An Efficient Memetic Algorithm for Parameter Tuning of PID Controller in AVR System

Ankush Mandal[1], Hamim Zafar[1], Pradipta Ghosh[1], Swagatam Das[1]
[1]Department of Electronics and Telecomm. Engg., Jadavpur University, Kolkata 700032, India
E-mails: ankushmandal19@gmail, hmm.zafar@gmail.com, pghosh1990@gmail.com, swagatamdas19@yahoo.co.in

Ajith Abraham[2,3]
[2]Machine Intelligence Research Labs (MIR Labs)
Scientific Network for Innovation and Research Excellence, WA, USA
[3]IT For Innovations, EU Center of Excellence, VSB - Technical University of Ostrava, Czech Republic
E-mail: ajith.abraham@ieee.org

***Abstract*—In recent years, there has been a growing interest in real world application of heuristic methods. Memetic Algorithm (MA) is one of such effective heuristics. In this paper, we represent an efficient MA for determining optimal proportional-integral-derivative (PID) controller parameters of an AVR system. This MA is developed by combining a competitive variant of Deferential Evolution (DE) and a Local Search method. The proposed method has excellent features, such as easy implementation, stable convergence characteristic and good computational efficiency. Fast tuning of PID controller parameters results in far better performance of the controller. Performance of our proposed algorithm is compared with other famous heuristics and the simulation results clearly indicate that our proposed approach is indeed more efficient and robust in improving the step response of an AVR system.**

Keywords-Memetic Algorithm; Differential Evolution; Local Search; AVR System; PID controller

I. INTRODUCTION

The proportional-integral-derivative (PID) controller is the most popular controller, which has been widely used in the industry because of its simple structure, easy implementation and robust performance in a wide range of operating conditions. However, proper tuning of the parameters of PID controllers has been quite difficult because many industrial plants are often burdened with problems such as high order, time delays, and nonlinearities [1]-[3]. In recent years, many heuristic methods have been approached for the tuning of PID controllers. The first method used the classical tuning rules proposed by Ziegler and Nichols. But in reality, it is often hard to optimize PID parameters with Ziegler-Nichols method in many industrial plants [1]-[3].

So, it is reasonable to add new features to PID controllers to increase its efficiency. While retaining the basic characteristics, several artificial intelligence (AI) techniques, such as neural network, fuzzy system, and neural-fuzzy logic, have been employed to improve the controller performances [1], [2].

Differential Evolution (DE) algorithm [4] belongs to Evolutionary Algorithms (EAs) and now-a-days it is considered one of the most powerful tools in EAs. It can be interpreted as discrete dynamical system governing the movements of a set of vectors in the search space. Behavior of the individuals greatly influence the progress of the search process and therefore on the convergence of the algorithm. However, EAs are good for global optimization where the exploration of the entire search space is required within relatively small no of iteration. But they are not good for producing precise solutions.

Local Search (LS) algorithms [5] are used to explore the nearby regions around the current solutions with high intensity. So, by using a LS method, highly accurate solutions can be obtained. EAs hybridized with LS method are commonly called Memetic Algorithms (MAs) and these MAs [6], [7] are proven to be more efficient than the EAs themselves. The reason behind this is the combination of global exploration and local exploitation.

In this paper, we propose a MA, denoted by cDESW, to find optimal PID controller parameters. In our proposed algorithm, a competitive variant of DE is used for global exploration and the classic Solis Wet's algorithm [5] is used as the local search process. For the DE algorithm, we have developed a hybrid mutation strategy by hybridizing a modified "DE/current-to-best/2" mutation strategy with a modified "DE/rand/1"mutation strategy. We have discussed the mutation strategy later in sufficient details.

In practice, the generator excitation system maintains generator voltage and controls the reactive power flow using an automatic voltage regulator (AVR) [8]. Here, the role of an AVR system is to hold the terminal voltage magnitude of a synchronous generator at a specified level. Hence, the stability of the AVR system is of great concern because it can seriously affect the security of the power system. In this article, a practical high-order AVR system with a PID controller is adopted to evaluate the performance of the proposed cDESW-PID controller.

The rest of this paper is organized as follows: section II describes classical DE, linearized model of an AVR system with PID controller and the objective function. A detailed description of the proposed cDESW algorithm is given in section III. Section IV and V respectively describes the procedure of implementation of cDESW-PID controller and dynamic behavior estimation of the system. In section VI, experimental results are represented and analyzed thoroughly. Finally, section VII concludes the paper.

II. RELATED WORKS

A. *Linearized Model of an AVR System with PID controller*

In this section, we have represented a linearized model of a higher order AVR system compensated with a PID controller.

978-1-4577-2150-2

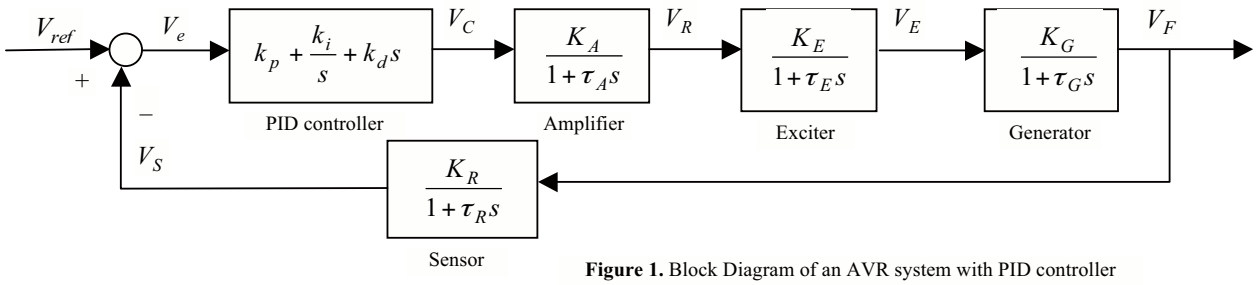

Figure 1. Block Diagram of an AVR system with PID controller

The block diagram of the whole system is given in Figure1.

1) PID Controller: The PID controller improves the dynamic response of the system and also reduce the steady-state error by rearranging the poles and zeros of the closed loop transfer function. The derivative controller adds a finite zero to the open-loop plant transfer function and improves the transient response. The integral controller adds a pole at the origin, thus increasing system type by one and reducing the steady-state error due to a step function to zero. The PID controller transfer function is as follows:

$$C(s) = k_p + \frac{k_i}{s} + k_d s \tag{1}$$

2) Linearized Model of an AVR System: An AVR is used to hold the terminal voltage magnitude of a synchronous generator at a specified level. A simple AVR system comprises four main components, namely amplifier, exciter, generator, and sensor. For mathematical modeling and determining transfer functions of the four components, these components must be linearized, which takes into account the major time constant and ignores the saturation or other nonlinearities.

The approximate transfer functions of these components may be represented, respectively, as follows [9].

a) Amplifier model: The amplifier model is represented by a gain K_A and a time constant τ_A. The transfer function is as follows:

$$\frac{V_R(s)}{V_C(s)} = \frac{K_A}{1+\tau_A s} \tag{2}$$

Where the value of K_A is in the range 10 to 400 and the value of the amplifier time constant τ_A is in the range 0.02s to 0.1s. In our simulation, K_A was set to 10 and τ_A was set to 0.1s.

b) Exciter model: After linearization, the tranfer function of a modern exciter can be represented by a gain K_E and a single time constant τ_E.

$$\frac{V_E(s)}{V_R(s)} = \frac{K_E}{1+\tau_E s} \tag{3}$$

Where the value of K_E is in the range 1 to 200 and the value of the amplifier time constant τ_E is in the range 0.5s to 1s. In our simulation, K_E was set to 1 and τ_E was set to 0.4s.

c) Generator model: In the linearized model of AVR system, 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.

$$\frac{V_F(s)}{V_E(s)} = \frac{K_G}{1+\tau_G s} \tag{4}$$

Here the constants depend on the load, the value of K_G varies between 0.7 to 1, and generator time constant τ_G in the range 1s to 2s from full load to no load. In our simulation, K_G was set to 1 and τ_G was set to 1s.

d) Sensor model: The sensor is modeled by a simple first order transfer function, given by

$$\frac{V_S(s)}{V_F(s)} = \frac{K_R}{1+\tau_R s} \tag{5}$$

Where the gain K_R is usually kept 1 and the time constant τ_R is very small, ranging from 0.01s to 0.06s. In our simulation, K_R was set to 1 and τ_R was set to 0.01s.

B. Performance Estimation of PID controller

In general, the PID controller design method using the integrated absolute error (IAE), or the integral of squared-error (ISE), or the integrated of time-weighted-squared-error (ITSE) is often applied in PID controller design because it can be evaluated analytically in the frequency domain [3]. These three integral performance criteria in the frequency domain have their own advantages and disadvantages. For example, a disadvantage of the IAE and ISE criteria is that by minimizing it, we will obtain a response with relatively small overshoot but a long settling time because the ISE performance criterion weights all errors equally independent of time. Although the ITSE performance criterion can overcome the disadvantage of the ISE criterion, the derivation processes are complex and time-consuming [6]. The IAE, ISE, and ITSE performance criterion formulas are as follows:

$$IAE = \int_o^\infty |e(t)|\, dt \tag{6}$$

$$ISE = \int_0^\infty e^2(t)dt \quad (7)$$

$$ITSE = \int_0^\infty te^2(t)dt \quad (8)$$

In this paper, we have developed a new performance criterion in the time domain for evaluating the performance of PID controller. Suitable values of control parameters k_p, k_i and k_d can yield a good step response and thus it will result in performance criteria minimization in the time domain. These performance criteria in the time domain include the overshoot M_p, rise time t_r, settling time t_s and steady-state error E_{ss}. The new performance criterion is defined as follows:

$$\min f(\vec{K}) = e^{-\alpha} \cdot (t_s + t_r) + (1 - e^{-\alpha}) \cdot (M_P + E_{ss}) \quad \text{where } \vec{K} \text{ satisfies the } \textbf{\textit{Routh-Hurwitz}} \text{ criterion.} \quad (9)$$

Here $\vec{K}$ is a row vector containing the control parameters, i.e. $\vec{K} = [k_p \quad k_i \quad k_d]$ and α is a weighting factor.

The performance criterion $f(\vec{K})$ is defined such a way that the designer can tune it according to system requirements just by changing the value of α. If α is smaller than 0.7 then rise time and settling time is reduced through the minimization process. On the other hand, we can set α to larger than 0.7 to reduce the overshoot and steady state error.

III. PROPOSED ALGORITHM

In this section we have discussed the proposed cDESW algorithm in sufficient details. In this multi-population based algorithm, we have developed a competitive variant of DE which is accompanied by a local search method. Furthermore, this algorithm employs a hybrid mutation strategy for DE to enhance the searching ability and to circumvent stagnation of the population at any local optima.

A. The modified DE algorithm

1) Competitive variant of DE: For a heuristic search process, it is useful to exploit the neighborhood because it is similar to information exchange between the neighbors which leads to better solutions. So, here we have incorporated a competition between the neighbors. Also the success rate is measured at each generation which helps in determining the new individual generation process for the next generation. Actually, depending on the success rate, either the current individual or its nearest neighbor is used for mutant vector formation. If the corresponding trial vector is chosen for next generation then the corresponding success rate is increased by 1 and if it is not chosen then the corresponding success rate is decreased by 1. If both the success rates for current individual and its nearest neighbor are equal then the current individual is used. At the time of initialization, all the success rates were set to 0. Using this method, we can get far better solutions with less function evaluations. Also the population does not converge to any local minima too quickly because we set the competition with nearest neighbor. Here, the nearest neighbor is selected on the basis of Euclidean distance between the current individual and the other individuals in the corresponding subpopulation.

2) Hybrid mutation strategy: As mentioned earlier, depending on the success rate, either the current individual or the nearest neighbor of the current individual is used for the mutant vector generation process. Let the chosen individual be $\vec{X}_{c,G}$.

In DE, greedy strategies like DE/current-to-best/*n* and DE/best/*n* benefit from their fast convergence property by guiding the search process with the best solution so far discovered, thereby converging faster to that point. But, as a result of such fast convergence tendency, in many cases, the population may lose its diversity and global exploration abilities within a relatively small number of generations, thereafter getting trapped to some locally optimal point in the search space. Taking into consideration these facts and to overcome the limitations of fast but less reliable convergence, we have developed a hybrid mutation strategy.

For constructing the final mutant vector, two mutant vectors generated by two different mutation strategies are combined with a weight factor ω. This way we developed a hybrid mutation strategy to prevent the algorithm from converging too quickly and at the same time exploring the whole search space to produce high quality results.

For the first mutation strategy, we have used a modified "DE/current-to-best/2" mutation strategy. For this modified mutation strategy, best individual of each subpopulation is stored in a memory archive; this memory archive is updated at each generation to store the new best individuals and delete the previous best individuals. During the mutation process, the nearest memory individual is used instead of the global best individual. The mutation process can be expressed as follows:

$$\vec{V}_{mut,1} = \vec{X}_{c,G} + F_{best} \cdot (\vec{X}_{m,G} - \vec{X}_{c,G}) + F \cdot (\vec{X}_{r1,G} - \vec{X}_{r2,G}) \quad (10)$$

where $\vec{X}_{m,G}$ is the nearest best individual as mentioned above. $\vec{X}_{r1,G}$ and $\vec{X}_{r2,G}$ are two distinct vectors randomly chosen from the subpopulation.

For the second mutation strategy, we have used "DE/current/1" mutation strategy. The mutation process can be expressed as follows: $\vec{V}_{mut,2} = \vec{X}_{c,G} + F \cdot (\vec{X}_{r1,G} - \vec{X}_{r2,G})$ (11)

where $\vec{X}_{r1,G}$ and $\vec{X}_{r2,G}$ are two distinct vectors randomly chosen from the subpopulation independently of first mutation process.

Now, the final mutant vector is a weighted sum of two above mentioned mutant vectors. If the weight factor for $\vec{V}_{mut,1}$ is ω then the final mutant vector is

$$\vec{V}_{mut} = \omega \cdot \vec{V}_{mut,1} + (1 - \omega) \cdot \vec{V}_{mut,2} \quad (12)$$

B. Local Search

1) Local Search method: As mentioned earlier, we have used Solis and Wet's algorithm as the LS method. The algorithm is a randomized hill climber. It starts from a initial point *i*. Then a deviation d is chosen from a normal distribution whose mean is 0 and variance is a parameter σ which is adaptive in nature. If *i+d* is better or *i-d* is better than the current individual in terms of fitness value then the current

individual is replaced by the better one and a success is recorded. Otherwise a failure is recorded. After some consecutive success (we set this value to 1) σ is increased (we set this increment as 1.5 times) to progress quicker. Similarly, after some consecutive failure (we set this value to 1) σ is decreased (we set this decrement as 1/3 times) to focus the search. Also a bias term is included to guide the search in the direction which gives success. As σ is adaptive, the step size of the search method is also adaptive which makes the algorithm very effective in exploiting the nearby region of the solution.

In our algorithm, the solutions got after executing the local search are also recorded. If the success exceeds the failure by at least *S* (we set it to 20) then the solution is recorded as a good solution and the corresponding σ value is also recoded. If the success does not exceed the failure by at least *S* then the solution is recorded as bad solution. When the local search algorithm runs again, the chosen individuals are compared with the previous good and bad solutions. If the current individual is a previous good solution then the search process starts with the previous value of σ. If the current individual is a previous bad solution then local search is not applied to the individual. We have incorporated this memory system to avoid unnecessary function evaluations which in turn increases the efficiency of the algorithm.

In our algorithm, during the local search process, the local search method is applied over the best individuals of all subpopulations.

IV. Implementation of a cDESW-PID Controller

We have developed a PID controller using the cDESW algorithm to improve the step transient response of AVR of a generator. The cDESW algorithm is mainly utilized to determine three optimal values of controller parameters k_p, k_i and k_d such that the control system obtains a good step response.

A. Defining Individuals

To apply the cDESW algorithm for searching the optimum values of controller parameters, we define each individual as a 3 dimensional vector $\vec{K}$ such that $\vec{K} = [k_p \quad k_i \quad k_d]$, i.e. each component of the vector is a controller parameter. We initialized the individuals by randomly choosing values from a uniform distribution having lower bound and upper bound of the values of the parameters. The range of the parameter values is given below.

TABLE I. Range of Three Controller Parameters

Controller Parameters	Min. Value	Max.Value
k_p	0	1.5
k_i	0	1
k_d	0	1

B. Individual Evalutation Process

We defined the evaluation function in eqn. (14) such a way that the problem becomes a minimization problem. In order to take system stability under consideration, we first evaluated the system stability for every set of parameter values by ***Routh-Hurwitz*** criterion. If a individual passes the ***Routh-Hurwitz*** criterion then we apply the evaluation function to get the fitness value of the individual and if the individual does not pass the ***Routh-Hurwitz*** criterion then the fitness value of the individual is penalized with a very large positive constant.

C. cDESW-PID controller

In this article, we represent a cDESW-PID controller for searching the optimal or near optimal controller parameters k_p, k_i and k_d with cDESW algorithm. The details of the parameter values and settings are given below.

1) Parameter values and Setting Used in Experiment

a) Population size: NP was kept fixed at 60 throughout the search process. We divided the whole population into 6 subpopulations, each containing 10 individuals.

b) Scaling factor: In this algorithm, scaling factor for each dimension of the difference vector is generated randomly depending on the value of the difference vector along the corresponding dimension. F is generated independently for each dimension of the difference vector. Scaling factor generation can be explained as follows:

$$F_j^d = rand(0,1) \cdot e^{-\left|x_j^d\right| / \left|x_R^d\right|} \tag{13}$$

where $d \in$[1, D(Dimension of the search space)]. We are generating scaling factor for the d th dimension of the j th individual. x_j^d is the value of the difference vector along d th dimension, x_R^d is the search range along that dimension.

c) Weight factor for hybrid mutation strategy: Weight factor ω for the first mutation scheme in hybrid mutation strategy was set to 0.7.

*d) Variance(σ):*Variance (σ) of the normal distribution from which the deviation is randomly chosen for the 1st stage of Local Search was updated at each generation as follows:

$$\sigma_d = 0.02 * (1 - (0.2 * (Gen / Max\_Gen))) * x_R^d \tag{14}$$

where σ_d is the value of variance for dimension *d*. *Gen* is the number of generations or iterations. *Max_Gen* is the number of maximum generations (here it is 150).

V. Experimental Results

A. Performance of AVR system without PID controller

AVR system shows a very poor performance without a PID controller. Figure 2 shows the terminal voltage step response of the AVR system without a PID controller. We found that M_p = 65.71%, E_{ss}=0.0909, t_r = 2608s, t_s = 6.99s.

B. Performance of cDESW-PID controller

In this section, we represent the performance of our proposed cDESW-PID controller in terms of overshoot M_p, steady state error E_{ss}, rise time t_r, settling time t_s. The performance was recorded after 50, 100 and 150 generations or iterations. In order to emphasize the advantage of the proposed cDESW method, we have also compared the results with other 2 state-of-the-art EAs, namely CLPSO [10] and

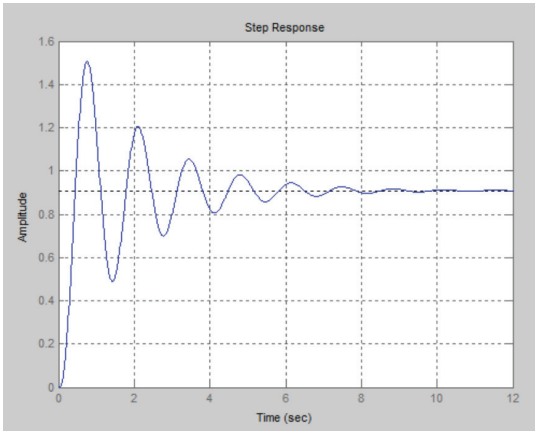

Figure 2. Terminal voltage step response of an AVR system without a PID controller

jDE [11]. An identical experimental environment was maintained for all the algorithms compared. We have executed the simulation 20 times for each value of α, i.e. for $\alpha = 1$ and $\alpha = 1.5$. The best results obtained over the independent 20 runs are summarized in Table II. A close inspection of Table II clearly reveals that our proposed cDESW-PID controller outperforms other CLPSO-PID and jDE-PID controller by producing far better step response characteristics.

1) Terminal Voltage Step Response: As mentioned earlier, the terminal voltage step response results are represented in Table 2. However, we have also given the step response curves of the AVR system with cDESW-PID controller in Figure 3 and Figure 4. These step response curves were obtained after 150 iterations; one corresponds to $\alpha = 1$ and the other corresponds to $\alpha = 1.5$. The step response curves clearly indicate that the system step response is greatly improved by introducing cDESW-PID controller to the AVR system.

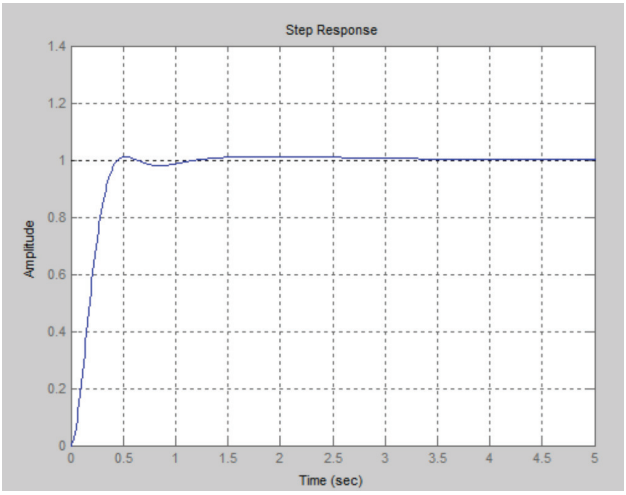

Figure 3. Terminal voltage step response of an AVR system with cDESW-PID controller ($\alpha = 1$)

2) Convergence Characteristics: The convergence characteristics graph is given in Figure 5. As we can see from the convergence characteristics of best evaluation value, cDESW algorithm reaches to a satisfactory result within 50 iterations. Also the smooth convergence characteristic of the algorithm indicates that the proposed cDESW-PID controller is able to produce very perfect step response of the AVR system without much fluctuation.

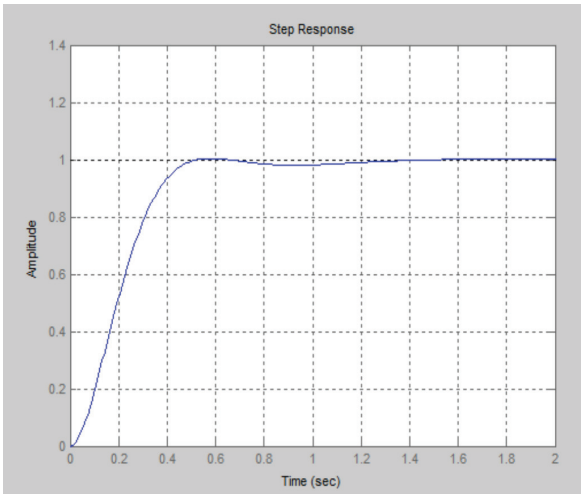

Figure 4. Terminal voltage step response of an AVR system with cDESW-PID controller ($\alpha = 1.5$)

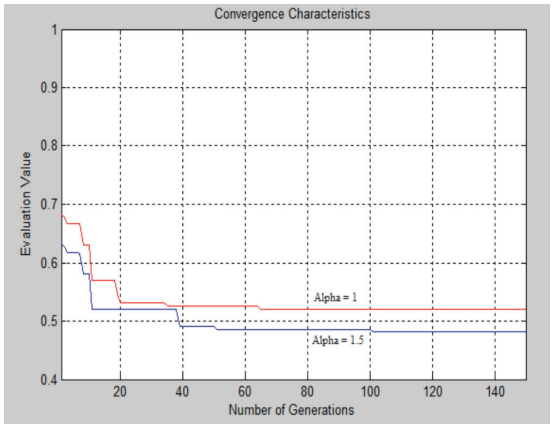

Figure 5. Convergence Characteristics of cDESW-PID controller (In the figure, Alpha means α)

3) Dynamic Behavior: During the simulation process, after each iteration, the mean value (Mean) and the standard deviation (STD) of the evaluation values of all individuals in the population were recorded for observing the dynamic convergence behavior of the individuals in population.

Figure 6 displays the variation of Mean and STD with iterations. As evident from the Figure 6 that though the mean value converges almost smoothly for all the algorithms, jDE and CLPSO bring premature convergence. This indicates that the proposed cDESW algorithm is able to produce high quality solutions by preventing the premature convergence. Also the convergence tendency of STD is much faster for cDESW than for jDE or CLPSO. This proves that cDESW has better convergence efficiency.

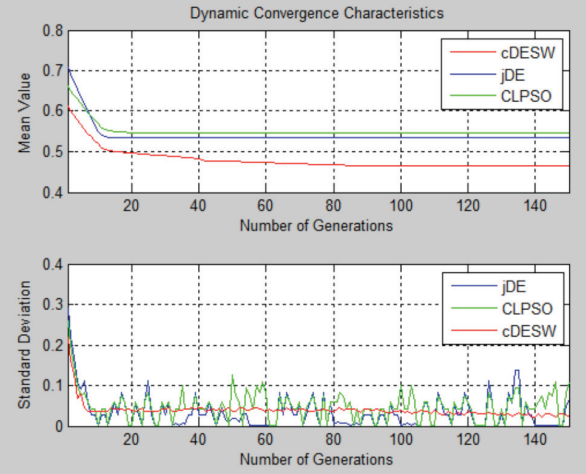

Figure 6. Convergence characteristics of Mean and STD of evaluation values in different algorithms (here α was taken as 1.5)

VI. CONCLUSION

This article represents an efficient design method for determining the PID controller parameters using the proposed cDESW algorithm. The simulation results obtained over a practical AVR system clearly indicates that the proposed method can perform an efficient search for optimal value of the controller parameters.

In order to verify the superiority of our proposed method, many performance estimation schemes have been approached, such as

- Step response characteristics.
- Convergence characteristics of best evaluation value.
- Dynamic convergence behavior of all the individuals.

It is clear from the results that cDESW algorithm can easily obtain higher quality solutions and it performs much better than jDE or CLPSO. So, the proposed cDESW-PID controller has more robust stability and efficiency in tuning PID controller parameters.

TABLE II. COMPARISON BETWEEN CDESW-PID, JDE-PID AND CLPSO-PID CONTROLLR

α	Number of generations	Type of controller	k_p	k_i	k_d	M_p (%)	E_{ss}	t_s (s)	t_r (s)
1	50	**cDESW-PID**	**0.6821**	**0.6288**	**0.2710**	**1.1602**	**0**	**0.3741**	**0.2504**
		jDE-PID	0.6861	0.5984	0.2891	1.2082	0	0.3951	0.2752
		CLPSO-PID	0.6568	0.5393	0.2458	1.1738	0	0.4027	0.2768
1	100	**cDESW-PID**	**0.6820**	**0.6287**	**0.2711**	**1.1688**	**0**	**0.3740**	**0.2503**
		jDE-PID	0.6722	0.6201	0.2996	1.1967	0	0.3960	0.2819
		CLPSO-PID	0.6571	0.5980	0.2630	1.7152	0	0.3795	0.2648
1	150	**cDESW-PID**	**0.6822**	**0.6297**	**0.2714**	**1.1570**	**0**	**0.3736**	**0.2601**
		jDE-PID	0.6663	0.5531	0.2365	1.1703	0	0.4076	0.2803
		CLPSO-PID	0.6570	0.5389	0.2458	1.1619	0	0.4025	0.2767
1.5	50	**cDESW-PID**	**0.6292**	**0.4680**	**0.2223**	**0.5013**	**0**	**0.4439**	**0.2864**
		jDE-PID	0.6301	0.4673	0.2239	0.6219	0	0.4450	0.2943
		CLPSO-PID	0.6271	0.4652	0.2209	0.5573	0	0.4489	0.3025
1.5	100	**cDESW-PID**	**0.6287**	**0.4678**	**0.2212**	**0.7714**	**0**	**0.4113**	**0.2966**
		jDE-PID	0.6266	0.4731	0.2219	0.6932	0	0.4427	0.3019
		CLPSO-PID	0.6271	0.4652	0.2209	0.9206	0	0.4168	0.3025
1.5	150	**cDESW-PID**	**0.6214**	**0.4534**	**0.2160**	**0.4021**	**0**	**0.4426**	**0.3013**
		jDE-PID	0.6467	0.4916	0.2375	0.4248	0	0.4492	0.3065
		CLPSO-PID	0.6254	0.4577	0.2187	0.4386	0	0.4528	0.3070

REFERENCES

[1] A. Visioli, "Tuning of PID controllers with fuzzy logic," Proc. Inst. Elect. Eng. Contr. Theory Applicat., vol. 148, no. 1, pp. 1-8, Jan. 2001.

[2] T. L. Seng, M. B. Khalid, and R. Yusof, "Tuning of a neuro-fuzzy controller by genetic algorithm," IEEE Trans. Syst., Man, Cybern. B, vol.29, pp. 226-236, Apr. 1999.

[3] R. A. Krohling and J. P. Rey, "Design of optimal disturbance rejection PID controllers using genetic algorithm," IEEE Trans. Evol. Comput.,vol. 5, pp. 78-82, Feb. 2001.

[4] R. Storn and K. Price, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization,* vol. 11, 341-359, 1997.

[5] F. J. Solis and R. J. Wets, "Minimization by random search techniques", mathematical Operations Research, 6:19–30, 1981.

[6] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization", in IEEE Congree on Evolutionary Computation, 2010.

[7] N. Shahidi, H. Esmaeilzadeh, M. Abdollahi, E. Ebrahimi, and C. Lucas, "Self-adaptive memetic algorithm: an adaptive conjugate gradient approach", IEEE Conference on Cybernetics and Intelligent Systems, 2004, 6 – 11.

[8] H. Saadat, *Power System Analysis*. New York: McGraw-Hill, 1999.

[9] H. Yoshida, K. Kawata, and Y. Fukuyama, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," IEEE Trans. Power Syst., vol. 15, pp. 1232–1239, Nov. 2000.

[10] J.J. Liang, A.K. Qin, P.N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions", IEEE Trans. On Evol. Compt., vol. 10, 281-295, 2006.

[11] J. Brest, V. Zumer, and M.S. Maucec, "Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization", IEEE Cong. On Evol. Compt., 2006.