer function and can improve the transient response in most cases.

The integral term adds a pole at origin resulting in increasing the system type and therefore reducing the steady-state error. Furthermore, this controller is often regarded as an almost robust controller. As a result, they may also control uncertain processes. The well-known PID controller transfer function is as follows:

$$C(s) = k_p + \frac{k_i}{s} + k_d s \tag{18}$$

One of the important approaches used to design and tune the PID controllers including those are being used in AVR systems is the well-known Ziegler and Nichols (ZN) approach (Guillermo *et al*., 2005). ZN is a well popular and interesting approach originated by Ziegler and Nichols in 1942 (Guillermo, 2005) and later extended in 1984 by Astrom and Hagglund (Astrom , 2006). In this paper the proposed EDARLA is used as an automatic technique for optimal designing the PID parameters for a practical high order AVR system. The design method is fast, robust and with adaptation ability. The application results and comparisons with DARLA and ZN methods are given in the next section.

AVR System Under Study

The responsibility of an AVR is to hold the terminal voltage of a synchronous generator at a specified level. Hence, the performance and stability of the AVR seriously affects the security of the power system. In this paper, the AVR system under study has been modeled based on IEEE standard 421.5 (Gaing, 2004). The model takes into account all the major time constants and saturation effect and other nonlinearities.

The transfer functions of the AVR components can be represented as follow (Gaing, 2004):

Amplifier Model

The amplifier model is represented by a gain K_A and a time constant τ_A, the transfer function is given by:

$$\frac{V_R(s)}{V_e(s)} = \frac{K_A}{1 + \tau_A s} \tag{19}$$

Typical value of K_A is in the range of 10 to 400. The amplifier time constant is very small ranging form 0.02 to 0.1sec.

Exciter Model

The transfer function of a modern exciter may be represented by a gain K_E and a single time constant τ_E as:

$$\frac{V_F(s)}{V_R(s)} = \frac{K_E}{1 + \tau_E s} \tag{20}$$

Typical value of K_E is in the range of 10 to 400. The time constant τ_E is in the range of 0.5 to 1.0 sec.

Generator Model

In the linearized model, the transfer function relating the generator terminal voltage to its field voltage can be represented by a gain K_G and a time constant τ_G as:

$$\frac{V_t(s)}{V_F(s)} = \frac{K_G}{1 + \tau_G s} \tag{21}$$

These constants are load dependent, K_G may vary between 0.7 to 1.0 and τ_G between 1.0 and 2.0 sec from full load to no load, respectively.

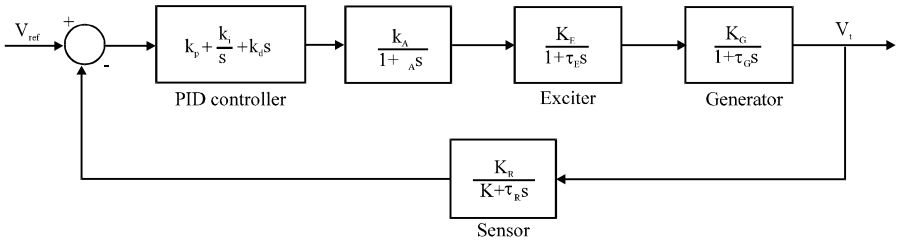

Fig. 6: Block diagram of the AVR System with PID controller

Sensor Model

The sensor is modeled by a simple first-order transfer function, given by:

$$\frac{V_S(s)}{V_t(s)} = \frac{K_R}{1+\tau_R s} \tag{22}$$

where, τ_R is very small, ranging from of 0.001 to 0.06 sec.

The block diagram of the AVR system with the PID controller is shown in Fig. 6. The Fig. 6 does not show the saturation effects, but they are fully considered in all design steps and simulations.

Performance Index

Generally, in many traditional optimal PID controller design approaches, some well-known performance indexes or performance criteria such as the integrated absolute error (IAE), the integral of squared-error (ISE), or the integrated of time-weighted- squared-error (ITSE) are widely used. Each of the three integral performance criteria has its own features. For example, a disadvantage of the IAE and ISE criteria is that its minimization can result in a response with relatively small overshoot but long settling time. That is because the ISE performance criterion equally weights the error over time. The ITSE criterion can overcome this disadvantage, but analytically derivation of the controller is rather complicated and time-consuming. The aforementioned classic IAE, ISE and ITSE performance criteria are shown as follow, where e(t) is the error between the desired and real output quantities:

$$IAE = \int_0^\infty |r(t) - y(t)|dt = \int_0^\infty |e(t)| \tag{23}$$

$$ISE = \int_0^\infty |e^2(t)|dt \tag{24}$$

$$ITSE = \int_0^\infty te^2(t)dt \tag{25}$$

Despite the existence of some classic performance indexes addressed, a more effective performance criterion in time domain is here suggested for designing the PID controller. The new performance criterion J is defined as follows and considers some broader requirements in more explicit manner:

$$\min_k J(K) = G_e \int_0^T e^2(t)dt + G_u \int_0^T u_c^2(t)dt + G_M M_P + G_s E_{ss} + G_d \sup\left|\frac{de(t)}{d(t)}\right| \tag{26}$$

where, T is the total simulation time and it is T = 20 sec for this study, e(t) is the tracking error , $u_c(t)$ is the control input , M_p is the amount of the overshoot, E_{SS} is the steady-state error at t = T and G-coefficients are the weighting elements ($G_e = 10$, $G_u = 1$,$G_M = 10$, $G_s = 10$, $G_d = 5$). The parameters of the PID controller are calculated based on the following approaches and the results are compared:

- Conventional DARLA
- Extended DARLA (EDARLA)
- Conventional Ziegler-Nichols
- Genetic Algorithm (GA)
- Conventional CARLA (Kashki *et al*., 2006)

SIMULATION RESULTS

Performance of the EDARLA

For the practical AVR system under study, the results show that the proposed method can perform an excellent search for the optimal PID controller parameters quickly. As will be shown through extensive simulation studies, one of drawbacks of the DARLA is that it is quite sensitive to the initial ranges considered for the design variables forming the overall search space. The proposed method is more efficient as well as more robust against the search space volume. For this comparative study, fortunately, the normal ranges of three controller parameters that is K_p, K_i and K_d considered in other reference (Gaing, 2004) is available. The AVR system parameters are show in Table 1 (Gaing, 2004). Some other symbols used in this study are shown in Table 2. Figure 7 shows the original step response of the AVR system without controller. In this case, $M_p\% = 50.51$, $E_{ss}\% = 9.06$, $T_r = 0.273$sec and $T_s = 5.565$ sec. As seen, the results are not satisfactory for a practical system.

Also, The PID controller parameters calculated by Ziegler-Nichols method are as follow:

$$K_p = 1.02 \quad K_i = 0.31 \quad K_d = 0.1847 \tag{27}$$

Here, the parameters of the PID controller are calculated by the proposed EDARLA. Considering the results already reported by Kashki *et al*. (2006), the ranges of the three parameters K_p, K_i and K_d

Table 1: The AVR system parameters

Description	Parameter	Value
Amplifier gain	K_A	10.00
Amplifier time constant	τ_A	0.10
Exciter gain	K_E	1.00
Exciter time gain	τ_E	0.40
Generator gain	K_G	1.00
Generator time constant	τ_G	1.00
Sensor gain	K_R	1.00
Sensor time constant	τ_R	0.01

Table 2: The used symbols

Variable name	Description
Loop-k	k-th of loop (k = 1,2,3)
F-Iter-k	No. of iteration in loop k (k = 1,2,3)
J_{min}	Minimum value of performance index
Ts	Settling time
Tr	Rise time
M_P (%)	Percent of overshoot
Ess (%)	Percent of steady state error

are taken as [0,1.5], [0,1] and [0 1], respectively. For calculation of the PID parameters, each interval is divided into 5 slots in the first, second and three loops. The simulation results of the EDARLA for different number of loops and different number of total iterations are summarized in Table 3. It can be seen that the final results are quite interesting and excellent results can be obtained through even less than 50 iterations. Moreover, comparing cases 2 and 3 (with the same number or iterations equal 20) reveals that having two loops rather than one loop can clearly result in better results as reflected in the final cost functions of the addressed case studies. Also, Fig. 8 and 9 show the convergence characteristics of the proposed method and terminal voltage step response of the AVR system respectively for different simulation conditions. As seen, through about 50 iterations, the EDARLA method successfully converges and provides good performance. The results prove that the proposed method (EDARLA) can search for optimal PID controller parameters quickly and efficiently.

Table 3: The Results of Simulation for proposed method (EDARLA)

Case	No. of iterations	Loop-1	Loop-2	Loop-3	Design time (sec)	J_{min}	M_P (%)	Tr (sec)	Ts (sec)	Ess (%)	K_p	K_i	K_d
1	10	10	0	0	0.272	17.446	19.45	0.509	2.850	0.00	0.600	0.600	0.200
2	20	20	0	0	0.561	14.390	7.44	0.396	1.135	0.07	0.900	0.400	0.200
3	20	10	10	0	0.535	13.730	0.25	1.146	1.691	0.08	0.744	0.160	0.344
4	50	20	15	15	1.386	13.370	0.00	0.518	0.962	0.00	1.044	0.240	0.296
5	100	50	25	25	2.856	10.020	0.00	0.854	1.036	0.00	0.636	0.176	0.224
6	150	50	50	50	4.320	8.980	0.00	0.692	1.087	0.00	0.574	0.176	0.144
7	200	100	50	50	5.467	8.420	0.00	0.433	0.673	0.00	0.948	0.256	0.232

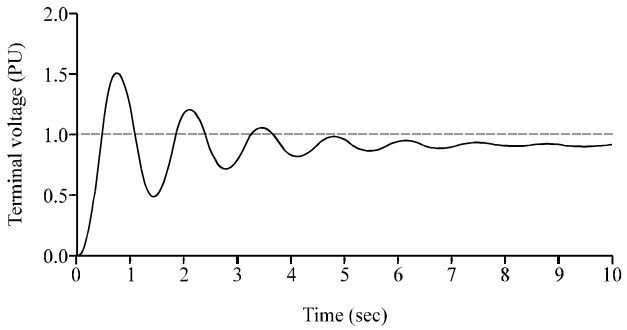

Fig. 7: Step response of the AVR system without controller

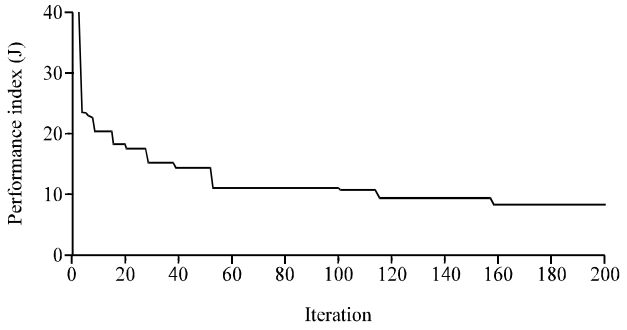

Fig. 8: Convergence tendency of the EDARLA (case 7)

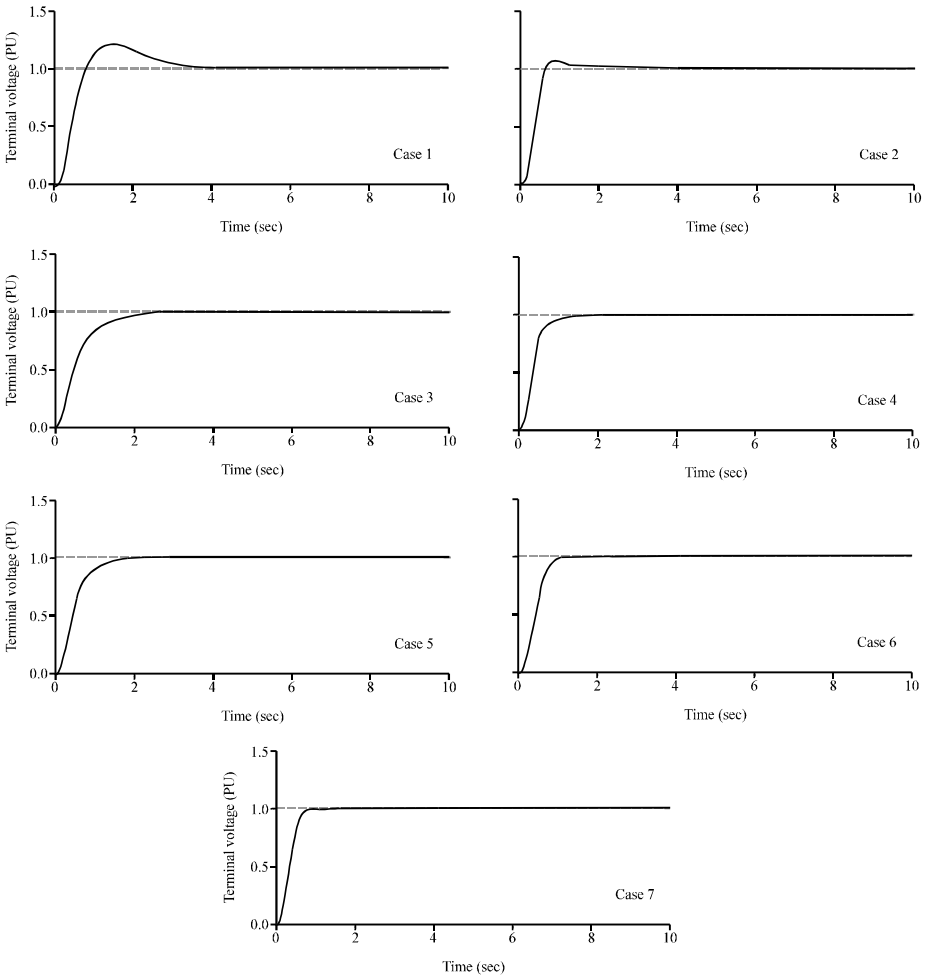

Fig. 9: Terminal voltage step response of the AVR system with optimal PID controller (EDARLA)

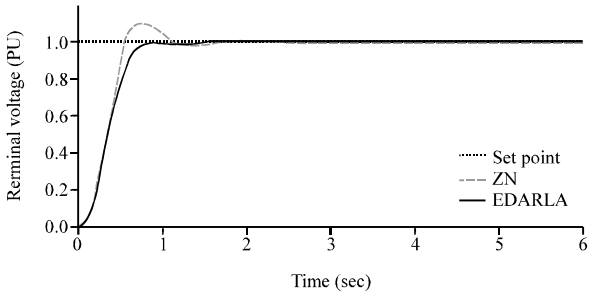

Fig. 10: Comparison of ZN and EDARLA methods

Comparison between the Proposed EDARLA, DARLA and ZN Methods

Figure 10-12 compare the results ZN, DARLA and EDARLA methods. The EDARLA has totally run for 50 iterations only while the DARLA has trained for 200 iterations and as seen

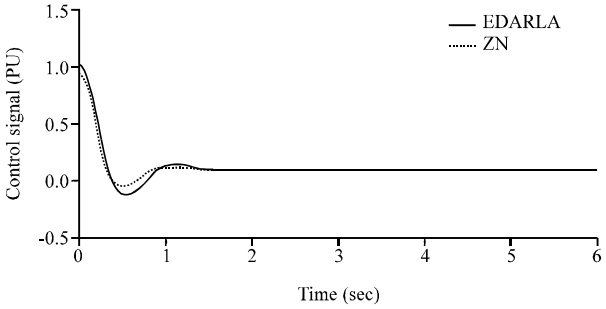

Fig. 11: The Control Signal of ZN and EDARLA methods

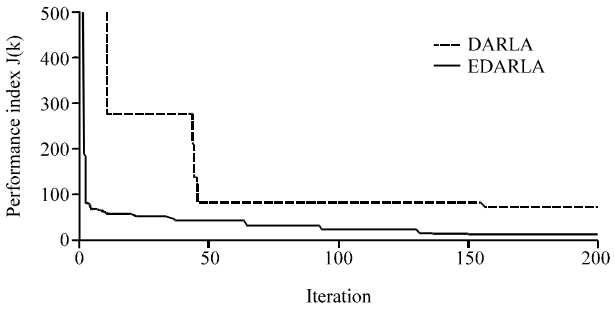

Fig. 12: Performance index of DARLA and EDARLA

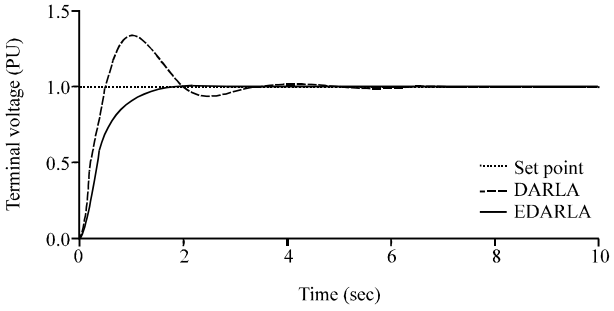

Fig. 13: The responses of DARLA and EDARLA methods

EDARLA provides much better results. Also, Fig. 13 shows the convergence curves and step response of the AVR for both DARLA and EDARLA methods after 200 iterations. Again, one can clearly see that the EDARLA has provided much better results with faster convergence.

Robustness to the Design Variables Ranges

As Already addressed, one of key issues in both DARLA and CARLA methods is the range of each design variable need to be pre-defined. This is not a trivial task, as the designer cannot always guess some appropriate ranges. If the ranges are too small, that makes the overall search space too small that the procedure may fail to find acceptable optimal points. On the other side, large ranges can make the algorithm too time consuming. Even worth, for a fixed number of slots (N_{div}), larger ranges

Table 4: Comparison of proposed method (EDARLA) and conventional DARLA run for 200 iterations

	Conventional DARLA ($\lambda = 1$)		EDARLA (proposed method)	
Interval of parameters	J_{min}	Simulation time (sec)	J_{min}	Simulation time (sec)
[0 1.5]	9.762	7.315	8.42	5.467
[0 5]	15.798	8.323	10.66	7.560
[0 10]	26.737	16.848	13.38	16.555
[0 15]	40.575	22.060	14.03	16.928
[0 20]	76.424	23.860	15.85	18.792

Table 5: Effect of the search space: A comparison between the proposed EDARLA and the conventional DARLA

	Conventional DARLA						EDARLA (Proposed Method)					
Interval of parameter	M_P (%)	Ts (sec)	Ess (%)	K_p	K_i	K_d	M_P (%)	Ts (sec)	Ess (%)	K_p	K_i	K_d
[0 1.5]	0.00	2.22	0.71	0.432	0.120	0.240	0.0	0.433	0.67	0.948	0.256	0.2320
[0 5]	4.21	0.77	0.18	1.151	0.427	0.339	0.0	1.330	0.00	0.812	0.184	0.2880
[0 10]	6.32	2.30	0.15	2.202	0.904	0.610	0.0	1.070	0.12	0.9331	0.137	0.5107
[0 15]	22.15	2.135	0.00	2.440	3.444	0.791	0.346	1.336	0.11	1.0607	0.232	0.4148
[0 20]	33.19	2.85	0.00	3.748	9.141	1.600	0.15	1.234	0.00	1.259	0.217	0.4640

may lead to even worth results as shown in Table 4 and 5. The tables show the comparative results of the conventional DARLA and the proposed EDARLA. Both methods have run for 200 iterations. As the results show, the DARLA fails to reach good results even for small ranges. Moreover, as the ranges increase, the overall performance index increases. The changes in the performance index, rise time, overshoot and overshoot is notably less for the EDARLA. This feature is quite important, because as already mentioned, in most cases, the best ranges for the search space cannot be determined exactly. Hence, a need for a heuristic approach such as EDARLA is apparent. The Results prove the good robustness of the EDARLA and explain how the conventional DARLA is too sensitive to the ranges taken for its variable. So, the most efficient, robust and optimal results can be gained by the proposed method (EDARLA). In the next subsections performance, efficiency and sensitivity of GA and CARLA approaches are also investigated.

Comparison between the Proposed EDARLA and GA Methods

In order to more highlight the advantages of the proposed method, we also implemented the GA (Genetic Algorithm) method (Gaing, 2004). The characteristics of the two controllers using the same performance index as defined by (22) are compared. The GA parameters are as follow:

Population size = 25
Crossover rate (Pc) = 0.75
Mutation rate (Pm) = 0.0075

Table 6 shows the GA approach results for different ranges of the parameters. As seen the GA does not provide good results even over 200 populations. It is also more sensitive to the variables ranges. Figure 14-16 compare the step responses of the AVR system when GA and EDARLA methods are used for training the PID controller for some different ranges of the parameters. As seen, EDARLA provides the best results in comparison with the results of the other methods discussed. Also, Fig. 17 compares the convergence curve of GA and EDARLA methods. It must be noted that while both methods are executed for 200 iterations, the EDARLA needs 200 fitness function evaluations, but the GA needs 200×25 function evaluations, because each population has 25 elements. That makes the GA too time consuming in comparison to DARLA and EDARLA.

Comparison Between the Proposed EDARLA and CARLA Methods

To give further insight to the horizon of the evolutionary optimization approaches used for design of controllers as whole and PID controllers of AVR systems in particular, the optimization

Table 6: Simulation results for designing PID controller by GA: sensitivity analysis

Case	Interval of parameter	Simulation time (sec)	J_{min}	M_P (%)	Tr (sec)	Ts (sec)	Ess (%)	K_p	K_i	K_d
1	[0 1.5]	128.96	10.110	5.780	0.962	3.21	0.00	0.3407	0.1917	0.1337
2	[0 5]	155.04	14.126	13.640	1.026	4.76	0.00	0.4904	0.3180	0.3047
3	[0 10]	346.10	25.613	15.970	0.753	4.19	0.03	3.6070	2.1950	2.0400
4	[0 15]	366.72	34.190	17.740	1.036	5.68	0.50	2.6490	0.9580	1.8070
5	[0 20]	365.57	47.990	36.568	1.742	12.24	0.00	1.5897	1.3058	3.4340

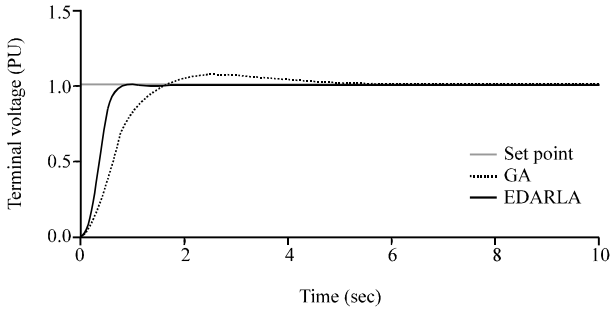

Fig. 14: The responses of GA and EDARLA methods (range of parameters : [0 1.5])

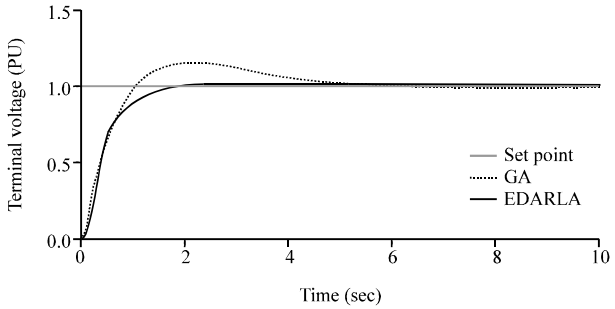

Fig. 15: The responses of GA and EDARLA methods (range of parameters : [0 10])

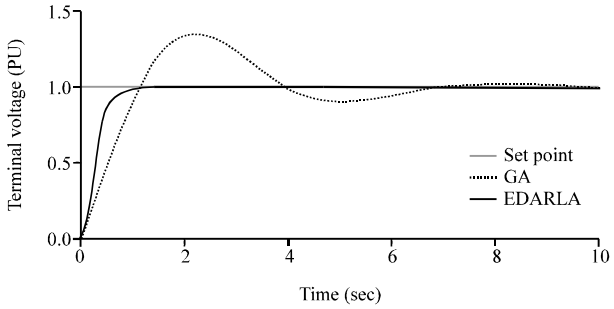

Fig. 16: The responses of GA and EDARLA methods (range of parameters : [0 20]) (Number of Iterations for EDARLA and Number of Generations for GA : 200)

performance of the classic CARLA (Kashki *et al*., 2006) is also evaluated. The results can be compared to the other methods already discussed including the EDARLA. Table 7 depicts the performance of

Table 7: Simulation results for designing PID controller by CARLA

Case	Interval of parameter	Simulation time (sec)	J_{min}	M_p%	Tr (sec)	Ts (sec)	Ess (%)	K_p	K_i	K_d
1	[0 1.5]	9.07	9.56	2.750	0.662	1.012	0.05	0.4540	0.2124	0.1220
2	[0 5]	10.19	11.84	0.000	1.728	3.223	1.36	0.4677	0.0875	0.2351
3	[0 10]	101.69	20.46	19.340	1.650	7.520	0.91	0.7552	0.3801	0.9902
4	[0 15]	122.63	32.17	22.471	1.275	5.956	1.40	0.7802	0.5401	0.8612
5	[0 20]	264.34	43.31	31.710	0.521	3.987	0.00	2.0880	3.5800	1.2300

$g_h = 0.75$, $g_w = 0.008$

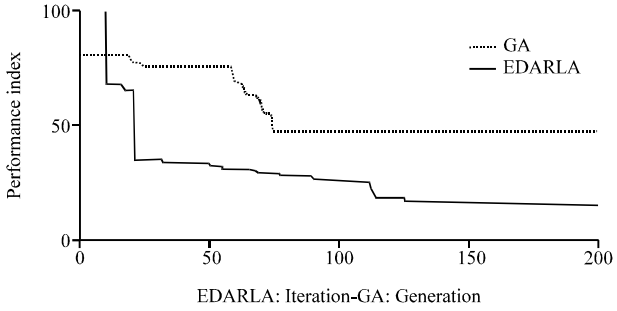

Fig. 17: Performance index of GA and EDARLA methods (range of parameters : [0 20]) (Number of Iterations for EDARLA and Number of Generations for GA : 200)

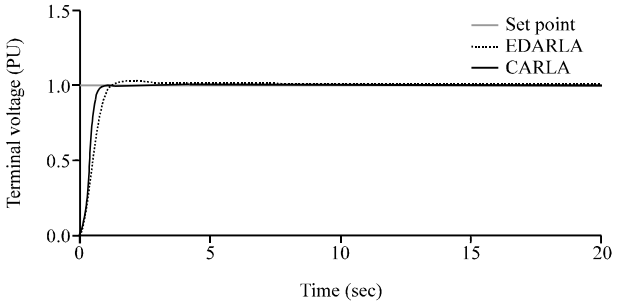

Fig. 18: The responses of CARLA and EDARLA (range of parameters : [0 1.5])

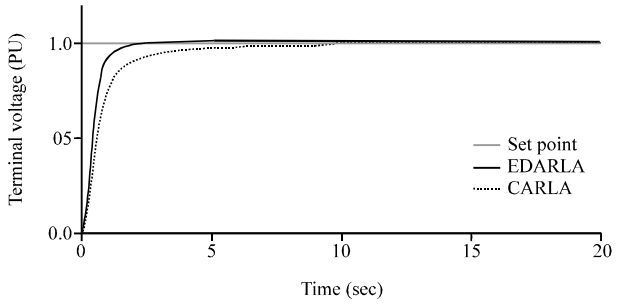

Fig. 19: The responses of CARLA and EDARLA (range of parameters : [0 5])

the CARDA run for 200 iterations, as seen, it provides good results when the search space is properly small (cases 1 and 2). It is however very sensitive to the ranges taken for the design parameters and provides poor results for large ranges. However, it provides better results than GA. It should also be noted that each iteration of the CARDA is a bit more time consuming than DARLA's. Figure 18-20

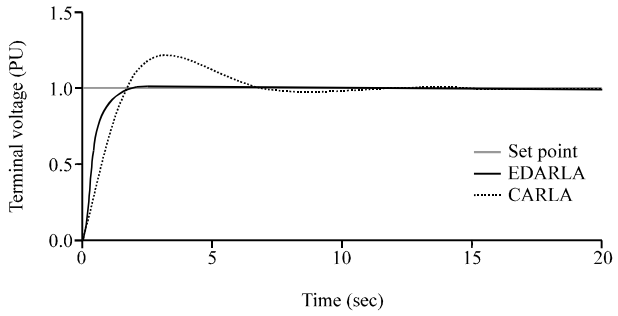

Fig. 20: The responses of CARLA and EDARLA (range of parameters : [0 15])

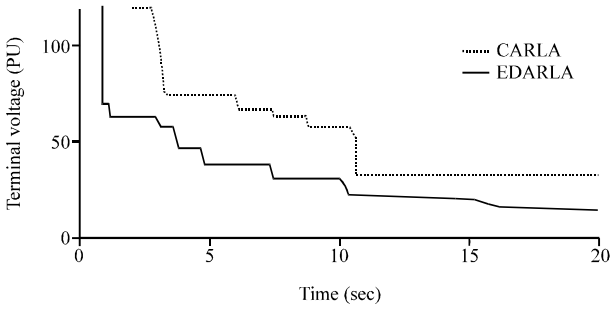

Fig. 21: Comparison of the performance index for CARLA and EDARLA range of parameters : [0 15])

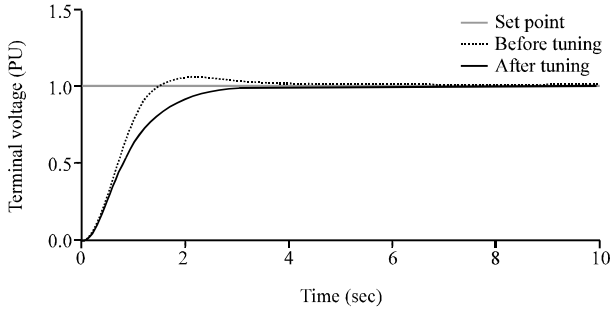

Fig. 22: Step response of the system before and after retuning by EDARLA

compare the step responses of the AVR system when CARLA and EDARLA methods are used. As seen, EDARLA provides the best results in comparison with the results of the other methods. Also, Fig. 21 compares the convergence curve of CARLA and EDARLA methods confirming the superior behavior of the proposed method.

Adaptive PID Controller Tuning Using EDARLA

Thank to the efficiency of the EDARLA, it is quite easy to quickly retune the controller parameters once a change occurs in the plant parameters. That change, of course, should be identified by a proper parameter identification method. The generator parameters are change as $K_G = 0.7$ and $T_G = 2$ which is a major change that is the generator is considered no-load with a high time-constant

significantly affects the overall transient response of the AVR system. The EDARLA algorithm has run for only 20 more iteration to retune the PID controller. The updated PID parameters are: $K_p = 0.9293$, $K_i = 0.1507$ and $K_d = 0.452$. Figure 22 shows the step response of the system before and after retuning and as seen the adaptation mechanism has provided much good results.

CONCLUSION

This study introduces a new intelligent method for optimizing the parameters of the PID controller for an AVR system. The proposed method is an extended version of DARLA method called EDARLA which optimizes the controller parameters while the variables are not considered independent opposed to the classic DARLA. By using matrix calculation, the speed of convergence can be increased and the system can be used for many real time applications. Superior performance, robustness and efficiency of the proposed method have been proved through extensive simulation results including comparisons with CARLA, GA, Ziegler-Nichols and DARLA methods. The extensive studies carried out clearly proves that the proposed approach is an excellent candidate for optimizing various control problems including adaptive control system thank to its high efficiency, speed and robustness.

ACKNOWLEDGMENT

The authors wish to thank the Fuzzy Systems and Applications Center of Excellence, Shahid Bahonar University of Kerman, Kerman, Iran.

REFERENCES

Astrom, K.J. and T. Hagglund, 2006. Advanced PID Control. illustrated Edn., ISA., USA., ISBN-10: 1556179421, pp: 461.

Charvillat, V. and R. Grigoras, 2007. Reinforcement learning for dynamic multimedia adaptation. J. Net. Comput. Appl., 30: 1034-1058.

Chou, C.H., 2006. Genetic algorithm-based optimal fuzzy controller design in the linguistic space. IEEE Trans. Fuzzy Syst., 14: 372-385.

Duan, Y., Q. Liu and X. Xu, 2007. Application of reinforcement learning in robot soccer. Eng. Appl. Artific. Intell., 20: 936-950.

Gaing, Z.L., 2004. A particle swarm optimization approach for optimum design of PID controller in AVR system. IEEE Trans. Energy Conver., 19: 384-391.

Guillermo, J.S., D. Aniruddha and S.P. Bhattacharyya, 2005. PID Controllers for Time-Delay Systems. 1st Edn., Birkhauser, Boston, ISBN: 0-8176-4266-8, pp: 330.

Ho, S.J., S. Li-Sun and H. Shinn-Ying, 2006. Optimizing fuzzy neural networks for tuning PID controllers using an orthogonal simulated annealing algorithm OSA. Fuzzy Syst. IEEE Trans., 14: 421-434.

Howell, M.N., G.P. Frost, T.J. Gordon and Q.H. Wu, 1997. Continuous action reinforcement learning applied to vehicle suspension control. Mechatronics, 7: 263-276.

Howell, M.N. and M.C. Best, 2000. On-line PID tuning for engine idle-speed control using continuous action reinforcement learning automata. Control Eng. Practice, 8: 147-154.

Howell, M.N. and T.J. Gordon, 2001. Continuous action reinforcement learning automata and their application to adaptive digital ?lter design. Eng. Appl. Arti?c. Intell., 14: 549-561.

Kamal, M.A.S. and J. Murata, 2008. Reinforcement learning for problems with symmetrical restricted states. Robotics Autonomous Syst., 56: 717-727.

Kashki, M., A.A. Gharaveisi and F. Kharaman, 2006. Application of cdcarla technique in designing takagi- sugeno fuzzy logic power system stabilizer (PSS), IEEE International Conference on Power and Energy, PECon '06, Nov. 28-29, IEEE Computer Society Press, pp: 280-285.

Kim, S.M. and W.Y. Han, 2006. Induction motor servo drive using robust PID-like neuro-fuzzy controller. Cont. Engin. Pract., 14: 481-487.

Lin, C.J. and Y.J. Xu, 2006. A novel genetic reinforcement learning for nonlinear fuzzy control problems. Neurocomputing, 69: 2078-2089.

Liu, B., 2008. Theory and Practice of Uncertain Programming. 3rd Edn., Uncertainty Theory Laboratory(UTLAB) , Department of Mathematical Sciences Tsinghua University Beijing, China.

Oh, S.Y., J.H. Lee and D.H. Choi, 2000. A new reinforcement learning vehicle control architecture for vision-based road following. IEEE Trans. Vehicular Technol., 49: 997-1005.

Seng, T.L., M.B. Khalid and R. Yusof, 1999. Tuning of a neuro-fuzzy controller by genetic algorithm. IEEE Trans. Syst., Man, Cybern. Part B, 29: 226-236.

Ying, H., 2000. Theory and application of a novel fuzzy PID controller using a simplified takagi-sugeno rule scheme. Inform. Sci., 123: 281-293.