An Optimal PID Control Algorithm for Training Feedforward Neural Networks

Xingjian Jing and Li Cheng

***Abstract*—The training problem of feedforward neural networks (FNNs) is formulated into a proportional integral and derivative (PID) control problem of a linear discrete dynamic system in terms of the estimation error. The robust control approach greatly facilitates the analysis and design of robust learning algorithms for multiple-input–multiple-output (MIMO) FNNs using robust control methods. The drawbacks of some existing learning algorithms can therefore be revealed clearly, and an optimal robust PID-learning algorithm is developed. The optimal learning parameters can be found by utilizing linear matrix inequality optimization techniques. Theoretical analysis and examples including function approximation, system identification, exclusive-or (XOR) and encoder problems are provided to illustrate the results.**

***Index Terms*—Feedforward neural networks, linear matrix inequality (LMI), proportional integral and derivative (PID) controller, robust learning.**

I. Introduction

TRAINING of a feedforward neural network (FNN) has been extensively studied in the literature [1], [2], [16], [20], [24]–[29]. The gradient descent algorithm such as back propagation (BP), as a basic method for training FNNs in many areas, for example function approximation, system identification, pattern recognition and control, etc., searches the parameter space (weights and thresholds) of the network in the steepest descent way to minimize the error between the network output and the desired output. The main drawbacks could be its slow convergence speed (unstable in some cases) and its inability to ensure global minimum. For the noisy input-output data in system identification and function approximation, the performance of a traditional BP may be even worse. Improvement methods, including introduction of a momentum term, utilization of standard optimization techniques [e.g., quasi-Newton, conjugate-gradient, recursive least square, and Levenberg-Marquardt (LM)], and adaptability of learning rates, have been frequently reported in the literature [2]–[7]. However, many improved methods may incur additional limitations for the algorithms, for example, dependence on heuristic knowledge, high storage and memory requirements, complex computation costs, or difficulty in scheduling the learning rate or ensuring stability, etc. The methods based on Lyapunov stability are also studied recently such as in [8] to guarantee the global convergence of the learning algorithm but without considering noise effects. Importantly, the learning problem is still treated as an optimization but not as a control problem. Extended Kalman filter and H_∞ filter methods are also utilized for the development of new learning algorithms to cope with noisy data and more robust convergence [9], [10]. However, the computation cost is obviously increased, and the global convergence is not ensured.

In this paper, the robust learning problem of FNNs is treated by a novel robust proportional integral and derivative (PID) control approach in order to achieve faster, global, and more robust convergence (for noisy data particularly in function approximation and nonlinear system identification). The training problem of FNNs is formulated into a robust control problem of a linear discrete dynamic system in terms of the estimation error of the network. The weight update law is transformed into a "virtual" control to be designed, while the noise in the data is mapped into bounded uncertainties. Therefore, this can greatly facilitate the analysis and design of robust learning algorithms for FNNs using many available robust control methods and optimization techniques such as linear matrix inequality (LMI). With this approach, an optimal PID training algorithm is consequently proposed, and the learning parameters can be determined optimally by minimizing a performance index using LMI techniques. Compared with existing learning algorithms such as BP and LM-BP, etc, the new optimal PID-learning (PID-L) algorithm can provide more robust and faster convergence particularly in dealing with noisy data in function approximation and system identification, and is easy to implement as a BP algorithm. Several existing BP-type algorithms can be regarded as special cases of the new PID-L algorithm. Simulations and comparisons are provided to illustrate the effectiveness of the proposed PID-L method.

It should be noted that the PID robust control approach proposed in this study is different from the existing techniques used in the design of adaptive controllers or filters using FNNs [17]. In the latter methods, the FNN model is usually incorporated in a closed-loop feedback control system such that online adaptive parameter estimation can be achieved in the context of Lyapunov stability. However, the proposed approach is aimed at directly training the FNN in an open-loop situation with only the available input output data and therefore can be used for any purposes (e.g., system identification, function approximation, pattern recognition, and control, etc). Moreover, PID-like learning algorithms for SIMO NNs were already studied in [23] and [26], where either the NNs were reconstructed to

Manuscript received September 26, 2011; revised January 9, 2012; accepted March 20, 2012. Date of publication April 17, 2012; date of current version February 6, 2013. This work was supported in part by a GRF project of Hong Kong RGC (Ref 517810) and internal research funds of Hong Kong Polytechnic University.

The authors are with the Department of Mechanical Engineering, Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: xingjian.jing@polyu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2012.2194973

act as a PID form function, or the PID is used in a simple gradient descent manner. However, the PID training algorithm of this study is developed in a novel robust control scheme by casting the FNN training into a "virtual" control system. The robust control approach was already preliminarily investigated in [30]. The PID-L in this study is much simpler and easier to implement, generic, and flexible to different training tasks, applicable to MIMO FNN problems, and shown to be globally convergent.

II. PID-Learning Approach

A. Problem Formulation

PID controllers are extensively accepted in practice for its simplicity in design and implementation and robustness to noise and disturbance. A faster and more robust learning algorithm for FNNs is derived by applying the PID control method in this section.

Consider a FNN with p inputs and r outputs described by

$$y = f(C_1, C_2, X) \tag{1}$$

where $X = [x_1, x_2, \ldots, x_{\mathrm{p}}]^{\mathrm{T}}$ is supposed to be in a compact set $U \subset R^{\mathrm{p}}$, $y \in R^{\mathrm{r}}$, C_1 and C_2 are real-valued vectors denoting different weights of the network, which can be different weights in different layers or just different weights by heuristic grouping or simply written into one vector denoted by C. Equation (1) may represent any multilayer FNN with appropriate nonlinear activation functions of bounded derivatives (with respect to X, C_1 and C_2). It is supposed that for any nonlinear function h: $U \to R^{\mathrm{r}}$ and $\varepsilon_q > 0$, there exist optimal weights C_1^* and C_2^* such that

$$|h(X) - f(C_1^*, C_2^*, X)| < \varepsilon_q \tag{2}$$

($|*|$ is the Euclidean norm). This is a reasonable assumption based on the approximation theory [11], [18], [22]. At time n, the measured output of the nonlinear function can be written as

$$y(n) = f(C_1^*, C_2^*, X_n) + \varepsilon(n) \tag{3}$$

where $\varepsilon(n)$ is a noise process. Suppose $\varepsilon(n)$ is uncorrelated to the input and upper bounded. Given a series of input-output data, the task is to find a weight update law

$$\hat{C}_1(n+1) = \hat{C}_1(n) + \Delta\hat{C}_1(n) \tag{4a}$$
$$\hat{C}_2(n+1) = \hat{C}_2(n) + \Delta\hat{C}_2(n) \tag{4b}$$

with any initial values $\hat{C}_1(0)$ and $\hat{C}_2(0)$ for the estimated FNN $\hat{y}(n) = \hat{f}(\hat{C}_1(n), \hat{C}_2(n), X_n)$ such that the estimation error

$$e(n) = \hat{y}(n) - y(n) \tag{5}$$

is asymptotically convergent to a small region around zero.

For a variable $x(n)$, $\Delta x(n) = x(n+1) - x(n)$ in this study. It can be derived from (5) that

$$\begin{aligned} e(n+1) &= e(n) + \hat{y}(n+1) - \hat{y}(n) - y(n+1) + y(n) \\ &= e(n) + \hat{f}\left(\hat{C}_1(n+1), \hat{C}_2(n+1), X_{n+1}\right) \\ &\quad - \hat{f}\left(\hat{C}_1(n), \hat{C}_2(n), X_n\right) - \Delta y(n). \end{aligned} \tag{6}$$

The term $\Delta y(n)$ is unknown at n. Noting that (4a) and (4b) and $X_{\mathrm{n}+1} = X_{\mathrm{n}} + \Delta X_{\mathrm{n}}$, the term $\hat{f}(\hat{C}_1(n+1), \hat{C}_2(n+1), X_{n+1})$ can be expanded by Taylor series around the previous estimated weights and input $(\hat{C}_1(n), \hat{C}_2(n), X_n)$ as

$$\begin{aligned} &\hat{f}\left(\hat{C}_1(n+1), \hat{C}_2(n+1), X_{n+1}\right) \\ &\quad = \hat{f}\left(\hat{C}_1(n), \hat{C}_2(n), X_n\right) + \hat{f}_{\hat{C}_1}\Delta\hat{C}_1(n) + \hat{f}_{\hat{C}_2}\Delta\hat{C}_2(n) \\ &\qquad + \hat{f}_{X_n}\Delta X_n + \sigma_n\left(\hat{C}_1(n), \hat{C}_2(n), X_n\right) \end{aligned} \tag{7}$$

where $\hat{f}_x = \partial\hat{y}(n)/\partial x(n)$, $\sigma_n(\hat{C}_1(n), \hat{C}_2(n), X_n)$ represents all the remaining terms in Taylor series expansion. To ensure the accuracy of the linear approximation in (7) with Taylor series, the difference terms $\Delta\hat{C}_1(n)$, $\Delta\hat{C}_2(n)$, and ΔX_n should be around zero. However, the residual $\sigma_n(\cdot)$ can always be upper bounded in the compact set$\mathbb{U}$ (see more discussions in [12], [19]). Then, from (6) and (7), it can be derived that

$$e(n+1) = e(n) + B(n) \cdot u(n) + w(n) \tag{8}$$

where $B(n) = [B_1(n)B_2(n)] = [\hat{f}_{\hat{C}_1}\hat{f}_{\hat{C}_2}]$

$u(n) = [\Delta\hat{C}_1(n)^T\Delta\hat{C}_2(n)^T]^T$,

$$w(n) = \hat{f}_{X_n}\Delta X_n - \Delta y(n) + \sigma_n(\cdot).$$

Equation (8) can be regarded as a discrete, linear, time-varying system. The control input $u(n)$ to be designed is the weight update law. The term $\sigma_n(\cdot)$ denotes an unknown disturbance input, which can be designed as small as possible if the control law $u(n)$ is properly chosen and ΔX_n is sufficiently small. Since ΔX_n and $\Delta y(n)$ are usually not known at n, the term $\hat{f}_{X_n}\Delta X_n - \Delta y(n)$ can be regarded as the other disturbance input. Obviously, $w(n)$ inevitably affect the dynamic evolution of the error system. From the control point of view, a simple control law $u(n)$ (i.e., the weight update law) without careful consideration of these disturbance effects will definitely result in bad performance including instability.

To illustrate this point, consider (8) when the control input $u(n)$ is designed as

$$u(n) = -\eta_{BP}B(n)^T e(n). \tag{9}$$

This is the traditional BP algorithm. For convenience in discussion, consider (8) as a scalar system first. Clearly, if the term $w(n)$ can be neglected and $|(1 - \eta_{BP}B(n)B(n)^T)| \leq \lambda < 1(\exists\lambda)$ holds, the learning algorithm (9) can work well, because in this case, the error dynamics will be

$$e(n+1) \approx \left(I - \eta_{BP}B(n)B(n)^T\right)e(n)$$

which is asymptotically stable if $|(1 - \eta_{BP}B(n)B(n)^T)| \leq \lambda < 1$ holds. For dynamic system identification or function approximation with changing input-output and noisy data, the term $w(n)$ cannot be simply neglected, and it is practically difficult to find an appropriate η_{BP} to guarantee $|(1 - \eta_{BP}B(n)B(n)^T)| < 1$ all the time. Similarly, when $e(n)$ in (8) is a vector instead of a scalar, it will be more difficult for (9) to stabilize such an uncertain time-varying system.

Thus, the convergence of the algorithm (9) could not always be guaranteed with a simple setting for the learning rate η_{BP}. Obviously, (8) provides a useful insight into the drawbacks of the existing learning algorithms for FNNs such as some BP-type algorithms. The robust control point of view offers the possibility of addressing these important (but not fully investigated or nearly neglected) issues by adopting robust control techniques.

B. Robust PID-Learning Algorithm

As discussed before, a better controller for (8) should consider carefully the effects incurred by the uncertain disturbance term $w(n)$. For this reason, the PID controller is adopted for its well-known simplicity and robustness in practice. It is known that the PID controller is an extensively used method in practical applications of many fields, which is robust to noise and disturbance of different properties and easy to implement. A PID controller has three terms, i.e., proportional (P), integral (I), and derivative (D) parts. Properly tuning the controller parameters can result in satisfactory performance for many linear or nonlinear systems. The continuous PID controller can be written as

$$V(s) = (K_{\rm p} + k_{\rm i}/s + k_{\rm d}s)E(s) \tag{10}$$

where V(s) and E(s) are the Laplace transforms of v(t) and e(t), respectively, $K_{\rm p}$, $k_{\rm i}$, and $k_{\rm d}$ are all real-valued diagonal matrices of r dimensions. To apply it to the discrete system (8), considering the commonly used backward difference method, i.e., $s = (1 - z^{-1})/T_{\rm s}$, the corresponding discrete PID is

$$\begin{aligned} v(n) = v(n-1) &+ K_{\rm p}\left[e(n) - e(n-1)\right] + k_{\rm i}T_{\rm s}e(n) \\ &+ (k_{\rm d}/T_{\rm s})\left[e(n) - 2e(n-1) + e(n-2)\right] \end{aligned} \tag{11a}$$

which can be rewritten as

$$\begin{aligned} v(n) = v(n-1) &+ K_{\rm p}\Delta e(n-1) + K_{\rm i}e(n) \\ &+ K_{\rm d}\left[\Delta e(n-1) - \Delta e(n-2)\right] \end{aligned} \tag{11b}$$

where $K_{\rm i} = k_{\rm i}T_{\rm s}$, $K_{\rm d} = k_{\rm d}/T_{\rm s}$, $\Delta e(n-1) = e(n) - e(n-1)$.

With the PID controller (11b), the PID-L algorithm for the error dynamic system (8) is given by

$$u(n) = -\begin{bmatrix} (1-\alpha)B_1^T\left(B_1B_1^T\right)^{-1}\cdot I_{r,r} \\ \alpha B_2^T\left(B_2B_2^T\right)^{-1}\cdot I_{r,r} \end{bmatrix} v(n) \tag{12}$$

where $I_{r,r}$ represents a unit diagonal matrix of $r \times r$ dimensions, and $1 > \alpha > 0$. Note that the network weights to be estimated are divided into different groups with different weighting parameters α and $1-\alpha$ in the learning rate in (12) (when $\alpha = 0.5$ it will be the usual case). In case that $(B_1B_1^T(n))^{-1}$ is singular, it can be replaced by $(B_1B_1^T(n) + \delta\cdot I)^{-1}$, where δ is a small positive number.

An important task in the implementation of the weight update law (12) is to determine the optimal PID parameters such that the controlled error dynamics in (8) is robust to the unknown disturbance represented by the term $w(n)$. Note that $\Delta y(n)$ can be approximated by the first-order Taylor series expansion $f_{X_n}\Delta X_n$. Thus, it can be obtained that $w(n) = (\hat{f}_{X_n} - f_{X_n})\Delta X_n + \sigma_n(\cdot)$. If the input is slowly changing or the estimated weights are around the real values, $(\hat{f}_{X_n} - f_{X_n})\Delta X_n$ will be around zero. The terms $|\Delta X_n|$ and $|\hat{f}_{X_n} - f_{X_n}|$ can both be bounded in the compact set$\mathbb{U}$, which is a common assumption in the literature [11], [12]. Therefore, it can be supposed that

$$w(n) \in L_{2e}. \tag{13}$$

It is noted that $w(n)$ will go to zero when the network weights approach to the ideal weights. Thus, it could also hold that $w(n) \in L_2$. It should be noted that, with different knowledge on $w(n)$, different robust control strategy can be attempted to design a robust learning algorithm (including static or dynamic ones) for a specific convergence performance of the error dynamics (8). This is an advantage of the methodology proposed in this paper.

To guarantee the asymptotical stability of the error dynamics with the learning law (12) and to find the optimal PID parameters, the following result can be achieved using the PID controller (11b).

Theorem 1: Given a scalar $\gamma > 0$, and sufficient number q of hidden units in the estimated FNN, the PID-L algorithm in (11b) and (12) can drive the estimation error in (8) to globally asymptotically converge to zero satisfying the performance $\|e(n)\|_2 < \gamma\|w_n\|_2$, if there exist any matrices of appropriate dimensions $Q > 0$, U_1, U_2, and Y, such that

$$\begin{bmatrix} U_1^T+U_1 & U_2-U_1^T+Q\overline{A}^T+Y^T\overline{B}^T & 0 & QG^T & U_1^T \\ * & -U_2-U_2^T & \overline{B}_1 & 0 & U_2^T \\ * & * & -\gamma^2 I & 0 & 0 \\ * & * & * & -I & 0 \\ * & * & * & * & -Q \end{bmatrix} < 0 \tag{14}$$

and the optimal PID parameters are given as: α is a small number satisfying $1 > \alpha > 0$, and

$$\begin{aligned} \begin{bmatrix} K_p \\ K_i \\ K_d \end{bmatrix} &= \begin{bmatrix} 0_{r,r} & -I_{r,r} & -2I_{r,r} \\ I_{r,r} & I_{r,r} & I_{r,r} \\ 0_{r,r} & 0_{r,r} & I_{r,r} \end{bmatrix} \\ &\times \left\{ \left(YQ^{-1} - \begin{bmatrix} I_{r,r} \\ I_{r,r} \\ I_{r,r} \end{bmatrix}^T \right)^T - \begin{bmatrix} -2I_{r,r} \\ I_{r,r} \\ 0_{r,r} \end{bmatrix} \right\} \end{aligned} \tag{15}$$

where

$$\overline{A} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \otimes I_{r,r}, \quad \overline{B} = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \otimes I_{r,r},$$

$$\overline{B}_1 = \begin{bmatrix} 1 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \otimes I_{r,r},$$

$$G = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}^T \otimes I_{r,r}, \quad w_n = \begin{bmatrix} w(n) \\ w(n-1) \end{bmatrix}.$$

Proof: See Appendix A.

If $w(n) \in L_{2e}$, then the norm in $\|e(n)\|_2 < \gamma\|w_n\|_2$ is L_{2e} norm. If $w(n) \in L_2$, then the norm is L_2 norm. Note that U_1, U_2, and Y can be any matrix variables and $\overline{A}, \overline{B}, \overline{B}_1, C$ are some auxiliary matrices, which are derived from the error dynamics (8) for the computation of the PID parameters with the LMI optimization technique (referred to Appendix A). Therefore, the optimal robust PID-L algorithm is given by (11), (12), and (15). Although the PID controller (11b) is focused in this study, similar results can be established for discrete PID controllers of other forms (e.g., via the bilinear transformation).

C. Analysis and Discussions

The PID-L algorithm above has the following properties.

(i) The algorithm in (12) includes several existing BP-type algorithms as special cases. Note that the BP algorithm in (9) simply acts as a proportional controller with $K_{\mathrm{p}} = \eta_{BP} B(n) B(n)^{\mathrm{T}}$. The PID controllers in (11b) and (11c) both have a term $v(n-1)$ or $v(n-2)$, which correspond to the momentum terms in the existing momentum BP-type algorithms [13]. The BP algorithm using Newton's method with quadratic approximation can be written as

$$u(n) = -\left(B(n)^T B(n)\right)^{-1} B(n)^T e(n)$$

which can be regarded as a proportional controller in the PID-L with $K_{\mathrm{p}} = B(n)(B(n)^{\mathrm{T}}B(n))^{-1}B(n)^{\mathrm{T}}$. Similarly, the LM-BP algorithm can be simply written as

$$u(n) = -\left(B(n)^T B(n) + \mu I\right)^{-1} B(n)^T e(n).$$

When μ is small, it is the quasi-Newton's algorithm above, and when μ is large, it is a gradient descent one. Obviously, the LM-BP is also included in the general PID-L algorithm (12). Therefore, the new PID-L algorithm in (12) provides a general scheme for the learning algorithm design and takes into consideration the noise effect and also the uncertain effects that are incurred by the changing input and output. The weight update law (12) with the PID parameters given by Theorem 1 can drive the estimated error of the FNN to around zero satisfying a predefined H_∞ level. Hence, the PID-L algorithm in (12) should have faster and more robust convergent performance theoretically.

Importantly, improved versions of these BP-type algorithms mentioned above can always be found by replacing the error $e(n)$ with the composite PID error $v(n)$ [in (11b) or (11c)]. In this case, the BP-type algorithms mentioned above are still included in the corresponding improved ones as special cases. For example, an improved PID LM-BP can be given by

$$u(n) = -\left(B(n)^T B(n) + \mu I\right)^{-1} B(n)^T v(n).$$

When K_{i} and $K_{\mathrm{d}} = 0$, it becomes the original one. This provides another new robust control approach to the design of learning algorithms.

It should be emphasized that although the error dynamics in (8) is a time-varying system, the LMI in (14) is independent of the time-varying parameters because all the involved system matrices are constant. That is, the LMI in (14) needs only run one time to find the optimal PID parameters before a learning task instead of repeating computation at each step. Hence, the computation cost of the proposed PID-L algorithms is as simple as the well-known gradient descent BP algorithm.

(ii) Substituting (12) into (8) gives $e(n+1) = e(n) - v(n) + w(n)$ (also see it in Appendix A). Therefore, it can be seen clearly that the error dynamics is mainly dependent on the PID parameters but not directly on the parameter α. However, by choosing different values of α, the learning rates for different model parameters of the neural network will be changed accordingly. Note that the learning rates for different weights of neural networks are usually determined by a common parameter in most existing learning algorithms (e.g., BP algorithms). A larger value of the parameter will result in larger learning rates for all the weights. However, a more reasonable way in practice could be to adjust different learning rates for different weights. Because it is better to have a slow learning rate for some sensitive weights in order to avoid oscillation of the network, while the total convergence speed of the algorithm is not affected simultaneously. The PID-L algorithm (12) can achieve this by providing a method to explore the heuristic knowledge on different weights of the neural network. The weights can be divided into different groups heuristically although only two groups are shown in (12). The parameters in C_1 can have a higher or lower learning rate than the parameters in C_2. For example, if the parameters in C_2 correspond to sensitive parameters in the neural network, α can be chosen to be less than 0.5. If $\alpha = 0.5$ by default, this corresponds to the commonly used case.

(iii) The results in Theorem 1 provide the best static parameter values for the PID-L algorithm in (12) such that an optimal performance and a globally asymptotical convergence could be achieved by exploiting the LMI optimization techniques. This is another advantage of the new method proposed in this paper. This could be the first result in the literature to address the learning problems of FNNs by using LMI techniques via a PID robust control method. Moreover, the parameters K_{p}, K_{i}, and K_{d} can also be determined such that the characteristic equation of the controlled error system has the characteristic roots of smallest magnitudes. After some manipulation, the controlled error dynamics can be written as (also see in Appendix A)

$$e(n+1)+(-2I_{r,r}+K_p+K_i+K_d)e(n)+(I_{r,r}-K_p-2K_d) \times e(n-1)+K_d e(n-2)=w(n)-w(n-1).$$

Therefore, the roots of the characteristic equation of the system above determine the dynamic evolution characteristics of the error system. For convergence, the roots must be within the unit circle. Obviously, for a faster

convergence, the roots should have the smaller magnitudes. This can be achieved systematically through the LMI optimization in Theorem 1 considering the disturbance effects.

(iv) Assuming that the noise is uncorrelated to the input, a correlation analysis can show that the noise effects on the parameter estimation through the PID-L algorithm in (12) will become trivial as the time going to infinity. For parameter C_1, applying the weight update law (12) results in

$$\hat{C}_1(n) = \hat{C}_1(0) + \sum_{i=1}^{n} \overline{\alpha} \cdot \left(\hat{f}_{\hat{C}_1(i)} \hat{f}^T_{\hat{C}_1(i)}\right)^{-1} \hat{f}^T_{\hat{C}_1(i)} \left[e(i) + e(i-1) + e(i-2)\right]$$

where $\overline{\alpha}$ is a function of the corresponding coefficients resulting from the recursive expansion. Therefore, the noise included in the error has a cross-correlation with $\hat{f}^T_{\hat{C}_1(i)}$, and this cross-correlation has effect on the parameter estimation. Note that $\hat{f}^T_{\hat{C}_1(i)}$ is a function of the input, which has no relation with the noise that is included in the error. Therefore, the cross-correlation between the noise and $\hat{f}^T_{\hat{C}_1(i)}$ will approach zero as $n \to \infty$. This implies that the noise in the error has no effect on the parameter estimation. For this reason, the estimated parameters can be further improved by $\hat{C}(t) = (1/N)\sum_{n=t-N+1}^{t} \hat{C}(n)$ provided that $\sum_{n=t-N+1}^{t} \varepsilon(n) = 0$.

III. Examples

In simulations, the parameters of the PID-L algorithm can be determined by the following process: 1) determine a sufficiently large number q; 2) given a disturbance attenuation level $\gamma > 0$ (3.6 by default in this paper), find the best PID parameters $K_{\rm p}$, $K_{\rm i}$, and $K_{\rm d}$ according to Theorem 1 (note that they can also be tuned heuristically); 3) α can be an appropriate value in a closed set [0 1].

A. Function Approximation

A RBF network is adopted to approximate a 2-D Gabor function, a bench-mark problem [8], which is given by

$$g(x_1, x_2) = \frac{1}{c\pi} \exp\left(-\left(x_1^2 + x_2^2\right)/\left(2(0.5)^2\right)\right) \cos\left(2\pi(x_1 + x_2)\right) \tag{16}$$

where $c = 0.1$. There are two $(p = 2)$ inputs $X = [x_1, x_2]^{\rm T}$. Given q hidden units, the basis function is chosen as $g_{\rm i}(x_1, x_2) = \exp(-(X - \xi_j)^T \Sigma_j (X - \xi_j))(j = 1, \ldots, q)$, and the output can be written as

$$Y = f(\Gamma, \xi, \Sigma; X) = \sum_{i=1}^{q} \Gamma_j g_j(\zeta_{ji}, \Sigma_{ji}; x_1, x_2) = \Gamma^T \overline{g}$$

where Γ, ξ, Σ are the parameters to be tuned, and $\overline{g}$ is consisting of $g_j(\zeta_j, \Sigma_j; x_1, x_2)$. The weights can be divided into two groups, i.e., $C_1 = \Gamma$ and $C_2 = [\xi^T, \Sigma^T]^T$, then the matrix B_1 and $B_2 = [B_{21}^{\rm T}, B_{22}^{\rm T}, \ldots]$ in (12) can be achieved, e.g.,

$$\begin{aligned} B_1 &= \partial \hat{f}(\cdot)/\partial \hat{C} = \overline{g}^T, \\ B_{2j} &= \partial \hat{f}(\cdot)/\partial \hat{\Sigma}_j \\ &= \Gamma_j g_j(\cdot) \left[-(x_1 - \xi_{j1})^2 \quad -(x_2 - \xi_{j2})^2\right]^T \cdots . \end{aligned}$$

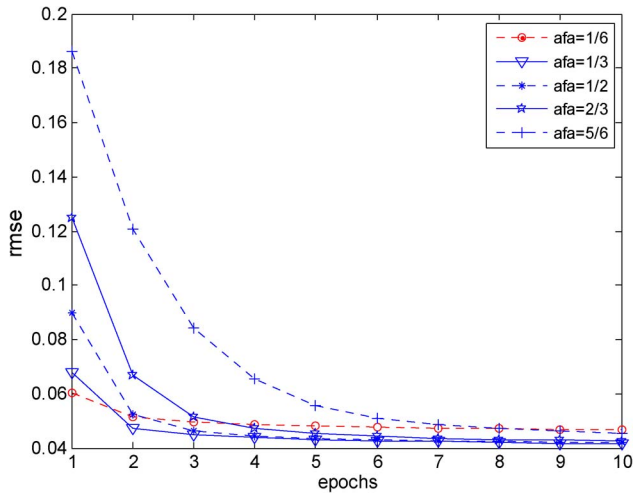

Fig. 1. Performance with different values of α.

Case (I). The effect of the parameter α on the convergence performance of the PID-L algorithm is studied. The number of hidden neurons is $q = 30$, and the initial weights are given as: $\Gamma_{\rm j} = 1/q$; $\Sigma_{j1} = \Sigma_{j2} = 15$; and the input space is regularly gridded to find q points for ξ_j (for example, if $q = 30$, it can be gridded evenly with six rows and five columns). 2000 training data X are generated randomly with uniform distribution in the range $[-0.5, 0.5]$, while 1600 validation data are generated regularly in the same range with an interval 0.025. The accuracy of the estimated RBF network is evaluated by the root mean square (rms) error over the 1600 validation data. Run the PID-L algorithm with $K_{\rm p} = 0.9989$, $K_{\rm i} = 0.9200$, $K_{\rm d} = -0.0001$ (by Theorem 1, resulting in the poles of the closed-loop system to be 0.0836, $-0.0012 \pm 0.0396i$) and different values of α, the results are shown in Fig. 1.

As discussed before, although the value of α has no significant effect on the final rmse value after sufficient learning, it has obvious effect on the convergence speed of the error dynamics. In this example, a smaller value of α, a faster convergence speed can the algorithm achieve. However, too small value of α may result in a large rmse (i.e., approximation error) after learning. In the following simulations, the parameter α is set to 1/3 by default. With the same initial weights, the performance of the PID-L algorithm is compared with the known LM-BP algorithm (mu = 1) and the other newly developed Lyapunov function (LF)-based algorithm [8] which is shown to outperform the BP and EKF algorithms. Here, mu represents the learning rate or a parameter to tune in the corresponding algorithms here and in what follows, and is chosen to have as fast as possible convergence speed. The results are shown in Fig. 2, indicating that the new PID-L algorithm is both faster in convergence and better in rmse after 10 epochs learning. Note that the LF with mu = 20 is as good as the LM-BP which has the worst rmse after 10 rounds of training, and the LF with mu = 30 can

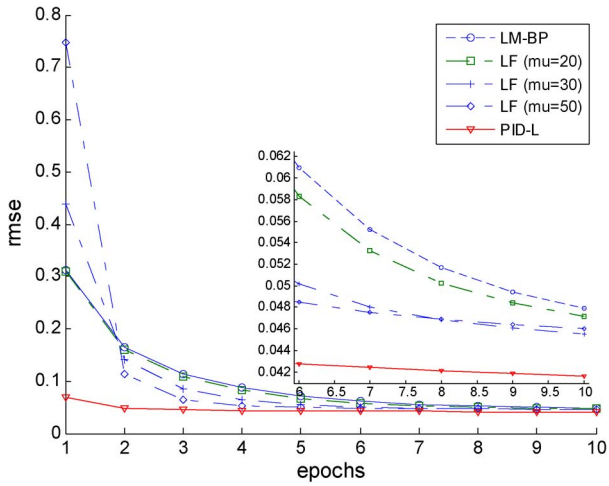

Fig. 2. Performance comparisons with the same initial weights.

TABLE I
PERFORMANCE OF DIFFERENT ALGORITHMS

method	rmse in validation after n rounds of training	
	n=3	n=5
BP (mu=0.8)	1.7547 (±0.0726)	1.7310 (±0.0508)
BP-adaptive (mu=1)	1.7355 (±0.0562)	1.7243 (±0.0494)
BP-momentum-adaptive	1.7391 (±0.0461)	1.7273 (±0.0313)
Scaled conjugate-BP	1.3221 (±0.1801)	0.9871 (±0.1654)
LM-BP (mu=1)	0.1026 (±0.0052)	0.0681 (±0.0035)
LF (mu=50) [8]	0.0518 (±0.0019)	0.0456 (±0.0014)
PID-learning	0.0465 (±0.0026)	0.0399 (±0.0007)

reach its smallest rmse after 10 rounds of training but converge slowly.

Case (II). For more comparisons, the data is generated in the same way as Case I, but the initial weights are random generated uniformly in $[-1\ 1]$, and each algorithm is run 50 times independently to find the averaged rmse. The parameters of the PID-L are the same as Case I. The results are summarized in Table I. The PID-L algorithm is the faster one compared with those BP-type algorithms and the newly developed LF-based method in [8]. With the same series of data but adding some white noise $(N(0, 0.3^2))$ into the output, 50 simulations are conducted for each algorithm with different series of noise. The simulation results are shown in Fig. 3, indicating that the PID-L algorithm still converges faster in terms of the averaged rmse and also has much smaller variation in rmse at each epoch.

Case (III). Another test is conducted to all the mentioned algorithms above when the parameter c in (16) is set to $c = 0.001$ and the data is generated in the same way as Case I. In this case, the output is changing very fast with respect to the input. The simulation shows that the PID-L algorithm with the same parameter settings as before can still guarantee stability, converge much faster, and achieve much better error performance after several rounds of training, while the other algorithms either fail to be stable or become obviously worse even when the corresponding learning rate is changed to a proper number (see Table II and Fig. 4). This test demonstrates the highly robust adaptability of the PID-L algorithm with respect to the fast changing output.

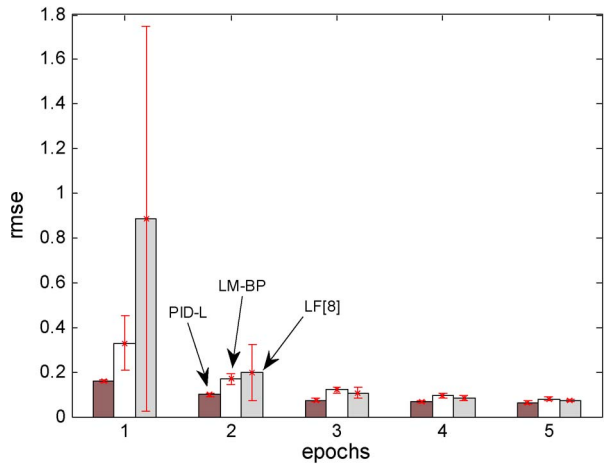

Fig. 3. Comparisons of the rmse (with std) in validation for noisy data between LF[8], LM-BP, and PID-learning (the learning rate (parameter) is 30 for LF[8], 1 for LM-BP and $\alpha = 0.65$, $K_{\rm p} = 0.9989$, $K_{\rm i} = 0.9200$, and $K_{\rm d} = -0.0001$ for the PID learning).

TABLE II
RMSE PERFORMANCE WHEN PARAMETERS ARE CHANGED

Methods			rmse in validation after *n* rounds of training			
			n=1	n=3	n=5	n=20
BP	**c=0.1**	mu=0.1	0.2261	0.1315	0.0851	0.0437
	c=0.001	mu=0.1	unstable	unstable	unstable	unstable
		mu=0.001	146.6537	118.4324	103.2759	53.6422
		mu=0.0001	163.8421	159.7308	155.8232	129.9086
LM-BP	**c=0.1**	mu=1	0.3118	0.1141	0.0705	0.0436
	c=0.001	mu=1	158.5998	149.4015	145.2096	132.2575
		mu=0.01	161.0285	152.3607	147.1048	132.9999
LF [8]	**c=0.1**	mu=30	0.4382	0.0856	0.0544	0.0433
	c=0.001	mu=30	547.2235	459.9189	421.1681	322.3571
		mu=2	165.5852	157.3001	151.8553	130.9917
		mu=0.1	159.9323	157.1734	155.5496	149.5962
PID-L	**c=0.1**	α =1/3	0.0681	0.0448	0.0432	0.0404
	c=0.001	K_p=0.9989 K_i=0.9200 K_d=-0.0001	6.7755	4.4903	4.3292	4.0502

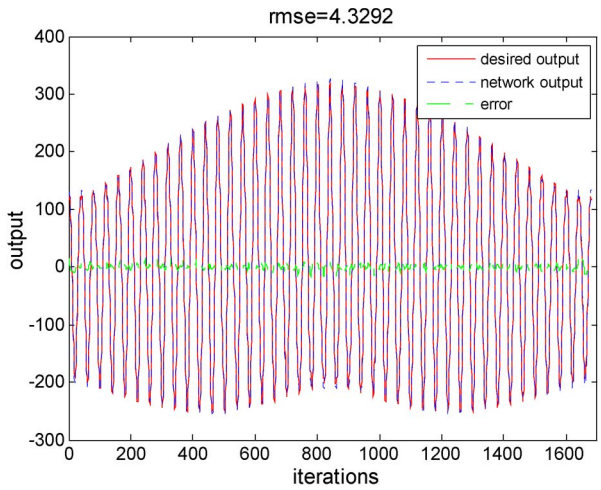

Fig. 4. Network output in validation still matches the real output well (rmse = 4.3292) using the PID-learning of the same parameter setting when the parameter c in (16) is changed from 0.1 to 0.001.

TABLE III
ESTIMATIONS WITH DIFFERENT ALGORITHMS

Real values	BP mu=0.002	LFa[8] mu=0.2	LFb [15] mu=1.5	LM-BP mu=0.1	OLS [14]	PID-L Kp =1, Ki =-0.0001, Kd =-0.5
1	0.7233 (±0.0040)	1.0047 (±0.0286)	0.1793 (±0.0014)	0.9954 (±0.0412)	1.002408	**0.9995 (±0.0248)**
0.5	0.7057 (±0.0042)	0.4952 (±0.0251)	0.1666 (±0.0014)	0.5033 (±0.0375)	0.496879	**0.5017 (±0.0210)**
0	0.0592 (±0.0031)	0.0212 (±0.0963)	0.0426 (±0.0042)	-0.0368 (±0.0969)		**-0.0019 (±0.0749)**
0.25	**0.0760** (±0.0008)	**0.2001** (±0.1979)	**0.0454** (±0.0031)	**0.3104** (±0.1836)	0.253131	**0.2502 (±0.1446)**
0	0.1003 (±0.0031)	0.0246 (±0.1000)	0.0580 (±0.0042)	-0.0251 (±0.0785)		**-0.0014 (±0.0675)**
-0.3	-0.2666 (±0.0021)	-0.2954 (±0.0299)	0.3554 (±0.0026)	-0.2940 (±0.0288)	−0.2999 78	**-0.3001 (±0.0237)**

B. System Identification With Nonwhite Noise

Consider a system identification example [14], i.e.,

$$x(t)=u(t-1)+0.5u(t-2)+0.25u(t-1)u(t-2) -0.3u^3(t-1)$$

$$y(t)=x(t)+\frac{1}{1-0.8q^{-1}}w(t),\quad w(t)\sim N(0,0.02^2). \quad (17a\text{-}b)$$

The noise process in the output is nonwhite. The input used to actuate the system is chosen as a low-frequency process, i.e., $u(t)=1.6u(t-1)-0.6375u(t-2)+0.16\zeta(t)$ with $\zeta(t)\sim N(0,1)$. In this case, it will be more difficult to identify the real model. A RBF NN model is used for identification

$$y(n)=\sum_{i=1}^{M}\kappa_j g_j\left(X(n)\right)=C^T G_X(n) \quad (18)$$

with the basis function vector chosen as

$$G_X(n)=\left[u(t-1),u(t-2),u^2(t-1), u(t-1)u(t-2),u^2(t-2),u^3(t-1)\right]^T.$$

The weight vector C is to be determined, whose real value is $C^*=[1\ 0.5\ 0\ 0.25\ 0\ -0.3]^{\mathrm{T}}$ according to (17a). The initial weight is chosen as $\kappa_j(0)=1/M$ for $j=1\ldots M$. A data set of 1000 input-output samples was generated, and the first 500 samples were used for training while the others for model validation.

Case (I). With the same input-output data but different additive noise, the network is trained for more than 50 times independently to find the estimated $\hat{C}$ and terminated when the training is up to 20 epochs or no obvious improvement in rmse is observed in successive 5 epochs or the rmse is becoming worse. The results are summarized in Table III, which indicate that the estimation of the PID-L is unbiased and much better than those of the BP (the learning rate is 0.002 in this case), the LF-based methods [8], [15], and the LM-BP. The result of the PID-L is also comparable to the result of [14] but with a BP-level computation. The method in [14] is an improved version of the orthogonal least square method which is known as an effective algorithm for offline identification of NARX models but with much computation cost.

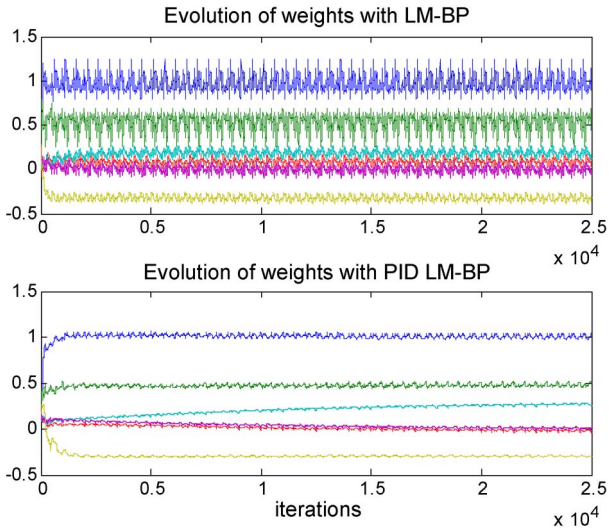

Fig. 5. Comparisons between the LM-BP and the improved PID LM-BP.

Case (II). As discussed before, the new PID-L algorithm integrates all the properties of several existing BP-type algorithms such as the basic BP, Newton-BP, and LM-BP, etc. This implies that the PID-L could always behave better or not worse than those BP-type algorithms. Importantly, improved versions of these BP-type algorithms can be achieved by replacing the error $e(n)$ with the composite PID error $v(n)$ [in (11b) or (11c)]. For example, if the LM-BP is used as $u(n)=-(B(n)^TB(n)+\mu\cdot I)^{-1}B(n)^Te(n)$, then an improved PID LM-BP can be written as $u(n)=-(B(n)^TB(n)+\mu\cdot I)^{-1}B(n)^Tv(n)$. If $K_{\mathrm{p}}=1$, $K_{\mathrm{i}}=0$, and $K_{\mathrm{d}}=0$, then the improved PID LM-BP becomes the original LM-BP. If appropriately tuning the PID parameters, then better performance could be achieved. A simulation result is shown in Fig. 5, where the parameter μ of LM-BP is tuned to 0.2 first to achieve a better performance, then using the same setting for μ and additionally tuning the other parameters of PID LM-BP to $K_{\mathrm{p}}=0.11$, $K_{\mathrm{i}}=0$ and $K_{\mathrm{d}}=0.07$, the evolution of the estimated model parameter C with the PID LM-BP is much better than that with the LM-BP.

TABLE IV
EPOCHS IN TRAINING TO ACHIEVE RMSE $= 0.0001$ IN XOR PROBLEM

Algorithm	BP	BP with adaptive learning rate	BP with momentum and adaptive learning rate	Scaled Conjugate Gradient BP	LF [8]	LM-BP	PID-L
Averaged epochs	762	715	900	399	97	104	66
STD of epochs	118	159		137	41	33	47
Max epoch	960	960		900	267	220	216
Min epoch	560	420		200	40	60	18
Fail ratio%	90	67	99	14	<5	<5	<5

TABLE V
EPOCHS USED TO ACHIEVE RMSE $= 0.0001$ IN ENCODER PROBLEM

Algorithm	BP	BP with adaptive learning rate	BP with momentum and adaptive learning rate	LF[8]	LM-BP	PID-L
Averaged epochs	347	216	413	235	145	134
STD of epochs	103	108	104	100	39	40
Max epoch	680	450	650	519	260	251
Min epoch	190	85	260	38	41	45

C. XOR Problem

For the exclusive-or (XOR) problem, the input series are: 0 0; 0 1; 1 0; 1 1 and the output series are: 0; 1; 1; 0. Consider a three-layer neural network with two inputs, four hidden neurons (as that in [8], [10] etc.), and one output for this problem and the transfer function in the hidden layer is chosen as tangent sigmoid functions. The network is trained for 100 times with random initial weights, and the training is terminated when the rmse per epoch reaches 0.0001 or the maximum epoch 1000 is reached. Several existing BP-type algorithms [21] are compared with learning rates (or initial learning rate) 0.85 to have a faster convergence. The (initial) learning rate μ is set to 0.1 for LM-BP and 0.5 for the LF-based method in [8] to have a faster convergence. The parameters of the PID-L are $\alpha = 0.5$, $Kp = 0.9965$, $Ki = 1.0057$, and $Kd = 0.0022$ (Theorem 1 $(\gamma = 2.6)$). The simulation results are given in Table IV. From Table IV, it can be seen that the new PID-L algorithm used fewer epochs in average to reach the training objective. Similar results could be concluded for more difficult parity-N problem. Moreover, it is noted that for a given neural network structure with sufficient hidden neurons and given initial network weights, the PID-L can always be tuned by its three parameters $(K_{\rm p}, K_{\rm i}, K_{\rm d})$ to have a better convergence.

D. 4-2 Encoder

To demonstrate the effectiveness of the new PID-L algorithm in application to multiple output cases, a 4-2 encoder problem is considered. The 4-D input series are given as [0 0 0 1; 0 0 1 0; 0 1 0 0; 1 0 0 0], and the 2-D outputs [0 0; 0 1; 1 0; 1 1]. A 4-8-2 FNN structure is adopted for this learning task. The parameters of the PID-L algorithm are given as $\alpha = 0.5$, $K_{\rm p} = 1.4720$, $K_{\rm i} = 1.6100$, and $K_{\rm d} = 0.1124$. Several existing well-known or newly developed algorithms are compared whose parameters are chosen in such a way that an as fast as possible speed can be achieved. For example, the learning rate for the gradient descent BP algorithm is 0.9, 0.5 for the LF method in [8], and the initial learning rate is 0.1 for LM-BP. The network is trained for more than 100 times using each algorithm with random initial weights, and the training is terminated when the rmse per epoch reaches 0.0001 or the maximum epoch 1000 is reached. The results are summarized in Table V, which shows that the PID-L algorithm is obviously much better than some other well-known BP-type algorithms. Although the difference among the performances of the LF method [8], the LM-BP algorithm and the PID-L algorithms may not be statistically significant, the PID-L algorithm still used fewer epochs in average to reach the same learning objective.

Based on the example studies above, it can be seen that the PID-L algorithm is faster and more robust in learning different tasks, particularly for function approximation and system identification. This could provide a totally new approach to nonlinear system identification subject to noise corruption. One advantage of the PID-L compared with the other algorithms also lies in that the algorithm parameter setting could be given by Theorem 1 automatically while the parameter setting of the other algorithms must be tuned manually many times to find a relatively better one for each specific problem. Improved versions of some existing BP-type algorithms can be easily developed as that demonstrated in case II of Example B. Different robust learning algorithms can also be designed based on the general robust control approach established in Section II.

Note that the computation cost of the PID-L algorithm is basically similar to that of the commonly used BP algorithm. In Matlab, to generate the incremental weight update term $u(n) = [\Delta\hat{C}_1(n)^T \Delta\hat{C}_2(n)^T]^T$ in (12), the code needs 6.2252e-005 $\pm$ 6.7889e-005s with the PID learning, while it needs 4.2997e-006 $\pm$ 2.8935e-006s with the BP algorithm. The PID algorithm needs more memory to carry out the integral and derivative computations than the BP but is known to be easy to implement in practice.

IV. CONCLUSION

An optimal PID control approach is proposed in this study to address the learning problems of FNNs. The training problem of a FNN is cast into a robust control problem of a linear discrete dynamic system. This greatly facilitates the analysis and design of a desired weight update law for the FNN, which actually acts as a virtual control law for the discrete dynamic system in terms of the estimation error. The advantages of the present method include:

(a) The learning problem is cast into a robust control problem, which demonstrates a powerful insight into the analysis and design of learning algorithms for FNNs. The advantage of the robust control approach could be that many existing robust control theories can readily be

available to use to cope with different kind of noise or disturbance in the data. For different noise process of different properties, the robust control approach provides a powerful tool or an alternative viewpoint to achieve a specific design in the learning algorithm.

(b) The determination of learning parameters is transformed into an optimization problem in terms of a simple LMI, which can achieve an optimal robust performance. This could remove the burden in manual tuning of multiple parameters as those in many existing learning algorithms.

(c) The robust PID controller is introduced in the robust control scheme to deal with the learning problem of FNNs; compared with several existing algorithms, the new learning algorithm is more generic and robust, particularly for function approximation and system identification with noisy data.

APPENDIX

Proof of Theorem 1

Using the PID-L algorithm (12), (8) can be written as

$$e(n+1) = e(n) - v(n) + w(n) \tag{A0}$$

which yields $E(z)(z-1) = -V(z) + W(z)$. Note that (11b) gives $V(z) = [K_p + K_i(1/1-z^{-1}) + K_d(1-z^{-1})]E(z)$. Then,

$$\left\{(z-1) + \left[K_p + K_i\frac{1}{1-z^{-1}} + K_d(1-z^{-1})\right]\right\} E(z) = W(z).$$

That is

$$e(n+1) + (-2I_{r,r} + K_p + K_i + K_d)e(n) + (I_{r,r} - K_p - 2K_d) \times e(n-1) + K_d e(n-2) = w(n) - w(n-1). \tag{A1}$$

Define

$$\begin{bmatrix} K_p \\ K_i \\ K_d \end{bmatrix} = \begin{bmatrix} I_{r,r} - b - 2c \\ I_{r,r} + a + b + c \\ c \end{bmatrix}. \tag{A2}$$

Equation (A1) can be written as

$$e(n+1) + ae(n) + be(n-1) + ce(n-2) = w(n) - w(n-1)$$

which further yields

$$\begin{aligned} e_{n+1} &= (A + \overline{B}K)e_n + \overline{B}_1 w_n \\ z(n) &= Ge_n \end{aligned} \tag{A3}$$

where

$$e_n = \begin{bmatrix} e(n) \\ e(n-1) \\ e(n-2) \end{bmatrix},\ A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \otimes I_{r,r}$$
$$\overline{B} = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \otimes I_{r,r},\ K = \begin{bmatrix} a + I_{r,r} \\ b + I_{r,r} \\ c + I_{r,r} \end{bmatrix}^T$$
$$\overline{B}_1 = \begin{bmatrix} 1 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \otimes I_{r,r},\ w_n = \begin{bmatrix} w(n) \\ w(n-1) \end{bmatrix}$$
$$G = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}^T \otimes I_{r,r}.$$

Now, (A3) is a typical robust control system, and the control law K is to be designed so that the output z(n) is stabilized. Note that system (A, B, G), which is the error dynamics of the learning system in (3)–(5), is fully controllable and observable through the PID controller. The H_∞ robust control theory is applied to find the optimal control law K, which corresponds to the optimal parameters of the PID learning. Given a disturbance attenuation level γ, and supposing zero initial conditions for (A3), the H_∞ robust control is to find the optimal K so that $\|z(n)\|_2 < \gamma\|w_n\|_2$ for any nonzero $w_n \in L_{2/2e}$. Define the LF as

$$V(n) = e_n^T P e_n \tag{A4}$$

where P is a positive definite matrix, i.e., $P > 0$. Letting $y_{\rm n} = e_{\rm n+1} - e_{\rm n}$ and using (A3), it can be derived that

$$0 = (A + \overline{B}K - I)e_n - y_n + \overline{B}_1 w_n = (\overline{A} + \overline{B}K)e_n - y_n + \overline{B}_1 w_n \tag{A5}$$

where $\bar{A} = A - I$ and I is a unit matrix. Using (A4) and (A5)

$$\begin{aligned} \Delta V(n) &= V(n+1) - V(n) = e_{n+1}^T P e_{n+1} - e_n^T P e_n \\ &= y_n^T P y_n + e_n^T P y_n + y_n^T P e_n \\ &= y_n^T P y_n + 2e_n^T P y_n \\ &\quad + 2\left(e_n^T T_1 + y_n^T T_2\right)\left[(\overline{A} + \overline{B}K)e_n - y_n + \overline{B}_1 w_n\right] \\ &= y_n^T \left(P - T_2 - T_2^T\right) y_n \\ &\quad + 2e_n^T \left[P - T_1 + (\overline{A} + \overline{B}K)^T T_2^T\right] y_n \\ &\quad + e_n^T \left[T_1(\overline{A} + \overline{B}K) + (\overline{A} + \overline{B}K)^T T_1^T\right] e_n \\ &\quad + 2e_n^T T_1 \overline{B}_1 w_n + 2y_n^T T_2 \overline{B}_1 w_n \end{aligned}$$

which further yields

$$\begin{aligned} &\Delta V(n) + z(n)^T z(n) - \gamma^2 w_n^T w_n \\ &= y_n^T \left(P - T_2 - T_2^T\right) y_n + 2e_n^T \left[P - T_1 + (\overline{A} + \overline{B}K)^T T_2^T\right] y_n \\ &\quad + e_n^T \left[T_1(\overline{A} + \overline{B}K) + (\overline{A} + \overline{B}K)^T T_1^T + G^T G\right] e_n \\ &\quad + 2e_n^T T_1 \overline{B}_1 w_n + 2y_n^T T_2 \overline{B}_1 w_n - \gamma^2 w_n^T w_n \\ &= X_n^T \Theta X_n \end{aligned} \tag{A6}$$

where

$$X_n=\begin{bmatrix}e_n^T & y_n^T & w_n^T\end{bmatrix}^T,\quad \Xi=T_1(\overline{A}+\overline{B}K)+(\overline{A}+\overline{B}K)^TT_1^T$$

$$\Theta=\begin{bmatrix}\Xi+G^TG & P-T_1+(\overline{A}+\overline{B}K)^TT_2^T & T_1\overline{B}_1\\ * & P-T_2-T_2^T & T_2\overline{B}_1\\ * & * & -\gamma^2I\end{bmatrix}.$$

Obviously, if $\Theta<0$, then $\Delta V(n)+z(n)^Tz(n)-\gamma^2w_n^Tw_n<0$ for any nonzero $X_n=[e_n^T\ y_n^T\ w_n^T]^T$. Then, noting the zero initial conditions, it can be obtained that

if $w(n)\in L_2$

$$\sum_{n=0}^{\infty}\left[z(n)^Tz(n)-\gamma^2w_n^Tw_n\right]<V(0)-V(\infty)=-V(\infty)<0$$

if $w(n)\in L_{2e}$,

$$\sum_{n=0}^{N_t}\left[z(n)^Tz(n)-\gamma^2w_n^Tw_n\right]<V(0)-V(N_t)=-V(N_t)<0$$

where N_t is a positive constant. Both cases yield

$$\|z(n)\|_2<\gamma\|w_n\|_2.$$

Using Schur complement, $\Theta<0$ is equivalent to

$$\begin{bmatrix}\Xi+G^TG & P-T_1+(\overline{A}+\overline{B}K)^TT_2^T & T_1\overline{B}_1 & G^T\\ * & P-T_2-T_2^T & T_2\overline{B}_1 & 0\\ * & * & -\gamma^2I & 0\\ * & * & * & -I\end{bmatrix}<0. \tag{A7}$$

Define

$$Q=P^{-1},\quad U_1=-T_2^{-T}T_1^TQ,\quad U_2=T_2^{-T}$$

$$\Gamma=\begin{bmatrix}Q & 0 & 0 & 0\\ U_1 & U_2 & 0 & 0\\ 0 & 0 & I & 0\\ 0 & 0 & 0 & I\end{bmatrix}.$$

Premultiplying and postmultiplying both sides of (A7) with Γ^T and Γ, inequality (A7) is equivalent to (see equation at the bottom of the page), where $\Xi_1=U_2-U_1^T+Q\bar{A}^T+QK^T\bar{B}^T+U_1^TQ^{-1}U_2$. Using Schur complement again, the inequality above is equivalent to

$$\begin{bmatrix}U_1^T+U_1 & U_2-U_1^T+Q\overline{A}^T+Y^T\overline{B}^T & 0 & QG^T & U_1^T\\ * & -U_2-U_2^T & \overline{B}_1 & 0 & U_2^T\\ * & * & -\gamma^2I & 0 & 0\\ * & * & * & -I & 0\\ * & * & * & * & -Q\end{bmatrix}<0$$

where $Y=KQ$. With $K=YQ^{-1}$, the desired PID parameters can be obtained from (A2).

Moreover, it can be seen that,

$$\Delta V(n)+z(n)^Tz(n)-\gamma^2w_n^Tw_n<0$$

(i.e., $\|z(n)\|_2<\gamma\|w_n\|_2$) holds for any nonzero $X_{\rm n}=[e_n^T\ y_n^T\ w_n^Tt]^T$. Note that the vector $X_{\rm n}=0$ implies that

$$e(n)=e(n-1)=e(n-2)=0 \tag{A8}$$

$$e_{\rm n+1}=e_{\rm n} \tag{A9}$$

$$w(n)=w(n-1)=0. \tag{A10}$$

(A8) and (A9) yield that

$$e(n+1)=e(n)=e(n-1)=e(n-2)=0. \tag{A11}$$

Using (A10) and (A11) in (A0) gives

$$v(n)=v(n-1)=0. \tag{A12}$$

Using (A11) and (A12) in (11b) yields

$$v(n+1)=v(n)=v(n-1)=u(n+1)=u(n)=u(n-1)=0. \tag{A13}$$

Thus, $X_{\rm n}\equiv0$ gives that

$$e(n+1)\equiv e(n)\equiv0\quad\text{and}\quad u(n+1)\equiv u(n)\equiv0.$$

That is, the modeling error keeps being zero, and the network weights come to constant values. This shows a global minimum reached. This completes the proof.

$$\Gamma^T\begin{bmatrix}\Xi & P-T_1+(\overline{A}+\overline{B}K)^TT_2^T & T_1\overline{B}_1 & G^T\\ * & P-T_2-T_2^T & T_2\overline{B}_1 & 0\\ * & * & -\gamma^2I & 0\\ * & * & * & -I\end{bmatrix}\Gamma$$

$$=\begin{bmatrix}U_1^T+U_1+U_1^TQ^{-1}U_1 & \Xi_1 & 0 & QG^T\\ * & U_2^TQ^{-1}U_2-U_2-U_2^T & \overline{B}_1 & 0\\ * & * & -\gamma^2I & 0\\ * & * & * & -I\end{bmatrix}$$

$$<0$$

REFERENCES

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[2] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*. Boston, MA: PWS-Kent, 1996.

[3] D. Sarkar, "Methods to speed up error back propagation learning algorithm," *ACM Comput. Surv.*, vol. l27, no. 4, pp. 519–544, Dec. 1995.

[4] C. Charalambous, "Conjugate gradient algorithm for efficient training of artificial neural networks," *Proc. Inst. Elect. Eng.—Circuits, Devices, Syst.*, vol. 139, no. 3, pp. 301–310, Jun. 1992.

[5] S. Osowski, P. Bojarczak, and M. Stodolski, "Fast second order learning algorithm for feedforward multilayer neural network and its applications," *Neural Netw.*, vol. 9, no. 9, pp. 1583–1596, Dec. 1996.

[6] J. Bilski and L. Rutkowski, "A fast training algorithm for neural networks," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, no. 6, pp. 749–753, Jun. 1998.

[7] G. Lera and M. Pinzolas, "Neighborhood based Levenberg-Marquardt algorithm for neural network training," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1200–1203, Sep. 2002.

[8] L. Behera, S. Kumar, and A. Patnaik, "On adaptive learning rate that guarantees convergence in feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1116–1125, Sep. 2006.

[9] Y. Iiguni, H. Sakai, and H. Tokumaru, "A real-time learning algorithm for a multilayered neural netwok based on extended Kalman filter," *IEEE Trans. Signal Process.*, vol. 40, no. 4, pp. 959–966, Apr. 1992.

[10] K. Nishiyama and K. Suzuki, "H_∞-learning of layered neural networks," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1265–1277, Nov. 2001.

[11] J. Park and I. W. Sandberg, "Universal approximation using radial basis function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, Jun. 1991.

[12] H. Han, C. Y. Su, and Y. Stepanenko, "Adaptive control of a class of nonlinear systems with nonlinearly parameterized fuzzy approximators," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 2, pp. 315–323, Apr. 2001.

[13] P. Sastry, "Backpropagation algorithm with momentum," *IEEE Trans. Neural Netw.*, vol. 5, no. 3, pp. 505–506, 1994.

[14] H. L. Wei and S. A. Billings, "Model structure selection using an integrated forward orthogonal search algorithm assisted by squared correlation and mutual information," *Int. J. Model., Identif. Control*, vol. 3, no. 4, pp. 341–356, 2008.

[15] Z. Man, H. R. Wu, S. Liu, and X. Yue, "A new adaptive backpropagation algorithm based on Lyapunov stability theory for neural networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1580–1591, Nov. 2006.

[16] J. H. Park, S. H. Kim, and C. J. Moon, "Adaptive neural control for strict-feedback nonlinear systems without backstopping," *IEEE Trans. Neural Netw.*, vol. 20, no. 7, pp. 1204–1209, 2009.

[17] S. S. Ge, C. C. Hang, T. H. Lee, and T. Zhang, *Stable adaptive Neural Network Control*. Boston, MA: Kluwer, 2001.

[18] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signal Syst.*, vol. 2, no. 4, pp. 303–314, 1989.

[19] H. Shao and W. Wu, "Convergence of BP algorithm with variable learning rates for FNN training," in *Proc. 5th Mexican Int. Conf. Artif. Intell.*, 2006, pp. 245–252.

[20] S. Huang and K. Tan, "Intelligent friction modeling and compensation using neural network approximations," *IEEE Trans. Ind. Electron.*, vol. 59, no. 8, pp. 3342–3349, Aug. 2012.

[21] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural Network Design*. Boston, MA: PWS-Kent, 1996.

[22] F. Scarselli and A. C. Tsoi, "Universal approximation using feedforward neural networks: A survey of some existing methods and some new results," *Neural Netw.*, vol. 11, no. 10, pp. 15–37, Jan. 1998.

[23] S. Cong and Y. Liang, "PID-like neural network nonlinear adaptive control for uncertain multivariable motion control systems," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 3872–3879, Oct. 2009.

[24] H. Yu, T. Xie, S. Paszczynski, and B. M. Wilamowski, "Advantages of radial basis function networks for dynamic system design," *IEEE Trans. Ind. Electron.*, vol. 58, no. 12, pp. 5438–5450, Dec. 2011.

[25] E. Kayacan, O. Cigdem, and O. Kaynak, "Sliding mode control approach for online learning as applied to type-2 fuzzy neural networks and its experimental evaluation," *IEEE Trans. Ind. Electron.*, vol. 59, no. 9, pp. 3510–3520, Sep. 2012.

[26] R. Vitthal, P. Sunthar, and C. D. Rao, "The generalized proportional-integral-derivative (PID) gradient descent back propagation algorithm," *Neural Netw.*, vol. 8, no. 4, pp. 563–569, 1995.

[27] M. A. Khanesar, E. Kayacan, M. Teshnehlab, and O. Kaynak, "Extended Kalman filter based learning algorithm for type-2 fuzzy logic systems and its experimental evaluation," *IEEE Trans. Ind. Electron.*, vol. 59, no. 11, pp. 4443–4455, Nov. 2012.

[28] R. H. Abiyev and O. Kaynak, "Type 2 fuzzy neural structure for identification and control of time-varying plants," *IEEE Trans. Ind. Electron.*, vol. 57, no. 12, pp. 4147–4159, Dec. 2010.

[29] A. Slowik, "Application of an adaptive differential evolution algorithm with multiple trial vectors to artificial neural network training," *IEEE Trans. Ind. Electron.*, vol. 58, no. 8, pp. 3160–3167, Aug. 2011.

[30] X. J. Jing, "An H_∞ control approach to robust learning of feedforward neural networks," *Neural Netw.*, vol. 24, no. 7, pp. 759–766, Sep. 2011.

Xingjian Jing received the B.S. degree from Zhejiang University, Hangzhou, China, in 1998, the M.S. degree from Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China, in 2001, and Ph.D. degree in nonlinear systems and signal processing from the Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U.K., in 2008.

He is currently an Assistant Professor with the Department of Mechanical Engineering, the Hong Kong Polytechnic University (PolyU), Kowloon, Hong Kong. Before joining in PolyU, he was a Research Fellow with the Institute of Sound and Vibration Research, University of Southampton, Southampton, U.K., from August 2008 to November 2009, where he worked on biomedical signal processing funded by a BBSRC (UK) project. He has published about 60 papers in refereed journals and conference proceedings. His current research interests include: system identification, signal processing, and control of complex nonlinear systems; nonlinear analysis in the frequency domain; intelligent computing methods; and their applications in nonlinear mechanical systems (sound and vibration control), nonlinear physiological systems (neural systems), and robotic systems, etc.

Li Cheng received the B.S. degree in applied mechanics from Xi'an Jiaotong University, Xi'an, China, in 1984, and DEA and Ph.D. degrees from the Institut National des Sciences Appliquees de Lyon, Villeurbanne Cedex, France, in 1986 and 1989, respectively.

He joined Laval University, in Canada, as an Assistant Professor in the Department of Mechanical Engineering, in 1992. In the following years, he was promoted to Associate Professor and Full Professor, before joining the Department of Mechanical Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong, in 2000. He is currently a Chair Professor and the Director of the Consortium for Sound and Vibration Research. His research interests mainly include noise and vibration control, fluid-structure interaction, damage detection, and smart material/structure.

Dr. Cheng has been a Member of board directors and committees of different learning organizations such as Acoustical Society of America, Canadian Acoustical Society, Acoustical Society of China, and Chinese Society of Noise and Vibration Engineering. He also serves in the editorial boards or advisory committee for several international journals.