Optimal Tuning of PID Parameters Using Iterative Learning Approach

Jian-Xin Xu *, Deqing Huang *, and Srinivas Inidi **

Abstract: Proportional-integral-derivative (PID) controller is the most predominant industrial controller that constitutes more than 90% feedback loops. Time domain performance of PID, including peak overshoot, settling time and rise time, is directly relevant to PID parameters. In this work we propose an iterative learning tuning method (ILT) – an optimal tuning method for PID parameters by means of iterative learning, PID parameters are updated whenever the same control task is repeated. The first novel property of the new tuning method is that the time domain performance or requirements can be incorporated directly into the objective function to be minimized. The second novel property is that the optimal tuning does not require as much the plant model knowledge as other PID tuning methods. The new tuning method is essentially applicable to any plants that are stabilizable by PID controllers. The third novel property is that any existing PID auto-tuning methods can be used to provide the initial setting of PID parameters, and the iterative learning process guarantees that a better PID controller can be achieved. The fourth novel property is that the iterative learning of PID parameters can be applied straightforward to discrete-time or sampled-data systems, in contrast to existing PID auto-tuning methods which are dedicated to continuous-time plants. In this paper, we further exploit efficient searching methods for the optimal tuning of PID parameters. Through theoretical analysis, comprehensive investigations on benchmarking examples, and real-time experiments on the level control of a coupled-tank system, the effectiveness of the proposed method is validated.

Key Words: PID tuning, iterative learning, objective functions, searching methods, iterative learning tuning (ILT).

1. Introduction

Among all the known controllers, the proportional-integral-derivative (PID) controllers are always the first choice for industrial control processes owing to the simple structure, robust performance, and balanced control functionality under a wide range of operating conditions. However, the exact workings and mathematics behind PID methods vary with different industrial users. Tuning PID parameters (gains) remains a challenging issue and directly determines the effectiveness of PID controllers [1]–[3].

To address the PID design issue, much effort has been invested in developing systematic auto-tuning methods. These methods can be divided into three categories, where the classification is based on the availability of a plant model and model type, (i) non-parametric model methods; (ii) parametric model methods; (iii) model free methods.

The non-parametric model methods use partial modelling information, usually including the steady state model and critical frequency points. These methods are more suitable for closed-loop tuning and applied without the need for extensive prior plant information [1]. Relay feedback tuning method [4]–[7] is a representative method of the first category.

The parametric model methods require a linear model of the plant – either transfer function matrix or state space model. To obtain such a model, standard off-line or on-line identification methods are often employed to acquire the model data. Thus parametric model methods are more suitable for off-line PID tuning [8]. When the plant model is known with a parametric structure, optimal design methods can be applied [9],[10]. As for PID parameter tuning, it can be formulated as the minimization of an objective function with possible design specifications such as the nominal performance, minimum input energy, robust stability, operational constraints, etc.

In model free methods, no model or any particular points of the plant are identified. Three typical tuning methods are unfalsified control [11], iterative feedback tuning [12] and extreme seeking [13]. In [11], input-output data is used to determine whether a set of PID parameters meets performance specifications and these PID parameters are updated by an adaptive law based on whether or not the controller falsifies a given criterion. In [12], the PID controller is updated through minimizing an objective function that evaluate the closed-loop performance and estimating the system gradient. In [13], adaptive updating is conducted to tune PID parameters such that the output of the cost function reaches a local minimum or local maximum.

In practice, when the plant model is partially unknown, it would be difficult to compute PID parameters even if the relationship between transient specifications and PID parameters can be derived. In many of existing PID tuning methods, whether model free or model based, test signals will have to be injected into the plant in order to find certain relevant information for setting controller parameters. This testing process may however be unacceptable in many real-time control tasks. On the other hand, many control tasks are carried out repeatedly, such as in batch processors. The first objective of this work is to explore the possibility of fully utilizing the task repetitiveness property, consequently provide a learning approach to improve PID controllers through iteratively tuning parameters when the transient behavior is of the main concern.
In most learning methods including neural learning and iterative learning [14], the process Jacobian or gradient plays the key role by providing the greatest descending direction for the learning mechanism to update inputs. The convergence property of these learning methods is solely dependent on the availability of the current information on the gradient. The gradient between the transient control specifications and PID parameters, however, may not be available if the plant model is unknown or partially unknown. Further, the gradient is a function of PID parameters, thus the magnitude and even the sign may vary. The most difficult scenario is when we do not know the sign changes a priori. In such circumstances, traditional learning methods cannot achieve learning convergence. The second objective of this work is to extend the iterative learning approach to deal with the unknown gradient problem for PID parameter tuning.

It should be noted that in many industrial control problems such as in process industry, the plant is stable in a wide operation range under closed-loop PID, and the major concern for a PID tuning is the transient behaviors either in the time domain, such as peak overshoot, rise time, settling time, or in the frequency domain such as bandwidth, damping ratio and undamped natural frequency. From the control engineering point of view, it is one of the most challenges to directly address the transient performance, in comparison with the stability issues, by means of tuning control parameters. Even for a lower order linear time invariant plant under PID, the transient performance indices such as overshoot could be highly nonlinear in PID parameters and an analytical inverse mapping from overshoot to PID parameters may not exist. In other words, from the control specification on overshoot we are unable to decide the PID parameters analytically. The third objective of this work is to link these transient specifications with PID parameters and give a systematic tuning method.

Another issue is concerned with the redundancy in PID parameter tuning when only one or two transient specifications are required, for instance when only overshoot is specified, or only the integrated absolute error is to be minimized. In order to fully utilize the extra degrees of freedom of the controller, the most common approach is to introduce an objective function and optimize the PID parameters accordingly. This traditional approach is however not directly applicable because of the unknown plant model, and in particular the unknown varying gradient. A solution to this problem is still iterative learning. An objective function, which is accessible, is chosen as the first step for PID parameter optimization. Since the goal is to minimize the objective function, the control inputs will be updated along the greatest descending direction, namely the gradient, of the objective function. In other words, the PID parameters are chosen to directly reduce the objective function, and the objective function is treated as the plant output and used to update the PID parameters. When the gradient is varying and unknown, extra learning trials will be conducted to search the best descending direction.

The paper is organized as follows. Section 2 gives the formulation of PID auto-tuning problem. Section 3 introduces the comparative studies on benchmark examples. Furthermore, Section 5 addresses the real-time implementation on a laboratory pilot plant. Section 6 concludes the work.

2. Formulation of PID Auto-Tuning Problem

2.1 PID Auto-Tuning

In this work, we consider a fundamental PID controller in continuous or discrete-time

\[ C(s) = k_p + k_i \frac{1}{s} + k_d s \]

\[ C(z) = k_p + k_i \frac{T_i z}{z - 1} + k_d \frac{z - 1}{T_d z} \]

where \( k_p \) is the proportional gain, \( k_i \) the integral gain, \( k_d \) is the derivative gain, \( s \) is the Laplace operator, \( z \) is the Z operator, \( T_i \) is the sampling period. Denote \( \mathbf{k} = [k_p, k_i, k_d]^T \).

The aim of PID tuning is to find appropriate values for PID parameters such that the closed-loop response can be significantly improved when comparing with the open-loop response. Since the PID control performance is determined by PID parameters \( \mathbf{k} \), by choosing a set of performance indices \( \mathbf{x} \), for instance overshoot and settling time, there exists a unique relationship or mapping \( \mathbf{f} \) between \( \mathbf{x} \) and \( \mathbf{k} \)

\[ \mathbf{x} = \mathbf{f}(\mathbf{k}). \]

The PID auto-tuning problem can be mathematically formulated as to look for a set of \( \mathbf{k} \) such that \( \mathbf{x} \) meet the control requirements specified by \( \mathbf{x}_0 \). If the inverse mapping is available, we have \( \mathbf{k} = \mathbf{f}^{-1}(\mathbf{x}_0) \). The mapping \( \mathbf{f} \), however, is a vector valued function of \( \mathbf{x} \), \( \mathbf{k} \) and the plant, and is in general a highly nonlinear mapping. Thus its inverse mapping in general is either not analytic solvable or not uniquely existing. Above all, the most difficult problem in PID auto-tuning is the lack of plant model, hence the mapping \( \mathbf{f} \) is unknown or partially unknown.

The importance of PID and challenge in PID auto-tuning attracted numerous researchers and come up with various auto-tuning or detuning methods, each has unique advantages and limitations. In this work we propose a new auto-tuning method using iterative learning and optimization, which complements existing PID tuning methods.

2.2 Performance Requirements and Objective Functions

An objective function, or cost function, quantifies the effectiveness of a given controller in terms of the closed-loop response, either in time domain or frequency domain. A widely used objective function in PID auto-tuning is the integrated square error (ISE) function

\[ J(\mathbf{k}) = \frac{1}{T - t_0} \int_{t_0}^{T} e^2(t, \mathbf{k}) dt, \]

where the error \( e(t, \mathbf{k}) = r(t) - y(t, \mathbf{k}) \) is the difference between the reference, \( r(t) \), and the output signal of the closed-loop system, \( y(t) \). \( T \) and \( t_0 \), with \( 0 \leq t_0 < T < \infty \), are two design parameters. In several auto-tuning methods such as IFT and ES, \( t_0 \) is set approximately at the time when the step response of the closed-loop system reaches the first peak. Hence the cost function effectively places zero weighting on the initial transient portion of the response and the controller is tuned to minimize the error beyond the peak time. Similar objective functions, such as integrated absolute error, integrated time weighted absolute error, integrated time weighted square error,

\[ \frac{1}{T - t_0} \int_{t_0}^{T} |e| dt, \quad \frac{1}{T - t_0} \int_{t_0}^{T} t |e| dt, \quad \frac{1}{T - t_0} \int_{t_0}^{T} t e^2 dt, \]
have also been widely used in the process of PID parameter tuning.

In many control applications, however, the transient performance, such as overshoot \( M_p \), settling time \( t_s \), rise time \( t_r \), could be of the main concern. An objective function that can capture the transient response directly, is highly desirable. For this purpose, we propose a quadratic function

\[
J = (x_d - x)^T Q (x_d - x) + k^T R k,
\]

where \( x = [100M_p, t_s, t_r] \), \( x_d \) is the desired \( x \), \( Q \) and \( R \) are two constant non-negative weighting matrices. The selection of \( Q \) and \( R \) matrices will yield effects on the optimization results, that is, produce different closed-loop responses. A larger \( Q \) highlights more on the transient performance, whereas a larger \( R \) gives more control penalty.

2.3 A Second Order Example

Consider a second order plant under a unity feedback with a PD controller. The PD controller and plant are respectively

\[
C(s) = k_p + k_d s, \quad G(s) = \frac{k}{s^2 + as + b},
\]

where \( k_p, k_d, k, a, b \) are all non-negative constants. The closed-loop system is stable if \( a + d > 0 \) and \( b + p > 0 \), where \( p = kk_p \) and \( d = kk_d \).

The nonlinear mapping \( f \) between the closed-loop transient response \( x = [M_p, t_s] \) and the PD parameters \( k = [k_p, k_d] \) is derived in Appendix, and shown in Figs. 1 and 2, where \( a = 0.1, b = 0, \) and \( k = 1 \). It can be seen that the mapping \( f \) is nonlinear or even discontinuous.

The nonlinear mapping \( f \) for discrete-time control system can also be derived but omitted here due to the complexity. Discretizing the plant in (3) with a sampling time \( T_s = 0.1 \) s, the nonlinear mapping between \( M_p \) and \( (k_p, k_d) \) is shown in Fig. 3. It can be seen that there exist local minima in the surface, and the gradient may vary and take either positive or negative values.

On the other hand, it can also be seen from those figures that the transient responses may vary drastically while the control parameters only vary slightly. This indicates the importance for PID parameter auto-tuning and the necessity for finding an effective tuning method.

3. Iterative Learning Approach

Iterative learning is adopted to provide a solution to the PID auto-tuning. The iterative learning offers a desirable feature that it can guarantee the learning convergence even if the plant model is partially unknown.

3.1 Principal Idea of Iterative Learning

The concept of iterative learning was first introduced in control to deal with a repeated control task without requiring the perfect knowledge such as the plant model or parameters[15]. It learns to generate a control action directly instead of doing a model identification. The iterative learning mechanism updates the present control action using information obtained from previous control actions and previous error signals.

Let us first give the basic formulation of iterative learning in PID auto-tuning. From preceding discussions, the PID auto-tuning problem can be described by the mapping

\[
x = f(k)
\]
where $k \in \Omega_k \subset \mathbb{R}^m$ and $x \in \Omega_x \subset \mathbb{R}^n$, where $n$ and $m$ are integer numbers. The learning objective is to find a suitable set $k$ such that the transient response $x$ can reach a given region around the control specifications $x_d$.

The principal idea of iterative learning is to construct a contractive mapping $A$

$$x_{d} - x_{i+1} = A(x_{d} - x_{i}),$$

where the norm of $A$ is strictly less than 1, and the subscript $i$ indicates that the quantity is in the $i$th iteration or learning trial. To achieve this contractive mapping, a simple iterative learning law is

$$k_{i+1} = k_i + \Gamma_i(x_{d} - x_i)$$

(4)

where $\Gamma_i \in \mathbb{R}^{m \times m}$ is a learning gain matrix. It can be seen that the learning law (4) generates a set of updated parameters, $k_i$, and previous performance deviations $x_{d} - x_i$. The schematic of the iterative learning mechanism for PID auto-tuning is shown in Fig. 4.

When $n = m$, define the process gradient

$$F(k) = \frac{\partial f(k)}{\partial k},$$

we can derive the condition for the contractive mapping $A$

$$x_{d} - x_{i+1} = x_{d} - x_{i} - (x_{i+1} - x_{i})$$

$$= x_{d} - x_{i} - \frac{\partial f(k)}{\partial k}(k_{i+1} - k_i)$$

$$= [I - F(k_i')\Gamma_i](x_{d} - x_i)$$

(5)

where $k_i' \in [\min[k_i, k_i+1], \max[k_i, k_i+1]] \subset \Omega_k$. Therefore we have a contractive mapping $A$

$$x_{d} - x_{i+1} = A(x_{d} - x_i)$$

as far as the magnitude

$$|A| = |I - F(k_i')\Gamma_i| \leq \rho < 1.$$  (6)

When $n > m$, there exists an infinite number of solutions because of redundancy in control parameters. With the extra degrees of freedom in PID, optimality can be exploited, for instance the shortest settling time, minimum peak overshoot, minimum values of control parameters, etc. A suitable objective function to be minimized could be a non-negative function $J(x, k) \geq 0$, where $J$ is accessible, such as the quadratic one (2). The minimization is in fact a searching task

$$\min_{k \in \Omega_k} J(x, k)$$

where $x = f(k)$. The optimal parameters $k_{opt}$ can be obtained by differentiating $J$ and computing

$$\frac{dJ}{dk} = \frac{\partial J}{\partial x} \frac{dx}{dk} + \frac{\partial J}{\partial k} F + \frac{\partial J}{\partial k} = 0.$$  (7)

In most cases $k_{opt}$ can only be found numerically due to the highly nonlinear relationship between $k$ and quantities $J, x$. A major limitation of optimization methods is the demand for the complete plant model knowledge or the mapping $f$. On the contrary, iterative learning only requires the bounding knowledge of the process gradient. The principal idea of iterative learning can be extended to solving the optimization problem under the assumption that all gradient components with respect to $k$ have known limiting bounds. Since the learning objective now is to directly reduce the value of $J$, the objective function $J$ can be regarded as the process output to be minimized. The new iterative learning tuning law is

$$k_{i+1} = k_i - \gamma_i J(k_i)$$

(8)

where $\gamma_i = [\gamma_{ij}, \cdots, \gamma_{i,n}]^T$ and $J(k_i)$ denotes $J(k_i, f(k_i))$. To show the contractive mapping, note that

$$J(k_{i+1}) = J(k_i) + [J(k_{i+1}) - J(k_i)]$$

$$= J(k_i) + \left[ \frac{dJ(k_i)}{dk} \right]^T (k_{i+1} - k_i)$$

where $k_i' \in \Omega_k$ is in a region specified by $k_i$ and $k_{i+1}$. By substitution of the ILT law (8), we have

$$J(k_{i+1}) = \left[ 1 - \left( \frac{dJ(k_i')}{dk} \right)^T \gamma_i \right] J(k_i).$$  (9)

The convergence property is determined by the learning gains $\gamma_i$ and the gradient $dJ/dk$.

3.2 Learning Gain Design Based on Gradient Information

To guarantee the contractive mapping (9), the magnitude relationship must satisfy $|1 - D_i^T \gamma_i| \leq \rho < 1$, where

$$D_i = \left( \frac{dJ(k_i'\gamma)}{dk} \right)^T$$

is the gradient of the objective function. The selection of learning gain $\gamma_i$ is highly related to the prior knowledge on the gradient $D_i$. Consider three scenarios and assume $D_i = (D_{ij1}, D_{ij2}, D_{ij3})$, that is, a PID controller is used.

When $D_i$ is known $a priori$, we can choose $\gamma_{ij} = D_{ij}^{-1}/3$. Such a selection produces the fastest learning convergence speed, that is, convergence in one iteration because $|1-D_i^T \gamma_i| = 0$.

When the bounding knowledge and the sign information of $D_{ij}$ are available, the learning convergence can also be guaranteed. For instance assume $0 < \alpha_i \leq D_{ij} \leq \beta_i < \infty$ for
In such circumstances, choosing \( \gamma_{i} = 1/3\beta_{i} \), the upper bound of the magnitude is \( \rho = 1 - \alpha_{i}/3\beta_{i} = 2/\beta_{i} - 3/3\beta_{i} < 1 \).

The most difficult scenario is when bounding functions or signs of \( D_{i,j} \) are unknown. In order to derive the iterative learning convergence, it can be seen from (9) that we do not need the exact knowledge about the mapping \( f \). It is adequate to know the bounding knowledge (amplitude and sign) of the gradient \( D(k) \) or \( F(k') \). Although an invariant gradient is assumed for most iterative learning problems, the elements in \( D(k) \) may change sign and take either positive or negative values. Without knowing the exact knowledge of the mapping \( f \), we may not be able to predict the varying signs in the gradient. Now, in the iterative learning law (8) the learning gains will have to change signs according to the gradient. Here the question is how to change the signs of the learning gains when we do not know the signs of the gradient components, i.e., how to search the direction of the gradient, which obviously can only be done in a model free manner if \( f \) is unknown.

A solution to the problem with unknown gradient is to conduct extra learning trials to determine the direction of gradient or the signs of the learning gains directly

\[
\gamma_{i} = [\pm \gamma_{1}, \ldots, \pm \gamma_{n}]^{T}.
\]

where \( \gamma_{i} \) are positive constants. From the derivation (5), when learning gains are chosen appropriately, \(|A| < 1 \) and the learning error reduces. On the other hand, if learning gains are chosen inappropriately, then \(|A| > 1 \) and the error increases after this learning trial. Therefore, several learning trials are adequate for ILT mechanism to determine the correct signs of the gradient.

In general, when there are two gradient components (\( D_{1,j}, D_{2,j} \)), there are 4 sets of signs \( \{1,1\}, \{1,-1\}, \{-1,1\}, \{-1,-1\} \), corresponding to all possible signs of the gradient \( (D_{1,j}, D_{2,j}) \). In such circumstances, at most 4 learning trials are sufficient to find the greatest descending among the four control directions, as shown in Fig. 5.

Similarly, if there are three free tuning parameters, there will be 8 sets of signs in the gradient \( (D_{1,j}, D_{2,j}, D_{3,j}) \), as shown in Fig. 6. In general, if there are \( n \) control tuning parameters, there will be \( n \) gradient components \( D_{i,j} \). Since each \( \gamma_{i} \) takes either positive or negative sign, there will be \( 2^{n} \) combinations and at most \( 2^{n} \) learning trials are required.

In addition to the estimation of the gradient direction or learning direction, the magnitudes of learning gains \( \gamma_{i} \) should also be adjusted to satisfy the learning convergence condition \(|1 - D'_{i} \gamma_{i}| \leq \rho < 1 \). Since the gradient \( D_{i} \) is a function of PID parameters \( k_{i} \), the magnitude of \( D_{i} \) varies at different iterations. When the magnitudes of the gradient is unknown, extra learning trials will be needed to search for suitable magnitudes of learning gains.

In this work, we adopt a self-adaption rule [16] which scales learning gains up and down by a factor \( \zeta = 1.839 \), that is, each component \( \gamma_{i,j} \) will be adjusted to \( \gamma_{i,j}^{*} \) \( / \zeta \). Extra learning trials will be performed with the scaled learning gains. It is reported that the choice of such a scaling factor \( \zeta \) will lead to a linear convergence rate [16], [17].

To facilitate searching and reduce the number of trials, we can also estimate gradient components \( D_{i,j} \) numerically using the values of the objective function and PID parameters obtained from previous two iterations

\[
\hat{D}_{i,j} = \frac{J(k_{i+1}) - J(k_{i+2})}{k_{j+1} - k_{j+2}}.
\]

The learning gain can be revised accordingly as \( \gamma_{i,j} = \lambda_{i,j} \hat{D}_{i,j} \) where \( \lambda_{i,j} \) is a constant gain in the interval of \((0, 1)\). This method is in essence the Secant method along the iteration axis, which can effectively expedite the learning speed[18].

Since gradient direction is a critical issue in searching, and the approximation (11) may not always guarantee a correct sign, we can use the estimation result in (11) partially by retaining the magnitude estimation, while still searching the correct direction

\[
\gamma_{i,j} = \pm \lambda_{i,j} \left| \frac{k_{j+1} - k_{j+2}}{J(k_{i+1}) - J(k_{i+2})} \right|.
\]

### 3.3 Iterative Searching Methods

Three iterative searching methods are considered for ILT in this work with verifications and comparisons. They are \( M_{0} \) – an exhaustive searching method in directions and magnitude, \( M_{1} \) – an exhaustive searching method in directions, \( M_{2} \) – a lazy searching method.

\( M_{0} \) does exhaustive searching in all \( 2^{n} \) directions and exhaustive searching in all \( 2^{n} \) magnitudes using self-adaptation with the factor \( \zeta \). Then PID parameters will be updated using the set that generates the best closed-loop response or yields the
biggest drop of $J$ in that trial. With the best tuned PID parameters as the initial setting, the ILT mechanism enters another run of exhaustive searching for the best response and best PID parameters. The searching process repeats until the stopping criterion is met.

This is the worst case searching where neither the gradient directions nor the gradient magnitudes are available. For each run of searching the greatest descending direction, $4^n$ trials are performed. For PID tuning where $n = 3$, 64 trials are needed for one run of searching. Clearly, this searching method is not efficient.

In $M_1$ the entire searching and updating process is similar to $M_0$ except that the magnitudes of learning gains are determined using the formula (12). Hence it reduces the total number of trials from $4^n$ in $M_0$ to $2^n$ for each run of searching. For PID tuning where $n = 3$, only 8 trials are needed.

$M_2$ does exhaustive searching in directions in the first run of searching, and the magnitudes of learning gains are determined using the formula (12). Then the greatest descending direction will be used for subsequent runs of searching, with the assumption that the mapping $f$ is in general smooth and drastic variations in gradient rarely occur. The exhaustive searching in directions will be activated again when the stopping criterion is met. The searching process will permanently stop if the stop criterion is still met after exhaustive searching in all directions.

The initial magnitudes of learning gains can be set as

$$\gamma(1,0,0,0) = \frac{\gamma_0}{0} [k_p,0,0,0,0],$$

where $\gamma_0$ is a positive constant, and chosen to be 0.1 in this work. $k_p,0,0,0,0$ are initial values of PID parameters determined using any existing PID auto-tuning methods. $\gamma_0$ is calculated with $k_p,0,0,0,0$ and the corresponding closed-loop response.

4. Comparative Studies on Benchmark Examples

In this section we conduct comprehensive tests on 8 benchmark plant models. Four plant models $G_1$ to $G_4$ were used in [13] to compare and validate several iterative tuning methods

$$G_1(s) = \frac{1}{1 + 20s} e^{-5s},$$

$$G_2(s) = \frac{1}{1 + 20s} e^{-20s},$$

$$G_3(s) = \frac{1}{1 + 10s},$$

$$G_4(s) = \frac{1}{1 + 10s + 20s},$$

where $G_1$ and $G_2$ have relatively large normalized time delay (NTD), $G_3$ has high-order repeated poles, and $G_4$ is nonminimum phase. The other four plant models $G_5$ to $G_8$ were used in [19],[20] to validate their auto-tuning methods

$$G_5(s) = \frac{1}{(s + 1)(s + 5)} e^{-0.5s},$$

$$G_6(s) = \frac{1}{(25s^2 + 5s + 1)(5s + 1)} e^{-s},$$

$$G_7(s) = \frac{1}{(s^2 + 2s + 3)(s + 5)} e^{-0.3s},$$

$$G_8(s) = \frac{1}{(s + s + 1)(s + 2)} e^{-0.1s},$$

where $G_5$ is a high-order plant with medium NTD, $G_6$ is a high-order and moderately oscillatory plant with short NTD, $G_7$ is a high-order and heavily oscillatory plant with short NTD, and $G_8$ has both oscillatory and repeated poles.

To make fair comparisons, we choose initial learning gains with form (13) for PID parameters in all case studies. In all searching results, we use $N$ to denote the total number of trials.

4.1 Comparisons between Objective Functions

Objective function ISE (1) has been widely investigated and adopted in PID tuning. The quadratic objective function (2), on the other hand, has more weights to prioritize transient performance requirements. When control requirements are directly concerned with transient response such as $M_p$ or $t_s$, we can only use the quadratic objective function (2). To show the effects of different objective functions and weight selections, we use plants $G_1 - G_4$. To make fair comparisons, the learning process starts from the same initial setting for PID gains generated by Ziegler Nichols (ZN) tuning method [8]. Exhustive searching method $M_0$ is employed. The tuning results through iterative learning are summarized in Table 1. For simplicity only $M_p$ and $t_s$ are taken into consideration in (2). The learning process stops when the drop of an objective function between two consecutive iterations is lower than $\epsilon = 10^{-4}$ for (1) and $\epsilon = 0.01$ for (2). In ISE, the parameters are $T = 100, 300, 500, 200$ s and $t_s = 10, 50, 140, 30$ s respectively [13].

Usually settling time is much greater than overshoot, thus $100M_p$ is used. When plants have much bigger settling time, we can choose $t_s$ instead of $(t_s)^2$ in the objective function, as shown in cases of $G_2$ and $G_3$. By scaling down $t_s$ in objective functions, overshoot decreases as it is weighted more. Meanwhile $t_s$ increases as it is weighted less.

Comparing ISE and quadratic objective function, we can see that latter offers more choices.

4.2 Comparisons between ILT and Existing Iterative Tuning Methods

Now we compare iterative learning based tuning with other iterative tuning methods such as extremum seeking (ES) tuning and iterative feedback tuning (IFT). ZN tuning method is also included for comparison. $G_1 - G_4$ and the same ISE [13] for each model are used. The PID controller parameters given by ZN are used as a starting point for ILT tuning. Exhaustive searching method $M_0$ is employed in ILT.

The results verify that ILT, ES and IFT give very similar responses which are much superior than that of ZN tuning method. Figure 7 shows the ILT results for $G_1$, (a) shows the decreasing objective function $J$, (b) shows decreasing performance indices $M_p$ and $t_s$, (c) shows the variations of the PID parameters, and (d) compares step responses with four tuning methods. Figure 8 shows the searching results of the gradient directions and the variations of the learning gains through self-adaptation. It can be seen that the gradients undergo changes in signs, hence it is in general a difficult and challenging optimization problem.

Table 2 summarizes the comparative results with all four plants $G_1 - G_4$ when 4 tuning methods were applied. The iteration numbers when applying ILT for $G_1 - G_4$ are 786, 512, 1024 and 1216 respectively.

Although ILT shares similar performance as ES and IFT, it
is worth to highlight some important factors in the tuning process. In ES tuning, there are more than 10 design parameters to be set properly. From [13], design parameters take rather specific values. In IFT, the initial values of the PID parameters must be chosen in such a way as to give an initial response that is very slow and with no overshoot. Further, in IFT the transient performance is purposely excluded from the objective function by choosing a sufficiently large $\tau$. On the contrary, in ILT only initial learning gains need to be preset, and we choose all three learning gains with a uniform value $\gamma_0 = 0.1$ in (13) for the ease of ILT design. In fact, by choosing initial learning gains with different values, we can achieve much better responses than those in Table 2. Needless to mention that ILT can handle both types of objective functions (1) and (2) with the flexibility to highlight transient behaviors.

4.3 Comparisons between ILT and Existing Auto-Tuning Methods

PID auto-tuning methods [8],[19],[20] provided several effective ways to determine PID parameters. In this subsection we compare ILT with the auto-tuning method [8] based on internal model control (IMC), and the auto-tuning method [20] based on pole-placement (PPT) which shows superior performance than [19]. Comparisons are made based on plants $G_5 - G_8$ which were used as benchmarks. Using ISE as the objective function and searching method $M_s$ in ILT, the tuning results are summarized in Table 3, where the start points of PID parameters are adopted from the tuning results in [19]. Comparing with the results auto-tuned by the IMC method and PPT method, ILT achieves better performance after learning.

Table 1 Control performances of $G_1 - G_4$ using the proposed ILT method.

<table>
<thead>
<tr>
<th>Plant</th>
<th>$J$</th>
<th>PID Controller</th>
<th>$100M_p$</th>
<th>$\tau_s$</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_1$</td>
<td>ISE</td>
<td>$3.59 + 0.13s^{-1} + 7.54s^{-2}$</td>
<td>4.48</td>
<td>21.35</td>
<td>768</td>
</tr>
<tr>
<td></td>
<td>$(100M_p)^2 + 0.5t_s$</td>
<td>$3.38 + 0.13s^{-1} + 7.05s^{-2}$</td>
<td>1.99</td>
<td>12.17</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>$(100M_p)^2 + 0.1t_s$</td>
<td>$3.25 + 0.12s^{-1} + 6.30s^{-2}$</td>
<td>0.63</td>
<td>12.83</td>
<td>832</td>
</tr>
<tr>
<td></td>
<td>$(100M_p)^2 + 0.01t_s$</td>
<td>$3.71 + 0.11s^{-1} + 9.11s^{-2}$</td>
<td>0.53</td>
<td>22.24</td>
<td>512</td>
</tr>
<tr>
<td>$G_2$</td>
<td>ISE</td>
<td>$0.93 + 0.03s^{-1} + 5.67s^{-2}$</td>
<td>0.71</td>
<td>50.67</td>
<td>512</td>
</tr>
<tr>
<td></td>
<td>$(100M_p)^2 + t_s$</td>
<td>$0.99 + 0.032s^{-1} + 6.87s^{-2}$</td>
<td>1.06</td>
<td>47.99</td>
<td>1600</td>
</tr>
<tr>
<td></td>
<td>$(100M_p)^2 + 0.2t_s$</td>
<td>$1.05 + 0.028s^{-1} + 9.79s^{-2}$</td>
<td>0.29</td>
<td>82.74</td>
<td>512</td>
</tr>
<tr>
<td></td>
<td>$(100M_p)^2 + 0.1t_s$</td>
<td>$1.03 + 0.029s^{-1} + 9.18s^{-2}$</td>
<td>0.20</td>
<td>83.61</td>
<td>640</td>
</tr>
<tr>
<td>$G_3$</td>
<td>ISE</td>
<td>$0.64 + 0.012s^{-1} + 11.3s^{-2}$</td>
<td>0.49</td>
<td>137.13</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>$(100M_p)^2 + t_s$</td>
<td>$0.76 + 0.013s^{-1} + 16.65s^{-2}$</td>
<td>1.93</td>
<td>120.56</td>
<td>576</td>
</tr>
<tr>
<td></td>
<td>$(100M_p)^2 + 0.2t_s$</td>
<td>$0.85 + 0.014s^{-1} + 25.77s^{-2}$</td>
<td>0.66</td>
<td>212.04</td>
<td>192</td>
</tr>
<tr>
<td></td>
<td>$(100M_p)^2 + 0.1t_s$</td>
<td>$0.83 + 0.014s^{-1} + 24.91s^{-2}$</td>
<td>0.62</td>
<td>212.76</td>
<td>192</td>
</tr>
<tr>
<td>$G_4$</td>
<td>ISE</td>
<td>$5.01 + 0.092s^{-1} + 25.59s^{-2}$</td>
<td>3.05</td>
<td>212.76</td>
<td>1216</td>
</tr>
<tr>
<td></td>
<td>$(100M_p)^2 + 0.25t_s$</td>
<td>$4.33 + 0.075s^{-1} + 22.19s^{-2}$</td>
<td>1.81</td>
<td>18.63</td>
<td>512</td>
</tr>
<tr>
<td></td>
<td>$(100M_p)^2 + 0.1t_s$</td>
<td>$3.89 + 0.071s^{-1} + 22.28s^{-2}$</td>
<td>1.70</td>
<td>20.56</td>
<td>384</td>
</tr>
<tr>
<td></td>
<td>$(100M_p)^2 + 0.01t_s$</td>
<td>$4.51 + 0.075s^{-1} + 23.96s^{-2}$</td>
<td>0.06</td>
<td>19.27</td>
<td>1216</td>
</tr>
</tbody>
</table>

Fig. 7 ILT performance for $G_1$. (a) The evolution of the objective function; (b) The evolution of overshoot and settling time; (c) The evolution of PID parameters; (d) The comparisons of step responses among ZN, IFT, ES and ILT, where IFT, ES and ILT show almost the same responses.
learning is summarized in Tables 4 and 5. It can be seen that $M_1$ and $M_2$ achieve similar performance, which is slightly inferior to $M_0$. However, comparing with $M_0$ the learning trial numbers in $M_1$ and $M_2$ have been significantly reduced.

4.5 ILT for Sampled-Data Systems

A promising feature of ILT is the applicability to sampled-data or discrete-time systems. To illustrate how ILT works for digital systems, consider plant $G_1$ which can be discretized using sampler and zero order hold.

$$G_4(z) = \frac{(-2.5e^{-0.05T_s} + 1.5e^{-1T_s} + 1)z}{z^2 - (e^{-0.1T_s} + e^{-0.05T_s})z + e^{-0.15T_s}},$$

where the sampling period $T_s = 0.1$ s. The digital PID controller is used. Choose again (14) as the objective function, and use ZN to generate the initial values for PID parameters. The closed-loop responses using ILT are summarized in Table 6. For comparison all three searching methods $M_0$, $M_1$ and $M_2$ are used.

It can be seen that in all cases the control responses have been improved drastically, especially the reduction in overshoot.

5. Real-Time Implementation

In order to show the applicability and effectiveness of the proposed ILT, real-time experiment has been carried out on a couple tank.

5.1 Experimental Setup and Plant Modelling

The couple tank equipment consists of two small perspex tower-type tanks and interconnected through an hole which yields a hydraulic resistance (Fig. 9). The experimental setup in this work was configured such that the level is measured from the tank-2 while the water is pumped into the tank-1 as the control input. The outlet of tank-2 is used to discharge the water into the reservoir. The measurement data is collected from couple tank using NI data acquisition card USB-6088. The control method is programmed using Labview. A window-type smoothing filter using 100 samples is implemented to mitigate measurement noise. The sampling period is 0.125 second.
A step response is conducted to approximate the couple tank dynamics with a first order plus time delay model:\cite{18}

\[ G(s) = \frac{k}{\tau s + 1} e^{-\tau L} \]  

(15)

where \( k \) is the plant DC gain, \( \tau \) is the time constant, \( L \) is the transportation lag. As shown in Fig. 10, after conducting a step response the obtained plant is

\[ G(s) = \frac{168}{1 + 65s} e^{-3.6s} \]  

(16)

5.2 Application of ILT Method

In the experiments, we conducts a series of tests to investigate ILT. The quadratic objective function is

\[ J = (100M_p)^2 + q\tau_i^2 \]

where two values \( q = 0.1 \) and \( q = 0.01 \) are used. The ISE is

\[ \frac{1}{T - t_0} \int_{t_0}^{T} e^2 dt \]
where $T = 124$ s and $t_0 = 27.4$ s. All three search methods $M_0$, $M_1$ and $M_2$ are applied.

It is observed in the experiment that the calculated gradient components (11) or (12) may be singular some times. This is due to the presence of measurement noise and the closeness of the values of $J$ at two adjacent iterations. On the other hand, the learning process may become sluggish when the PID parameters at two adjacent iterations are too close, yielding a very lower learning gains $\gamma_{ij}$. To solve these two problems, a constraint is applied when updating the learning gains

$$c_1 |k| \leq \gamma_{ij} \leq c_2 |k|,$$

where $0 \leq c_1 < c_2 \leq 1$. The upper and lower learning gain bounds are made in proportion to PID parameters. The rationale is clear. When a controller parameter is bigger, we can update it with a larger bound without incurring drastic changes. If setting absolute bounds for the learning gain, an overly small bound would limit the parameter updating speed, and an overly large bound would make the constraint ineffective. In the real-time application we consider 2 sets of boundaries

- $C_1$: $c_1 = 0.02$  $c_2 = 0.2$,
- $C_2$: $c_1 = 0.05$  $c_2 = 0.4$.

The initial PID controller is found using ZN tuning method

$$0.11 + \frac{0.011}{s} + 0.26s.$$

The transient performance with ZN tuned PID is $100M_p = 30.02$ and $t_s = 58.73$ s. ILT is applied to improve the performance.

### 5.3 Experimental Results

The results are summarized in Table 7. It can be seen that searching methods $M_1$ and $M_2$ can significantly reduce the number of learning iterations while $M_p$ and $t_s$ can be maintained at almost the same level. By changing the weight for $t_s$, the final overshoot $M_p$ and settling time $t_s$ can be adjusted. The constraint $C_1$ can also reduce the trial number, because higher limits in learning gains will expedite the learning progress. In experiments, it is observed that $M_2$ can be further simplified by removing the last run of exhaustive searching, so long as the variation of $J$ reaches the preset threshold $\varepsilon$.

### 6. Conclusion

A new PID auto-tuning method is developed and compared with several well established auto-tuning methods including ZN, IFT, ES, PPT, IMC. Iterative learning tuning provides a sustained improvement in closed-loop control performance, and offers extra degrees of freedom in specifying the transient control requirements through a new objective function.

The iterative learning tuning method proposed in this work can be further extended to various optimal designs for controllers, owing to its model free nature. For instance, by iterative searching and task repeatability, we can tune the parameters of lead and lag compensators, filters, observers, and use both time domain and frequency performance indices in objective functions.

The future issues associated with the iterative learning tuning approach we are looking into include the robustness of ILT when the control process is not completely repeatable, the incorporation of various practical constraints, improvement of searching efficiency for high dimension multi-loop control systems, as well as ILT for closed-loop systems with lower sta-
bility margins such as the sampled-data system with large time delays.

References

Appendix
The nonlinear mapping f will be explored for three cases with underdamped, overdamped and critical damped characteristics. Underdamped Case
When the system parameters satisfy \((a+kk_p)^2-4(b+kk_p)<0\), the closed-loop system shows underdamped response with two complex poles. The system response in the time domain is
\[
y(t) = A - A \sqrt{1 + A^2 e^{-\alpha t}} \cos(\beta t - \tan^{-1} \lambda),
\]
where
\[
A = \frac{p}{b + p}, \quad \alpha = \frac{a + d}{2}, \quad \beta = \frac{1}{2} \sqrt{4(b + p) - (a + d)^2}, \quad \lambda = \frac{4p^2 + 2bd - pd}{p \sqrt{4(b + p) - (a + d)^2}}
\]
with \(p = kk_p\) and \(d = kk_d\). It is easy to see that the steady-state output is \(A\) and
\[
M_p = -\sqrt{1 + A^2} e^{-\alpha t}, \quad \cos(\beta t - \tan^{-1} \lambda),
\]
\[
t_s = \frac{1}{\alpha} \ln \left(\frac{50}{\sqrt{1 + A^2}}\right),
\]
where the peak time is
\[
t_p = \frac{1}{\beta} \tan^{-1} \frac{\lambda / \beta - \alpha / \beta}{(\alpha - \beta) / \beta + d / \beta}, \quad \text{if} \quad \frac{(\lambda - d) / \beta + (a - d) / \beta}{(\alpha - \beta) / \beta + d / \beta} \geq 0,
\]
\[
t_p = \frac{1}{\beta} \pi + \tan^{-1} \frac{\lambda / \beta - \alpha / \beta}{(\alpha - \beta) / \beta + d / \beta}, \quad \text{if} \quad \frac{(\lambda - d) / \beta + (a - d) / \beta}{(\alpha - \beta) / \beta + d / \beta} < 0.
\]
Note that the settling time \(t_s\), corresponding to 2% of the final steady state value, is calculated from the worst case condition
\[
\left|\sqrt{1 + A^2} e^{-\alpha t} \cos(\beta t - \tan^{-1} \lambda)\right| < 0.02
\]
by ignoring the factor \(\cos(\beta t - \tan^{-1} \lambda)\).

Overdamped Case
When the system parameters satisfy \((a+kk_p)^2-4(b+kk_p) > 0\), the closed-loop shows overdamped response and the closed-loop system has two distinct real poles. The system response in the time domain is
\[
y(t) = A - \frac{1}{2} \left[\frac{d}{\beta} - \frac{p}{\beta(a + \beta)}\right] e^{-\alpha t} + \frac{1}{2} \left[\frac{d}{\beta} - \frac{p}{\beta(a - \beta)}\right] e^{-\beta t},
\]
where
\[
\alpha = \frac{a + d}{2}, \quad \beta = \frac{1}{2} \sqrt{(a + d)^2 - 4(b + p)}.
\]
The relationship \(\alpha > \beta\) always holds since \(b + p > 0\). Therefore the steady state is still \(A\).
First derive the peak overshoot. From
\[
\frac{dy}{dt} = -\frac{1}{2\beta} \left[(da + db - p)e^{-\alpha t} + (da + db + p)e^{-\beta t}\right],
\]
and let \(dy/dt = 0\) we can obtain the peak time
\[
t_p = \frac{1}{2\beta} \ln \left(\frac{da - db - p}{da + db - p}\right).
\]
The output has overshoot if and only if \(a, b, p, d\) lie in the domain...
When the system parameters lie in the domain $\mathcal{D}$,

$$M_p = \frac{y(t_p) - A}{A} = \frac{1}{2\beta p} \left( \frac{d\alpha - d\beta - p}{d\alpha + d\beta - p} \right)^{\frac{3}{2}} \left( \alpha + \beta \right) \left( d\alpha - d\beta - p \right) - \left( \alpha - \beta \right) \left( d\alpha + d\beta - p \right) \right).$$

Now derive the settling time. Let $\frac{\sinh A}{A} < 0.02$ for all $t \geq t_s$. If the overshoot exceeds 2%, then $t_s$ is the bigger real root of the following equation

$$-\frac{1}{2} \left[ \frac{d}{\beta} - \frac{p}{\beta(\alpha + \beta)} \right] e^{-\alpha t} + \frac{1}{2} \left[ \frac{d}{\beta} - \frac{p}{\beta(\alpha - \beta)} \right] e^{-\beta t} = 0.02 \frac{p}{b + p};$$

otherwise, $t_s$ is the unique real root of the following equation

$$-\frac{1}{2} \left[ \frac{d}{\beta} - \frac{p}{\beta(\alpha + \beta)} \right] e^{-\alpha t} + \frac{1}{2} \left[ \frac{d}{\beta} - \frac{p}{\beta(\alpha - \beta)} \right] e^{-\beta t} = -0.02 \frac{p}{b + p}.$$

Note that in neither cases can we obtain the analytic expression of $t_s$.

### Critical Damped Case

When the system parameters satisfy $(a + k_k^2 - 4(b + k_k^2) = 0$, the closed-loop system shows critical damped response with two identical real poles. The system response in the time domain is

$$y(t) = A - \frac{p + (p - da)t}{b + p} e^{-at} \quad (A.5)$$

with the steady-state $A$.

First derive the peak overshoot. Letting

$$\frac{dy}{dt} = \frac{\alpha^2(d + (p - da)t)}{b + p} e^{-at} = 0$$

yields a unique solution

$$t_p = \frac{-d}{p - da} = -\frac{2d}{2p - ad - d^2},$$

implying that the overshoot exists if and only if $a^2 < 4b + d^2$.

The overshoot is

$$M_p = \frac{d^2(a + d)^2 - p(-a^2 + 4b + 3d^2 + 2ad)}{p(a - a^2 + 4b + d^2)} e^{\frac{2d}{a - a^2 + 4b + d^2}}.$$