An Adaptive Particle Swarm Optimization Applied to Optimum Controller Design for AVR Power Systems

M. Pourmahmood Aghababa
University of Tabriz
Control Engineering Department, Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran

A.M. Shotorbani
Azerbaijan University of Tarbiat Moallem
Department of Electrical Engineering, Azerbaijan University,35 km of Tabriz-Maraghe Rd. E.Azerbaijan, Iran

R. M. Shotorbani
University of Tabriz
Department of Mechanical Engineering, University of Tabriz, Tabriz, Iran

ABSTRACT
This paper describes an improved version of particle swarm optimization (PSO) method, called adaptive particle swarm optimization (APSO), for solving engineering optimization problems especially in power system fields. This algorithm uses a novel PSO algorithm to increase convergence rate and avoid being trapped in local optimum. The APSO algorithm efficiency is verified using some benchmark functions. Numerical simulation results demonstrate that the APSO is fast and has much less computational cost. Then, the proposed APSO method is used for determining the parameters of the optimal proportional-integral-derivative (PID) controller for an AVR power system. The proposed approach has superior features including easy implementation, stable and fast convergence characteristics and good computational efficiency. Also, the proposed method is indeed more efficient and robust in improving the step response of the AVR system**.**

Keywords
Particle Swarm Optimization, Fast Convergence, Local Optimum, PID Controller, AVR Power System.

1. INTRODUCTION
The objective of optimization is to seek values for a set of parameters that maximize or minimize objective functions subject to certain constraints. In recent years, many optimization algorithms are introduced. Traditional optimization algorithms use exact methods to find the best solution. The idea is that if a problem can be solved, then the algorithm should find the global best solution. As the search space increases the cost of these algorithms increases. Therefore, when the search space complexity increases the exact algorithms can be slow to find global optimum. Linear and nonlinear programming, brute force or exhaustive search and divide and conquer methods are some of the exact optimization methods.

Calculus provides the tools and elegance for finding the optimum of many objective functions. It quickly finds a single optimum but requires a search scheme to find the global optimum. Continuous functions with analytical derivatives are necessary (unless derivatives are taken numerically, which results in even more function evaluations plus a loss of accuracy). If there are too many variables, then it is difficult to find all the extrema. The gradient of the objective function serves as the compass heading pointing to the steepest downhill path. It works well when the optimum is nearby, but cannot deal with cliffs or boundaries, where the gradient cannot be calculated.

Other optimization algorithms are stochastic algorithms, consisted of intelligent, heuristic and random methods. Stochastic algorithms have several advantages compared to other algorithms as follows: [13]

1) Stochastic algorithms are generally easy to implement.

2) They can be used efficiently in a multiprocessor environment.

3) They do not require the problem definition function to be continuous.

4) They generally can find optimal or near-optimal solutions.

There are several stochastic algorithms such as: Genetic Algorithms (GA), Guided Local Search (GLS), Tabu Search (TS), Variable Neighbourhood Search (VNS), Iterated Local Search (ILS), Simulated Annealing (SA), Greedy Randomized Adaptive Search Procedure (GRASP), Memetic Algorithms (MA), Scatter Search (SS), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Shuffled Frog Leaping algorithm (SFL), etc. Each of these algorithms has its characteristics. Especially, particle swarm optimization (PSO) is an efficient and well known stochastic algorithm which has found many successful applications in engineering problems.

Particle swarm optimization is a population-based searching technique proposed in 1995 [7] as an alternative to genetic algorithm (GA) [5]. Its development is based on the observations of social behavior of animals such as bird flocking, fish schooling, and swarm theory. Compared with GA, PSO has some attractive characteristics. First, PSO has memory, that is, the knowledge of good solutions is retained by all particles, whereas in GA, previous knowledge of the problem is destroyed ones the population is changed. Second, PSO has constructive cooperation between particles, that is, particles in the swarm share their information.

Recently, PSO has gained attention and applications by more and more researchers [14]. Please refer to [7, 9, 16] for recent survey on PSO, where some improvements and applications of PSO are provided.

In this paper, a modified PSO, named Adaptive PSO (APSO), with fast convergence to optimal or near optimal solution, is proposed. In APSO, two additional coefficients are added to the standard PSO velocity updating formula (equation 2). The coefficients will cause the APSO to move to the optimal or near optimal solution faster than the standard PSO. Also, a new procedure is proposed for escaping from local optimum traps.

PID (Proportional-Integral-Derivative) control is one of the earliest control strategies. It has been widely used in the industrial control fields. Its widespread acceptability can be recognized by:

the familiarity with which it is perceived amongst researchers and practitioners within the control community, simple structure and effectiveness of algorithm, relative ease and high speed of adjustment with minimal down-time and wide range of applications where its reliability and robustness produces excellent control performances. However, successful applications of PID controllers require the satisfactory tuning of three parameters - which are proportional gain (K_P), integral time constant (K_I), and derivative time constant (K_D) - according to the dynamics of the process. Unfortunately, it has been quite difficult to tune properly the gains of PID controllers because many industrial plants are often burdened with problems such as high order, time delays, and nonlinearities [4].

Traditionally, these parameters are determined by a trial and error approach. Manual tuning of PID controller is very tedious, time consuming and laborious to implement, especially where the performance of the controller mainly depends on the experiences of design engineers. In recent years, many tuning methods have been proposed to reduce the time consumption on determining the three controller parameters. The most well known tuning method is the Ziegler-Nichols tuning formula [22]; it determines suitable parameters by observing a gain and a frequency on which the plant becomes oscillatory.

Considering the limitations of the Ziegler-Nichols method and some empirical techniques in raising the performance of PID controller, recently artificial intelligence techniques such as fuzzy logic [18, 21], fuzzy neural network [2, 10], and some stochastic search and optimization algorithms such as simulated annealing [20], genetic algorithm [11, 19], particle swarm optimization approach [4], immune algorithm [14], and ant colony optimization [6] have been applied to improve the performances of PID controllers. In these studies, it has been shown that these approaches provide good solutions in tuning the parameters of PID controllers. However, there are several causes for developing improved techniques to design PID controllers. One of them is the important impact it may give because of the general use of the controllers. The other one is the enhancing operation of PID controllers that can be resulted from improved design techniques. Finally, a better tuned optimal PID controller is more interested in real world applications.

The generator excitation system maintains generator voltage and controls the reactive power flow using an Automatic Voltage Regulator (AVR) [4]. The role of an AVR is to hold the terminal voltage magnitude of a synchronous generator at a specified level. Hence, the stability of the AVR system would seriously affect the security of the power system. In this paper, a practical high-order AVR system with a PID controller is adopted to test the performance of the proposed PSO-PID controller.

In this paper, the proposed APSO method is applied for determining the optimal values of the parameters of PID controllers. Here, we formulate the problem of designing PID controller as an optimization problem and our goal is to design a controller that has well performance by adjusting four performance indexes, the maximum overshoot, the settling time, the rise time and the integral absolute error of step response. After designing PID controllers for some simple benchmark transfer functions, an optimal PID controller is designed for an AVR system using APSO algorithm. The advantages of this methodology are that it is a simple method with less computation burden, high-quality solution and stable and fast convergence specifications.

2. APSO ALGOEITHM

In this section, first the procedure of the standard PSO algorithm is briefly reviewed. Then, the proposed APSO algorithm is introduced.

2.1. The Standard PSO Algorithm

A particle swarm optimizer is a population based stochastic optimization algorithm modeled based on the simulation of the social behavior of bird flocks. PSO is a population-based search process where individuals initialized with a population of random solutions, referred to as particles, are grouped into a swarm. Each particle in the swarm represents a candidate solution to the optimization problem, and if the solution is made up of a set of variables, the particle can correspondingly be a vector of variables. In a PSO system each particle is "flown" through the multidimensional search space, adjusting its position in the search space according to its own experience and that of neighboring particles. The particle therefore makes use of the best position encountered by itself and that of its neighbors to position itself toward and optimal solution. The performance of each particle is evaluated using a predefined fitness function, which encapsulates the characteristics of the optimization problem.

Generally, a numerical optimization problem can be described as follows:

$$\begin{aligned} &\min \quad F(X), \quad X=[x_1,x_2,\ldots,x_N]^T \\ &\text{s.t.} \quad x_i \in [a_i,b_i], \quad i=1,2,\ldots,N. \end{aligned} \tag{1}$$

The core operation of PSO is the updating formulae of the particles, i.e. the velocity updating equation and position updating equation. The global optimizing model proposed by Shi and Eberhart (1999) is as follows [16]:

$$v_{i+1} = w \times v_i + RAND \times c_1 \times (P_{best} - x_i) + rand \times c_2 \times (G_{best} - x_i) \tag{2}$$

$$x_{i+1} = x_i + v_{i+1} \tag{3}$$

where v_i is the velocity of particle i, x_i is the particle position, w is the inertia weight factor, c_1 and c_2 are two positive constant parameters called acceleration coefficients, RAND and rand are the random functions in the range [0, 1], P_{best} is the best position of the i^{th} particle and G_{best} is the best position among all particles in the swarm.

2.2. APSO Algorithm

Globally optimize an objective function in a given search domain consists of finding its' global optimum fast without being trapped in any local optimum. Slow convergence of PSO before providing an accurate solution is a drawback, closely related to its lack of any adaptive accelerators in the velocity updating formulae. In equation 2, c1 and c2 determine the step size of the particles movements through the Pbest and Gbest, respectively. In the original PSO, these step sizes are constant and for the all particles are the same. For doing more sensitive and faster movements, new step sizes can be modified, which they should accelerate the convergence rate.

In each iteration, the value of the objective function is a criterion that presents the relative improvement of this movement with respect to the previous iteration movement. Thus, the difference between the values of the objective function in the different iterations can be selected as the accelerators. Adding two additional coefficients to the original step sizes in equation 2, it

causes to adaptive movements. Therefore, velocity updating formulae turns to the following form.

$$v_{i+1} = w\times v_i + RAND\times c_1\times (f(P_{best}) - f(x_i))\times (P_{best} - x_i) + rand\times c_2\times (f(G_{best}) - f(x_i))\times (G_{best} - x_i) \quad (4)$$

where $f(P_{best})$ is the best fitness function that is found by i^{th} particle and $f(G_{best})$ is the best objective function that is found by swarm up to now and other parameters are chosen the same as section A.

On the other hand, when strongly multi-modal problems are being optimized, PSO algorithm usually suffers from the premature suboptimal convergence (simply premature convergence or stagnation) which occurs when some poor particles attract the swarm, due to a local optimum or bad initialization, preventing further exploration of the search space. According to [1], although PSO finds good solutions faster than other evolutionary algorithms, it usually can not improve the quality of the solutions as the number of iterations is increased. The rationale behind this problem is that particles converge to a single point, which is on the line between the global best and personal best positions. This point is not guaranteed to be even a local optimum. Proofs can be found in [17]. Another reason for this problem is the high rate of information flow between particles, resulting in the creation of similar particles (with a loss in diversity) which increases the possibility of being trapped in local minima [15]. This feature prevents standard PSO from being real practical interest for lots of applications. In general, any mechanism that can increase diversity will help in preventing premature convergence. Inspired with simulated annealing technique, letting worse solutions with a probability to exist in the next population can be one way for improving diversity property of the algorithm. Therefore, in APSO algorithm, we name every point, which is found by equation (5), the temporary point x_t, ($x_t = x_{i+1}$). If x_t is better than x_i ,it will be accepted and if it is worse than x_i, we will accept it with probability of $e^{-\Delta f/T}$, where $\Delta f = f(x_t) - f(x_i)$ and T is a decreasing function of iteration number to be determined later. This process is performed for all particles. When a temporary point is rejected, that we name it a deviated particle xd, it is given back in the opposite direction of the previous movement. These descriptions are formulated by the following equations.

$$x_t = x_i + v_i$$
$$\Delta f = f(x_t) - f(x_i) \quad (5)$$

$$\text{If} \quad \Delta f < 0 \quad \text{then } x_{i+1} = x_t$$
$$\text{If} \quad \Delta f \geq 0 \quad \text{then } x_d = x_t + v_i,\ x_{i+1} = x_d \quad (6)$$

where $\alpha = \begin{cases} +\varepsilon & \text{probability} = e^{-\Delta f/T} \\ -\varepsilon & \text{otherwise} \end{cases}$ and $0 < \varepsilon \leq 1$ is a constant.

In general the proposed APSO algorithm works as follows. First, the algorithm parameters such as number of particles, initial particles and velocities, c_1 and c_2 constants and any other parameters are initialized. Then the algorithm starts with the initial swarm as initial solutions. Computing new velocities using APSO algorithm, temporary positions are calculated. For each particle, Δf is calculated, if $\Delta f < 0$ then the solution would be accepted as a better solution, otherwise worse solution would be accepted with probability of $e^{-\Delta f/T}$, and deviated particle is turned back to the opposite direction of the traveled route, equations 5 and 6. This procedure causes diversification and escaping from local optimum. This process is iterated for all the particles in the swarm. Afterwards, the annealing schedule is performed. If one of the termination conditions is satisfied then the algorithm stops, otherwise the proposed procedure is iterated. The general pseudo-code for APSO algorithm is given in Appendix A.

Remark 1. The terms $f(P_{best}) - f(x_i)$ and $f(G_{best}) - f(x_i)$ are named local and global adaptive coefficients, respectively. In each iteration, the former term defines the movement step size in the direction of best position which is found by i^{th} particle and the later term defines movement step size in the direction of the best optimum point which ever had been found by the swarm, adaptively. In other words, the adaptive coefficients decrease or increase the movement step size relative to being close or far from the optimum point, respectively. By means of this method, velocity can be updated adaptively instead of being fixed or changed linearly. Therefore, using the adaptive coefficients, the convergence rate of the algorithm will be increased that it is performed by the proportional large or short steps.

Remark 2. Stochastic optimization approaches have problem dependent performance. This dependency usually results from the parameter setting of each algorithm. Thus using different parameter settings for APSO algorithm, which is a stochastic optimization algorithm, result in high performance variances. In general, no single parameter setting exists which can be applied to all problems. Therefore, all parameters of APSO should be determined optimally, by trial and error.

Remark 3. There are three stopping criteria. The first criterion is related to the maximal number of iterations of the algorithm, the second one is when no improvement has been made for a certain number of iterations in the best solution and the third one is when a satisfactory solution is found.

Remark 4. The adaptive version of PSO is proposed for continuous variable functions. Moreover, the main idea of fasting can be applied to the discrete form of the PSO [8]. It can be a future work of the authors.

Remark 5. Increasing the value of the inertia weight, w, will increase the speed of the particles resulting in more exploration (global search) and less exploitation (local search). On the other hand, decreasing the value of w will decrease the speed of the particle resulting in more exploitation and less exploration. Thus, an iteration-dependent weight factor often outperforms a fixed factor. The most common functional form for this weight factor is linear, and changes with step i as follows:

$$w_{i+1} = w_{max} - \frac{(w_{max} - w_{min})}{N_{iter}} \times i \quad (7)$$

where N_{iter} is the maximum number of iterations and w_{max} and w_{min} are selected to be 0.9 and 0.1, respectively.

Remark 6. The initial value of T (denoted by T_0) is set by the following formula:

$$T_0 = \beta \times f(G_{best}) \quad (8)$$

where β is a positive constant and $f(G_{best})$ is the objective value of the best position among all particles in the initial swarm (population). And the function T is decreased by $T(i+1) = \theta \times T(i)$, ($T(0)=T_0$), where $0.5 < \theta < 1$ is a constant and i represents the iteration number.

Remark 7. Stop condition typically would happen, when no improvement has been made for a certain number of iteration or the maximum number of iteration has been reached or when T_0 get to be smaller than the smallest typical value (T_{min}).

Remark 8. The proposed APSO is still a general optimization algorithm that can be applied to any real world continuous optimization problems.
In next section, we will apply such an approach for several benchmark functions and compare the obtained results from APSO with the standard PSO and GA algorithms. Then it is employed to design optimal PID controller for AVR system.

3. LINEARIZED MODEL OF AN AVR SYSTEM WITH PID CONTROLLER

The role of an AVR is 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 the sensors. For mathematical modeling and transfer function 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 reasonable transfer function of these components may be represented, respectively, as follows [4].

• Amplifier model.

The amplifier model is represented by a gain and a time constant; the transfer function is

$$\frac{V_R(s)}{V_e(s)} = \frac{K_A}{(1+\tau_A s)} \quad (9)$$

Typical values of K_A are in the range of 10 to 400. The amplifier time constant τ_A is very small ranging from 0.02 to 0.1 s.

• Exciter model.

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

$$\frac{V_F(s)}{V_F(s)} = \frac{K_E}{(1+\tau_E s)} \quad (10)$$

Typical values of K_E are in the range of 10 to 400. The time constant τ_E is in the range of 0.5 to 1.0 s.

• Generator model.

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

$$\frac{V_t(s)}{V_F(s)} = \frac{K_G}{(1+\tau_G s)} \quad (11)$$

These constants are load dependent. K_G may vary between 0.7 to 1.0, and τ_G may vary between 1.0 and 2.0 sec.

• Sensor model.

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

$$\frac{V_S(s)}{V_t(s)} = \frac{K_R}{(1+\tau_R s)} \quad (12)$$

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

The above models provide an AVR system compensated with a PID controller block diagram, which is shown in Figure 1.

4. OBJECTIVE FUNCTION DEFINITION

In the design of a PID controller, the performance criterion or objective function is first defined based on some desired specifications and constraints under input testing signal. Some typical output specifications in the time domain are overshoot, rise time, settling time, and steady-state error. In general, three kinds of performance criteria, the integrated absolute error (IAE), the integral of squared-error (ISE), and the integrated of time-weighted-squared-error (ITSE) are usually considered in the control design under step input testing, as they can be evaluated analytically in the frequency domain. It is worthy to notice that using different performance indices probably makes different solutions for PID controllers. The 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 their minimization can result in a response with relatively small overshoot but a long settling time. Although the ITSE performance criterion can overcome the disadvantage of the ISE criterion, the derivation processes of the analytical formula are complex and time-consuming [4]. The IAE, ISE, and ITSE performance criteria formulas are as follows:

$$IAE = \int_0^\infty |r(t) - y(t)| \, dt = \int_0^\infty |e(t)| \, dt \quad (13)$$

$$ISE = \int_0^\infty e^2(t) \, dt \quad (14)$$

$$ISTE = \int_0^\infty t e^2(t) \, dt \quad (15)$$

In this paper, another new time domain performance criterion is defined by:

$$\min_K W(K) = (1/(1+e^{-\alpha})) \times (T_s + T_r) + (e^{-\alpha}/(1+e^{-\alpha})).(M_p + E_{ss}) \quad (16)$$

and it is used for evaluating the PID controller.

where $K = [K_P, K_I, K_D]$, and $\alpha \in [-5,5]$ is the weighting factor. The optimum selection of α depends on the designer's requirement and the characteristics of the plant under control. One can set α to be smaller than 0 to reduce the overshoot and steady-state error. On the other hand, one other can set α to be larger than 0 to reduce the rise time and settling time. If α is set to be 0, then all performance criteria (i.e. overshoot, rise time, settling time, and steady-state error) will have the same worth.

5. EXPERIMENTS AND RESULTS OF SIMULATIONS

In this section, the efficiency and effectiveness of the introduced APSO is validated using a set of test functions. Afterwards, the APSO is applied to design an optimum PID controller for an AVR system.

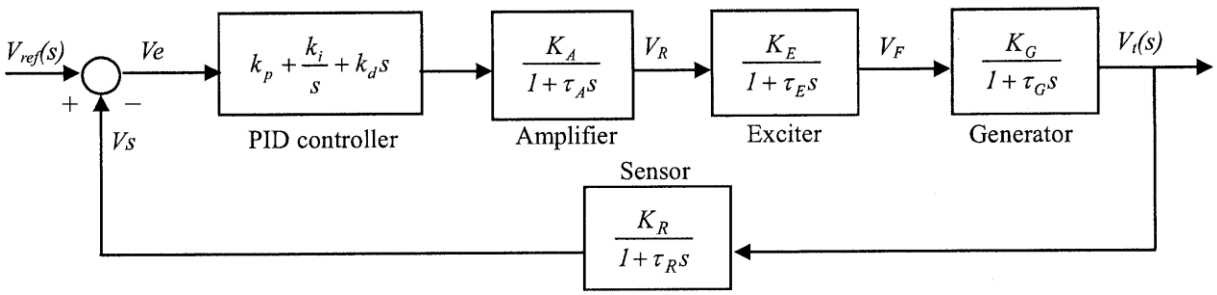

Figure 1. Block diagram of an AVR system with a PID controller

5.1. Testing Using Benchmark Functions

The efficiency of APSO is verified using a set of test functions. To avoid any misinterpretation of the optimization results, related to the choice of any particular initial populations, we performed each test 100 times, starting from various randomly selected solutions, inside the hyper rectangular search domain specified in the usual litterateur.

The results of APSO tests performed on 15 functions listed in Appendix B are shown in Table 1. To evaluate the efficiency of the proposed APSO algorithm, we retained the following criteria summarizing results from 100 minimizations per function: the rate of successful minimizations ($RATE_{SM}$), the average of the objective function evaluation numbers ($AVERAGE_{OBJEN}$) and the average error ($AVERAGE_{ERROR}$). These criteria are defined precisely in the following.

When at least one of the termination tests is verified, APSO stops and provides the coordinates of a located solution, and the objective function value "OBJ_{FPSO}" at this solution. We compared this result with the known analytical minimum" OBJ_{ANAL}"; we considered this result to be "successful" if the following inequality held:

$$|OBJ_{FPSO} - OBJ_{ANAL}| < \varepsilon_{rel}|OBJ_{INIT}| + \varepsilon_{abs} \quad (17)$$

where $\varepsilon_{rel} = 10^{-2}$, $\varepsilon_{abs} = 10^{-4}$ and OBJ_{INIT} is an empirical average of the objective function value, calculated over typically 100 solutions, randomly selected inside the search domain, before running the algorithm. The average of the objective function evaluation numbers is evaluated in relation to only the successful minimizations and it shows the convergence rate of the algorithm. In fact, this criterion measures the speed of the algorithm and shows that if it is fast or slow. The average error is defined as the average of OBJ gaps between the best successful solution found and the known global optimum. This criterion shows the accuracy of the algorithm in finding the global optimum.

As shown in Table 1, when the search space is more complicated the rate of successful minimization is decreased. Hence, APSO can escape from local minima trap because of its stochastic and intelligent nature. For all functions, the average of the objective function evaluation numbers does not exceed 100 with a suitable accuracy. This shows that the algorithm is fast in convergence. For all functions, average of the OBJ gaps between the best successful solution found and the known global optimum is less than 0.01. This accuracy is acceptable for many real world optimization problems.

A typical convergence diagram for Z_{10} function is depicted in Figure 2. One can see that the convergence rate is superior.

Table 1. Results of APSO for 15 benchmark functions

Benchmark function	$RATE_{SM}$ (%)	$AVERAGE_{OBJEN}$	$AVERAGE_{ERROR}$
RC	100	35	0.001
ES	100	38	0.003
GP	100	34	0.005
B2	100	29	0.001
SH	100	41	0.002
R_2	100	31	0.003
Z_2	100	33	0.005
DJ	100	25	0.002
$H_{3,4}$	99	28	0.005
$S_{4,5}$	99	36	0.0009
$S_{4,7}$	100	40	0.004
$S_{4,10}$	100	44	0.004
R_5	99	45	0.0043
$H_{6,4}$	98	42	0.006
Z_{10}	99	46	0.005

5.2. APSO Based PID Controller for AVR System

For designing an optimal PID controller, determination of vector K with regards to the minimization of performance index is the main issue. Here, the minimization process is performed using the proposed APSO algorithm. For this purpose, step response of the plant is used to compute four performance criteria overshoot (M_p), steady-state error (E_{ss}), rise time (T_r) and settling time (T_s) in the time domain. At first, the lower and upper bounds of the controller parameters should be specified. 0 and 1 are selected as these bounds. Then a population of particles is initialized, randomly in the specified range. Each particle represents a solution (i.e. controller parameters K) that its performance index should be evaluated. This work is performed by computing M_p, E_{ss}, T_r, and T_s using the step response of the plant, iteratively. Then, by using the four computed parameters, the performance index is evaluated for each particle according to these performance criteria. Now the main procedure of APSO algorithm performs the optimization procedure. The process is repeated until

a stopping criterion is satisfied. In this stage, the best particle is the optimal vector K.

Figure 3 shows AVR system step response without PID controller while Figure 4 demonstrates step response of AVR system with APSO based optimal PID controller. As the figures shows, step response of APSO based controller is smooth and has less overshoot. APSO found K_P, K_I and K_D equal to 0.6570, 0.4189 and 0.2458, respectively.

6. CONCLUSIONS

The problem of optimization is the active subject between researchers and many different algorithms introduced yet. Particle swarm optimization (PSO) is an effective and well-known intelligent optimization method. However, PSO suffers from two major drawbacks: low speed of convergence toward optimum and trapping in local optimum. In this paper, with introducing two modifications on traditional PSO algorithm, we generated an improved adaptive version of PSO algorithm and named it Adaptive PSO (APSO). APSO has two important advantages:

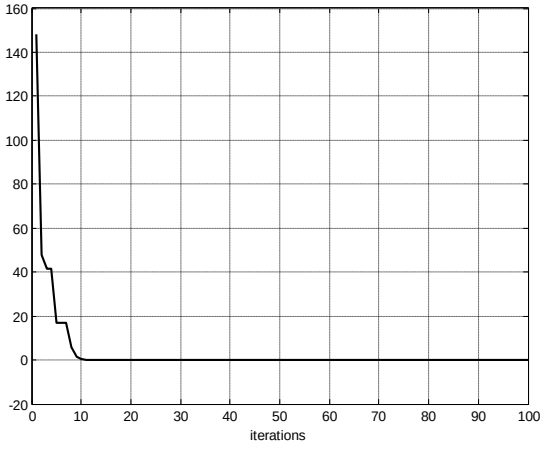

Figure 2. Typical convergence rate diagram of APSO for Z_{10}

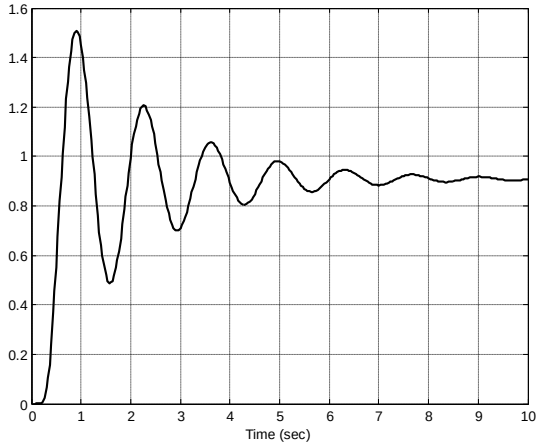

Figure 3. Terminal voltage step response of AVR system without PID controller.

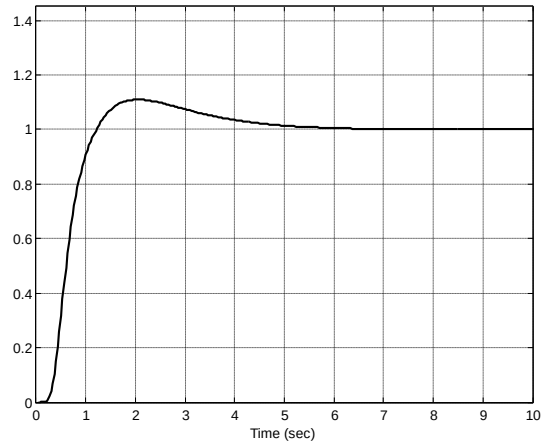

Figure 4. Terminal voltage step response of an AVR system with the APSO based PID controller.

high convergence rate and escaping from local optimum, illustrated by simulation results of some benchmark functions. As a practical application of the proposed method, APSO is used to optimal tuning of PID controllers. Through the simulation of a practical AVR system, the results show that the proposed controller can perform an efficient search for the optimal PID controller parameters.

7. APPENDIXES

7.1 Appendix A.

The pseudocode for APSO algorithm:
Initialize APSO parameters.
Loop:
REPEAT
For each particle i;
Evaluate the objective function of the particle i, i.e. $f(x_i)$;
Update the global and local best positions and their objective function values;
Calculate the velocity by equation (4);
Calculate the temporary position x_t by equation (5);
Using the equation (6) calculate the new position;
Decrease the function T;
END of Loop
If stop condition is true then stop else go to Loop;

7.2 Appendix B.

Some well-known benchmark functions of optimization problems:

7.2.1 Branin RCOS (RC) (2 variables):

$$RC(x_1,x_2) = (x_2 - (5/4\pi^2)\,x_1^2 + (5/\pi\pi x_1 - 6)^2 + 10(1-(1/8\pi))\cos(x_1) + 10$$

Search domain: $-5 < x_1 < 10, 0 < x_2 < 15$

no local minimum
3 global minima:

$$(x_1,x_2)^* = (-\pi,12.275),(\pi 2.275),(9.424,2.475)$$

$$RC((x_1,x_2)^*) = 0.397887$$

7.2.2 B2 (2 variables):

$B2(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$

Search domain: $-100 < x_j < 100, j = 1,2$

several local minima (exact number unspecified in usual literature)

1 global minimum: $(x_1, x_2)^* = (0,0)$

$B2((x_1, x_2)^*) = 0$

7.2.3 Easom (ES) (2 variables):

$ES(x_1, x_2) = -\cos(x_1)\cos(x_2)\exp(-((x_1 - \pi) + (x_2 - \pi)^2))$

Search domain: $-100 < x_j < 100, j = 1,2$

several local minima (exact number unspecified in usual literature)

1 global minimum: $(x_1, x_2)^* = (\pi, \pi)$

$B2((x_1, x_2)^*) = -1$

7.2.4 Goldstein and Price (GP) (2 variables):

$GP(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2 \times$

$(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times$

$[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2$

$- 36x_1x_2 + 27x_2^2)]$

Search domain: $-2 < x_j < 2, j = 1,2$

4 local minima

1 global minimum: $(x_1, x_2)^* = (0,-1)$

$GP((x_1, x_2)^*) = 3$

7.2.7 Shubert (SH) (2 variables):

$SH(x_1, x_2) = \{\sum_{j=1}^{5} j\cos[(j+1)x_1 + j]\} \times$

$\{\sum_{j=1}^{5} j\cos[(j+1)x_2 + j]\}$

Search domain: $-10 < x_j < 10, j = 1,2$

760 local minima

18 global minimum

$SH((x_1, x_2)^*) = -186.7309$

7.2.7 Hartmann ($H_{3,4}$) (3 variables):

$H_{3,4}(x_1, x_2, x_3) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2]$

Search domain: $0 < x_j < 1, j = 1,2,3$

4 local minima: $P_i = (p_{i1}, p_{i2}, p_{i3})$

$H_{3,4}(P_i) \cong -c_i$

1 global minimum: $X^* = (0.11, 0.555, 0.855)$

$H_{3,4}((x_1, x_2, x_3)^*) = -3.86278$

7.2.8 Hartmann ($H_{6,4}$) (6 variables):

$H_{6,4}(x_1, \ldots, x_6) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{6} a_{ij}(x_i - p_{ij})^2]$

Search domain: $0 < x_j < 1, j = 1, \ldots, 6$

6 local minima: $P_i = (p_{i1}, \ldots, p_{i6})$

$H_{6,4}(P_i) \cong -c_i$

1 global minimum

$X^* = (0.201, 0.150, 0.476, 0.275, 0.311, 0.657)$

$H_{6,4}(X^*) = -3.3223$

7.2.9 De Joung (DJ) (3 variables):

$DJ(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2$

Search domain: $-5.12 < x_j < 5.12, j = 1,2,3$

1 single minimum (local and global)

$(x_1, x_2, x_3)^* = (0,0,0)$

$DJ((x_1, x_2, x_3)^*) = 0$

7.2.10 Shekel ($S_{4,n}$) (3 variables):

$S_{(4,n)}(X) = -\sum_{j=1}^{n} [(X - a_i)^T (X - a_i) + c_i]^{-1}$

$X = (x_1, x_2, x_3, x_4)^T$

$a_i = (a_i^1, a_i^2, a_i^3, a_i^4)^T$

3 functions were considered: $S_{4,5}$, $S_{4,7}$ and $S_{4,10}$

Search domain: $0 < x_j < 10, j = 1,2,3,4$

n local minima: a_i : ith local minimum approximation

$S_{4,n}(a_i^T) \cong -1/c_i$

1 global minimum for each function

$S_{4,5}(X) = -10.15, S_{(4,7)}(X) = -10.40, S_{(4,10)}(X) = -10.53$

7.2.11 Rosenbrock (R_n) (n variables):

$R_n(X) = -\sum_{j=1}^{n} -[100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2]$

Search domain: $-5 < x_j < 10, j = 1, \ldots, n$

several local minima (exact number unspecified in usual literature)

1 global minimum: $X^* = (1, \ldots, 1)$;

$R_n(X^*) = 0$

7.2.12 Zakharov (Z_n) (n variables):

$Z_n(X) = (\sum_{j=1}^{n} x_j^2) + (\sum_{j=1}^{n} 0.5jx_j)^2 + (\sum_{j=1}^{n} 0.5jx_j)^4$

Search domain: $-5 < x_j < 10, j = 1, \ldots, n$

several local minima (exact number unspecified in usual literature);

1 global minimum: $X^* = (0, \ldots, 0)$;

$Z_n(X^*) = 0$

8. REFRENCES

[1] Angeline, P. 1998. Using Selection to Improve Particle Swarm Optimization, In Optimization Conference on Evolutionary Computation, Piscataway, New Jersey, USA, pp. 84-89, IEEE service center.

[2] Chu, S.Y., and Teng, C.C. 1999. "Tuning of PID controllers based on gain and phase margin specifications using fuzzy neural network," Fuzzy Sets and Systems, 101(1), pp. 21-30.

[3] Fogel, L. 1994. Evolutionary Programming in Perspective: Top-Down View. Computational Intelligence: Imitating Life, Piscataway, New Jersey, USA, IEEE.

[4] Gaing, Z. L. 2004. "A Particle Swarm Optimization Approach For Optimum Design of PID controller in AVR system," IEEE Transactions on Energy Conversion, 9(2):384-391.

[5] Holland, J. H. 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, Internal Report.

[6] Hsiao, Y. T., Chuang, C. L., and Chien, C. C. 2004. "Ant colony optimization for designing of PID controllers," in proceedings of the 2004 IEEE Conference on Control Applications/International Symposium on Intelligent Control/International Symposium on Computer Aided Control Systems Design, Taipei, Taiwan.

[7] Kennedy, J., and Eberhart, R. 1995. Particle Swarm Optimization. In Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, vol.4, pp.1942-1948.

[8] Kennedi, J., and Eberhart, R. 1997. A Discrete Binary Version of the Particle Swarm Algorithm. In Proceedings of the Conference on Systems, Man, and Cybernetics, pp. 4104-4109.

[9] Kennedy, J. Eberhart, R. C., and Shi, Y. 2001. Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco.

[10] Kim, D. H. 2001. "Tuning of a PID controller using a artificial immune network model and local fuzzy set", in Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference, vol.5, pp. 2698 – 2703.

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

[12] Li, L. L., Wang, L., and Liu, L. H. 2005. An effective hybrid PSOSA strategy for optimization and its application to parameter estimation, Applied Mathematics and Computation.

[13] Lovberg, M., and Krink, T. 2002. Extending Particle Swarm Optimizers with Self-Organized Criticality. In Proceeding of Forth Congress on Evolutionary Computation, vol. 2, pp.1588- 1593.

[14] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller A., and Teller, E. 1953, Journal of Chemical Physics, vol. 21, pp. 1087–1092.

[15] Riget, J., and Vesterstrom, J. 2002. A Diversity-Guided Particle Swarm Optimizer- The ARPSO, EVALife Technical Report, no 2002-2.

[16] Shi, Y., and Eberhart, R. 1998. A modified particle swarm optimizer. Proceedings of the IEEE international conference on evolutionary computation. Piscataway, NJ: IEEE Press; p. 69–73.

[17] Van den Bergh, F. and Engelbrecht, A. P. 2002. A New Locally Convergent Particle Swarm Optimization, In Proceedings of the IEEE Conference on Systems, Man, and Cybernetics, Hammamet, Tunisia.

[18] Visioli, A. 1999. "Fuzzy logic based set-point weight tuning of PID controllers", IEEE Trans. System, Man, and Cybernetics – Part A: System and Humans, vol. 29, no. 6, pp. 587-592.

[19] Wang, P., and Kwok, D.P. 1994. "Optimal design of PID process controllers based on genetic algorithms", Control Engineer Practice, vol. 2, no. 4, pp.641-648.

VII: Proc. EP98, New York, pp. 591–600.

[20] Zhou, G., and Birdwell, J. D. 1994. "Fuzzy logic-based PID autotuner design using simulated annealing, in Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design, 7-9, pp. 67 – 72.

[21] Zhao, Z.Y., Tomizuka, M. and Isaka, S. 1993. "Fuzzy gain scheduling of PID controllers", IEEE Trans. System, Man, and Cybernetics, vol. 23, no. 5, pp. 1392-1398.

[22] Ziegler, J.G., and Nichols, N.B. 1942. "Optimum settlings for automatic controllers", Trans. On ASME., vol. 64, pp.759-768.