Tuning of Optimum PID Controller Parameter Using Particle Swarm Optimization Algorithm Approach

Wan Azhar Wan Yusoff
Nafrizuan Mat Yahya
Azlyna Senawi
Fakulti Kejuruteraan Mekanikal,
Universiti Malaysia Pahang,
Karung Berkunci 12, 25000 Kuantan, Pahang.

Abstract: In this paper, an artificial intelligence method, Particle Swarm Optimization (PSO) algorithm is presented for determining the optimal proportional-integral-derivative (PID) controller parameters of a typical servo motion system. This paper demonstrates in detail on how to employ the PSO method to search efficiently the optimal PID controller parameters of a typical servo motion system. In order to assist estimating the performance of the proposed PSO-PID controller, a new time-domain performance criterion function has been used. The proposed approach yields better solution in term of rise time, settling time, maximum overshoot and steady state error condition of the system. Compared to conventional Ziegler – Nichols method, the proposed method was indeed more efficient and robust in improving the step response of a typical servo motion system.

INTRODUCTION

Even in a decade where advanced control algorithms mostly based on some kind of optimization procedure have achieved a high degree of maturity, Proportional Integral Derivative (PID) controllers are still widely used in industrial applications even though many new control techniques have been proposed [1-2]. The reason is that it has a simple structure which is easy to be understood by the engineers, and under practical conditions, it has been performing more reliably compared to more advanced and complex controllers [3-4]. The main propose of designing a PID controller is to determine the three gains and they are proportional gain (k_p), integral gain (k_i) and derivative gain (k_d) of the controller [2]. However, the three adjustable PID controller parameters should be tuned appropriately [1].

Over the years, several heuristic methods have been developed for the tuning of PID controllers. The first method used the classical tuning rules proposed by Ziegler and Nichols [5]. Generally, it is always hard to determine optimal or almost optimal PID parameters with the Ziegler-Nichols method in many industrial plants [5]. Other than original works done by Ziegler and Nichols, a great number of methods have been proposed to obtain optimal gains of the PID such as by Cohen and Coon in 1953, Åström and Hägglund in 1984 or by Zhuang and Atherton in 1993 [1].

To obtain the optimal parameter tuning, it is highly desirable to increase the capabilities of PID controllers by adding new features. Most in common, artificial intelligence (AI) techniques have been employed to improve the controller performances for a wide range of plants while retaining the basic characteristics. AI techniques such as artificial neural network, fuzzy system and neural-fuzzy logic have been widely applied in order to get proper tuning of PID controller parameters [6-10].

Recently, a new evolutionary technique, Particle Swarm Optimization (PSO) was first introduced in 1995 by Kennedy and Eberhart for unconstrained continuous optimization problems [11-12]. Its development was based on observations of the social behavior of animals such as bird flocking, fish schooling and swarm theory. The PSO is initialized with a population of random solutions. The PSO has some attractive characteristics where it has memory and therefore, knowledge of good solutions is retained by all particles. There exist constructive cooperation between particles where particles in the swarm share information between them. The theoretical framework of PSO is very simple and PSO is easy to be coded and implemented using computer [11]. In fact, the PSO technique can generate a high quality solution within shorter calculation time and stable convergence characteristics than other stochastic methods [13]. Thus, this technique has gained much attention and wide applications in various fields recently [14-20].

The rest of the paper is organized as follows. In topic follows, a brief discussion about PID controller, basic PID servo motion system and performance estimation of PID controller are presented. Next, the PSO method and its implementation into the PSO-PID controller are viewed in detail. Further, the simulation results are presented in table form and discussed. Finally, the discussion of the results followed by conclusion of the research is provided.

PID CONTROLLER [5, 21]

The PID controller is used to improve the dynamic response as well as to reduce or eliminate the steady-state error. The derivative controller adds a finite zero to the open-loop plant transfer function and improves the transient response. The integral controller adds a pole at the origin, thus increasing system type by one and reducing the steady state-error due to a step function to zero.

The continuous form of a PID controller, with input $e(\cdot)$ and output $u_{pid}(\cdot)$, is generally given as :

$$u_{pid}(t) = k_p\left[e(t) + \frac{1}{T_i}\int_t^t e(\tau)\,d\tau + T_d\frac{d}{dt}e(t)\right] \quad (1)$$

where k_p is the proportional gain, T_i is integral time constant and T_d is the derivative time constant. We can also rewrite as

$$u_{pid}(t) = k_p e(t) + k_i\int_0^t e(\tau)\,d\tau + k_d\frac{d}{dt}e(t) \quad (2)$$

where $k_i = k_p / T_i$ is the integral gain and $k_d = k_p T_d$ is the derivative gain.

In simple form, the PID controller transfer function is

$$C(s) = k_p + \frac{k_i}{s} + k_d s \quad (3)$$

BASIC PID SERVO MOTION SYSTEM [22]

The basic components of a typical servo motion system are depicted in Fig. 1. According to this figure, the servo drive closed a current loop and is modeled simply as a linear transfer function *G*(*s*). In their most basic form, servo drives receive a voltage command that represents a desired motor current. Motor shaft torque, *T* is related to motor current, *I* by the torque constant, K_τ as show in (4).

$$T \approx K_\tau I \quad (4)$$

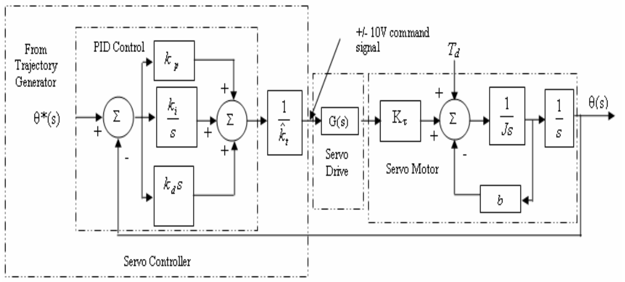

Fig. 1: Basic P.I.D Servo Control Topology [22]

For the purposes of this discussion, the transfer function of the current regulator or really the torque regulator can be approximated as unity for the relatively lower motion frequencies that are interested in and therefore the following approximation as shown in (5) has been made.

$$G(s) \approx 1 \quad (5)$$

The servomotor is modeled as a lump inertia, *J*, a viscous damping term, *b*, and a torque constant, K_τ. The lump inertia term is comprised of both the servomotor and load inertia. It is also assumed that the load is rigidly coupled such that the torsional rigidity moves the natural mechanical resonance point well out beyond the servo controller's bandwidth. This assumption allows us to model the total system inertia as the sum of the motor and load inertia for the frequencies that will be control.

The actual motor position, *θ(s)* is usually measured either by an encoder or resolver coupled directly to the motor shaft. Again, the underlying assumption is that the feedback device is rigidly mounted such that its mechanical resonant frequencies can be safely ignored. External shaft torque disturbances, T_d are added to the torque generated by the motor's current to give the torque available to accelerate the total inertia, *J*.

Around the servo drive and motor block is the servo controller that closes the position loop. A basic servo controller generally contains both a trajectory generator and PID controller. The trajectory generator typically provides only position set-point commands labeled in Fig. 1 as *θ*(s)*. The PID controller operates on the position error and outputs a torque command that is sometimes scaled by an estimate of the motor's torque constant, $\hat{K}_t$. If the motor's torque constant is not known, the PID gains are simply re-scaled accordingly. Due to the exact value of the motor's torque constant is generally not known, the symbol "^" is used to indicate it is an estimated value in controller. In general, equation (6) holds with sufficient accuracy so that the output of the servo controller (usually +/- 10 Volts) will command the correct amount of current for a desired torque.

$$\hat{K}_t \approx K_t \quad (6)$$

There are three gains to adjust in the PID k_p, k_i and k_d. These gains all act on the position error defined as (7). The output of the PID controller is a torque signal.

$$error\ (t) = \theta^*(t) - \theta(t) \quad (7)$$

PERFORMANCE ESTIMATION OF PID CONTROLLER [5, 23]

Controller design attempts to minimize the system error produced by certain anticipated inputs. The system error is defined as the difference between the desired response of the system and its actual response. Performance criteria are mainly based on measures of the system error. Basically, PID controller design method using criterion as tabulate in TABLE I.

TABLE I
PERFORMANCE ESTIMATION OF PID CONTROLLER [23]

Name of Criterion	Formula
Integral of the Absolute Error (IAE)	$IAE = \int_0^\infty \lvert e(t) \rvert \, dt$
Integral of Square Error (ISE)	$ISE = \int_0^\infty e(t)^2 \, dt$
Integral of Time-weighted Square Error (ITSE)	$ITSE = \int_0^\infty t e(t)^2 \, dt$
Integral of Time-weighted Absolute Error (ITAE)	$ITAE = \int_0^\infty t \lvert e(t) \rvert \, dt$

A disadvantage of the IAE and ISE criteria is that its minimization can result in a response with relatively small overshoot but a long settling time because IAE and ISE performance criterion weights all errors equally independent of time. Although the ITSE performance criterion weights errors with time, the derivation processes of the analytical formula are complex and time consuming.

In this paper, the performance criterion in the time domain was used as proposed by [5] for evaluating the PID controller. A step of good control parameters k_p, k_i and k_d can yield a good step response that will result in performance criteria minimization in the time domain. These performance criteria in the time domain include the overshoot *Mp,* rise time, t_r, settling time, t_s and steady state error E_{ss}. Therefore, a new performance criterion *W(K)* is defined as [5]:

$$W(K) = \left(1 - e^{(-\beta)}\right) \bullet \left(M_p + E_{ss}\right) + e^{-\beta} \bullet \left(t_s - t_r\right) \tag{8}$$

where K is $[k_p, k_i, k_d]$ and β is the weighting factor.

The performance criterion *W(K)* can satisfy the designer requirements using the weighting factor, β value. β can set larger than 0.7 to reduce the overshoot and steady-state error or smaller than 0.7 to reduce the rise time and settling time.

OVERVIEW OF PARTICLE SWARM OPTIMIZATION [5, 11, 13, 24]

The particle swarm optimization (PSO) has been introduced by Kennedy and Eberhart in 1995. PSO is derived from the social-psychological theory, and has been found to be robust in complex systems. Each particle is treated as a valueless particle in g-dimensional search space, and keeps track of its coordinates in the problem space associated with the best solution (evaluating value) and this value is called *pbest.* The overall best value and its location obtained so far by any particle in the group that was tracked by the global version of the particle swarm optimizer *gbest*. The PSO concept consists of changing the velocity of each particle toward its *pbest* and *gbest* locations at each time step. As example, the *j*th particle is represented as $x_j = (x_{j,1}, xj_{,2}, \ldots, x_{j,g})$ in the g-dimensional space. The best previous position of the *j*th particle is recorded and represented as $pbest_j = (pbest_{j,1}, pbest_{j,2}, \ldots, pbest_{j,g})$. The index of best particle among all particles in the group is represented by the $gbest_g$. The rate of the position change (velocity) for particle *j* is represented as $v_j = (v_{j,1}, v_{j,2}, \ldots, v_{j,g})$. The modified velocity and position of each particle can be calculated using the current velocity and distance from $pbest_{j,g}$ to $gbest_g$ as shown in the following formulas:

$$v_{j,g}^{(t+1)} = w \bullet v_{j,g}^{(t)} + c_1 * rand(\ ) * \left(pbest_{j,g} - x_{j,g}^{(t)}\right) + c_2 * rand(\ ) * \left(gbest_g - x_{j,g}^{(t)}\right) \tag{9}$$

$$x_{j,g}^{(t+1)} = x_{j,g}^{(t)} + v_{j,g}^{(t+1)} \tag{10}$$

$j = 1, 2, \ldots, n$

$g = 1, 2, \ldots, m$

where

n	number of particles in a group;
m	number of members in a particle;
t	pointer of iterations(generations);
$v_{j,g}^{(t)}$	velocity of particle *j* at iteration *t,* $v_g^{\min} \le v_{j,g}^{(t)} \le v_g^{\max}$;
w	inertia weight factor;
c_1, c_2	acceleration constant;
rand( )	random number between 0 and 1;
$x_{j,g}^{(t)}$	current position of particle *j* at iteration *t*;
$pbest_j$	*pbest* of particle *j*;
gbest	*gbest* of the group

The parameter $v_g^{\max}$ determined the resolution, or fitness, with which regions were searched between the present position and the target position. If $v_g^{\max}$ is too high, particles might fly past good solutions but if $v_g^{\max}$ is too low, particles may not explore sufficiently beyond local solutions.

The constant C_1 and C_2 represent the weighting of the stochastic acceleration terms that pull each particle toward *pbest* and *gbest.* C_1 and C_2 were often set to be 2.0 according to past experience. This because low values allow particle to fly far from the target region before being tugged back while high values result in abrupt movement toward or past target regions.

Generally, the inertia weight w is set according to equation (11) below. Suitable selection of w provides a balance between global and local explorations, thus requiring less iteration on average to find a sufficiently optimal solution.

$$w = \frac{w_{\max} - w_{\min}}{iter_{\max}} \times iter \tag{11}$$

where $iter_{max}$ is the maximum number of iterations or generations and *iter* is the current number of iterations.

IMPLEMENTATION OF A PSO-PID CONTROLLER [5, 13]

The PID controller using the PSO algorithm was developed to improve the step transient response of typical servo motion system. It was also called the PSO-PID controller. The PSO algorithm was mainly utilized to determine three optimal controller parameters k_p, k_i, and k_d, such that the controlled system could obtain a good step response output.

In this paper, to apply the PSO method for searching the controller parameter, we use the "individual" to replace the "particle" and the "population" to define the "group". The three controller parameters k_p, k_i, and k_d, composed an individual K by $K \equiv [k_p, k_i, k_d]$; hence there are three members in an individual. These members are assigned as real values. If there are n individuals in a population, then the dimension of a population is n x 3. A set of good control parameters k_p, k_i, and k_d, can achieve a good step response and result in minimization of performance criteria in the time domain including the settling time (t_s) , rise time (t_r), maximum overshoot (M_p) and steady state error (E_{ss}). In the same time, we defined the evaluation value, f as in (12) which is reciprocal of the performance criterion $W(K)$ as in (8).

$$f = \frac{1}{W(K)} \tag{12}$$

It employs the smaller $W(K)$ the value of individual K, the higher its evaluation function.

In order to limit the evaluation value of each individual of the population within a reasonable range, the ***Routh-Hurwitz*** criterion must be utilized to test the closed-loop system stability before evaluating the evaluation value of an individual. The feasible individual and small value of $W(K)$ if the individual satisfied the ***Routh-Hurwitz*** criterion stability test applied to the characteristic equation of the system.

The searching procedures of the proposed PSO-PID controller were shown as follows [5, 24, 25]:

Step 1: Specify the lower and upper bounds of the three controller parameters and initialize randomly the individuals of the population including searching points, velocities, *pbests* and *gbest*.

Step 2: For each initial individual K of the population, employ the **Routh-Hurwitz** criterion to test the closed-loop system stability and calculate the values of the four performance criteria in the time domain, namely M_p, E_{ss}, t_r and t_s.

Step 3: Calculate the evaluation value of each individual in the population using the evaluation value, f given by (12).

Step 4: Compare evaluation value of each individual with its *pbest*. The best evaluation value among the *pbest* is denoted as *gbest.*

Step 5: Modify the member velocity v of each individual K according to (13)

$$\begin{aligned} v_{j,g}^{(t+1)} &= w \bullet v_{j,g}^{(t)} \\ &\quad + c_1 * rand\,() * \left(pbest_{\,j,g} - k_{j,g}^{(t)}\right) \\ &\quad + c_2 * rand\,() * \left(gbest_{\,g} - k_{j,g}^{(t)}\right) \\ &\quad j = 1, 2 \ldots n, \qquad g = 1, 2 \ldots 3 \end{aligned} \tag{13}$$

Step 6: If $v_{j,g}^{(t+1)} > V_g^{\max}$, then $v_{j,g}^{(t+1)} = V_g^{\max}$

If $v_{j,g}^{(t+1)} < V_g^{\max}$; then $v_{j,g}^{(t+1)} = V_g^{\max}$

Step 7: Modify the member position of each individual K according to (14)

$$k_{j,g}^{(t+1)} = k_{j,g}^{(t)} + v_{j,g}^{(t+1)} \tag{14}$$

such that $\quad k_g^{\min} \le k_{j,g}^{(t+1)} \le k_g^{\max}$

where $k_g^{\min}$ and $k_g^{\max}$ represent the lower and upper bounds, respectively, of member g of the individual K. For example, when g is 1, the lower and upper bounds of the k_p controller parameter are $k_g^{\min}$ and $k_g^{\max}$ respectively.

Step 8: If the number of iterations reaches the maximum then, go to **Step 9.** Otherwise, go to **Step 2**.

Step 9: The individual that generates the latest *gbest* is an optimal controller parameter.

SIMULATION EXAMPLES RESULTS

The simulation of the Typical Servo Motion System without PID controller, PSO-PID controller and Ziegler and Nichols-PID Controller were implemented by MATLAB Version 7.2 and executed on the Pentium 4 2.66GHz personal computer with 1GB RAM.

Typical Servo Motion System without PID Controller

The block diagram of the Typical Servo Motion System without PID is shown in Fig. 2 below. Result of this step response of the Typical Servo Motion System without PID controller has shown in following Fig. 3. To simulate this case, we found that performance criteria of the system in the time domain as the TABLE II below.

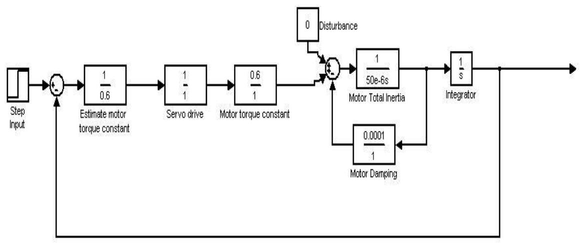

Fig. 2: Block Diagram of the Typical Servo Motion System without PID Controller

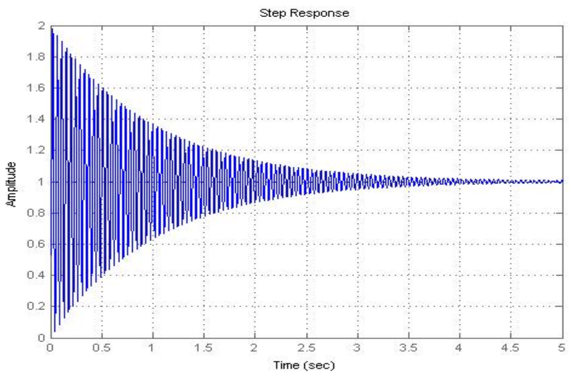

Fig. 3: Step response of the Typical Servo Motion System without PID controller

TABLE II
PERFORMANCE CRITERIA OF TYPICAL SERVO MOTION SYSTEM WITHOUT PID CONTROLLER

Performance Criteria	**Value**
Settling time (t_s)	3.91 s
Rise time (t_r)	0.0070 s
Maximum overshoot (M_p)	97.7578 %
Steady state error (E_{ss})	0

Typical Servo Motion System With PSO-PID Controller

According to the trial, the following PSO parameters are used for verifying the performance of the PSO-PID controller in searching the PID controller parameters:

- the member of each individual is k_p, k_i and k_d;
- population size = 25;
- inertia weight factor w is set by (11), where w_{max} = 0.9 and w_{min} = 0.4;
- Range of three controller parameter:
  - o k_p : minimum value = 0 , maximum value = 1.50;
  - o k_i : minimum value = 0 , maximum value = 1.00;
  - o k_d: minimum value = 0 , maximum value = 1.00;
- The limit of change in velocity :
  - o $V_{k_p}^{\max} = k_p^{\max} / 2$
  - o $V_{k_i}^{\max} = k_i^{\max} / 2$
  - o $V_{k_d}^{\max} = k_d^{\max} / 2$
- Acceleration constant C_1 and C_2 = 2

The block diagram of the Typical Servo Motion System with PSO-PID controller is shown in Fig. 4 below. Results of this step response of the Typical Servo Motion System with PSO-PID controller with β = 1.0 and β = 1.5 has shown in following Fig. 5 and Fig. 6 respectively. The simulation results that showed the best solution were summarized in the Table III below.

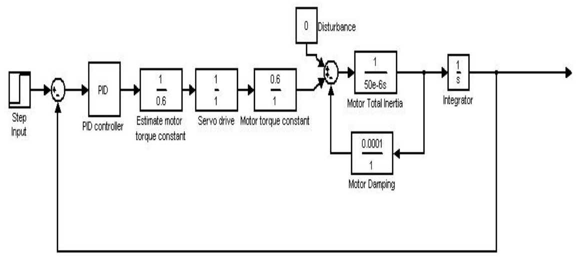

Fig. 4: Block Diagram of the Typical Servo Motion System with PSO-PID Controller

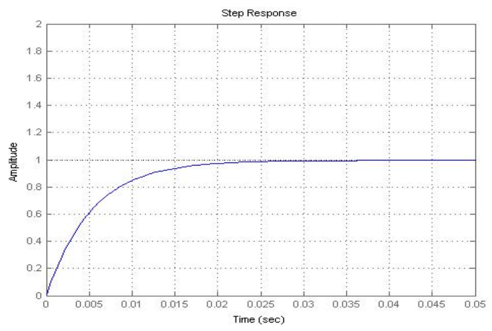

Fig. 5: Step response of the Typical Servo Motion System with PSO-PID controller with β =1.0

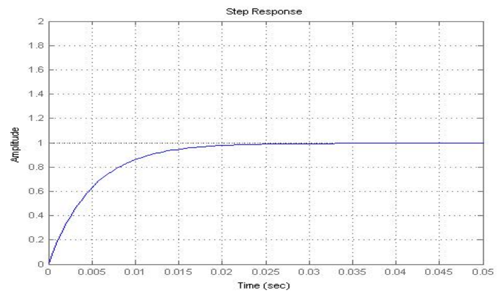

Fig. 6: Step response of the Typical Servo Motion System with PSO-PID controller with β = 1.5

TABLE III
BEST SOLUTION OF TYPICAL SERVO MOTION SYSTEM WITH PSO-PID CONTROLLER WITH THE DIFFERENT VALUES OF *B*

$\boldsymbol{\beta}$	1.0	1.5
Number of Iteration	25	25
$\boldsymbol{k_p}$	0.0092	0.0099
$\boldsymbol{k_i}$	0.0015	0.0017
$\boldsymbol{k_d}$	0.0095	0.0100
$\boldsymbol{t_r}$	0.0120	0.0110
$\boldsymbol{t_s}$	0.0220	0.0200
$\boldsymbol{M_p}$	0.1115	0.1084
$\boldsymbol{E_{ss}}$	0	0
Evaluation Value	13.3876	11.6415

As can be seen, through about 25 iterations or generations, the PSO method can prompt convergence and obtain good evaluation value, thus achieve better performance criterion that are rise time, settling time, percentage of overshoot and steady state error condition. These results show supremacy of the PSO-PID controller that can search optimal PID controller parameter quickly and efficiently.

Typical Servo Motion System with Ziegler and Nichols-PID Controller

In order to emphasize the advantages of the proposed PSO-PID controller, we also implemented the Ziegler and Nichols-PID Controller into the Typical Servo Motion System. The Ziegler and Nichols method basically boils down to these two steps (fundamental of servo motor):

Step 1: Set k_i and k_d to zero. Excite the system with a step command. Slowly increase k_p until the shaft position to oscillate. At this point, record the values of k_p and set k_o equal to this value. Record the oscillation frequency, f_o. Fig. 7 shows the result of slowly increasing on the proportional term. The system begins to oscillate at approximately 0.5Hz (f_o = 0.5Hz) with k_o approximately 5e^{-5} *Nm/rad*

Step 2: Set the final PID gains using equation (15)

$$k_p = 0.6\,k_0 \quad \textit{Nm/rad}\,;$$
$$k_i = 2\,f_o k_p \quad \textit{Nm/(rad.sec)}\,; \qquad (15)$$
$$k_d = \frac{k_p}{8 f_o} \quad \textit{Nm/(rad.sec)}\,;$$

Using the values got in *Step 1* into the equation (15), the optimum PID gains according Ziegler and Nichols method are then:

k_p = 3.0e^{-4} *Nm/rad*
k_i = 3.0e^{-4} *Nm/(rad.sec)*
k_d= 7.40e^{-5} *Nm/(rad.sec)*

Fig. 8 shows the result of Typical Servo Motion System with Ziegler and Nichols-PID Controller. Table IV summarize the performance criteria of the system in the time domain.

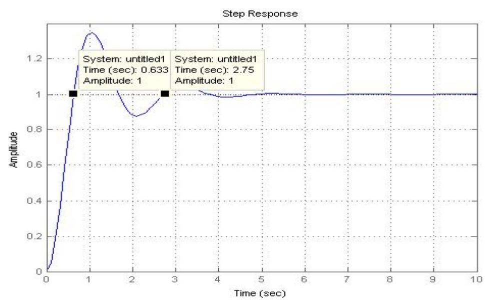

Fig. 7: Determining f_o and K_o of Ziegler and Nichols method

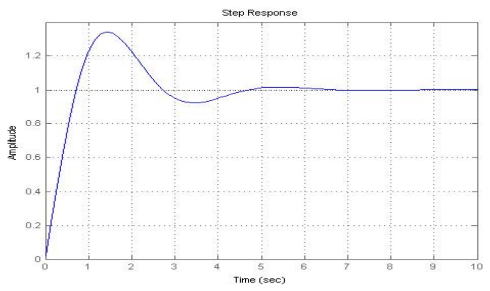

Fig. 8: Step response of the Typical Servo Motion System with Ziegler and Nichols-PID Controller

TABLE IV
PERFORMANCE CRITERIA OF TYPICAL SERVO MOTION SYSTEM WITH ZIEGLER AND NICHOLS-PID CONTROLLER

Performance Criteria	**Value**
Settling time (t_s)	4.3980 s
Rise time (t_r)	0.5160 s
Maximum overshoot (M_p)	34.2869 %
Steady state error (E_{ss})	0

DISCUSSION AND CONCLUSION

This paper presents a novel design method for determining the PID controller parameters using the PSO method. The proposed method integrates the PSO algorithm with the new time-domain performance criterion into a PSO-PID controller. Through the simulation of a typical servo motion system, the results show that the proposed controller can perform an efficient search to obtain optimal PID controller parameter that achieve better performance criterion that are rise time, settling time, percentage of overshoot and steady state error condition.

This PSO-PID controller has been compared Ziegler-Nichols PID method to verify it being more superior .The comparison from rise time show that the PSO-PID achieves less time that is 0.01*sec* compared to Ziegler-Nichols PID that need 0.52*sec*. The system using Ziegler-Nichols method takes about 4.4*sec* to finally settle between 2% of the final value making it very difficult to incorporate into any high performance motion control application. In contrast, the PSO method gives a quicker settling time that it takes only 0.02*sec* to settle. The PSO method instantaneously reduces the maximum overshoot of the system to only 0.1% compared to the Ziegler-Nichols method that recorded 34.3% of maximum overshoot. However, there is no steady state error for both methods.

Therefore, it is clear from the results that the proposed PSO method has more robust stability and efficiency and can solve the searching and tuning problems of PID controller parameters more easily and quickly than the Ziegler-Nichols method.

REFERENCES

[1] Pedret, C., Vilanova, R., Moreno, R. and Serra, I. (2002). A refinement procedure for PID controller tuning. *Computer and Chemical Engineering, Vol.26, 903-908.*

[2] Chang, W.-D., Hwang, R.-C. and Hsieh, J. –G. (2003). A multivariable on-line adaptive PID controller using auto-tuning neurons. *Engineering Applications of Artificial Intelligence, Vol.16, 57-63.*

[3] Tan, K.K., Huang, S. and Ferdous, R. (2002). Robust self-tuning PID controller for nonlinear systems. *Journal of Process Control, Vol.12, 753-761.*

[4] Valério, D. and Costa, J. S. (2006). Tuning of fractional PID controllers with Ziegler-Nichols-type rules. *Signal Processing, Vol. 86, 2771-2784.*

[5] Gaing, Z.L. (2004). A Particle Swarm Optimization approach for optimum design of PID controller in AVR system. *IEEE Transactions on Energy Conversion, Vol.19(2), 384-391.*

[6] Chen, M. and Linkens, D.A. (1998). A hybrid neuro-fuzzy PID controller. *Fuzzy Sets and Systems, Vol.99, 27-36.*

[7] 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, Vol. 101, 21-30.*

[8] Lan, L. H. (2006). Stability analysis for a class of Takagi-Sugeno fuzzy control systems with PID controllers. *International Journal of Approximate Reasoning, In Press.*

[9] Kim, S.-M. and Han, W. Y. (2006). Induction motor servo drive using robust PID-like neuro-fuzzy controller. *Control Engineering Practice, Vol.14, 481-487.*

[10] D'Emilia, G., Marra, A. and Natale, E. (2006). Use of neural networks for quick and accurate auto-tuning of PID controller. *Robotic and Computer-Integrated Manufacturing, In Press.*

[11] Kennedy, J. and Eberhart, R.C. (1995). Particle swarm optimization. *Proceedings of the International Conference on Neural Networks, 1942-1948.*

[12] Fan, H. (2002). A modification to particle swarm optimization algorithm. *Engineering Computations, Vol. 19, 970-989.*

[13] Kao, C. –C, Chuang, C.-W and Fung, R. –F (2006). The self-tuning PID control in a slider-crank mechanism system by applying particle swarm optimization approach. *Mechatronics, Vol.16, 513-522.*

[14] Tayal, M. (2003). Particle swarm optimization for mechanical design. *M. Sc. thesis,* The University of Texas at Arlington, United States.

[15] Wang, K.-P., Huang, L., Zhou, C. –G. and Pang, W. (2003). Particle swarm optimization for traveling salesman problem. *Proceedings of the 2nd International Conference on Machine Learning and Cybernetics, 1583-1585.*

[16] Ahmed, T. (2004). Adaptive particle swarm optimizer for dynamic environments. *M. Sc. thesis,* The University of Texas at Arlington, United States

[17] Baumgartner, U., Magele, Ch. and Renhart, W. (2004). Pareto optimality and particle swarm optimization. *IEEE Transactions on Magnetics, Vol. 40(2), 1172-1175.*

[18] Ning, L., Fei, L., Debao, S. and Chang, H. (2004). Particle swarm optimization for constrained layout optimization. *Proceedings of the **5th** World Congress on Intelligent Control and Automation, 2214 -2218.*

[19] Nunez, J.J.J. (2004). Particle swarm optimization applications in power system engineering. *M.Sc. thesis*, University of Puerto Rico, Mayagues Campus, Puerto Rico.

[20] Asokan, P., Baskar, N., Babu, K., Prabhaharan, G. and Saravanan, R. (2005). Optimization of surface grinding operations using particle swarm optimization technique. *Journal of Manufacturing Science and Engineering, Vol. 127, 885-892.*

[21] Chang, W.-D. and Yan, J.-J. (2005). Adaptive robust PID controller design based on a sliding mode for uncertain chaotic systems. *Chaos, Solitons and Fractals, Vol. 26, 167-175.*

[22] Kaiser, D., Fundamentals of Servo Motion Control. http://www.parkermotion.com/whitepages/ServoFundamentals.pdf

[23] Wang, X., Wang, Y., Zhou, H. and Hui, X. (2006). PSO-PID: A novel controller for AQM Routers. *Proceedings of IEEE and IFIP International Conference on Wireless and Optical Communication Network (IEEE/IFIP WOCN 2006), 1-5.*

[24] Yoshida, H., Fukuyama, Y., Kawata, K., Takayama, S. and Nakanishi, Y. (2001). A Particle Swarm Optimization for reactive power and voltage control considering voltage security assessment. *IEEE Transactions on Power Systems, Vol.15(4), 1232-1239.*

[25] Liu, Y., Zhang, J. and Wang, S. (2004). Optimization design based on PSO algorithm for PID controller. *Proceedings of 5th World Congress on Intelligent Control and Automation, Vol. 3, 2419-2422.*