A Hybrid of ε-Constrained and Particle Swarm Optimization for Designing of PID Controllers

Ying-Tung Hsiao*, Cheng-Long Chuang#, and Joe-Air Jiang#

*Department of Electrical Engineering
Tamkang University, Taipei, Taiwan, 251, R.O.C.
Tel: +886-2-26215656 Ext. 2786
E-mail: h siao@ee.tku.edu.tw

#Department of Bio-Industrial Mechatronics Engineering
National Taiwan University, Taipei, Taiwan, 106, R.O.C.
Tel: +886-2-33665341
E-mail: jiang@ntu.edu.tw

Abstract—In this paper, an optimum approach to design PID controllers has been proposed. PID control schema based on classical control theory has been widely used in industrial control processes. Since most of the control systems have nonlinear properties, it is difficult to determine optimal parameters for a given PID controller. This study utilizes the particle swarm optimization algorithm as solution method to search for PID parameters that capable of minimizing the integral absolute control error. At the same time, the transient response is guaranteed by minimizing the maximum overshoot, settling time, rising time of step response. Moreover, we also utilize ε-constraint method to improve the performance of the controllers. Finally, experimental results demonstrate that better control performance and robustness can be achieved in comparison with known methods.

I. INTRODUCTION

The proportional-integral-derivative (PID) controllers are based on classical control theory. Due to their simple control structures and robustness in a wide range of operating conditions, PID controllers have been widely used in various industrial control systems. The design of a PID controller requires specification of three parameters, which are proportional gain ($K_p$), integral time constant ($K_i$), and derivative time constant ($K_d$), respectively. Traditionally, these parameters are determined by a trial and error approach. Manual tuning of PID controller is very tedious and time consuming, especially the performance of the controller is mainly depends on the experiences of the design engineers.

In recent years, many tuning methods have been proposed to reduce the time consuming on optimizing the choice of the three controller parameters. The most well known tuning method is the Ziegler-Nichols tuning formula [1]; it determines the parameters by observing the gain at which the plant becomes oscillatory and the frequency of this oscillatory. Several other similar simple methods have been developed, either [2] – [4]. Panagopoulos et al. presented a design method for PID controllers based on optimization of load disturbance rejection with constraints on robustness to model uncertainties [5]. In [6], bilinear transformation and linear programming are used to determine the set of all PID gains that can stabilize a given discrete-time plant of arbitrary order. Grassi et al. presented a frequency loop shaping methods based on a sensitivity function for tuning PID controllers [7]. Ho et al. [8], Wang et al. [9] and Shen [10] proposed new tuning methods for PID controllers with plants with under-damped step response. Moreover, several new methods, including some evolutionary computing methods, have been developed to reduce the complexity of tuning the parameters of PID controllers; such as genetic algorithm [11] – [12], fuzzy logic [13] – [16], immune algorithm [17], simulated annealing [18], pattern recognition [19] and ant colony optimization [20]. In these studies, it has been shown that these approaches provide good solutions in tuning the parameters of PID controllers. However, there are several reasons for us to develop better methods to design PID controllers. One is the significant impact it may give because of the widespread use of the controllers. The other is the enhancing performance of PID controllers can derive from improved design methods.

In this work, we formulate the problem of designing of PID controllers as an optimization problem. The optimization objective function consists of four performance indexes used for measuring the quality of the selected parameters for PID controller, which are the maximum overshoot, settling time, rise time and integral absolute error of step response. Moreover, if we consider the load disturbance response, one extra parameter and two extra performance indexes will be adopted into the objective function, and will be described in further section. In this study, the primary design goals are to obtain good step response performance and recovery performance of load disturbance response by minimizing the integral absolute control error. Meanwhile, the transient response is guaranteed by minimizing the other performance indexes. Furthermore, we employ a solution algorithm based on a hybrid method that combines the ε-constraint and particle swarm optimization (PSO) technique. PSO is a general-purpose optimization technique that has been recently developed and recognized as effective for game theory [21], economic dispatch problem in power system [22], multi-objectives optimization problems [23], hydroelectric generator scheduling problem [24], and travel salesman problem [25] with successful results. Moreover, we also utilized the ε-constraint technique to enhance the feature of multi-objectives optimization for the proposed method.
The feature of presenting technique different from other methods is that it can be implemented easily. Furthermore, it allows design engineers to find a global optimum solution for the problem of design PID controllers, and then apply ε-constraint technique to find another weighted optimum solution until the solution satisfies the particular preferences of design engineers. Finally, several step response and load distribution response simulation results demonstrate that better control performance can be achieved in comparison with the known methods.

This paper is organized as follows. Section II presents the problem formulation for the problem of design PID controllers. Section III introduces the PSO technique. The ε-constraint technique is introduced in Section IV. Applying the proposed hybrid algorithm for designing the PID controllers is proposed in Section V. Section VI discusses the experimental results as illustrations. Conclusions are given in the last section.

II. PROBLEM FORMULATION

A PID controller can be formulated as a transfer function as

\[ G_c(s) = K_p + \frac{K_i}{s} + K_d s \]  

(1)

where \( K_p \), \( K_i \), and \( K_d \) are the proportional, integral and derivative gains, respectively. The output \( U(t) \) of PID controller is

\[ U(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{de(t)}{dt} \]

(2)

where \( e(t) \) is the error between the reference input \( r(t) \) and the output \( y(t) \) at time \( t \). The parameters of PID controller \( K_p \), \( K_i \), and \( K_d \) can be manipulated to produce various response curves from a given plant. As shown in Fig. 1, these parameters are generated by the tuning algorithm for a given plant. There is another parameter, which is well known as set-point weighting, is useful in shaping the responses of the set point charges [26]. To improve the set-point response of the system, the set-point weighting \( b \) is introduced into the PID controller as shown in fig. 2, and can be formulated as follow

\[ U(t) = K_p (br(t) - y(t)) + K_i \int e(t) dt + K_d \frac{de(t)}{dt} \]

(3)

The transient response of a control system is necessarily important, since both the amplitude and time duration of the transient response must be kept within tolerable or prescribed limits. Hence, four key index performances of the transient response are used to measure the performance of the PID controller systems and defined as follows:

A. Maximum Overshoot: \( f_{mo} \)

Let \( y \) be the unit-step response, \( y_{max} \) denotes the maximum value of \( y \), and \( y_{ss} \) represents the steady-state value of \( y \), the maximum overshoot is defined as

\[ f_{mo} = y_{max} - y_{ss} \]

(4)

B. Rise Time: \( f_{rt} \)

The rise time \( t_r \) is defined as the time required for the step response to rise from 10 to 90 percent of its final value, we have

\[ f_{rt} = t_r \]

(5)

C. Settling Time: \( f_{st} \)

The settling time \( t_s \) is defined as the time required for the step response to decrease and stay within a specified percentage of its final value. A frequently used settling time is within an interval between 3 to 5 percent.

\[ f_{st} = t_s \]

(6)

D. Integral Absolute Control Error: \( f_{iae} \)

The integral of the absolute magnitude of control error (IAE) is written as

\[ f_{iae} = \int_0^T |e(t)| dt \]

(7)

For higher-order system, IAE must be computed numerically, since it is not practicable to integrate up to infinite, the limit \( T \) is replaced by \( t \) which is chosen sufficiently large so that \( e(t) \) for \( t > T \) is negligible. Usually \( T \) is estimated to be settling time \( t_s \) or multiple of settling time.

In summary, the optimal design for transient response of PID controllers can be formulated in the following way:

For a given plant, find the parameters \( K_p^*, K_i^*, K_d^* \), and \( b^* \) of the PID control system that minimize the performance indexes on the transient response. That is

\[ \text{minimize } f = f_{mo} + f_{rt} + f_{st} + f_{iae} \]

(8)

If we need to consider the recovery performance of load disturbance response while designing the PID controllers, following two extra performance indexes are also used to measure the performance of the PID controller systems and defined as follows:

E. Maximum Overshoot under Load Disturbance: \( f_{mod} \)

Let \( y_d \) be the load disturbance response, \( y_{maxd} \) denotes the maximum value of \( y_d \), and \( y_{ss} \) represents the steady-state value
of \( y_d \), the maximum overshoot under load disturbance is defined as
\[
f_{\text{mod}} = y_{\text{max}} - y_{sd}
\] (9)

F. Settling Time under Load Disturbance: \( f_{sd} \)
The settling time \( t_{sd} \) is defined as the time required for the load disturbance response to decrease and stay within a specified percentage of its final value. A frequently used settling time is within an interval from 3 to 5 percent,
\[
f_{sd} = t_{sd}
\] (10)

In summary, the optimal design for load disturbance response of PID controllers can be formulated in the following way:

For a given plant, find the parameters \( K_p^*, K_i^*, K_d^* \), and \( b^* \) of the PID control system that minimize the performance indexes on the load disturbance response and transient response. That is minimize \( f = f_{\text{mo}} + f_{rt} + f_{tu} + f_{mod} + f_{sd} \) (11)

III. PARTICLE SWARM OPTIMIZATION ALGORITHM

The particle swarm optimization (PSO) algorithm is first introduced by Kennedy and Eberhart [27] – [28]. It is a stochastic optimization technique that can likely simulate the behavior of a flock of birds, or the sociological behavior of a group of people. Several modified versions of PSO algorithm have been proposed and attempted to improve the performance of the original PSO algorithm.

The PSO is a population based optimization technique. It consists of a population, called swarm, in the solution space. The algorithm tries to find optimal solutions by using a prior experience of each particle. Each individual particle represents a possible solution to the optimization task at current time. By sharing information between the particles of the swarm, each particle not only accelerates in the direction of its own personal best solution found, but also in the direction of the global best solution discovered so far by any of the particles of the swarm. On the other hand, when a particle finds a promising new global optimal solution, all the other particles will trend toward the new solution. In the treading process, other particles may explore other regions and search for better solutions.

Let \( S \) denotes the swarm size, where \( i = 1, 2, \ldots, S \) represents the number of each individual particle. We denote current position, current velocity, and a personal best solution in search space as \( x_i, v_i, \) and \( y_i \), respectively. Assume that the function \( f \) is the cost function, which needs to be minimized. The operation of the PSO algorithm can be described as follows

\[
v_{i,j}(t+1) = wv_{i,j}(t) + c_1r_1(j)[y_{i,j}(t) - x_{i,j}(t)] + c_2r_2(j)[\hat{y}_{i,j}(t) - x_{i,j}(t)]
\] (12)

for all \( j = 1, 2, \ldots, n \), thus, \( v_{i,j} \) is the velocity of the \( j \)-th dimension of particle \( i \), \( w \) represents the inertia weight, \( c_1 \) and \( c_2 \) are acceleration constant, \( r_1 \) and \( r_2 \) \( \sim U(0,1) \), and \( \hat{y}_{i,j} \) represents the discovered global best solution. The new position of a particle is updated using
\[
x_{i}(t+1) = x_{i}(t) + v_{i}(t+1)
\] (13)

And the personal best solution of each particle is updated using
\[
y_{i}(t+1) = \begin{cases} y_{i}(t) & \text{if } f(x_i(t+1)) \geq f(y_i(t)) \\ x_{i}(t+1) & \text{if } f(x_i(t+1)) < f(y_i(t)) \end{cases}
\] (14)

Finally, the global best solution is found by any particle, \( \hat{y} \), is defined as follow
\[
\hat{y}(t+1) = \arg \min_{y_i} f(y_i(t+1)), 1 \leq i \leq S.
\] (15)

The value of \( v_i \) can be restricted in a range \([-v_{\text{max}}, v_{\text{max}}] \), to prevent the position of particles moving too far away from the search space to reduce redundant calculation. The value of \( v_{\max} \) is typically chosen to be \( k \times x_{\text{max}} \), with \( 0.1 \leq k \leq 1.0 \) [28], to limit the maximum distance that a particle is able to move during one iteration.

The value of \( w \), called the inertia weight, is typically chosen from 1 to near 0. During the algorithm processing, \( w \) will decrease when an iteration has completed. The idea of inertia weight is similar to the momentum term in training algorithm of gradient descent neural network. It can be found in simulated annealing algorithms, as well.

The acceleration constant \( c_1 \) and \( c_2 \) control the dependency of the movement for each particle. \( c_1 \) controls the incidence of personal best solution for deciding the distance of next movement of the particle. In the other hand, \( c_2 \) controls the incidence of global best solution for deciding the distance of next movement of the particle. Typically, these are both presetted to a value of 2.0 [28]. However, the value of \( c_1 \) and \( c_2 \) are not restricted. Different values to \( c_1 \) and \( c_2 \) may leads to improved performance [29]. A new idea, called constriction factor, was recently invented into PSO algorithm for help to ensure convergence. The operation of the constriction factor is formulated as follow
\[
v_{i,j}(t) = \chi[v_{i,j}(t) + c_1r_1(y_{i,j}(t) - x_{i,j}(t))] + c_2r_2(\hat{y}_{i,j}(t) - x_{i,j}(t))
\] (16)

where
\[
\chi = \frac{2}{2 - \phi - \sqrt{\phi^2 - 4\phi}}
\] (17)

and \( \phi = c_1 + c_2 \). \( \phi > 4 \).

Since the PSO algorithm been proposed, several improvements have been suggested. There are many different changes were made based on the equation shown above. In this paper, we still utilize the original PSO algorithm with inertia weight as our base algorithm. And we have hybrid \( \varepsilon \)-constraint technique and the PSO algorithm to enhance the performance of the PSO algorithm, which is described in Section IV.

IV. \( \varepsilon \)-CONSTRAINT TECHNIQUE

In multi-objectives optimization problem, sometimes one objective cannot be improved without sacrificing other objectives. Typically, better optimal solutions can be found by two different methods, one is weights method, and another is \( \varepsilon \)-constraint method. In this paper, we use \( \varepsilon \)-constraint method as our post-optimization method.
There is a general multi-objectives optimization problem with \( m \) objective functions, and each of them are usually conflict with each other. Typically, it is impossible to obtain an optimal solution with all objective functions are optimized at the same time. In \( \varepsilon \)-constraint technique, we optimize one objective and set the other objectives as constraints, thus, it converts the problem into a single objective optimization problem. The detail description of \( \varepsilon \)-constraint technique is as follows:

A general multi-objectives optimization problem can be expressed as follows:

\[
\min C(x) = [C_1(x), C_2(x), \ldots, C_m(x)]^T
\]

such that

\[
x \in \Omega
\]

where \( x \) is a decision vector, \( \Omega \) is a non-empty constraint domain, and \( C_i(x) \), where \( i = 1, 2, \ldots, m \), are the objective functions. For some reason, that it is needed to improve the \( i \)th objective as design engineers desired. While the \( \varepsilon \)-constraint provides a basis for the second stage to search for an optimal solution. If the result from the second stage still does not satisfy the design engineers’ request, we can increase the trade-off tolerances for other objective functions, and then execute the second stage, again. After several time of iterations for second stage, it allows design engineers to find an acceptable and non-inferior solution. A summarized version of these two stage solution algorithm is shown as below:

Stage 1: Original PSO

The primary goal of this stage is to find a global, non-inferior point for a multi-objectives optimization problem using particle swarm optimization technique. Let \( x \) represents the point that obtained in this stage, and also set \( C_N \) as the primary objective function.

Stage 2: Hybrid of \( \varepsilon \)-constraint and PSO

This stage uses the result obtained from the first stage as basis, and applies particle swarm optimization technique, again. During the optimization process, every objective function, except \( C_N \), has a trade-off tolerance \( \Delta \varepsilon \). Then solve the problem as a single objective optimization problem as follows:

\[
\min C_N(x)
\]

such that

\[
x \in \Omega
\]

\[
C_i(x) \leq \varepsilon_i + \Delta \varepsilon_i, \quad i = 1, 2, \ldots, m; \quad i \neq N.
\]

The following presents the solution algorithm for the optimal design of PID controllers.

Step 1. Initialization. **(First stage)**

An initial population (swarm) of particles \( x_{i,d} \) is generated, where \( i = 1, 2, \ldots, S \). Each particle contains parameters \( K_p^*, K_i^*, K_d^*, b^* \), and \( b^* \), and randomly be deployed within a presetted solution space with a randomly generated initial velocity. The current position of each particle is settled to the personal best solution \( y_{i,d} \). The velocity for each particle \( v_{i,d} \) is randomly initialized as well. Then evaluate every particle, and select the particle that has the lowest cost to be the global best solution of the swarm \( g_d \). The cost function is denoted by \( f \), and maximum iteration time is \( n_{\text{max}} \). And set iteration counter \( n_i = 0 \).

Step 2. Update personal and global best solutions.

For particle counter \( i = 1 \) to \( S \)

If \( f(x_{i,d}) \leq f(y_{i,d}) \)

For \( d = 1 \) to 4 // as four dimensional optimization

\[ y_{i,d} = x_{i,d} \]

End for

End if

If \( f(y_{i,d}) \leq f(g_d) \)

For \( d = 1 \) to 4

\[ g_d = y_{i,d} \]

End for

End if

Step 3. Calculate the velocity for each particle.

For particle counter \( i = 1 \) to \( S \)

For \( d = 1 \) to 4

Calculate and update the velocity value for each dimension of particle \( i_d \) by equation (12).

3066
End for
End for
Step 4. Update current position for each particle.
For particle counter \( i = 1 \) to \( S \)
For \( d = 1 \) to \( 4 \)
Calculate and update the position for each dimension of particle \( i \) by equation (13).
End for
End for
Step 5. Check for stop criterion.
If \( n_g \leq n_{\text{max}} \)
Jump to Step 6.
Else
\( n_g = n_g + 1 \) and Return to Step 2.
End if
**(End of first stage)**
Step 6. Initialization for \( \epsilon \)-constraint. **(Second stage)**
Suppose that \( f \) consists of \( P \) performance indexes, and design engineers need to improve \( N \)th performance index that obtained from evaluating the result of the first stage. According to the presetted trade-off tolerance \( \Delta_c p \), every performance indexes, except \( N \)th performance index, are be transferred into constraint by equation (20) as follow
For \( p = 1 \) to \( P \)
If \( p = N \)
Continue
End if
\( C_p = C_p \times (1 + \Delta_c p) \)
End for
Step 7. Initialization
An initial population (swarm) of particles \( x_{i,d} \) and \( v_{i,d} \) are generated, where \( i = 1, 2, \ldots, S \). Each particle contains parameters \( K_p^*, K_i^*, K_d^* \), and \( b^* \), and randomly be deployed near the global best solution \( g_d \) obtained from the first stage with a randomly generated initial velocity. The current position of each particle is settled to the personal best solution \( y_{i,d} \). And set iteration counter \( n_g = 0 \).
Step 8. Update personal and global best solutions.
For particle counter \( i = 1 \) to \( S \)
For \( p = 1 \) to \( P \)
If \( f(x_{i,d}) > f(y_{i,d}) \)
Set break_flag = true; break
End if
End for
If break_flag = true
Continue
End if
If \( f(x_{i,d}) \leq f(y_{i,d}) \)
For \( d = 1 \) to \( 4 \)
\( y_{i,d} = x_{i,d} \)
End for
End if
If \( f(y_{i,d}) \equiv f(g_d) \)
For \( d = 1 \) to \( 4 \)
\( g_d = y_{i,d} \)
End for
End if
End for
Step 9. Calculate the velocity for each particle.
For particle counter \( i = 1 \) to \( S \)
For \( d = 1 \) to \( 4 \)
Calculate and update the velocity value for each dimension of particle \( i \) by equation (12).
End for
End for
Step 10. Update current position for each particle.
For particle counter \( i = 1 \) to \( S \)
For \( d = 1 \) to \( 4 \)
Calculate and update the position for each dimension of particle \( i \) by equation (13).
End for
End for
Step 11. Check for stop criterion.
If \( n_g \leq n_{\text{max}} \)
Print out the solution \( g_d \), and step.
Else
\( n_g = n_g + 1 \) and Return to Step 8.
End if
**(End of second stage)**

VI. EXPERIMENTAL RESULTS
The numerical results presented in this section illustrate the performance of the proposed solution algorithm. All the computation is implemented with Matlab/Simulink on a P4 3.06 GHz computers with 1024 MB RAM. The values of the parameters in the proposed algorithm are \( S = 300, n_{\text{max}} = 100, w = 0.9, C_1 = 2, C_2 = 2, K_p_{\text{max}} = 5, K_i_{\text{max}} = 5, K_d_{\text{max}} = 5, K_b_{\text{max}} = 0.5, \) and \( \Delta c_p = 0.1 \).
This study utilized 3 plants list as follows to illustrate the transient response performance and efficiency of the proposed method.

\[
G_{P1}(s) = \frac{e^{-0.5s}}{(s+1)^2} \quad (21)
\]

\[
G_{P2}(s) = \frac{4.228}{(s+0.5)(s^2+1.64s+8.456)} \quad (22)
\]

\[
G_{P3}(s) = \frac{27}{(s+1)(s+3)^2} \quad (23)
\]

Table 1 summarizes the simulation results on the three plants. In this table, \( y_{ps} \) denotes the percent maximum overshoot, \( t_r \) represents the 5 percent settling time, \( t_s \) denotes the rise time and \( IAE \) are the integral of the absolute error.
Figure 3, 4 and 5 show the step response of \( G_{P1} \), \( G_{P2} \) and \( G_{P3} \) respectively. The results obtained by using Ziegler-Nichols [1] and Kitamori [32] are also presented for comparison. For \( G_{P1} \) and \( G_{P3} \), the Ziegler-Nichols’s method and Kitamori’s method may produce better performance on rising time, but the pro-
posed method outperforms the other methods on all other performance indexes significantly. For GP2, although Zeigler-Nichols’s method produces better response on maximum overshoot and integral of the absolute error, but the proposed method performs well on these performance indexes, and outperforms the Zeigler-Nichols’s method on other performance indexes apparently. By utilizing the $\varepsilon$-constraint method, the proposed method works well on optimizing response performance on rising time. Without utilizing the $\varepsilon$-constraint method, the responses performances on rising time that obtained from original PSO method in first stage might be worsen.

There are 4 extra plants list as follow to illustrate the transient response performance and efficient of the proposed method.

\[
G_{P_4}(s) = \frac{15}{(s+3)(s^2 + 0.9s + 5)} \tag{24}
\]
\[
G_{P_5}(s) = \frac{18}{(s+3)^2(s^2 + s + 2)} \tag{25}
\]
\[
G_{P_6}(s) = \frac{3e^{-0.1s}}{(s^2 + 1.2s + 1)(s+3)} \tag{26}
\]
\[
G_{P_7}(s) = \frac{20e^{-2s}}{(s^2 + 2.4s + 4)(s+5)} \tag{27}
\]

Table 2, 3, 4, and 5 summarizes the simulation results on the three plants. In this table, $y_{\text{m0}}$ denotes the percent maximum overshoot, $t_p$ represents the peak time, $t_s$ represents the 5 percent settling time, $t_r$ denotes the rise time, $IAE$ are the integral of the absolute error, $y_{\text{m0}_D}$ denotes the percent maximum overshoot of the load disturbance response, $t_p_D$ represents the peak time of the load disturbance response, and $t_s_D$ represents the 5 percent settling time of the load disturbance response. Note that Wang’s method and Ho’s method target on a standard PID controller without a set-point weighting parameter $b$.

Figure 6, 7, 8, and 9 show the step response and load disturbance response of GP4, GP5, GP6, and GP7, respectively. For GP4, the proposed method produces better response than that obtained using the other methods on most of the performance indexes. Although the rising time of Wang’s method has a better rising time on GP4, but the proposed method produces better transient response and load disturbance response on other performance indexes apparently. For GP5, although Wang’s method and Ho’s method has better percent maximum overshoot and rising time on GP5, but the proposed method outperforms other methods on all of the other performance indexes. For GP6, although the proposed method produces better responses than that obtained using the other methods on the six performance indexes. Note: The PID parameters of the Kitamori’s controller are not

<table>
<thead>
<tr>
<th>Plants</th>
<th>Zeigler-Nichols</th>
<th>Kitamori</th>
<th>Proposed Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{P_4}(s)$</td>
<td>$K_P=2.190$</td>
<td>$K_P=2.212$</td>
<td>$K_P=3.0155$</td>
</tr>
<tr>
<td></td>
<td>$K_i=1.151$</td>
<td>$K_i=1.085$</td>
<td>$K_i=0.92783$</td>
</tr>
<tr>
<td></td>
<td>$y_{\text{m0}}=42.95%$</td>
<td>$y_{\text{m0}}=14.51%$</td>
<td>$y_{\text{m0}}=4.78%$</td>
</tr>
<tr>
<td></td>
<td>$t_r=0.7500$</td>
<td>$t_r=1.0240$</td>
<td>$t_r=0.8490$</td>
</tr>
<tr>
<td></td>
<td>$t_c=3.8530$</td>
<td>$t_c=3.6400$</td>
<td>$t_c=1.7100$</td>
</tr>
<tr>
<td></td>
<td>$t_s=1.8304$</td>
<td>$t_s=1.5466$</td>
<td>$t_s=1.3687$</td>
</tr>
<tr>
<td>$G_{P_5}(s)$</td>
<td>$K_P=2.126$</td>
<td>$K_P=2.212$</td>
<td>$K_P=3.0155$</td>
</tr>
<tr>
<td></td>
<td>$K_i=0.565$</td>
<td>$K_i=1.085$</td>
<td>$K_i=0.92783$</td>
</tr>
<tr>
<td></td>
<td>$y_{\text{m0}}=16.47%$</td>
<td>$y_{\text{m0}}=4.78%$</td>
<td>$y_{\text{m0}}=6.15%$</td>
</tr>
<tr>
<td></td>
<td>$t_r=0.7300$</td>
<td>$t_r=1.0240$</td>
<td>$t_r=0.8490$</td>
</tr>
<tr>
<td></td>
<td>$t_c=5.3720$</td>
<td>$t_c=3.6400$</td>
<td>$t_c=1.7100$</td>
</tr>
<tr>
<td></td>
<td>$t_s=0.9595$</td>
<td>$t_s=1.5466$</td>
<td>$t_s=1.3687$</td>
</tr>
<tr>
<td>$G_{P_6}(s)$</td>
<td>$K_P=3.072$</td>
<td>$K_P=2.357$</td>
<td>$K_P=3.1619$</td>
</tr>
<tr>
<td></td>
<td>$K_i=2.272$</td>
<td>$K_i=1.429$</td>
<td>$K_i=1.1032$</td>
</tr>
<tr>
<td></td>
<td>$y_{\text{m0}}=32.55%$</td>
<td>$y_{\text{m0}}=10.77%$</td>
<td>$y_{\text{m0}}=3.72%$</td>
</tr>
<tr>
<td></td>
<td>$t_r=0.6650$</td>
<td>$t_r=0.8270$</td>
<td>$t_r=0.9040$</td>
</tr>
<tr>
<td></td>
<td>$t_c=3.7210$</td>
<td>$t_c=2.2980$</td>
<td>$t_c=1.5210$</td>
</tr>
<tr>
<td></td>
<td>$t_s=1.1245$</td>
<td>$t_s=0.8815$</td>
<td>$t_s=1.0883$</td>
</tr>
<tr>
<td>$G_{P_7}(s)$</td>
<td>$K_P=3.1619$</td>
<td>$K_P=2.357$</td>
<td>$K_P=3.1619$</td>
</tr>
<tr>
<td></td>
<td>$K_i=2.272$</td>
<td>$K_i=1.429$</td>
<td>$K_i=1.1032$</td>
</tr>
<tr>
<td></td>
<td>$y_{\text{m0}}=32.55%$</td>
<td>$y_{\text{m0}}=10.77%$</td>
<td>$y_{\text{m0}}=3.72%$</td>
</tr>
<tr>
<td></td>
<td>$t_r=0.6650$</td>
<td>$t_r=0.8270$</td>
<td>$t_r=0.9040$</td>
</tr>
<tr>
<td></td>
<td>$t_c=3.7210$</td>
<td>$t_c=2.2980$</td>
<td>$t_c=1.5210$</td>
</tr>
<tr>
<td></td>
<td>$t_s=1.1245$</td>
<td>$t_s=0.8815$</td>
<td>$t_s=1.0883$</td>
</tr>
</tbody>
</table>
Table 2
Summary of simulation results on $G_{P4}$

<table>
<thead>
<tr>
<th>Method</th>
<th>$K_p$</th>
<th>$K_i$</th>
<th>$K_d$</th>
<th>$y_{eo}$</th>
<th>$t_e$</th>
<th>$t_i$</th>
<th>$t_d$</th>
<th>$y_{eo,D}$</th>
<th>$t_e,D$</th>
<th>IAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang's Method</td>
<td>0.4631</td>
<td>0.4530</td>
<td>0.6090</td>
<td>63.31%</td>
<td>1.79</td>
<td>0.61</td>
<td>7.67</td>
<td>61.76%</td>
<td>16.19</td>
<td>3.2150</td>
</tr>
<tr>
<td>Shen's Method</td>
<td>0.916</td>
<td>1.8514</td>
<td>0.3200</td>
<td>12.61%</td>
<td>1.94</td>
<td>0.87</td>
<td>4.15</td>
<td>72.95%</td>
<td>16.3</td>
<td>2.0690</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>0.8552</td>
<td>1.6514</td>
<td>0.3020</td>
<td>3.83%</td>
<td>3.4910</td>
<td>0.9920</td>
<td>3.8500</td>
<td>55.08%</td>
<td>16.1720</td>
<td>1.7597</td>
</tr>
</tbody>
</table>

Table 3
Summary of simulation results on $G_{P5}$

<table>
<thead>
<tr>
<th>Method</th>
<th>$K_p$</th>
<th>$K_i$</th>
<th>$K_d$</th>
<th>$b$</th>
<th>$y_{eo}$</th>
<th>$t_e$</th>
<th>$t_i$</th>
<th>$t_d$</th>
<th>$y_{eo,D}$</th>
<th>$t_e,D$</th>
<th>IAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang's Method</td>
<td>0.223</td>
<td>0.9411</td>
<td>0.6186</td>
<td>0.322</td>
<td>30.09%</td>
<td>2.86</td>
<td>1.09</td>
<td>5.3790</td>
<td>66.36%</td>
<td>6.82</td>
<td>2.2190</td>
</tr>
<tr>
<td>Ho's Method</td>
<td>0.783</td>
<td>0.8558</td>
<td>0.7130</td>
<td>0.322</td>
<td>30.09%</td>
<td>2.86</td>
<td>1.09</td>
<td>5.3790</td>
<td>66.36%</td>
<td>6.82</td>
<td>2.2190</td>
</tr>
<tr>
<td>Shen's Method</td>
<td>0.92</td>
<td>1.0123</td>
<td>0.4078</td>
<td>0.322</td>
<td>30.09%</td>
<td>2.86</td>
<td>1.09</td>
<td>5.3790</td>
<td>66.36%</td>
<td>6.82</td>
<td>2.2190</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>1.0931</td>
<td>1.0283</td>
<td>0.4078</td>
<td>0.322</td>
<td>30.09%</td>
<td>2.86</td>
<td>1.09</td>
<td>5.3790</td>
<td>66.36%</td>
<td>6.82</td>
<td>2.2190</td>
</tr>
</tbody>
</table>

Table 4
Summary of simulation results on $G_{P6}$

<table>
<thead>
<tr>
<th>Method</th>
<th>$K_p$</th>
<th>$K_i$</th>
<th>$K_d$</th>
<th>$b$</th>
<th>$y_{eo}$</th>
<th>$t_e$</th>
<th>$t_i$</th>
<th>$t_d$</th>
<th>$y_{eo,D}$</th>
<th>$t_e,D$</th>
<th>IAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang's Method</td>
<td>0.466</td>
<td>0.5870</td>
<td>0.7130</td>
<td>0.322</td>
<td>2.4357</td>
<td>1.45</td>
<td>6.53</td>
<td>61.96%</td>
<td>17.09</td>
<td>17.0000</td>
<td></td>
</tr>
<tr>
<td>Ho's Method</td>
<td>0.6540</td>
<td>1.301</td>
<td>1.4613</td>
<td>0.322</td>
<td>2.4357</td>
<td>1.45</td>
<td>6.53</td>
<td>61.96%</td>
<td>17.09</td>
<td>17.0000</td>
<td></td>
</tr>
<tr>
<td>Shen's Method</td>
<td>0.92</td>
<td>1.0123</td>
<td>0.4078</td>
<td>0.322</td>
<td>2.4357</td>
<td>1.45</td>
<td>6.53</td>
<td>61.96%</td>
<td>17.09</td>
<td>17.0000</td>
<td></td>
</tr>
<tr>
<td>Proposed Method</td>
<td>2.3497</td>
<td>2.1463</td>
<td>0.4078</td>
<td>0.322</td>
<td>2.4357</td>
<td>1.45</td>
<td>6.53</td>
<td>61.96%</td>
<td>17.09</td>
<td>17.0000</td>
<td></td>
</tr>
</tbody>
</table>

Table 5
Summary of simulation results on $G_{P7}$

<table>
<thead>
<tr>
<th>Method</th>
<th>$K_p$</th>
<th>$K_i$</th>
<th>$K_d$</th>
<th>$b$</th>
<th>$y_{eo}$</th>
<th>$t_e$</th>
<th>$t_i$</th>
<th>$t_d$</th>
<th>$y_{eo,D}$</th>
<th>$t_e,D$</th>
<th>IAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang's Method</td>
<td>0.147</td>
<td>0.3729</td>
<td>0.4250</td>
<td>0.329</td>
<td>0.3729</td>
<td>0.4250</td>
<td>0.329</td>
<td>0.3729</td>
<td>0.4250</td>
<td>0.329</td>
<td>0.3729</td>
</tr>
<tr>
<td>Ho's Method</td>
<td>0.231</td>
<td>0.3929</td>
<td>0.3014</td>
<td>0.329</td>
<td>0.3929</td>
<td>0.3014</td>
<td>0.329</td>
<td>0.3929</td>
<td>0.3014</td>
<td>0.329</td>
<td>0.3929</td>
</tr>
<tr>
<td>Shen's Method</td>
<td>0.3729</td>
<td>0.4250</td>
<td>0.3014</td>
<td>0.329</td>
<td>0.3929</td>
<td>0.3014</td>
<td>0.329</td>
<td>0.3929</td>
<td>0.3014</td>
<td>0.329</td>
<td>0.3929</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>0.4250</td>
<td>0.3014</td>
<td>0.3014</td>
<td>0.329</td>
<td>0.3929</td>
<td>0.3014</td>
<td>0.329</td>
<td>0.3929</td>
<td>0.3014</td>
<td>0.329</td>
<td>0.3929</td>
</tr>
</tbody>
</table>

Figure 6. Comparison of step response of the $G_{P4}(s)$ plant.

Figure 7. Comparison of step response of the $G_{P5}(s)$ plant.

Figure 8. Comparison of step response of the $G_{P6}(s)$ plant.

Figure 9. Comparison of step response of the $G_{P7}(s)$ plant.
VII. CONCLUSION

This paper presents a novel approach for designing of the PID controller for getting a good performance under a given plant. The problem of design PID controllers is formulated as an optimal problem considering four performance indexes on the transient response, the maximum overshoot, settling time, rise time and integral absolute error. And there are two extra performance indexes on the load disturbance response, which are the maximum overshoot of the load disturbance response and the settling time of the load disturbance response. Furthermore, a hybrid of the ε-constraint method and PSO technique solution algorithm was proposed to derive the global optimal solution.

The feature of the presented technique different from other methods is that (i) it is easily to implement the proposed algorithm (perhaps that the entire procedure of the proposed method seems very long, but the main algorithm is still based on PSO), the ε-constraint method only plays as assistance role for PSO); (ii) it allows a more flexible problem formulation; (iii) it allows design engineers to find a global optimal solution for the problem of the design of PID controllers; (iv) it allows the design engineers to set a specific performance index to improve the previous obtained solution. Finally, the proposed method was implemented and tested on several plants with promising results that compare with known methods.

VIII. ACKNOWLEDGMENTS

The authors are grateful to the National Science Council of the Republic of China for financially supporting this research under contract no. NSC 93-2213-E-002-133.

REFERENCES
