Article ID: 1000 - 8152(2001)03 - 0456 - 05

Simple Recurrent Neural Network Control for Non-minimum Phase Nonlinear System *

LI Xiang, CHEN Zengqiang and YUAN Zhuzhi
(Department of Automation, Nankai University · Tianjin, 300071, P. R. China)

Abstract: We discuss a uniform structure of simple recurrent neural networks, based on which a novel neural control system is developed. With the introduction of the weighted control information into the neural controller's cost function, the method is an extension of the common neural networks controller proposed before. The stability of the whole neural control system is demonstrated and its effectiveness is verified via simulation.

Key words: simple recurrent neural networks; non-minimum phase system; nonlinear system; neural network control
Document code: A

非最小相位非线性系统的简单递归神经网络控制

李 翔 陈增强 袁著祉
(南开大学自动化系·天津, 300071)

摘要: 从简单递归神经网络的统一结构出发设计了简单递归神经网络控制器, 在引入了控制加权的目标函数下优化神经网络权值学习, 因此是通常意义的神经网络控制器的推广. 证明了整个系统的稳定性, 并通过仿真验证了控制器的有效性.

关键词: 简单递归神经网络; 非最小相位系统; 非线性系统; 神经网络控制

1 Introduction

Neural network theory has gained great achievements in the control literature these ten years. Owing to its simple structure and dynamical mappling capability, simplified recurrent neural network has attracted more and more attention, based upon which have researchers thrown their enthusiasm onto developing kinds of control paradigms[1~6]. For instance, there is a nonlinear neural controller with diagonal neural networks in [1], with some detailed modifications in [2] and similar control scheme proposed in [3]. In [4,5] there are studies on nonlinear modeling and adaptive predictive control based on Elman network, while in [6] a stable adaptive controller was designed. However, all those proposed works took no account of their control signal variation in their controllers' cost functions, and they have difficulties in controlling the rigid non-minimum phase nonlinear systems.

In this paper a uniform structure of simple recurrent neural networks (SRNN) is given. Based on it, a simple recurrent neural network control system (SRNNC) which consists of neural network identifier (NNI) and neural network controller (NNC) is devised in the following. Not the same as the usual NNC proposed before, a more general cost function for neural network controller is given, where a term including the weighted control signals is introduced into controller's cost function. Hence, the gained NNC is capable of controlling the non-minimum phase nonlinear systems. Given some conditions the learning rates satisfied, it is demonstrated in the following sections that the whole control system's stability can be guaranteed.

2 Simple recurrent neural network structure

In [1 ~ 6], kinds of simplified recurrent neural network structures have been discussed. However, they all can be unified into the uniform structure introduced in the following (Fig. 1).

We can get network descriptive equations for Fig. 1

* Foundation item: supported by National 863 CIMS Project Foundation (863 - 511 - 945 - 010), Natural Science Foundation of Tianjin (983602011) and University Key Teacher Foundation of Ministry of Education (0065).
Received date: 2000 - 07 - 07; Revised date: 2001 - 01 - 15.

as follows:

$$\begin{cases} O(k) = W^{o}S(k) \\ S(k) = h(W^{c}S^{c}(k) + W^{I}X(k-1)), \\ S^{c}(k) = S(k-1). \end{cases} \quad (2.1)$$

$h(\cdot)$ is sigmodal activation function. The context layer $S^{c}(k) = S(k-1)$ functions as the memory of the hidden layer state, and it also brings more abundant dynamics into SRNN than that of multi-layer feedforward neural network.

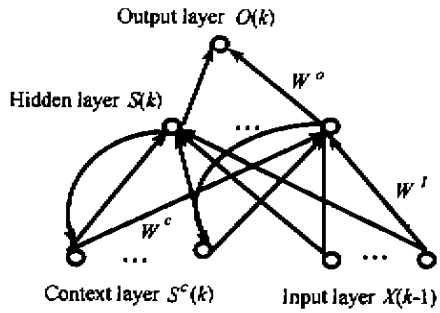


Fig. 1 SRNN uniform structure

From Fig.1, if the input layer $X(k-1)$ has only one node $u(k-1)$, SRNN will be Elman-structure networks in [4,6]. Furthermore, if $y(k-1),\cdots,y(k-n)$ are introduced into the input layer, it will be the extended Elman networks in [5]. Given the previous control sequence imported to the input layer, the network structure of [1,3] is a special case of our uniform SRNN. Various of the network structures mentioned above have the same character, i.e., their origin is a multi-layer feedforward neural network (MFNN), and they all have the context layer which saves the activation response of the hidden layer. In this paper, these kinds of networks have been unified into the uniform structure showed in Fig.1 named after simple recurrent neural networks.

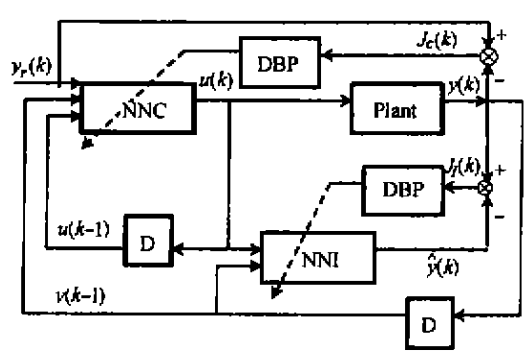


Fig. 2 SRNNC structure

Fig.2 is the structure of our proposed SRNNC. The system control signal is generated from NNC, and NNI models the nonlinear system to approximate its dynamics. As a more general cost neural controller function than that of [1,3], (2.3) takes account of the control signal variation with a term as the weighted control signals added into the cost function. In the case of system oscillation or divergence, our neural control system designed later will constrain the divergence and guarantee the whole system performance and its stability, so SRNNC will be more general applicable than those neural network control schemes proposed before. In Fig.2,

$$J_I(k) = \frac{1}{2}[y(k) - \hat{y}(k)]^2, \quad (2.2)$$

$$J_c(k) = \frac{1}{2}[y_r(k) - y(k)]^2 + \frac{\lambda}{2}u^2(k). \quad (2.3)$$

NNI and NNC both are the structure of SRNN (Fig.1), and their input/output mappings are detailed as follows:

NNI:

$$\hat{y}(k) = O_I(k) = \sum_{j=1}^{N_I} W_j^{o}(k)S_j(k), \quad (2.4)$$

$$S_j(k) = h\Big(\sum_{i=1}^{N_I} W_{i,j}^{c}(k)S_i^{c}(k) + \sum_{i=1}^{n} W_{i,j}^{I}(k)X_i^{I}(k)\Big); \quad (2.5)$$

NNC:

$$u(k) = O_c(k) = \sum_{j=1}^{N_c} V_j^{o}(k)T_j(k), \quad (2.6)$$

$$T_j(k) = g\Big(\sum_{i=1}^{N_c} V_{i,j}^{c}(k)T_i^{c}(k) + \sum_{i=1}^{m} V_{i,j}^{I}(k)X_i^{c}(k)\Big), \quad (2.7)$$

such that the active function $h(\cdot), g(\cdot)$ are specialized here as $h(x) = g(x) = \dfrac{1}{1+e^{-x}}$.

3 Simple recurrent neural network controller design

With the uniform SRNN structure shown in Section 2, our NNI and NNC, and their weight, W^{0}, W^{c}, W^{I} and V^{o}, V^{c}, V^{I}, will be adapted as follows,

$$W(k+1) = W(k) + \eta_I\Big(-\frac{\partial J_I(k)}{\partial W(k)}\Big), \quad (3.1)$$

$$V(k+1) = V(k) + \eta_c\Big(-\frac{\partial J_c(k)}{\partial V(k)}\Big). \quad (3.2)$$

Note $e_I(k) = y(k) - \hat{y}(k)$, the tuning for NNI is

$$\Delta W(k) = W(k+1) - W(k) = \eta_I e_I(k)\frac{\partial O_I(k)}{\partial W(k)}. \quad (3.3)$$

Let $h' = \left.\frac{\mathrm{d}h(x)}{\mathrm{d}x}\right|_{x=\sum_{i=1}^{N_I} W_{i,j}^c(k)S_i(k-1)+\sum_{i=1}^{m} W_{i,j}^I(k)X_i^I(k)}$,

$g' = \left.\frac{\mathrm{d}g(x)}{\mathrm{d}x}\right|_{x=\sum_{i=1}^{N_c} V_{i,j}^c(k)T_i(k-1)+\sum_{i=1}^{m} V_{i,j}^I(k)X_i^c(k)}$.

For all $W^o(k), W^c(k), W^I(k)$ in NNI (2.4), (2.5), $\frac{\partial O_I(k)}{\partial W(k)}$ are as follows:

$$\begin{cases} \dfrac{\partial O_I(k)}{\partial W_j^o(k)} = S_j(k), \\ \dfrac{\partial O_I(k)}{\partial W_{i,j}^I(k)} = \dfrac{\partial O_I(k)}{\partial S_j(k)}\dfrac{\partial S_j(k)}{\partial W_{i,j}^I(k)} = W_j^o(k)h'X_i^I(k). \end{cases} \tag{3.4}$$

$$\frac{\partial O_I(k)}{\partial W_{i,j}^c(k)} = \frac{\partial O_I(k)}{\partial S_j(k)}\frac{\partial S_j(k)}{\partial W_{i,j}^c(k)} = W_j^o(k)h'\Big(S_i(k-1) + \sum_{l=1}^{N_I} W_{l,j}^c(k)\frac{\partial S_l(k-1)}{\partial W_{i,j}^c(k)}\Big). \tag{3.5}$$

Generally, the modification of $W(k)$ in every time step is very small[4], so it is presumed

$$\frac{\partial S_l(k-1)}{\partial W_{i,j}^c(k)} = \frac{\partial S_l(k-1)}{\partial W_{i,j}^c(k-1)},$$

and (3.5) can be rewritten as:

$$\frac{\partial O_I(k)}{\partial W_{i,j}^c(k)} = \frac{\partial O_I(k)}{\partial S_j(k)}\frac{\partial S_j(k)}{\partial W_{i,j}^c(k)} = W_j^o(k)h'\Big(S_i(k-1) + \sum_{l=1}^{N_I} W_{l,j}^c(k)\frac{\partial S_l(k-1)}{\partial W_{i,j}^c(k-1)}. \tag{3.6}$$

As to the adapting for NNC, while

$$\frac{\partial y(k)}{\partial V(k)} \approx \frac{\partial \hat{y}(k)}{\partial V(k)}, e_c(k) = y_r(k) - y(k),$$

we obtain

$$\frac{\partial J_c(k)}{\partial V(k)} = e_c(k)\frac{\partial e_c(k)}{\partial V(k)} + \lambda u(k)\frac{\partial u(k)}{\partial V(k)} = -e_c(k)\frac{\partial \hat{y}(k)}{\partial V(k)} + \lambda u(k)\frac{\partial O_c(k)}{\partial V(k)}. \tag{3.7}$$

If $u_i(k)$ is the first element in the sequence $X^I(k)$ and $y_u = \sum_{j=1}^{N_I} W_j^o(k)h'W_{1,j}^I(k)$, then

$$\frac{\partial \hat{y}(k)}{\partial V(k)} = \frac{\partial \hat{y}(k)}{\partial u(k)}\frac{\partial u(k)}{\partial V(k)} = \sum_{j=1}^{N_I} W_j^o(k)h'W_{1,j}^I(k)\frac{\partial O_c(k)}{\partial V(k)} = y_u\frac{\partial O_c(k)}{\partial V(k)}.$$

Denote $e_c(k)$ as e_c. The network weights tuning for NNC is

$$\Delta V(k) = V(k+1) - V(k) = \eta_c[e_c y_u - \lambda u(k)]\frac{\partial O_c(k)}{\partial V(k)}. \tag{3.8}$$

For NNC $V^o(k), V^c(k), V^I(k)$, from (2.6), (2.7) and we can get $\frac{\partial O_c(k)}{\partial V(k)}$, similar to those of NNI, as:

$$\frac{\partial O_c(k)}{\partial V_j(k)} = T_j(k),$$

$$\frac{\partial O_c(k)}{\partial V_{i,j}^I(k)} = \frac{\partial O_c(k)}{\partial T_j(k)}\frac{\partial T_j(k)}{\partial V_{i,j}^I(k)} = V_j^o(k)g'X_i^c(k). \tag{3.9}$$

$$\frac{\partial O_c(k)}{\partial V_{i,j}^c(k)} = \frac{\partial O_c(k)}{\partial T_j(k)}\frac{\partial T_j(k)}{\partial V_{i,j}^c(k)} = V_j^o(k)g'\Big(T_i(k-1) + \sum_{l=1}^{N_c} V_{l,j}^c(k)\frac{\partial T_l(k-1)}{\partial V_{i,j}^c(k-1)}. \tag{3.10}$$

Since the dynamical backward propagation algorithm (DBP) is used here to tuning the networks, the learning rates η_I, η_c will influence the system performance. If the learning rate is much bigger, it will deteriorate the whole neural control system's stability, and if much smaller the perfect system performance will not be obtained, It's proved latrer in Theorem 1 and Theorem 2 that given η_I, η_c satisfied (3.11), (3.12) respectively, NNI and NNC will be convergent in exponential speed, so the whole system is asymptotic stable. Define $\|\cdot\|$ in Theorem 1,2 as the usual Euclidean norm in $\mathbb{R}^n$.

Theorem 1 If $W^o(k), W^c(k), W^I(k)$ for NNI (2.4), (2.5) are updated along (3.3), (3.4) and (3.6), then NNI (2.4), (2.5) will be convergent in exponential speed, given that its learning rate η_I holds the condition as follows

$$0 < \eta_I < \frac{2}{\left\|\frac{\partial O_I(k)}{\partial W(k)}\right\|^2}. \tag{3.11}$$

Proof Define the Lyapunov function $L(k) = \frac{1}{2}e_I^2(k) = \frac{1}{2}(y(k) - \hat{y}(k))^2$, the proof is similar to that of [1].

Theorem 2 If $V^o(k), V^c(k), V^I(k)$ for NNC (2.6), (2.7) are updated along (3.8) ~ (3.10), then NNC (2.6), (2.7) will be convergent in exponential speed, given that its learning rate η_c holds the condition

as follows

$$0 < \eta_c < \frac{2}{\left\|\frac{\partial O_c(k)}{\partial V(k)}\right\|^2 (\gamma_u^2 + \lambda)}. \quad (3.12)$$

Proof Define the Lyapunov function

$$L(k) = J_c(k) = \frac{1}{2}e_c^2(k) + \frac{\lambda}{2}u^2(k),$$

then

$$\Delta L(k) = [e_c(k) + \frac{1}{2}\Delta e_c(k)]\Delta e_c(k) + \lambda[u(k) + \frac{1}{2}\Delta u(k)]\Delta u(k) \equiv \Delta L_1(k) + \Delta L_2(k),$$

such that $e_c(k+1) = e_c(k) + \Delta e_c(k)$. From (3.8),

$$\Delta e_c(k) = \frac{\partial e_c(k)^T}{\partial V(k)}\Delta V(k) = -\left[\frac{\partial \hat{y}(k)}{\partial u(k)}\frac{\partial u(k)}{\partial V(k)}\right]^T \Delta V(k) = -\gamma_u \eta_c [e_c \gamma_u - \lambda u(k)]\left\|\frac{\partial O_c(k)}{\partial V(k)}\right\|^2,$$

so

$$\Delta L_1(k) = -\gamma_u e_c \eta_c [e_c \gamma_u - \lambda u(k)]\left\|\frac{\partial O_c(k)}{\partial V(k)}\right\|^2 + \frac{1}{2}\gamma_u^2 \eta_c^2 [e_c \gamma_u - \lambda u(k)]^2 \left\|\frac{\partial O_c(k)}{\partial V(k)}\right\|^4.$$

Similarly,

$$\Delta L_2(k) = \lambda u(k)\eta_c [e_c \gamma_u - \lambda u(k)]\left\|\frac{\partial O_c(k)}{\partial V(k)}\right\|^2 + \frac{\lambda}{2}\eta_c^2 [e_c \gamma_u - \lambda u(k)]^2 \left\|\frac{\partial O_c(k)}{\partial V(k)}\right\|^4,$$

$$\Delta L(k) = -\left\|\frac{\partial O_c(k)}{\partial V(k)}\right\|^2 [e_c \gamma_u - \lambda u(k)]^2 \left|\eta_c - \frac{1}{2}(\gamma_u^2 + \lambda)\eta_c^2 \left\|\frac{\partial O_c(k)}{\partial V(k)}\right\|^2\right|. \quad (3.13)$$

As a result, to ensure NNC be convergent in exponential speed, it should be $\Delta L(k) < 0$, i.e.

$$\eta_c \left\{2 - (\gamma_u^2 + \lambda)\eta_c \left\|\frac{\partial O_c(k)}{\partial V(k)}\right\|^2\right\} > 0.$$

So η_c must satisfy condition (3.12) to guarantee the asymptotical stability of NNC.

Remark In linear system theory, the general minimum-variance control (GMV; its cost function has the form of $J_{GMVC} = (y_r(k) - y(k))^2 + \lambda u^2(k)$) can select a sufficient big λ to guarantee the closed-loop system eigenfunction $B + \lambda A \doteq 0$ stable if B is unstable, i.e. non-minimum phase system. If $\lambda \approx 0$, GMV degrades to the *minimum-variance* control (MVC, its cost function has the form of $J_{MVC} = (y_r(k) - y(k))^2$) and the closed-loop system eigenfunction is $B = 0$, hence MVC can't control the non-minimum phase system. Nonlinear systems haven't their analytic closed-loop eigenfunctions, and the effect of λ in our neural system is not analytic either. Neural controller uses gradient-based learning algorithm to tune its weights, and this greedy optimization algorithm drives the whole system to its cost *function*'s local *minima*. For [1,3], *minimize* the cost *function* $J_c' = \frac{1}{2}(y_r(k) - y(k))^2$ to obtain its local minima, i.e.

$$\frac{\partial J_c'}{\partial u(k)} = 0 = -[y_r(k) - y(k)]\frac{\partial y(k)}{\partial u(k)} \approx -\frac{\partial \hat{y}(k)}{\partial u(k)}[y_r(k) - y(k)] = -\gamma_u[y_r(k) - y(k)]. \quad (3.14)$$

Generally, $\gamma_u \neq 0$, so (3.14) changes to $y_r(k) - y(k) = 0$, that is to say, under J_c' in [1,3], the weight tuning algorithms (gradient-tased) will drive the whole neural *system* to its local *minima* as $y_r(k) = y(k)$, while the nonlinear *system* is non-minimum phase, this kind of neural controller can't constrain its control signal's divergence. Similarly, minimize the cost function of our NNC

$$J_c = \frac{1}{2}(y_r(k) - y(k))^2 + \frac{1}{2}\lambda u^2(k)$$

as

$$\frac{\partial J_c}{\partial u(k)} = 0 \approx -\gamma_u[y_r(k) - y(k)] + \lambda u(k).$$

Here the local minima are $u(k) \approx \frac{[y_r(k) - y(k)]}{\lambda}\gamma_u$. In the case of non-minimum phase system, our neural controllers can constrain the control signal's divergence because of the influence from λ. In simulation studies followed, our neural controller with a nonzero λ succeeds in controlling those non-minimum phase examples while the controller with $\lambda = 0$ case fails.

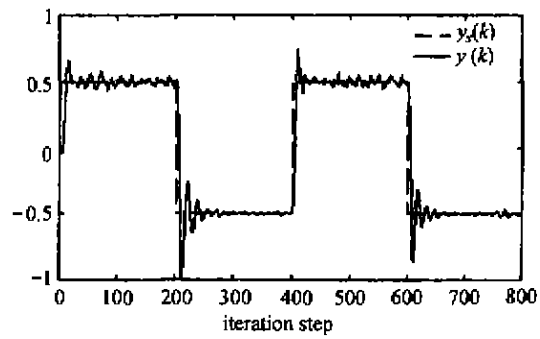
(3.11) and (3.12) give an upper bound for η_l and η_c to guarantee the wholeneural system's stability, and if select the optimal learning rate η_l^*, η_c^* as (3.15), the rapid or optimal learning convergence[1,7] will be guaranteed.

$$\eta_l^* = \frac{1}{\left\|\frac{\partial O_l(k)}{\partial W(k)}\right\|^2}, \quad \eta_c^* = \frac{1}{\left\|\frac{\partial O_c(k)}{\partial V(k)}\right\|^2 (\gamma_u^2 + \lambda)}. \quad (3.15)$$

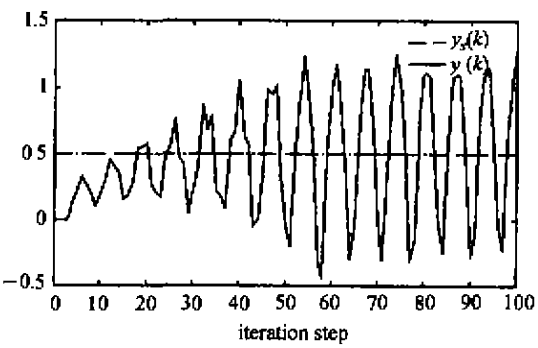
4 Simulation study

This non-minimun phase example is identified from a laboratory-scale liquid level system. The system consists of a d.c. water pump feeding a conical flask which in turn feeds a square tank, giving the system second-order dynamics. The controllable input is the voltage to the pump motor and the plant output is the height of the water in the conical flask. The identified plant model is as:

$$\begin{aligned} y(k) = {} & 0.97722y(k-1)+0.3578u(k-1)+ \\ & 0.4589u(k-2)-0.3103y(k-1)u(k-1)- \\ & 0.04228y^2(k-2)+0.1663y(k-2)u(k- \\ & 2)-0.03259y^2(k-1)y(k-2)- \\ & 0.3513y^2(k-1)u(k-2)+ \\ & 0.3084y(k-1)y(k-2)u(k-2)+ \\ & 0.1087y(k-2)u(k-1)u(k-2)+ \\ & 0.2573y(k-2)e(k-1)+ \\ & 0.2939y^2(k-2)e(k-1)+ \\ & 0.4770y(k-2)u(k-1)e(k-1). \quad (4.1) \end{aligned}$$



(a) The case of λ=8.5



(b) The case of λ=0

Fig. 3 System output of Example 1

Our simulation goal here is to make the height of the water in the conical flask follow the set values $y_s(k)$ which switch between 0.5 and −0.5 every 200 iterations in the presence of external random noise $e(k)$ bounded in $[-0.1, 0.1]$. To control this rigid liquid level system with rich nonlinearity and exterior noise, we select NNI and NNC structure as 3 − 7 − 1. The learning rate η_I, η_c both are initialized as 0.1, with inital network weight values in $[-0.5, 0.5]$. For the system (4.1), select $\lambda = 8.5$, after pre-training NNI in 1500 epochs, start closed-loop control and NNC works (see Fig.3(a)). While select $\lambda = 0$, i.e.using [1,3], Fig. 3(b) is the plant output in the first 100 iterations and the whole neural control system diverges after 500 iterations.

5 Conclusions

In the paper some new works were investigated in the following aspects:

The first is that a uniform simple recurrent neural network stucture was given in this paper. The second is a more general cost function was introduced into the neural network control, and for the non-minimum phase nonlinear system SRNNC, which we devised in the paper, showed its unique advantage over the usual neural network control schemes. We can say that the usual neural network control schemes such as [1,3] are the special case of our SRNNC.

References

[1] Ku Chao-Chee, Lee K and Wang Y. Diagonal recurrent neural networks for dynamic system control [J]. IEEE Trans. Neural Networks, 1995,6(1):144 − 156

[2] Chen Wei and Wu Jie. A modification to a recurrent neural networks algorithm [J]. Control Theory and Applications, 1998,15(5):814 − 816 (in Chinese)

[3] Wang Yaonan. An Intelligent control using a kind of dynamical recurrent neural networks [J]. Control and Decision, 1995,10(6):508 − 513 (in Chinese)

[4] Pham D T and Liu X. Training of Elman network and dynamic system modelling [J]. Int J. Systems Science, 1996,27(2):221 − 226

[5] Li Xiang, Chen Zengqiang and Yuan Zhuzhi. An extended Elman neural networks-based nonlinear self-tuning controller [J]. Automatic Instrument,1999,20(12):17 − 20 (in Chinese)

[6] Li Xiang, Chen Zengqiang and Yuan Zhuzhi. Nonlinear stable adaptive control based upon Elman networks [J]. Appl. Math. J. Chinese Univ. Ser.B, 2000,15(3):332 − 340

[7] Polycarpou M M and Ioannou P A. Learning and convergence analysis of neural-type structured networks [J]. IEEE Trans. Neural Networks, 1992,3(1):39 − 50

本文作者简介

李 翔 1975年生.南开大学信息技术科学学院自动化系博士研究生.研究兴趣包括神经网络自适应控制,智能控制,混沌同步与控制.

陈增强 1964年生.工学博士,南开大学信息技术科学学院自动化系教授.研究方向为自适应控制,预测控制,智能控制.曾获教育部及天津市科技进步三等奖.

袁著祉 1937年生.南开大学信息技术科学学院自动化系教授,博士生导师.研究方向为自适应过程控制,智能控制.曾获国防科工委光华一等奖和教育部科技进步一等奖.