Article ID: 1000 - 8152(2000)06 - 0861 - 07

PID Gradient Algorithm for Neural Network Based Generalised Nonlinear PID Controller *

TAN Yonghong and DANG Xuanju
(School of Computer Sciences, Guilin Institute of Electronic Technology · Guilin, 541004, P. R. China)
Achiel R. Van Cauwenberghe
(Department of Control Engineering and Automation, University of Ghent · Ghent, Belgium)
Mehrdad Saif
(School of Engineering Sciences, Simon Fraser University · Burnaby, BC, V5A1S6, Canada)

Abstract: An algorithm of PID gradient descent with momentum term (PIDGDM) is proposed. In this algorithm, the procedure of gradient optimization is considered as a feedback control system. Then, the convergent characteristic of the algorithm is presented. In this paper, a nonlinear PID controller is also proposed to handle some nonlinear control problems. The nonlinear PID control strategy is realized using neural networks. The PIDGDM algorithm is applied to the training of the neural nonlinear PID controller. Finally, simulation study of applying the neural nonlinear PID control strategy to a continuous-stirred-tank-reactor (CSTR) and a van de Vusse reactor is illustrated.

Key words: PID control; gradient descent; optimization; neural networks; nonlinear
Document code: A

用 PID 梯度算法训练基于神经网络的广义非线性 PID 控制器

谭永红 党选举
(桂林电子工业学院智能系统与工业控制研究室·桂林, 541004)
范·高温伯格 迈达德·萨义夫
(比利时根特大学控制工程与自动化系·根特, 比利时) (加拿大西蒙·弗雷泽大学工程科学系·加拿大)

摘要: 将梯度优化过程看作反馈控制系统, 从而提出了一种带动量项的 PID 梯度算法(PIDGDM). 进一步讨论了该算法的收敛特性. 提出了一种非线性 PID 控制器以对付一些非线性控制问题. 神经网络被用于实现该非线性 PID 控制器. PIDGDM 算法被用于训练所提出的神经网络非线性 PID 控制器. 最后, 给出了有关方法应用于两个化工过程的仿真研究结果.

关键词: PID 控制; 梯度; 优化; 神经网络; 非线性

1 Introduction

It is known that gradient descent (GD) algorithm is one of the simple and popular methods in nonlinear optimization and neural network training. However, GD algorithm usually suffers from the drawbacks of slow convergence as well as local minimum. In order to improve the convergent speed, a strategy called gradient descent algorithm with momentum term (GDM) has widely been used. In this paper, the procedure of the gradient descent optimization is investigated as a feedback control system. Then a PID gradient descent algorithm with momentum term (PIDGDM) is proposed. The corresponding convergent properties of the proposed algorithm is discussed in this article.

In this paper, a generalised nonlinear PID control (GNPIDC) strategy is also proposed to tackle some non-linear control problems. In this control scheme, the system error, the integral of the system error, and the derivative of the system error are employed as the inputs of the controller but the mapping from the inputs to the output of the controller is nonlinear. The implementation procedure of the control strategy using neural networks is

* Foundation item: supported by the Natural Science Foundation of Guangxi Province (9824030).
Received date: 1998 - 07 - 10; Revised date: 2000 - 04 - 18.

described. The determination of the control architecture can be considered as a procedure of system identification. Thus, the proposed PIDGDM algorithm is applied to the training of the neural network based GNPID controller. The simulation results on the processes that are respectively a contrinuous-stirred-tank-reactor (CSTR) and a van de Vusse reactor will be illustrated.

2 PID gradient descent algorithm

The gradient descent (GD) algorithm is a very popular method in nonlinear optimization and neural network training. However its drawback of slow convergence will keep it from being used for on-line training. In this section, a (PIDGDM) is proposed. Note that GD algorithm can be considered as a pure integral tuning of the gradient in terms of the feedback control. It is known that the integral tuning, while removing offset, may reduce the stability margin of the system. Such a problem is often met when we use a GD algorithm with a large learning rate to speed up the training. It is possible to improve the algorithm by introducing proportional tuning and derivative tuning which can stabilize the learning procedure and speed-up the convergence. Suppose a nonlinear function is described by

$$y_k = f(x_k), \tag{1}$$

where y_k is the output of the function, x_k is the variable matrix, and $f(\cdot)$ is a continuous function. Then, the GD algorithm used to search for minimum of $f(\cdot)$ at iteration k is given by

$$x_k = x_{k-1} - \lambda \partial f / \partial x_{k-1}, \tag{2}$$

where λ is the optimizing step-size or learning rate. Then the minimum or the local minimum of the objective function is found when

$$\partial f / \partial x_{k-1} = 0. \tag{3}$$

From (2), it is noted that the update of the variable matrix is actually based on a pure integral rule, i.e. it can be rewritten as

$$x_k = -\left(\lambda / (1 - z^{-1})\right)(\partial f / \partial x_{k-1}), \tag{4}$$

where $1/(1 - z^{-1})$ is the integral operator. This can explain why the choice of the optimizing step-size λ may often cause some conflict, i.e. a large λ can obtain fast learning but also easily cause oscillation and instability, whilst a small λ can result in stable learning but the learning procedure will be slow. Let us consider the learning procedure as a feedback control system shown in Fig. 1. In this system, the set-point is zero and the controller output is x_k. The gradient $-\partial f / \partial x_{k-1}$ is assumed to be a" system error ",i. e.

$$E_k = 0 - \partial f / \partial x_{k-1} = -\partial f / \partial x_{k-1}. \tag{5}$$

The purpose of the controller is to drive E_k to zero in a fast and stable way. It is known that a stable and fast control response can be obtained by choosing proper gains of the controller. Motivated by the PID control, the GD algorithm with incremental PID tuning is formulated as follows:

$$\begin{aligned} x_k = x_{k-1} &- \lambda_i(\partial f / \partial x_{k-1}) - \lambda_p[(\partial f / \partial x_{k-1}) - \\ &(\partial f / \partial x_{k-2})] - \lambda_d[(\partial f / \partial x_{k-1}) - \\ &2(\partial f / \partial x_{k-2}) + (\partial f / \partial x_{k-3})], \end{aligned} \tag{6}$$

where λ_p, λ_i, and λ_d are the proportional gain, the integral gain, and the derivative gain of the algorithm. Obviously, if λ_p and λ_d are zero, the algorithm is the standard GD algorithm. Therefore, the proposed algorithm provides more freedom to improve the convergence by choosing proper gains.

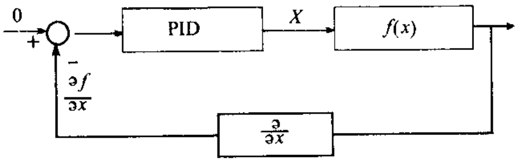

Fig. 1 Architecture of FID gradient descent method

It is well known that the addition of the momentum term to the gradient descent law may sometimes result in a faster learning. The gradient descent method with momentum term (GDM) is described by

$$\Delta x_k = -\lambda \partial f / \partial x_{k-1} + \alpha \Delta x_{k-1}, \tag{7}$$

where α is the momentum factor. This indicates that the variable change not only depends on the gradient but also the previous variable change. Formula (7) can be rewritten as

$$x_k = -\left(\lambda / (1 - z^{-1})(1 - \alpha z^{-1})\right)(\partial f / \partial x_{k-1}). \tag{8}$$

In this algorithm, the gradient passes an integrator and a low-pass filter to affect x_k. The low-pass filer can suppress the high frequency noise and smooth the learning process. Based on the same idea, we can obtain the PID gradient descent algorithm with momentum term (PIDGDM), i.e.

$$\begin{aligned} \Delta x_k = \alpha \Delta x_{k-1} &- \lambda_0(\partial f / \partial x_{k-1}) - \\ &\lambda_1(\partial f / \partial x_{k-2}) - \lambda_2(\partial f / \partial x_{k-3}), \end{aligned} \tag{9}$$

where $z^{-1}(\partial f / \partial x_{k-1}) = \partial f / \partial x_{k-2}$; and $\lambda_0 = \lambda_p + \lambda_i + \lambda_d$, $\lambda_1 = -(\lambda_p + 2\lambda_d)$, and $\lambda_2 = \lambda_d$. Thus, it leads to

$$x_k = -(\lambda_0+\lambda_1 z^{-1}+\lambda_2 z^{-2})(\partial f/\partial x_{k-1})/(1-z^{-1})(1-\alpha z^{-1}). \tag{10}$$

Note that (10) includes not only two poles (i. e. an integrator and a low-pass filter) but also two zeros. The good performance of the algorithm depends on choosing a proper combination of the poles and the zeros. One way is to tune these coefficients based on the Ziegler-Nichols (Z-N) method. Then, fix the zeros of the controller and tune the parameter α. To ensure the asymptotic stability, the pole $z = \alpha$ should be located in the unit circle of z-plane, i. e. $0 < \alpha < 1$. From another point of view, the algorithm can also be refered to as lag-lead compensator, in which $(1 - z_1 z^{-1})/(1 - z^{-1})$ is the lag section and $(1 - z_2 z^{-1})/(1 - \alpha z^{-1})$ is the lead section. In the lag section, z_1 is designed to satisfy the condition $|z_1| < 1$ while in the lead section, $|z_2|$ should be larger than α . This algorithm gives a phase lag at low frequencies to reduce the steady-state error and a phase lead at higher frequencies to increase the stability margins and to increase the bandwidth which leads to the speed-up of the response.

It is known than the GD method often gets stuck in a local minimum. Assume that a local minimum is met during the learning, it will lead to $\partial f/\partial x_{k-1} = 0$. Hence, the variable change is trapped in this local minimum. On the other hand, by using the GDM, it is possible to escape from a local minimum if Δx_{k-1} is not zero. The momentum term tries to keep the process of variable change moving. From (9), we see that the variable change is stuck only when $\partial f/\partial x_{k-1}$, $\partial f/\partial x_{k-2}$, $\partial f/\partial x_{k-3}$ as well as Δx_{k-1} are all equal to zero. Thus, the proposed PIDGDM approach is more likely to escape from a local minimum than the GDM method. In the following, the investigation of the convergent property of the PIDGDM algorithm is presented.

Theorem 1 Let X be a non-empty closed set in $\mathbb{R}^n$, and let the non-empty set $\Omega \subset X$ be the solution set. Let $A: X \to X$ be a point-to-set map. Given $x(1) \in X$, the sequence $\{x(k)\}$ is generated iteratively as follows:

If $x(k) \in \Omega$ then stops; otherwise, let $x(k+1) \in A(x(k))$, replace k by $k+1$ and repeat.

Suppose that the sequence $\{x(k)\}$ produced by the algorithm is contained in a compact subset of X, and suppose that there exists a continuous function f, called the descent function, such that $f(y) < f(x)$ if $x \notin \Omega$ and $y \in A(x)$. If the map A is closed over the complement of Ω then either the algorithm stops in a finite number of step with a point in Ω or it generates the infinite sequence $\{x(k)\}$ such that every convergent subsequence of $\{x(t)\}$ has a limit in Ω, i. e. all accumulation points of $\{x(t)\}$ belong to Ω and $f(x(k)) \to f(x)$.

Proof The detail of the proof can be found in [1].

Lemma 1 Suppose that both $\lambda_0, \lambda_2 > 0$ and the function $F(z^{-1}) = \varphi(z^{-1})/\Psi(z^{-1}) = (\lambda_0 + \lambda_1 z^{-1} + \lambda_2 z^{-2})/(1 - \alpha z^{-1})$ are asymptotically stable (i. e. $0 < \alpha < 1$), then

$$F(z^{-1}) \geqslant (4\lambda_0\lambda_2 - \lambda_1^2)/4\lambda_2. \tag{11}$$

Proof See Appendix A.

Theorem 2 Assume that $f: \mathbb{R}^n \to \mathbb{R}^1$ is continuously differentiable and that the algorithm

$$A(x_k) = x_{k-1} + \alpha\Delta x_{k-1} - (\partial f/\partial x_{k-1})(\lambda_0 + \lambda_1 z^{-1} + \lambda_2 z^{-2}), \tag{12}$$

where $\lambda_0, \lambda_2 > 0$ and $0 < \alpha < 1$, is applied to minimizing f. If the coefficients of the algorithm satisfy

$$4\lambda_0\lambda_2 \geqslant \lambda_1^2, \tag{13}$$

then f is a descent function.

Proof See Appendix B.

Theorem 3 Let $f: \mathbb{R}^n \to \mathbb{R}^1$ be continuously differentiable. In algorithm (9), let x^* be such that $\partial f/\partial x_{k-1}^* = 0$. If the coefficients are chosen as $\lambda_0, \lambda_2 > 0$, $0 < \alpha < 1$, and $4\lambda_0\lambda_2 \geqslant \lambda_1^2$, then the algorithm converges to x^*.

Proof See Appendix C.

Theorem 4 Suppose that both the PID gradient algorithm with momentum term (PIDGDM) and the gradient descent algorithm with momentum term (GDM) have the same factor of the momentum term α, and also have the same gradient at k. If the coefficients of the PIDGDM are chosen as $4\lambda_0\lambda_2 \geqslant \lambda_1^2 + 4\lambda\lambda_2$, where $\lambda > 0$ is the step-size of the gradient descent method, then the PIDGDM approach has faster speed of the convergence than the GDM method.

Proof From Theorem 2, it is known that the PID gradient descent method with momentum term may result in

$$\Delta f_{\text{pidgdm}} = - \| \partial f / \partial x_{k-1} \|^2 (\lambda_0 + \lambda_1 z^{-1} + \lambda_2 z^{-2}) / (1 - \alpha z^{-1}). \quad (14)$$

Considering the gradient descent method with momentum term, then

$$\Delta f_{\text{gdm}} = - \| \partial f / \partial x_{k-1} \|^2 \lambda / (1 - \alpha z^{-1}). \quad (15)$$

The ratio between $|f_{\text{pidgdm}}|$ and $|f_{\text{gdm}}|$ is

$$\| \Delta f_{\text{pidgdm}} \| / \| \Delta f_{\text{gdm}} \| = | \lambda_0 + \lambda_1 z^{-1} + \lambda_2 z^{-2} | \lambda^{-1}.$$

Considering Lemma 1, and $4\lambda_0\lambda_2 > \lambda_1^2 + 4\lambda\lambda_2$, then it results in

$$\| \Delta f_{\text{pidgdm}} \| / \| \Delta f_{\text{gdm}} \| > 1. \quad (16)$$

It means that, under this condition, the PIDGDM has derived a faster convergent speed than the GDM method.

To demonstrate the convergent procedures of both GDM (with $\lambda = 0.164, \alpha = 0.5$) and PIDGDM (with $\lambda_p = 0.164, \lambda_i = 0.164, \lambda_d = 0.0492, \alpha = 0.5$) in the estimation of the process $y(k) = 1.5y(k-1) + 0.75y(k-2) + u(k-1) - 0.5u(k-2)$, Fig. 2 shows the parameter estimation results. It illustrates that PIDGDM algorithm has obtained faster convergence.

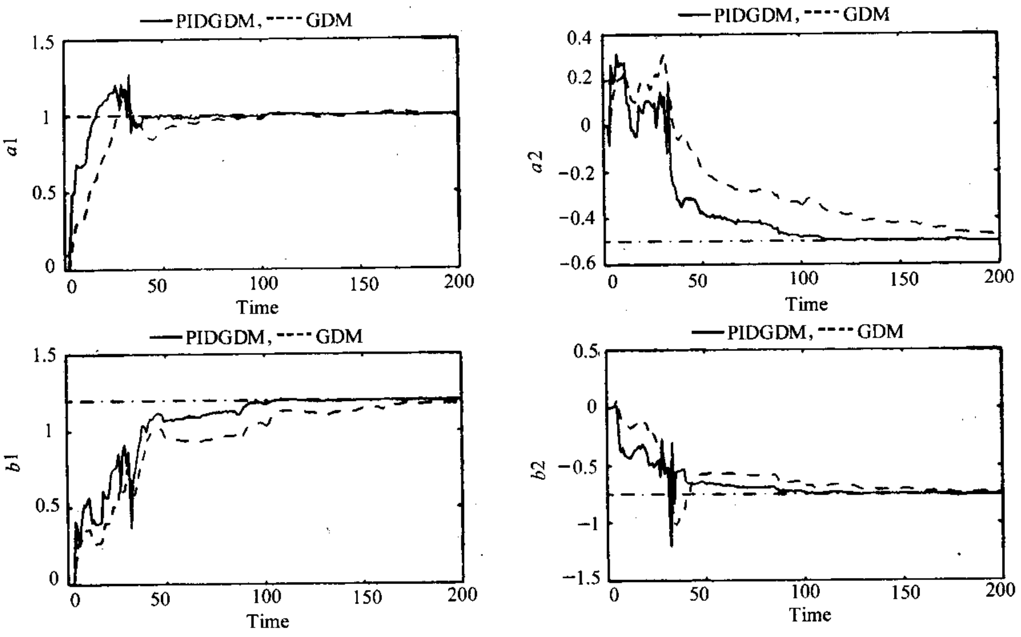

Fig. 2 Parameter estimates ofboth GDM and PIDGDM methods

3 Generalised nonlinear PID control

It is known that a nonliear PID controller will be much more attractive to deal with nonlinear control problems. In this paper, a generalized PID controller with the following definition is proposed.

Definition 1 Let $u \in \mathbb{R}, e \in \mathbb{R}$, and $W \subset \mathbb{R}^N$ $(W \neq 0)$. $N(\cdot, W)$ is a continuous nonlinear function which satisfies $N(x, W) = 0 (x = 0)$ and $N(x, W) \neq 0 (x \neq 0)$. Then, the controller is defined by

$$u(k) = N[e(k), \Delta e(k), \sum_{i=1}^{k} e(i), W], \quad (17)$$

where the elements of the input set $\{e(k), \Delta e(k), \sum_{i=1}^{k} e(i)\}$ are respectively the system error, the difference of the system error, and the the integral of the system error. The control mapping described in (17) is said to be a generalized nonlinear PID controller (GNPIDC). Based on defintion 1, one can also find the controller has the following features:

Proportional control:

$$u(k) = N[e(k), W_p]. \quad (18)$$

For the special case where $u(k) = k_p e(k)$, we have the traditional proportional control. Note that the gain of the traditional proportional control is a constant but the gain of the proposed controller is a nonlinear function.

Derivative control:

$$u(k) = N[\Delta e(k), W_d]. \quad (19)$$

The output of the controller $u(k)$ is not zero only when a change occurs in the system error. The linear case of this control strategy is $u(k) = k_d \Delta e(k)$.

Integral control:

$$u(k) = N[\sum_{i=1}^{k} e(i), W_i]. \quad (20)$$

When $e(i) = 0$ $(i = k, k+1, \cdots)$ but $\sum_{i=1}^{k-1} e(i) \neq 0$ $(i = 1, \cdots, k-1)$, $u(k)$ will be kept in a certain value C. This feature is very important for removing the steady-state system error.

It is noticed that the performance of the GNPID controller depends upon the mapping $N(\cdot)$. Assuming that the structure of the nonlinear mapping $N(\cdot)$ is pre-defined, the controller may be specified by adjusting the parameter matrix W to minimize the performance index of the feedback system, i.e.

$$W = \arg\min_{W} [r(k+1) - y(k+1)]^2, \quad (21)$$

where r is the set-point of the system, y is the output of the process. As soon as the parameter-matrix W is determined according to (21), the GNPID controller can be used to drive the output of the controlled process y to track the setpoint trajectory r.

4 Neural network based GNPIDC

We consider the determination of the architecture of the GNPID control as a system identification procedure using neural networks. Thanks to the universal approximation property of neural networks, it provides a powerful tool to solve the problem of the implementation of $N(\cdot)$. We can use an on-line or off-line learning algorithm to train the parameter-matrix W using the information depending on the system performance. In this approach, the structure of the neural GNPID controller is pre-specified and the performance of the control system is optimized with respect to the parameters of the neural controller. Assume that a three layer feedforward neural network is used to realize the mentioned nonlinear control strategy, i.e.

$$\begin{cases} u(k) = C^{\mathrm{T}}(k) M, \\ c_i(k) = s[n_i(k)] = s[\sum_{j=1}^{3} k_{ij} h_j(k)], \end{cases} \quad (22)$$

where $u \in \mathbb{R}$ is the output of the controller; $M = [m_1, \cdots, m_H]^{\mathrm{T}}$ where H is the number of hidden modes, c_i is the ith node of the hidden layer, $s(\cdot)$ is the sigmoid function and k_{ij} are the weights. To satisfy the requirement of Definition 1, here, we employ $s(x) = (1 - \exp(-x)/(1 + \exp(-x))$ in the neural network. The input variables h_j of the controller are defined as: $h_1(k) = e(k)$; $h_2(k) = e(k) - e(k-1)$; $h_3(k) = \sum_{i=0}^{k} e(i)$.

5 Simulation study

Example 1 The process is an exothermic continuous-stirred-tank-reactor (CSTR)[2]:

$$\frac{\mathrm{d}x_1(t)}{\mathrm{d}t} = -x_1(t) + D_a(1 - x_1(t))\exp(\frac{x_2(t)}{1 + x_2(t)/\varphi}),$$

$$\frac{\mathrm{d}x_2(t)}{\mathrm{d}t} = -x_2(t) + B \cdot D_a(1 - x_1(t))\exp(\frac{x_2(t)}{1 + x_2(t)/\varphi}) + \beta(u(t - \tau) - x_2(t)),$$

$$y(t) = x_2(t),$$

here x_1 and x_2 represent dimensionless reactant conversion and temperature, u is the coolant temperature which is used as a manipulated variable. The parameters in the process are: $B = 8$, $D_a = 0.072$, $\varphi = 20$, $\beta = 0.3$ and $\tau = 0.4$. The sampling time is 0.1. The process output is the reactor temperature x_2. A feedforward network with 5 inputs $\{y(t-1), y(t-2), u(t-5), u(t-6), 1\}$, 3 hidden neurons, and one linear output mode is constructed to model the process. Based on the derived neural model, a neural d-step-ahead predictor[3] $(d = 5)$ is constructed.

Then we use 4 − 3 − 1 neural network to construct the GNPID controller. The PIDGDM algorithm is used to train the controller. The tuning parameters are respectively $\lambda_p = 1.5$, $\lambda_i = 0.035$, $\lambda_d = 4.5$, and $\alpha = 0.5$. A sequence of random steps is used as the setpoint of the system for training. After 5000 steps of training, the MSE reaches the target and the training procedure is finished. The trained neural GNPID controller is implemented in the closed-loop system. The black line in Fig. 3 shows the response of the neural control. A conventional PID control with $K_p = 1.5$, $K_d = 0.005$, and $K_i = 0.045$ tuned by Z-N method is also used to control the process. The dotted line in Fig. 3 represents the response of the conventional PID controller. Comparing with the conventional PID control, the neural GNPID control has obtained faster response and better performance.

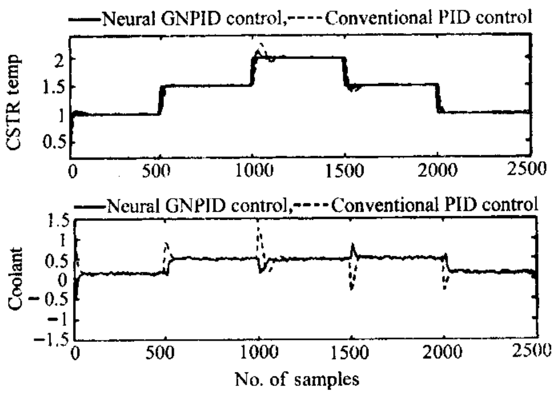

Fig. 3 Comparison between the GNPID and PID

Example 2 The van de Vusse reactor can be described by[4]

$$\frac{dC_A}{dt} = -k_1C_A - k_3C_A^2 + d(C_{Af} - C_A),$$

$$\frac{dC_B}{dt} = -k_1C_A - k_2C_B(-dC_B).$$

The isothermal series/paraller reactions:

$$A \to B \to C, \quad 2A \to D,$$

take place in the reactor. In the process, C_A and C_B are respectively the effluent concentration of component A and B, and d is the dilution rate. The values of the parameters are $k_1 = 50\ \mathrm{h}^{-1}$, $k_2 = 100\ \mathrm{h}^{-1}$, and $k_3 = 10\ \mathrm{l(mol^{-1}\ h)^{-1}}$. The concentration of A in the feed stream is given by C_{Af} and equals to 10 mol l^{-1}. Initially, the process is at steady-state with $C_A = 0.2143$ mol l^{-1} and $C_B = 0.1520$ mol l^{-1}. The objective of control is to control C_B by manipulation d. The time delay of the process is 0.052 h, The process is sampled at every 0.02 h. An external recurrent neural network with 4 hidden modes and 5 inputs is generated to describe the dynamic behavior of the process. A nonlinear recursive predictor based on the neural model[3] is produced to compensate the effect of time-delay.

The neural network based GNPID controller trained by the PIDGDM algorithm is used to control the process. The tuning parameters are $\lambda_p = 150$, $\lambda_i = 20$, $\lambda_d = 125$, and $\alpha = 0.75$. For comparison, the neural GNPID controller is also trained by the GDM algorithm with $\lambda = 30$ and $\alpha = 0.75$. In this simulation, both neural GNPID controllers have the same architecture, (i.e. $H = 100$) but the training algorithms are different. Fig. 4 shows the results of this comparison. Although the controller trained by the GDM algorithm presents more sluggish response than the response obtained by using PIDGDM method, it still results in relatively larger overshoot and undershoot.

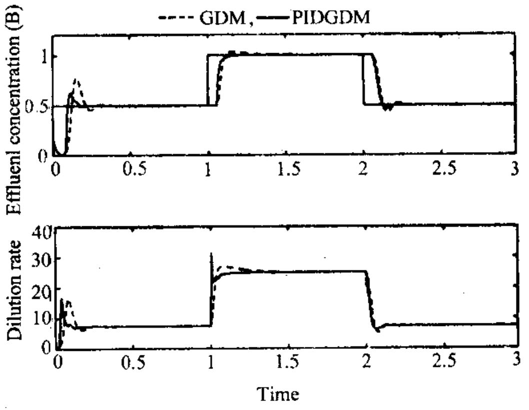

Fig. 4 Comparison between the PIDGDM and GDM

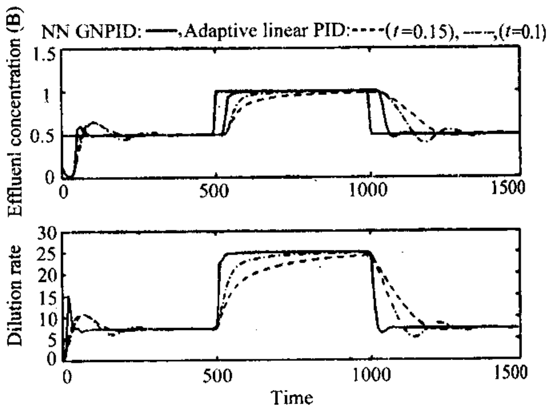

Fig. 5 Comparison between the GNPID and ALPID

The neural GNPID controller with PIDGDM scheme is also compared with a robust PID control strategy called adaptive linear PID control strategy. The adaptive linear PID (ALPID) controller is designed based on the principle of internal model control (IMC)[5]. The linear model used to describe the process is a second-order ARMA model which is on-line estimated by a recursive least squares (RLS) algorithm. Fig. 5 illustrates the comparison between the neural GNPID controller and the ALPID controller. The black line illustrates the control result of the NN GNPIDC. The

ALPID controller with $f = 0.15$ and 0.1 are respectively shown by (-·-·-) line and (------) line. The ALPID controller results in some oscillation at the operating point $r = 0.5$ when $f = 0.1$. At $r = 1.0$, the performance becomes better but the oscillation occurs again when the setpoint changes from 0.5 to 1.0. To reduce the oscillation, f is increased to 0.15, we see that the oscillation is damped at $r = 0.5$ but when $r = 1.0$ the response becomes very sluggish. However, in this case, the neural GNPID control obtains faster response and better performance.

6 Conclusion

In this paper, a PID gradient descent algorithm with momentum term (PIDGDM) has been proposed. The tuning degree of the algorithm is increased so that it is possible to find a group of tuning parameters to obtain a satisfactory training performance. The derived convergent condition for the PIDGDM provides a way to choose proper parameters so as to lead to stable and fast convergence. Moreover, a generalized nonlinear PID control strategy is proposed and is implemented using neural networks. Based on the system performance information, the learning mechanism can adjust the weights of the neural networks to approximate the mapping from the controller inputs to its outputs. Simulations have illustrated that the neural network based GNPID control can handle some nonlinear processes effectively.

References

[1] Bazaraa M S and Shetty C M. Nonlinear Programming: Theory and Algorithms [M]. New York: John Wiley & Sons Inc., 1979

[2] Ray W. New approaches to the dynamics of nonlinear systems with applications for process and control system design [J]. Chemical Process Control, 1981, (2): 245 – 268

[3] Tan Y and De Keyser R. Neural network based adaptive predictive control [A]. Clarke D ed. Advances in Model-Based Predictive Control [M], London: Oxford University Press, 1994, 358 – 369

[4] Aoyama A, Doyle F and Venkatasubramanian V. Control affine neural network approach for nonminimumphase nonlinear process control [J]. J. Proc. Contr., 1996, 6(1): 17 – 26

[5] Morari M and Zafirious E. Robust Process Control [M]. Englewood Cliffs, NJ: Prentice-Hall, 1989

[6] Bohn C and Atherton D P. An analysis package comparing PID anti-windup strategies [J]. IEEE Control Systems Magazine, 1995, 15(2): 34 – 40

Appendix A Proof of Lemma 1

Consider the relation:

$$F(z^{-1}) = \varphi(z^{-1})/\psi(z^{-1}) > \min\varphi(z^{-1})/\max\psi(z^{-1}). \tag{A1}$$

Note that if $F(z^{-1})$ is asymptotically stable then $0 < \alpha < 1$. Thus, it leads to

$$z^{-1} = 1/\alpha > 1. \tag{A2}$$

Then, it leads to $\max\psi(z^{-1}) = \lim\limits_{\alpha\to 0}\psi(z^{-1}) = 1$. Considering $z^{-1} = -\lambda_1/2\lambda_2$, it yields the stationary point

$$d\varphi(z^{-1})/dz^{-1} = 0.$$

Since $d^2\varphi(z^{-1})/dz^{-2} = 2\lambda_2 > 0$, the only minimum value of $\varphi(z^{-1})$ is

$$\min\ \varphi(z^{-1}) = (4\lambda_0\lambda_2 - \lambda_1^2)/4\lambda_2. \tag{A3}$$

Combining (A1), (A2) with (A3), the lemma is proved.

Appendix B Proof of Theorem 2

Suppose

$$\Delta f \approx (\partial f^{\mathrm{T}}/\partial x_{k-1})\Delta x_{k-1} = -\|\partial f/\partial x_{k-1}\|^2(\lambda_0 + \lambda_1 z^{-1} + \lambda_2 z^{-2})/(1 - \alpha z^{-1}).$$

Considering $\lambda_0, \lambda_2 > 0, 0 < \alpha < 1$, and Lemma 1, also, choosing $4\lambda_0\lambda_2 \geqslant \lambda_1^2$, it leads to

$$(\lambda_0 + \lambda_1 z^{-1} + \lambda_2 z^{-2})/(1 - \alpha z^{-1}) > (4\lambda_0\lambda_2 - \lambda_1^2)/4\lambda_2 \geqslant 0.$$

Then, it results in $\Delta f \leqslant 0$. So that f is a descent function. Also

$$d \triangleq -(\partial f/\partial x_{k-1})(\lambda_0 + \lambda_1 z^{-1} + \lambda_2 z^{-2})/(1 - \alpha z^{-1}) \tag{B1}$$

is defined as the descent direction.

Appendix C Proof of Theorem 3

Let $\Omega = \{x^* : \partial f/\partial x_{k-1}^* = 0\}$. If the coefficients are chosen as $\lambda_0, \lambda_2 > 0, 0 < \alpha < 1$, and $4\lambda_0\lambda_2 \geqslant \lambda_1^2$, then, in terms of Theorem 2, f is a descent function. Note also that the algorithm map:

$$A(x_k) = x_{k-1} + \alpha\Delta x_{k-1} - (\partial f/\partial x_{k-1})(\lambda_0 + \lambda_1 z^{-1} + \lambda_2 z^{-2})$$

is closed in the complement of Ω. If $x^* \notin \Omega$, then $[\partial f/\partial x_{k-1}^*]^{\mathrm{T}} d < 0$, where d is defined by (B1). Since d is a descent direction, hence $f(\eta) < f(x^*)$ for $\eta \in A(x_k)$. Assume that the sequence produced by the algorithm is contained in a compact set, then by Theorem 1, the algorithm converges to a point within $\Omega = \{x^* : \partial f/\partial x_{k-1}^* = 0\}$.

本文作者简介

谭永红 见本刊 2000 年第 3 期第 414 页.

党选举 1965 年生. 桂林电子工业学院副教授. 研究领域为智能控制及智能自动化仪表.

范·高温伯格 见本刊 2000 年第 3 期第 414 页.

迈达德·萨义夫 加拿大西蒙·弗雷泽大学工程科学系教授, 加拿大大不列颠哥伦比亚 IEEE 控制学会主席. 研究领域为非线性系统及控制, 故障检测与诊断和神经网络在建模与控制中的应用.