Research Article Open Access

Majid Moradi Zirkohi*

OPTIMAL PID CONTROLLER DESIGN USING ADAPTIVE VURPSO ALGORITHM

Abstract: The purpose of this paper is to improve the Velocity Update Relaxation Particle Swarm Optimization algorithm (VURPSO). The improved algorithm is called Adaptive VURPSO (AVURPSO) algorithm. Then, an optimal design of a Proportional-Integral-Derivative (PID) controller is obtained using the AVURPSO algorithm. An adaptive momentum factor is used to regulate a trade-off between the global and the local exploration abilities in the proposed algorithm. This operation helps the system to reach the optimal solution quickly and saves the computation time. Comparisons on the optimal PID controller design confirm the superiority of AVURPSO algorithm to the optimization algorithms mentioned in this paper namely the VURPSO algorithm, the Ant Colony algorithm, and the conventional approach. Comparisons on the speed of convergence confirm that the proposed algorithm has a faster convergence in a less computation time to yield a global optimum value. The proposed AVURPSO can be used in the diverse areas of optimization problems such as industrial planning, resource allocation, scheduling, decision making, pattern recognition and machine learning. The proposed AVURPSO algorithm is efficiently used to design an optimal PID controller.

Keywords: Particle Swarm Optimization, Velocity Update Relaxation Particle Swarm Optimization, Optimal PID Controller

DOI 10.1515/eng-2015-0015
Received August 25, 2014; accepted January 19, 2015

1 Introduction

So far, proportional-integral-derivative controller (PID) is the most commonly used controller in industry for its simplicity, robustness, and ease of implementation [1]. Many practical control applications are based on PID control [2–4]. In process control, a PID controller attempts to correct the error between a measured process variable and a desired set point by calculating and then outputting a corrective action that can adjust the process accordingly. Performance of the control system is based on selecting three parameters namely proportional, integral, and derivative gains.

Many process plants controlled by PID controllers have similar dynamics. It is possible to set satisfactory controller parameters from less plant information than a complete mathematical model [2]. In practice, systems usually have some complex features, such as nonlinearity and time delay, which make tuning of a PID controller difficult. Traditionally, the problem has been handled by the trial and error approach. In the past decade more systematic methods have been introduced. Ziegler-Nichols (ZN) tuning formula is an experimental one that is widely used [2]. One of the disadvantages of this method is the necessity of prior knowledge regarding plant model. Once tuning the controller by ZN method, a good but not optimum system response will be reached. Other methods for tuning the PID controller have been proposed by many researches in the area of control engineering [5–7].

The evolutionary computation technique has become gradually popular to obtain global optimal solution in many areas [8]. In order to find optimum parameters of the PID controller, evolutionary methods such as Genetic Algorithm (GA) [9], Particle Swarm Optimization (PSO) [10] and Ant Colony (AC) [11] have been proposed. Among them, the PSO algorithm proposed by Kennedy and Eberhart as a stochastic optimization strategy is a most promising method [12]. It is due to the good features such as combination of simplicity (in terms of its implementation), low computational cost and good performance [12]. PSO algorithm as a new attractive optimizer has been applied in variety of research fields [13]. The PSO algorithm is an excellent optimization methodology and a promising approach for finding the optimal PID controller parameters [14]. Performance of a control system is improved because of optimizing control design parameters using the PSO algorithm [15]. Despite the mentioned advantages, it has some shortages. For example, it is easy to be trapped in the local optimal and searching ability reduces significantly in

***Corresponding Author: Majid Moradi Zirkohi:** Department of Electrical Engineering, Behbahan Khatam Alanbia University of Technology, Behbahan, Iran, E-mail: moradi@bkatu.ac.ir

the later stages of optimization [16] that sometimes leads to slow convergence rates.

To obtain a better optimal solution and enhance performance of the PSO, incorporation of velocity update relaxation strategy in PSO namely VURPSO helps to have enhanced computational efficiency. In the traditional PSO algorithm, the velocity is updated at every iteration cycle. In the VURPSO, velocity should be updated only when a particle cannot further improve the fitness with its previous velocity rather than in every iteration. This philosophy of velocity updating enhances computational efficiency compared to the traditional PSO. Moreover, a momentum factor has been incorporated in position update equation which can limit the particles in defined search space without checking the boundary at every iteration [1]. Tuning parameters of a dual input power system stabilizer connected with a single-machine to infinite bus has been done with a help of the VURPSO algorithm and GA algorithm [1]. In [17] some well-known function has been minimized using the VURPSO algorithm. Simulation results have confirmed that the VURPSO algorithm is superior to the PSO algorithm. The value of momentum factor is very significant in order to ensure an optimal tradeoff between exploration and exploitation mechanisms of the swarm population. A larger value of momentum factor enhances the exploration whereas a smaller value improves the local exploitation.

Motivated by the aforementioned researches, as a novelty, this paper develops an adaptive mechanism for regulating the momentum factor in VURPSO, which provides a balance between global exploration and local exploitation to achieve a faster convergence speed and better solution accuracy with minimum incremental computational burden. We call the proposed method Adaptive VURPSO (AVURPSO). We use AVURPSO to optimize the PID coefficients by minimizing a new performance criteria used as a fitness function. Comparisons in terms of the speed of convergence and precision between VURPSO and AVURPSO are presented through simulations on a time delay system.

This paper is organized as follows: Section 2 describes the particle swarm optimization. Section 3 presents the Velocity Update Relaxation Particle Swarm Optimization. Section 4 develops the Adaptive VURPSO. Section 5 designs the PID controller. Section 6 presents the simulations and finally, Section 7 concludes the paper.

2 PSO Algorithm

The PSO algorithm is performed as follows: the unknown parameters are called the particles that form the population size denoted by . Starting with a random initialization, the particles will move in a searching space to minimize an objective function. The parameters are estimated through minimizing the objective function. The fitness of each particle is evaluated according to the objective function for updating the best position of particle and the best position among all particles as two goals in each step of computing. Each article is directed to its previous best position and the previous best position among particles. Consequently, the particles tend to fly towards the better searching areas over the searching space. The velocity of i^{th} particle v will be calculated as follows [12, 13]:

$$v_i(k+1) = v_i(k)+c_1r_1(pbest_i(k)-x_i(k))+c_2r_2(gbest-x_i(k)) \quad (1)$$

where for the i^{th} particle in the k–th iteration, x_i is the position, $pbest_i$ is the previous best position, $gbest$ is the previous global best position of particles, c_1 and c_2 are the acceleration coefficients namely the cognitive and social scaling parameters, r_1 and r_2 are two random numbers in the range of [0 1].

A new position of the ith particle is then calculated as

$$x_i(k+1) = x_i(k) + v_i(k+1) \quad (2)$$

The PSO algorithm performs repeatedly until the goal is achieved. Number of iterations denoted by $k_{\max}$ can be set to a specific value as a goal of optimization.

3 Velocity Update Relaxation Particle Swarm Optimization

In the traditional PSO, velocities of the particles are limited in the range of $[v_{\min}, v_{\max}]$. Usually values of $v_{\min}$ and $v_{\max}$ are set to $x_{\min}$ and $x_{\max}$, respectively. Because position of each particle is limited in the range of $[x_{\min}, x_{\max}]$ and velocity is updated at every iteration cycle, evaluating the obtained results according to the limits for confining or rejecting the results takes extra computational burden. In contrast, the VURPSO postulates the particles in a defined search space without checking the boundary at every iteration [1, 17]. Velocity of each particle is kept unchanged if its fitness at current iteration is better than one at the preceding iteration; otherwise velocity is updated. As a result, the computational efficiency is enhanced.

The new position of particle is then calculated as [1]:

$$x_i(k+1) = (1-mf)\times x_i(k) + (mf)\times v_i(k+1) \quad (3)$$

where mf is called momentum factor given in the range of $0 < mf < 1$. The new position vector is a point on the line

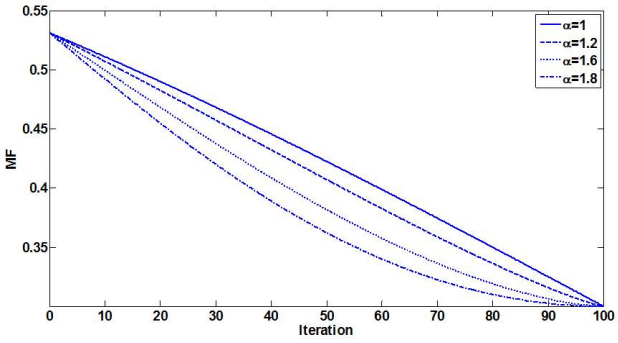

Figure 1: Momentum factor with respect to iteration in different values of α.

between the former position vector and the new velocity vector.

4 Adaptive Velocity Update Relaxation Particle Swarm Optimization

At the beginning of run, the particles are far away from the optimum point. Hence, a big velocity is needed to globally search the solution space and hence momentum factor must be a large value. At the end, only small movements are needed and thus momentum factor must set to a small value. This operation helps the system to reach the optimum of the fitness function quickly. In stochastic search, regulating a good balance between global exploration and local exploitation is very important [6]. Therefore, adaptive updating of the momentum factor can provide a good balance between global and local explorations. This is why adaptive VURPSO is superior to VURPSO that uses a constant mf.

This paper proposes a novel momentum factor in the form of a nonlinear function with respect to iteration as follows:

$$mf = \frac{0.8 - 0.2e^{-((k_{\max}-k)/k_{\max})^{\alpha}}}{1 + e^{-((k_{\max}-k)/k_{\max})^{\alpha}}} \tag{4}$$

where α is a nonlinear modulation index. Equation (4) implies that $0.3 \le mf < 0.53$.

Equations (3) and (4) confirm that velocity and position are rapidly changed at the beginning of run while this change is relatively less at the end of run. Thus, AVURPSO tends to have strong ability for global search at the beginning of run and for local search near the end of run.

A reasonable set of choice for nonlinear modulation index α is found in the range of [1, 2]. The momentum factor is decreasing as the iteration goes ahead as shown in Figure 1. By selecting a larger index factor, we have a larger change when starting and a smaller change at the end of run.

5 Optimal PID Controller Design

The continuous form of a PID controller, is given by

$$u(t) = K_p[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{d}{dt}e(t)] \tag{5}$$

where $e(t)$ is input of controller, $u(t)$ is output, K_p is the proportional gain, T_i is the integral time constant, and T_d is the derivative time constant. We can also rewrite equation (5) in the Laplace form of

$$U(s) = \left(K_p + \frac{K_i}{s} + K_d s\right) E(s) \tag{6}$$

where K_i and K_d stand for the integral gain and derivative gain, respectively [2].

A PID controller design process using the optimization algorithm is shown in Figure 2, where y_d is the desired output, y is the actual output, and u is the control input generated by the PID controller as defined in equation (5).

For a given plant, the problem of designing a PID controller is to adjust the parameters K_p, K_i, K_d such that the actual output follows the desired output. Performance of control system is evaluated based on a performance criterion which includes expected values of some factors such as the overshoot, rise time, settling time, steady state error, and integral of absolute errors. Two kinds of Performance criteria in output tracking, usually considered in the controller designing, are the integral squared error (ISE) and integral absolute error (IAE) of the desired output [18]. A

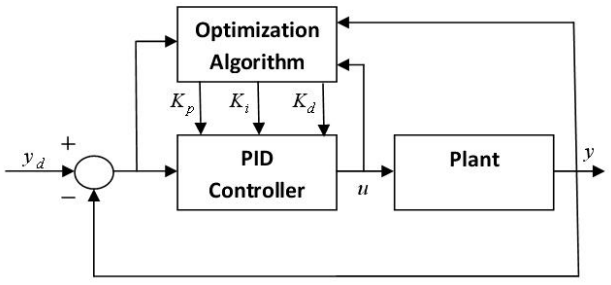

Figure 2: A PID controller using the optimization algorithm.

disadvantage of the ISE and IAE criteria is that they may result in a response with a relatively small overshoot but a long settling time because they weigh all errors uniformly over time.

We are interested in minimizing the integral of absolute error (IAE) and control effort, overshoot, settling time and control effort. To achieve this purpose, based on the concept of multi-objective optimization, a new performance criterion (fitness) in the time domain is proposed for evaluating the PID controller as follows:

$$F(x_i(k)) = \beta \left(\int_0^T (0.1 \left|e(t)\right| + \left|u(t)\right|)dt \right) + (1-\beta)(t_s + OS) \tag{7}$$

where $x_i(k)$ is the parameters $[K_p, K_i, K_d]$, β is a weight factor given $0 \le \beta \le 1$,$e(t) = y_r - y$, t_s is settling time, OS is overshoot, $\left|e(t)\right|$ is absolute error and $u(t)$ is output of controller (control effort). The integral of absolute error (IAE) and integral of absolute-input must be computed numerically up to T which is chosen sufficiently large so that $e(t)$ is negligible for $t > T$. Selection of β depends on the expected specifications for the control system and the characteristics of the plant under control. The performance criterion contains two terms. With the use of $\beta > 0.5$, importance of the first term will be more than the second term in the optimization algorithm.

The AVURPSO algorithm is summarized as follows:

1. Initialize positions and velocities of particles with $x_i(1)$ and $v_i(1)$.
2. Evaluate the fitness of each particle by computing $F(x_i(k))$ in (7).
3. Compare the fitness of each particle to the fitness of its best position to update the best position. This states that if $F(x_i(k)) < F(pbest_i(k))$ then $pbest_i(k) = x_i(k)$.
4. Compare the fitness of each particle to the fitness of the global best particle to update the global best position. This means that if $F(x_i(k)) < F(gbest)$ then $gbest = x_i(k)$.
5. If the fitness of the ith particle at $k-1$ iteration is better than at k iteration, then update the velocity according to the Equation (1).
6. Calculate the momentum factor using Equation (4).
7. Update the position of each particle according to Equation (3).
8. Go to step 2 and repeat until the stopping criteria is met.

6 Simulation Results

The proposed AVURPSO algorithm is compared with the VURPSO algorithm and a conventional method through the following simulations. The mentioned algorithms are applied to optimize a PID controller. The task is a three-dimensional optimization problem of determining the optimal coefficients $[K_p \quad K_i \quad K_d]$ to minimize the cost function (7). In both VURPSO and AVURPSO, the population size and maximum number of iteration are given to $n = 100$ and $k_{\max} = 100$, respectively. Moreover, In VURPSO, the design parameters are adopted as $c_1 = c_2 = 2$ and constant $mf = 0.3$ [1]. In order to have a comparison, in the proposed AVURPSO, we select mf as expressed by Equation (4) and we set $\alpha = 1.2$. The controller is applied on the two following plants as examples. The control objective is a set point application where the desired output given by $y_d = 1$. Each algorithm runs 20 times. Among the results, the best one is selected for each algorithm.

Example 1:

The plant is a resonant time delay system as follows [19]:

$$G(s) = \frac{e^{(-0.1s)}}{s^2 + 0.02s + 1} \tag{8}$$

The search space of PID gains is defined by: $K_{p\min} = 0$, $K_{p\max} = 10$, $K_{i\min} = 0$, $K_{i\max} = 10$, $K_{d\min} = 0$, $K_{d\max} = 10$. A comparison between algorithms on performance of the PID controller on different values of β is given in Table 1. The best results of all algorithms are achieved with $\beta = 0.54$ as presented in Table 1. The AVURPSO is superior to others as confirmed by comparisons. Statistical analysis in terms of the worst value, mean value, best value and standard deviation of fitness function are given by Table 2. The worst value of AVURPSO is better than the best value of the VURPSO and Conventional approach [20]. The low standard deviation value of the proposed algorithm ensures the degree of consistency in producing the global optimal value.

Figure 3 shows the speed of convergence versus iteration in these algorithms. The AVURPSO converges in less iteration and more quickly than another one. The results are compared in execution time in Table 3. The AVURPSO has a much more convergence speed than VURPSO. As a result, it can save the time required to find the global optimum. The AVURPSO reaches to a desired value after 21 iterations while VURPSO finds it after 33 iterations. So, the proposed method has a superior performance in terms of convergence speed and accuracy. This fact is verified by Equations (3) and (4) and shown by Figure 3. The velocity

Table 1: A comparison between algorithms on performance of the PID controller in Example 1.

	Algorithm	K_p	K_i	K_d	OS	t_s	IAE
0.67	AVURPSO	0.59	0.55	1.28	0	2	2.28
	VURPSO	3.89	0.55	5.4	0.03	4	2.36
0.6	AVURPSO	1.19	0.82	1.57	0	2	1.7
	VURPSO	1.029	0.79	1.37	0.01	2	1.75
0.54	AVURPSO	1.1555	0.8708	1.4661	0.0004	2	1.644
	VURPSO	1.1261	0.865318	1.4700	0.0003	2	1.648
	Conventional	0.144	0.3871	0.8082	0.011	10.13	11.33
0.36	AVURPSO	0.62	0.65	1.259	0.03	4	1.83
	VURPSO	0.85	1.019	1.41	0.1	5	2.1

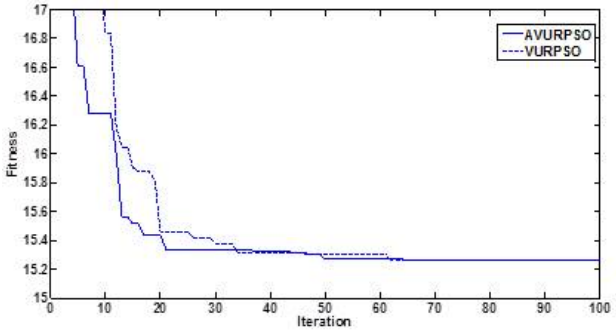

Figure 3: Comparing the VURPSO and AVURPSO on speed of convergence in Example 1.

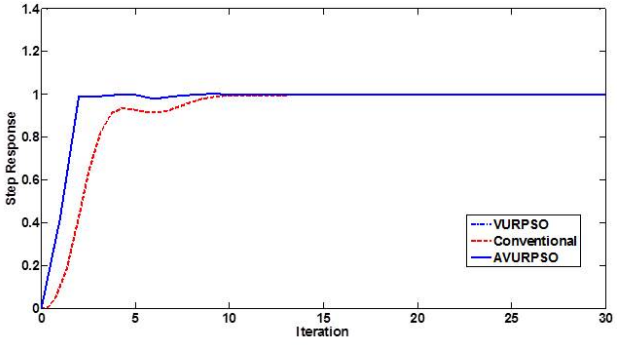

Figure 4: Comparing VURPSO, AVURPSO and Conventional design on performance of control in Example 1.

Table 2: A comparison between AVURPSO and VURPSO on statistical analysis of fitness function in Example 1.

Algorithm	worst	Mean	Best	Std.
AVURPSO	15.296	15.281	15.262	0.012
VURPSO	15.352	15.292	15.260	0.027

Table 3: A comparison between AVURPSO and VURPSO on execution time in Example 1.

β	Algorithm	Execution time
054	77.879	AVURPSO
	94.403	VURPSO

and position are rapidly changed at the beginning of run while this change is relatively less at the end of run. On the other words, the AVURPSO tends to have strong ability for global search at the beginning of run and for local search near the end of run. Both the accuracy and the speed of convergence have become higher with the use of adapting parameter. It can be concluded that the adaptive nature makes the proposed algorithm more efficient to apply for many problems than two other algorithms. This superiority is confirmed in step response of the obtained PID controller using different algorithms with $\beta = 0.54$ as shown in Figure 4.

Example 2:

Consider a third-order system whose transfer function is given by [11]:

$$G(s) = \frac{4.228}{s^3 + 2.14s^2 + 9.27s + 4.228} \tag{9}$$

Table 4: A comparison between algorithms on performance of PID controller in Example 2.

β	Algorithm	K_p	K_i	K_d	OS	t_s	IAE
0.54	AVURPSO	0.79378	0.65047	0.01	0.04	3.6803	11.379
	VURPSO	0.79272	0.65029	0.01	0.0410	3.6809	11.382
	Ant Colony	2.517	2.219	1.151	0.1649	4.9962	37.716

Table 5: A comparison between AVURPSO and VURPSO on statistical analysis of fitness function in Example 2.

Algorithm	worst	Mean	Best	Std.
AVURPSO	5.6742	5.490	5.384	0.0309
VURPSO	5.848	5.507	5.388	0.0806

Table 6: A comparison between AVURPSO and VURPSO on execution time in Example 2.

β	Algorithm	Execution time
0.54	62.21	AVURPSO
	89.04	VURPSO

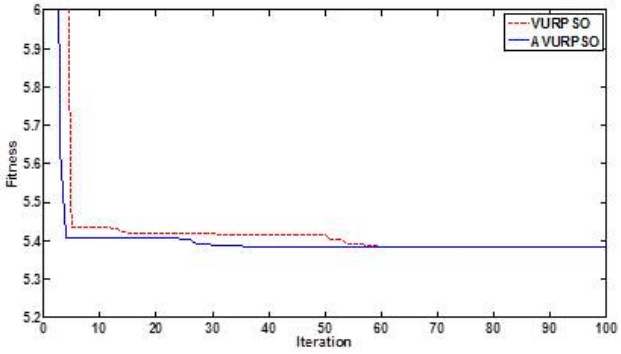

Figure 5: Comparing the VURPSO and AVURPSO on speed of convergence in Example 2.

The search space of PID gains is defined by: $K_{p\min} = 0$, $K_{p\max} = 7$, $K_{i\min} = 0, K_{i\max} = 7$, $K_{d\min} = 0, K_{d\max} = 7$ AVURPSO , VURPSO and Ant Colony [11] algorithms are compared through simulation for $\beta = 0.54$. The results are presented in Table 4 and the worst value, mean value, best value and standard deviation (Std.) are represented in Table 5. By comparing the algorithms we can conclude that the AVURPSO algorithm is not only better than others but also provides more reliable solutions because of having a lower standard deviation value. In addition, to compare execution time of these algorithms each algorithm runs 20 times. Average of the elapsed time is considered as a criterion for execution time. Among the three algorithms considered in this paper, the proposed algorithm spends less computation time to reach answer compared with other algorithms as confirmed by Table 6.

Figure 5 shows the speed of convergence in these algorithms versus iteration. AVURPSO converges quickly with less iteration as compared with other algorithm. Again, simulation results confirm the superiority of AVURPSO algorithm in terms of accuracy and speed of convergence without the premature convergence problem. The same implication is given for the set point application as shown in Figure 6.

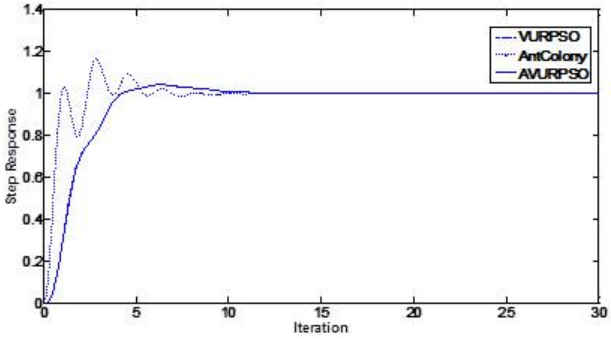

Figure 6: Comparing VURPSO, AVURPSO, and Ant Coloni on performance of control in Example 2.

7 Conclusion

In this paper, an adaptive version of the velocity update relaxation particle swarm optimization has been presented. The proposed algorithm employs an adaptive momentum factor to restrict the particles inside the defined search space without checking the boundary at every iteration step. Due to the fact that the momentum factor is adaptively decreased from a relatively large value to a small value, AVURPSO shows strong global search ability at the beginning of the run and strong local search ability near the end of the run. The proposed algorithm has been efficiently used in the optimal design of the PID controller.
To show the validity of the proposed algorithm, comparisons between the proposed algorithm and others mentioned in this paper have been presented through numerical simulations. The case study is an optimal PID controller design used to control a resonant time delay system in Example 1 and a third order linear system in Example 2. A new performance criterion has been introduced as a cost function for optimizing the performance of the PID controller in the set point application. Performance of the control system has been evaluated in terms of over shoot, settling time and integral of absolute error and control effort. Statistical analysis of the fitness function and executing time has been computed for each algorithm. Comparisons between algorithms confirm superiority of the AVURPSO algorithm to the VURPSO algorithm, Ant Colony algorithm and the conventional approach in terms of computation time and speed of convergence.

References

[1] A. Chatterjee, V. Mukherjee, and S. Ghoshal, Velocity relaxed and craziness-based swarm optimized intelligent PID and PSS controlled AVR system, International Journal of Electrical Power & Energy Systems, 2009, 31, 323-333.

[2] D. Xue, Y. Chen, D. P. Atherton, Linear feedback control: analysis and design with MATLAB, Siam, 2007.

[3] K. L. Rihem Farkh, Mekki Ksouri, Stabilizing Sets of PI/PID Controllers for Unstable Second Order Delay System, International Journal of Automation and Computing, 2014, 11, 210-222.

[4] L.-L. Ou, J.-J. Chen, D.-M. Zhang, W.-D. Zhang., Distributed H∞ PID feedback for improving consensus performance of arbitrary-delayed multi-agent system, International Journal of Automation and Computing, 2014, 11, 189-196.

[5] M. Chidambaram, R. Padma Sree, A simple method of tuning PID controllers for integrator/dead-time processes, Computers & chemical engineering, 2003, 27, 211-215.

[6] J.-C. Shen, New tuning method for PID controller, ISA Transactions, 2005, 41, 473-484.

[7] L.-H. L. Fu-Cai Liu, Juan-Juan Gao, Fuzzy PID Control of Space Manipulator for Both Ground Alignment and Space Applications, International Journal of Automation and Computing, 2014, 11,. 353-360.

[8] R. L. Johnston, H. M. Cartwright, Applications of evolutionary computation in chemistry, Springer, 2004.

[9] A. Herreros, E. Baeyens, J. R. Perán, Design of PID-type controllers using multiobjective genetic algorithms, ISA Transactions, 2002, 41, 457-472.

[10] P. J. Angeline, Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences, 7th International Conference EP98, (25-27 March 1998, San Diego, USA), 601-610.

[11] Y.-T. Hsiao, C.-L. Chuang, C.-C. Chien, Ant colony optimization for designing of PID controllers, 2004 IEEE International Symposium on Computer Aided Control Systems Design, (2-4 September 2004, Taipei, Taiwan), 321-326.

[12] R. C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, (4-6 October 1995, Nagoya, Japan), 39-43.

[13] Y.-P. Chang, C.-N. Ko, A PSO method with nonlinear time-varying evolution based on neural network for design of optimal harmonic filters, Expert Systems with Applications, 2009, 36, 6809-6816.

[14] W.-D. Chang, S.-P. Shih, PID controller design of nonlinear systems using an improved particle swarm optimization approach, Communications in Nonlinear Science and Numerical Simulation, 2010, 15, 3632-3639.

[15] M. M. Fateh, M. M. Zirkohi, Adaptive impedance control of a hydraulic suspension system using particle swarm optimisation, Vehicle System Dynamics, 2011, 49, 1951-1965.

[16] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation, 2002, 6, 58-73.

[17] Y. Liu, Z. Qin, X. He, Supervisor-student model in particle swarm optimization, Congress on Evolutionary Computation, 2004, (19-23 June 2004, Portland, USA), 542-547.

[18] R. A. Krohling, J. P. Rey, Design of optimal disturbance rejection PID controllers using genetic algorithms, IEEE Transactions on Evolutionary Computation, 2001, 5, 78-82.

[19] B. Kristiansson, B. Lennartson, Optimal PID controllers for unstable and resonant plants, Proceedings of the 37th IEEE Conference on Decision and Control, 1998 , (18 December 1998, Tampa, USA) 4380-4381.

[20] D. P. Atherton, S. Majhi, Limitations of PID controllers, Proceedings of the 1999 American Control Conference, (2-4 June 1999, San Diego, USA), 3843-3847.