INT J COMPUT COMMUN, ISSN 1841-9836
Vol.7 (2012), No. 2 (June), pp. 204-217

Application of Chaos Embedded PSO for PID Parameter Tuning

O.T. Altinoz, A.E. Yilmaz, G.-W. Weber

O. Tolga Altinoz
Hacettepe University Bala Vocational School
Electronics Technology Department, Ankara, Turkey
taltinoz@hacettepe.edu.tr

A. Egemen Yilmaz
Ankara University
Electronics Engineering Department, Ankara, Turkey
aeyilmaz@eng.ankara.edu.tr

G. Wilhelm Weber
Middle East Technical University
Institute of Applied Mathematics, Ankara, Turkey
gweber@metu.edu.tr

Abstract: Proportional-Integral-Derivative (PID) control is the most common method applied in the industry due to its simplicity. On the other hand, due to its difficulties, parameter tuning of the PID controllers are usually performed poorly. Generally, the design objectives are obtained by adjusting the controller parameters repetitively until the desired closed-loop system performance is achieved. This allows researchers to use more advanced and even some heuristic methods to achieve the optimal PID parameters. This paper focuses on application of the chaos embedded particle swarm optimization algorithm (CPSO) for PID controller tuning, and demonstrates how to employ the CPSO method to find optimal PID parameters in details. The method is applied to optimal PID parameter tuning for three typical systems with various ordered, and comparisons with the conventional PSO and the Ziegler-Nichols methods are performed. The numerical results from the simulations verify the performance of the proposed scheme.
Keywords: Particle Swarm Optimization (PSO); chaos theory; PID control; multidimensional optimization.

1 Introduction

Numerous control techniques have so far been introduced and applied in industrial problems. On the other hand, since its invention in 1910, Proportional-Integral-Derivative (PID) control has become the best known controller due to its simplcitiy and efficiency in various control problems. Industrial implementation of the PID controller is remarkable that its tuning still presents a challenge in many applications. Primary problem associated with such systems is to obtain the optimal PID controller parameters for a satisfactory control performance, which is a quite difficult task. Therefore, enormous amount of research results have been reported in the literature, among which the well-known tuning method was proposed by Ziegler and Nichols in 1942. Despite the huge number of proposed approaches for PID parameter tuning, most of them (as well as the Ziegler-Nichols method) occasionally yield poor performance in practice.

Generally, conventional tuning methods use the root locus and frequency response for PID parameter tuning. Early presentation of the most common conventional tuning method, the Ziegler-Nichols method, was based on the open-loop step response of the system. With such

definition, the performance was observed to be inadequate and poor. Later, the authors improved the method by making it dependent on the frequency response of the close-loop system. The procedure imposed by this method can be summarized as follows: i) First, the system is assumed to be under proportional control only. The proportion parameter is increased until the system becomes critically stable. The proportion parameter yielding this condition is recorded as well as the corresponding oscillation period; ii) Based on these values, other parameters (i.e. corresponding to the integral and derivative operations) are determined via some look-up-tables which can be found any Control Theory textbook. However, this method yields unsatisfactory phase and gain margins. In addition, due to its off-line unautomatized nature, the Ziegler-Nichols method is not quite applicable to processes in the working systems in practice. For this reason, recently, heuristic methods have been employed for PID parameter tuning in order to improve the controller performance.

In the last decade, heuristic approaches have received increased attention from researchers dealing with engineering problems. Over the years, several heuristic methods have been proposed for PID parameter tuning. In 1995, a novel heuristic approach called the Particle Swarm Optimization (PSO) was introduced by Eberhart and Kennedy [7]. Compared to other optimization algorithms, PSO has a very simple mathematical definition yielding easy-to-implement and short computer programs, which can generate high-quality solutions with reasonable speed. However, like other heuristic methods, conventional PSO suffers from premature convergence especially in higher order complicated problems. In order to improve PSO, so far numerous variants have been introduced. Compared to the conventional PSO, most of these variants show relatively better performance. Much research is still in progress for improvement of the performance of PSO especially in complicated problems. In this study, we present a simple method for PID controller parameter tuning by using the chaos embedded particle swarm optimization algorithm (CPSO).

PID parameter tuning achives in such a way that parameters are adjusted in real-time when the controller is active, which is called on-line tuning or the parameters are measured than the tuned controller is performed on a plant, which is called off-line tuning. Hence, in this study, the CPSO is applied to overall system to obtained the design objectives by adjusting the controller parameters at each iteration repetitively until the desired closed-loop system performance is achieved. This attitude is off-line tuning.

The performance of the closed-loop system can defined in terms of rise time, overshoot, settling time and steady state error. In general, the system with fast rise and settling time under no steady-state error and almost zero overshoot is desired. Hence, in this study to provide a desired performance, mean value of squared error signal, which is different between reference signal and the system output, is minimized by using CPSO and PSO.

These methods (CPSO, PSO and Ziegler-Nichols) are applied to three typical systems of different orders. The first system under investigation is the acceleration model of an object in a frictional environment. The second one is the inverted pendulum control problem, and the last issue is the field-controled of a DC motor problem. When these systems are controlled via PID controller, the closed-loop performance is sensitively dependent to the PID parameters. Hence, determination of these parameters becomes quite critical. Therefore, PID parameter tuning methods should be performed on the closed-loop system.

In Section 2 of the paper, the descriptions and formulations of the controller and problems under investigation are presented. Section 3 contains the definitions of the PSO and CPSO algorithms; and finally, Section 4 presents the numerical results and Section 5 contains the comparisons together with discussions and concluding remarks.

Table 1: Increase in each PID parameter and its impact on transient response

Parameter	Rise Time	Overshoot	Settling Time	Steady State Error
K_P	Decrease	Increase	Increase	Decrease
K_I	Decrease	Increase	Increase	Decrease
K_D	Decrease	Decrease	Decrease	Slight Change

2 Problem Statement

In this study, PID control is discussed and three systems of different orders are investigated for the performance comparison of the proposed CPSO method with the Ziegler-Nichols method and conventional PSO. Each system is summarized via pictorial description as well as its transfer function in the upcoming subsections.

2.1 The Proportional-Integral-Derivative (PID) Control

The basic form of the PID controller composes from sum of the multiplication, integration and differentiation of the error signal, and each operator is multiplied by three control parameters. Time domain representation of PID control equation is presented in Eq. 1, meanwhile the transfer function is given in Eq. 2:

$$u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de(t)}{dt}, \tag{1}$$

$$G(s) = \frac{U(s)}{E(s)} = \frac{K_D s^2 + K_p s + K_I}{s}, \tag{2}$$

where K_P, K_I and K_D are the control parameters, $u(t)$ is the controller output/system input, and $e(t)$ is error signal. In frequency domain, the transfer function of the PID controller has one pole at the origin, and two zeros with locations depending on the tuning strategy.

Table 1 presents the separate and isolated impact of change in each parameter on the system performance. However, in reality, the parameters must be mutually tuned. Therefore, these impacts listed in Table 1 should not be considered as rules-of-thumb, and applied directly for parameter tuning.

2.2 A System of First Order: Acceleration Control of an Object in a Frictional Environment

The first order system in this study results from the acceleration of an object against the friction due to the ground contact, which is one of the fundamental models in vehicular control. The Figure 1 depicts the pictorial description of the relevant system.

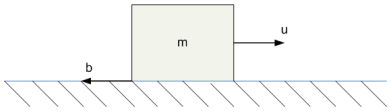

Figure 1: Accelerated object model in a frictional environment.

The transfer function of the system is given in Eq. 3. The model presents the equation of motion for the speed of an object with mass m. Assuming that the input u imposes a force;

Table 2: Setup1a and Setup1b descriptors

Descriptor	Symbol	Setup1a	Setup1b
Friction constant (N.sec/m)	b	50	50
Mass of the object (kg)	m	100	50
Desired velocity of the object (m/sec)	v	1	1

meanwhile the friction force bv restricts the movement of the object, where v is the velocity of the object as well as the output of the overall system. The aim of the controller is to make the object to reach to a desired velocity.

$$\frac{V(s)}{U(s)} = \frac{1/m}{s + \frac{b}{m}} \tag{3}$$

For this particular type of problem, two different setups are constructed and investigated. From now on, these setups will be referred to as Setup1a and Setup1b. The descriptor parameters of these setups are given in Table 2.

2.3 A System of Second Order: Inverted Pendulum

The inverted pendulum is one of the benchmark problems of control theory for illustration and comparison of different control methodologies. The problem comes from the motivation on development of missiles, rockets, robots and other transportation means. In general, the aim in the inverted pendulum problem is to maintain the pendulum at upright position. Therefore, in addition to the PID control, various techniques (such as optimal control [15], linear [9] control, nonlinear control [5], intelligent and adaptive control methods [3]) have also been applied to this problem.

Figure 4 shows the general model of the inverted pendulum. The pendulum is composed of a free moving pendulum with the mass m_p and length l attached to the cart with mass m_t, where a force f is applied to this setup.

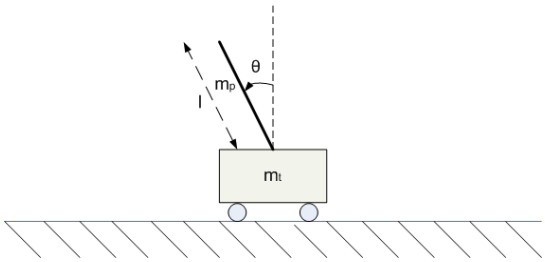

Figure 2: Inverted Pendulum Model.

Eq. 4 shows the transfer function of the inverted pendulum system. The moment of inertia I about the pendulum's mass center is defined as $I = (1/3)m_p l^2$.

$$\frac{\theta(s)}{U(s)} = \frac{m_p l}{((I + m_p l^2) - m_p^2 l^2)s^2 - m_p g l(m_p + m_t)}. \tag{4}$$

For this particular type of problem, four different setups are constructed and investigated. From now on, these setups will be referred to as Setup2a, Setup2b, Setup2c and Setup2d. The descriptor parameters of these setups are given in Table 3.

Table 3: Setup2a - Setup2d descriptors

Descriptor	Symbol	Setup2a	Setup2b	Setup2c	Setup2d
Mass of the cart (kg)	m_t	0.5	0.5	1	1
Mass of the pole (kg)	m_p	1	1	1	1
Length of the pole (m)	l	0.5	0.5	2	2
Initial value of the rod angle (rad)	θ	0.4	0.8	0.4	0.8

Table 4: Setup3a and Setup3b descriptors

Descriptor	Symbol	Setup3a	Setup3b
Motor constant (N.m/A)	k_m	50.10^{-3}	50.10^{-3}
Friction constant (N.sec/m)	b	50.10^{-3}	50.10^{-3}
Resistance (Ω)	R_f	1	10
Field time effect (msec)	τ_f	1	10
Rotor time constant (msec)	τ_l	100	10
Rotor angle (rad)	θ	2	2

2.4 A System of Third Order: Field-Controlled DC Motor

The third problem under investigation is the field-controlled DC motor problem depicted in Figure 1, which has a third order transfer function. The main purpose of this type of systems are to increase speed while reduce the torque. Eq. 5 gives the transfer function of the system.

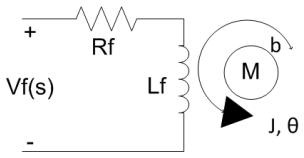

Figure 3: Field-controlled DC motor model.

$$\frac{\theta(s)}{v_f(s)} = \frac{k_m/(bR_f)}{s(\tau_f s+1)(\tau_l s+1)}, \tag{5}$$

where km is the motor constant, b is the friction constant, $\tau_f = L_f/R_f$ is the field time effect, $\tau_l = J/b$ is the rotor time constant, and J is the rotor inertia, v_f is the input and θ is the output of the system.

For this particular type of problem, two different setups are constructed and investigated. From now on, these setups will be referred to as Setup3a and Setup3b. The descriptor parameters of these setups are given in Table 4.

3 Particle Swarm Optimization (PSO)

The PSO algorithm depends on motions of particles (swarm members) searching for the global best in an N-dimensional continuous space. The position of each particle is nothing but a solution candidate, and every time, the fitness of this candidate is re-evaluated. In addition

to its exploration capability (i.e. the tendency for random search throughout the domain), each particle has a cognitive behavior (i.e. remembering its own good memories, and having tendency to return there); as well as a social behavior (i.e. observing the rest of the swarm and having tendency to go where most other particles go).

The original PSO formulation of Kennedy and Eberhart [7] depends on the update of the position $x_i[k]$ and the velocity $v_i[k]$ of the ith particle (swarm member) at the kth iteration as follows:

$$v_i[k+1] = v_i[k] + c_1 \times rand() \times (pbest_i - x_i[k]) + c_2 \times rand() \times (gbest_i - x_i[k]), \quad (6)$$

$$x_i[k+1] = x_i[k] + v_i[k+1] \times \Delta t, \quad (7)$$

where c_1 and c_2 are measures indicating the tendencies of approaching to *pbest* and *gbest*, which are the best positions achieved personally by the ith particle and the whole swarm, respectively. In other words, c_1 and c_2 are the measures of the cognitive and the social behaviors (called cognitive and social parameters), respectively. $rand()$ is a uniformly distributed pseudo-random number generator which produces random numbers between 0.0 and 1.0; and the time step size Δt is taken to be unity for simplicity.

This optimization algorithm demonstrates an outstanding performance under complicated problems. But still, it occasionally faces problems such as getting stuck at local optima and stagnation for multi-dimensional and complex problems. Thus, various improvements have been proposed in order to get rid of these problems. One of the important innovations was introduced by Shi and Eberhart [11], who proposed a term called "inertial weight" in order to improve the performance of the method (used for controlling local and global exploration behavior of the population). By introduction of this term, which puts an additional control on the current velocity of the particles, Eq. 6 is modified as:

$$v_i[k+1] = w[k] \times v_i[k] + c_1 \times rand() \times (pbest_i - x_i[k]) + c_2 \times rand() \times (gbest_i - x_i[k]), \quad (8)$$

which is referred to as the inertial weight PSO, and is currently accepted as the de-facto PSO formulation. Moreover, in a following study [11], Shi and Eberhart showed that the ideal choice for the inertial weight is to decrease it linearly from 0.9 to 0.4. The following pseudocode presents the PSO algorithm.

```
PSO Algorithm
initialize random velocity and position
do
  for i = 1 to swarm size
        1) Calculate fitness function (fit) aimed to be minimized.
        This is the function that the mean squre error of the
        difference between reference signal and the output of the
        system
        if fit < best pattern
             ith particle best position = ith particle position
             best pattern = fit
        end if
  end for
  find min best pattern and corresponding particle
  for i = 1 to swarm size
        Update velocity
```

```
      Update position
  end for
  check the limits of the maximum velocity
while maximum iterations are not exceeded or minimum error is not achieved
```

4 Chaos Embedded Particle Swarm Optimization (CPSO)

After PSO has been presented, various hybrid and innovative methods were introduced in order to get rid of its drawbacks such as premature convergence and stagnation for multi-dimensional and complicated problems. One of the ideas for preventing premature convergence is to embed chaotic maps into PSO. In recent years, chaotic maps have been utilized for generating pseudo-random numbers, and they have been applied to many areas like communication and control theory, since they can produce sufficiently random numbers enhancing the overall performance of the systems. For PSO, the chaotic maps can be helpful for prevention of premature convergence [2]; moreover, it can improve the global/local searching capabilities of the algorithm. Implementation of chaotic maps in PSO can be categorized into two different approaches:

In the first approach, variables and/or random number generators are reinforced with the chaotic maps without any radical changes in the algorithm. In a similar manner, any stochastic algorithm can be improved by using chaotic maps. For PSO, the performance of the algorithm is greatly dependent on the parameters w, c_1, c_2 as well as the random number generators. Thus, any improvement performed on these variables would slightly have influence on the overall performance.

In the second approach, chaos is used in order to interact with the PSO algorithm for searching the solution space. The modification in the PSO algorithm must be fulfilled. In addition to the conventional algorithm, the add-on code is evaluated in order to find a solution candidate with better fitness. At each iteration, the search space boundary is reduced for increasing the searching efficiency. By this way, the particles are kept away from local optimum. This phenomenon is called Chaos Search. In this study, due to its effective performance, the second approach is preferred to the first approach.

The Chaos Embedded Particle Swarm Optimization (CPSO) [8] differs from the conventional PSO with four new operators, which are: chaotic map, variable mapping, inverse variable mapping, and search range operators.

4.1 Chaotic Maps

The discrete-time dynamical system in the iteration form in Eq. 9 is called chaotic mapping or chaotic map.

$$cx_{i+1} = F(cx_i, P), \tag{9}$$

where $F : S \to S$ and $S \in \Re$, $S = [0, 1]$ or $S = [-1, 1]$. P is the chaos parameter, cx is a vector and F is a nonlinear transformation. The equation starting from the initial value cx_0 in the iterative map is obtained. Chaotic maps have sensitivity on initial value and they produce pseudo - random sequences based on cx_0.

One of the most common chaotic maps is the logistic map. The logistic map which is a model of population biology, is frequently used with PSO [8].

$$cx_{i+1}[k] = \mu cx_i[k](1 - cx_i[k]), \tag{10}$$

where μ (which is called bifurcation parameter) is set to 4 for ergodicity as seen in Figure 4. There are two fixed points (If the condition $cx^* = F(cx^*, P)$ holds cx^* is called fixed point) exists

at 3/4 and 0. Moreover, if the chaotic map will be used with CPSO, the fixed points must be checked at the algorithm.

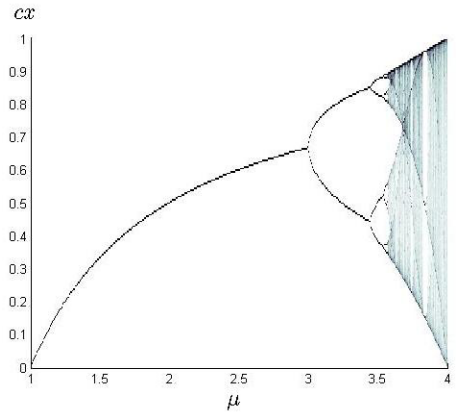

Figure 4: Bifurcation Diagram of Logistic Map.

The chaotic map is a function producing a random number based on the initial value. In CPSO, the initial value is the current position for a particle and chaotic map produces a new position. If the fitness value related to the new position is optimal than the initial position, then the position of a particle is changed as the new position. Thus, it is essential that the chaotic orbit must be ergodic, which means the initial value of the chaotic map must be varied. However, the range of the particle position is not between $[-1, 1]$ or $[0, 1]$. Hence, the particle position should be mapped into chaotic space. Therefore, the variable mapping operator is defined for this purpose.

4.2 Variable Mapping

The position of the particle should be mapped into the chaotic domain by using carrier equation as defined in Eq. 11 [14].

$$cx_i[k] = (x_i[k] - X_{min})/(X_{max} - X_{min}), \tag{11}$$

where cx_i is decision variable, which is the initial value of the chaotic map, x is the position of the particle and X_{min} and X_{max} are the boundaries of the search space. This function maps the range $[X_{min}, X_{max}]$ to the range of chaotic variable $[0, 1]$.

The procedure; variable mapping a particle position and then using this position as an initial value of a chaotic map is called Chaos Search. However, when the optimal point is obtained, it should be mapped to a search space. Hence, the inverse mapping operator is produced.

4.3 Inverse Variable Mapping

After determination of the chaotic variable from a chaotic map with the initial value from variable mapping, the chaotic variable is converted into the particle position by using Eq. 12. This function maps the range of chaotic variable $[0, 1]$ to the range of solution space $[X_{min}, X_{max}]$.

$$x_i[k] = X_{min} + cx_i[k](X_{max} - X_{min}) \tag{12}$$

The position, which is taken from the inverse mapping, is used in order to evaluate the new solutions. If the new solution is better than the non-chaos-search one, the new solution is put pursuant to chaos search.

4.4 Search Range

If the search space extends in a wide area, the searching cannot be completed in the optimal area in a short time. Hence, in order to obtain high performance, the chaotic search is run in a small range. This search area is changed in the current optimal solution neighborhood [4].

$$X_{\min} = \max(X_{\min}, x_g - r(X_{\max} - X_{\min})), \tag{13}$$

$$X_{\max} = \min(X_{\max}, x_g + r(X_{\max} - X_{\min})), \tag{14}$$

where x_g is the global best position so far and r is the variable between 0 and 1. In this study, r is chosen as 0.45, and x_g is chosen as 0 for simplicity. Decreasing the range of the chaotic search will increase the searching efficiency.

The outline of the CPSO used in this study can be listed as follows:

```
Chaos PSO Algorithm
initialize random velocity and position
do
  for i = 1 to swarm size
        1) Calculate the fitness function (fit) aimed to be minimized.
        This function is nothing but the mean square error of the
        difference between reference signal and the output of the
        system
        if fit < best pattern
             ith particle best position = ith particle position
             best pattern = fit
        end if
  end for
  find min best pattern and corresponding particle
  for i = 1 to swarm size
        Update velocity
        Update position
  end for
  check the limits of the maximum velocity
     1) Execute variable mapping that maps the positions into chaos
        variables
     (Section 4.2)
     2) Use Logistic map and find new chaos variable
     (Section 4.1)
     3) Execute Inverse Variable Mapping that converts chaos
     variable into positions
     (Section 4.3)
     if (new position < old position)
         Position = New Position
         break
     end if
     4) Change the search area
     (Section 4.4)
  end for
while maximum iterations are not exceeded or minimum error is not achieved
```

Table 5: Optimization algorithm parameters for the simulation

Property	Value
Number of particles in population	10
Number of iterations ($maxiter$)	50
Social and cognitive parameters (c_1 and c_2)	1.494
Inertia weight (w_{max} and w_{min})	0.9 down to 0.4

Table 6: PID controller parameter values for Setup1a

Parameter	Value Obtained by CPSO	Value Obtained by PSO	Value Obtained by Ziegler-Nichols
K_P	100	22.64	30.52
K_I	38.745	100	109.01
K_D	1	0.1	30.52

5 Simulation Results

In this study, a simulation environment is constructed in order to apply the tuning methods and make performance comparisons. Figure 5 shows the overall setups for all cases; and the selected parameters for the optimization algorithms are given in Table 5.

Figure 5: Overall Setups for all Cases.

The brief summary of the solution strategy to the control problems in this study is as follows: 1) initialize the overall system with random PID parameters; 2) obtain the error, which is the transient response of the system where the reference signal is zero; 3) calculate the fitness function; 4) update the parameters by using Ziegler-Nichols, PSO or CPSO; 5) return to step 2 until the maximum iteration is achieved.

5.1 Results for Setup1

For this type of problem, the PID parameters are tuned by using PSO, CPSO and Zeigler-Nichols methods. Figure 6 shows the transient responses of the closed loop system for Setup1a and Setup1b. Table 6 and 7 present the obtained PID parameters for Setup1a and Setup1b by means of different methods.

It can be observed from the figure that for Setup1a, PSO yields the worst PID parameters. CPSO has no overshoot and better rise time compared to PSO and Ziegler-Nichols. For Setup1b, Ziegler-Nichols yields the worst performance. Even though CPSO seems to outperform to PSO, it can be concluded that CPSO performance is almost the same with PSO.

In summary, for this particular type of problem, CPSO presents better performance compared to PSO and Ziegler-Nichols.

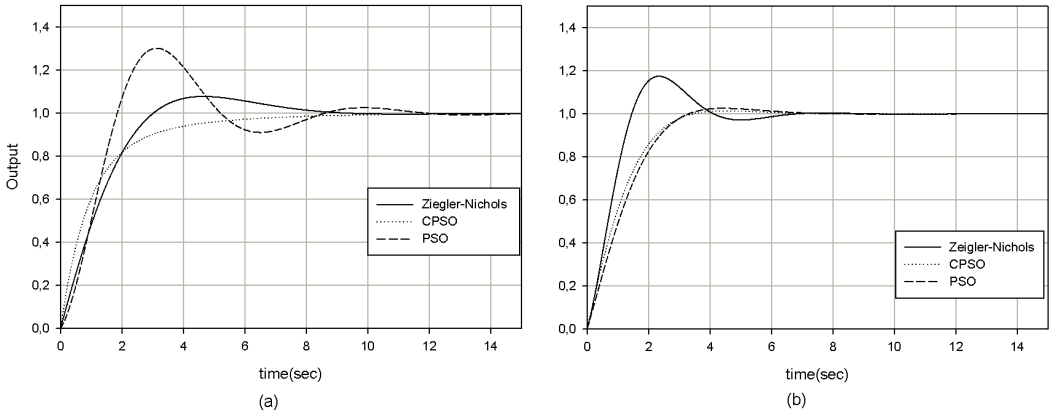

Figure 6: Transient response of the system for a) Setup1a and b) Setup1b.

Table 7: PID controller parameter values Setup1b

Parameter	Value Obtained by CPSO	Value Obtained by PSO	Value Obtained by Ziegler-Nichols
K_P	33.86	24.89	0.85
K_I	49.3	45.57	43.6
K_D	1	1	0.85

5.2 Results for Setup2

For this type of problem, again, the PID parameters are tuned by using PSO, CPSO and Ziegler-Nichols methods. Tables 8 and 9 give the obtained PID parameters, and Figure 3 shows the transient responses of designs.

The results as seen in Figure 3 indicates that CPSO outperforms to PSO and Ziegler-Nichols. However, in Case 2a and Case 2b CPSO performance is similar to that of Ziegler-Nichols and PSO. From the simulations, it is observed that, CPSO is better than PSO; on the other hand, it should be noted that for this particular type of problem, Ziegler-Nichols outperforms to conventional PSO. This result emphasizes the importance of the efforts spent for improvement of the conventional PSO.

5.3 Results for Setup3

The PID parameters obtained for this type of problem are presented in Tables 10 and 11, and illustrated in Figure 8.

Table 8: PID controller parameter values for Setup2a and Setup2b

Parameter	Value Obtained by CPSO	Value Obtained by PSO	Value Obtained by Ziegler-Nichols
K_P	93.6226	60.0136	60.3788
K_I	10.4097	0	4.997
K_D	10.1	3.6553	4.977

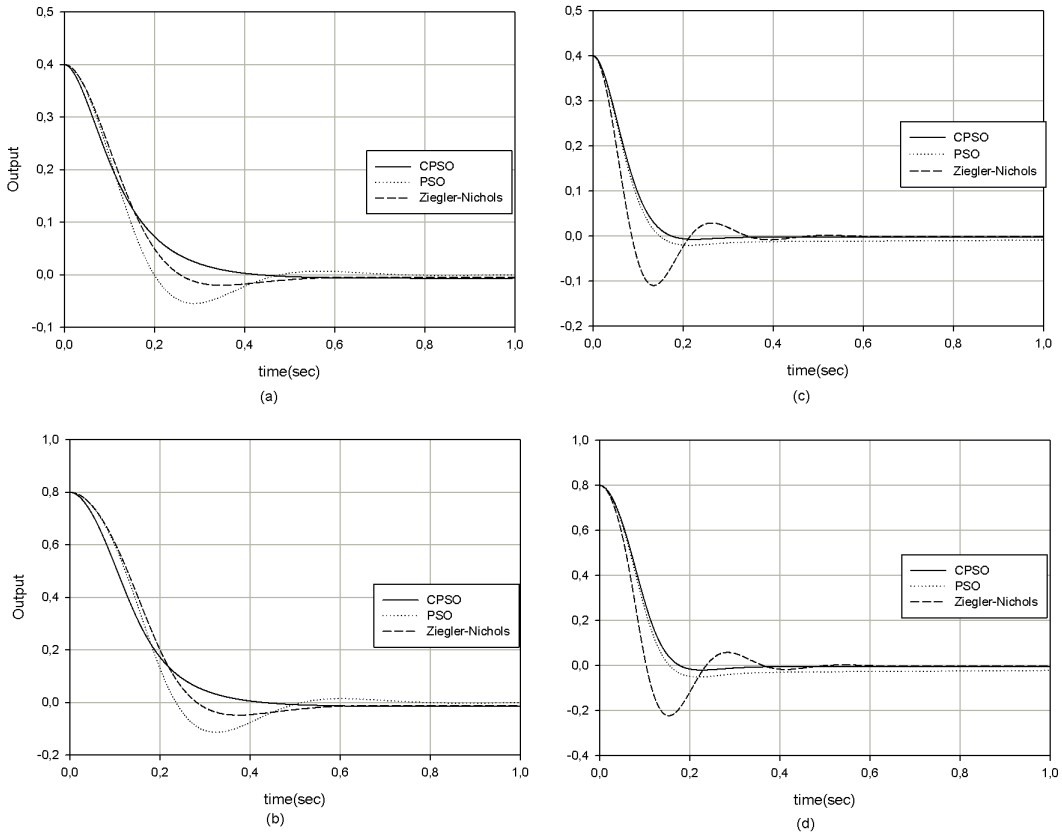

Figure 7: Transient response of the system for a) Setup2a, b) Setup2b, c) Setup2c and d) Setup2d.

Table 9: PID controller parameter values for the Setup2c and Setup2d

Parameter	Value Obtained by CPSO	Value Obtained by PSO	Value Obtained by Ziegler-Nichols
K_P	162.9	169.7	190
K_I	10	79.03	4.9
K_D	10	10	5.12

Table 10: PID controller parameter values for Setup3a

Parameter	Value Obtained by CPSO	Value Obtained by PSO	Value Obtained by Ziegler-Nichols
K_P	395.33	219.20533	1.08
K_I	1.1	55.2488	0.0314
K_D	41.6245	1	6.22

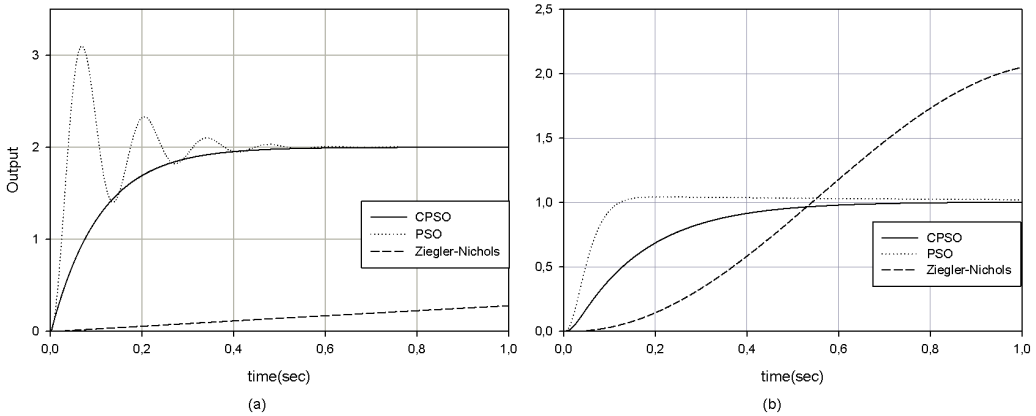

Figure 8: Transient response of the system for a) Setup3a and b) Setup3b.

Table 11: PID controller parameter values for the Setup3b

Parameter	Value Obtained by CPSO	Value Obtained by PSO	Value Obtained by Ziegler-Nichols
K_P	28.73	62.7446	0.2442
K_I	186.95	1.7	87.205
K_D	8.7	1	$1.7\ 10^{-4}$

The results once more indicate the efficiency of the CPSO algorithm. At this point, the following remark shall be made: Even though for Setup3b PSO seems to have a better rise time as well as a settling time compared to CPSO, it yields a steady state error. Therefore, it can still be claimed that CPSO has a better overall performance compared to PSO and Ziegler-Nichols.

6 Conclusion and Future Works

In this study, the chaos embedded particle swarm optimization algorithm (CPSO) is developed, and applied to the some popular engineering problems. The parameter tuning of the PID controller is performed by applying CPSO to three systems. At each iteration, the PID parameters are found off-line; then, the optimized controller is applied to the system. The effectiveness of the proposed method is exposed by comparing it to PSO and Ziegler-Nichols methods. It is observed and concluded that CPSO outperforms to PSO and Ziegler-Nichols, and it is quite efficient for usage in other systems. The scope of the current study is limited to off-line determination of the PID parameters; but simple and parallelizable structure of PSO (and also of CPSO) gives the impression that it is feasible to have real-time implementations of PSO (and also CPSO) in industrial applications for on-line PID controller parameter tuning.

Bibliography

[1] L.S. Coelho, A novel quantum particle swarm optimizer with chaotic mutation operator, *Chaos, Solitons and Fractals*, Vol.37, pp. 1409-1418, 2008

[2] L.S. Coelho, V.C. Mariani, A novel chaotic particle swarm optimization approach using Henon map and implicit filtering local search for economic load dispatch, *Chaos, Solitons and Fractals*, Vol.39, pp. 510-518, 2009

[3] S. Cong and Y. Liang, PID-Like neural network nonlinear adaptive control for uncertain multivariable motion control systems, *IEEE Transactions on Industrial Electronics*, 56(10):3872-3879, 2009

[4] X.Y. Gao, L.Q. Sun, D.S. Sun, An enhanced particle swarm optimization algorithm, *Information Technology Journal*, Vol.8, pp. 1263-1268, 2009

[5] H.N. Iordanou, B.W. Surgenor, Experimental evaluation of the robustness of discrete sliding mode control versus linear quadratic control, *IEEE Transactions on Control Systems Technology*, 5(2):254-260, 1997

[6] C. Jiejin, M. Xiaoqian, L. Lixiang, P. Haipeng, Chaotic particle swarm optimization for economic dispatch considering the generator constraints, *Energy Conversion and Management*, Vol.48, pp. 645-653, 2007

[7] J. Kennedy, R.C. Eberhart, Particle swarm optimization, *IEEE International Conference on Neural Networks*, pp. 1942-1948, 1995

[8] B. Liu, L. Wang, Y.H. Jin, F. Tang, C.X. Huang, Improved particle swarm optimization combined with chaos, *Chaos, Solitons and Fractals*, 25, pp. 1261-1271, 2005

[9] G.A. Medrano-Cersa, Robust computer control of an inverted pendulum, *IEEE Control Systems Magazine*, 19(3):58-67, 1999

[10] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, *IEEE International Conference on Evolutionary Computation*, pp. 69-73, 1998

[11] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, *Congress of Evolutionary Computing*, pp. 1945-1950, 1999

[12] Y. Song, Z. Chen, Z. Yuan, New chaotic PSO-based neural network predictive control for nonlinear process, *IEEE Transactions on Neural Networks*, 18(2):595-601, 2007

[13] J.M.T. Thompson, H.B. Stewart, *Nonlinear Dynamics and Chaos*, John Wiley and Sons, 2nd Edition, 2002

[14] T. Xiang, X. Liao, K.W. Wang, An improved particle swarm optimization combined with piecewise linear chaotic map, *Applied Mathematics and Computation*, pp. 1637-1645, 2007

[15] G.W. van der Linder, P.F. Lambrechts, H-inf control of an experimental inverted pendulum with dry friction, *IEEE Control Systems Magazine*, 13(4):44-50, 1993