Approximate Linearization of Nonlinear Systems A Neural Network Approach*

PEI Hailong and ZHOU Qijie

(Department of Automatic Control Engineerang, South China University of Technology · Guangzhou, 510641, PRC)

Abstract: Recent researches show that neural networks have the ablilty to approximate a function as well as its derivatives. This result offers a promising opportunity to introduce neural network theory into nonlinear system control. In this paper a novel method of approximate nonlinear system linearization with neural networks is proposed. The network approximator is designed to integrate the involutive equation of a nonlinear system whether the integrability condition is satisfied or not. Simulation results show that this method is feasible.

Key words: neural networks; approximation; linearization

1 Introduction

It is well known that differential geometric methods have become powerful techniques for understanding input-output and full state linearization of nonlinear systems[1,2]. However, the conditions for feedback linearization of nonlinear systems are restrictive. So, it is of practical interest to investigate extensions when these conditions can not be satisfied well but violated only slightly. There have been proposed some methods, among them, the approximate linearization has shown much feasibility[3].

Most results concerning approximate transformations for feedback linearizable systems or non feedback linearizable systems involve approximations of the systems themselves[4,5]. In [3], Krener introduces ρ-th order linear systems and the equivalent concepts of ρ-th order involutivity and ρ-th order integrability at a point for certain distributions. He also provided a constructive algorithm that yields an approximate transformation. Along this route, several algorithms have been proposed for the realization of approximate transformation[6~9].

It has become a general recognition that artificial neural networks (ANNs) can be applied to approach any given function with any given precision[10]. Recent research shows that, besides the given function, the derivative of the function can be also approximated as well with the same neural network[11,12]. This result offers us an opportunity to introduce neural networks to the powerful nonlinear system linearization theories, because the Lie algebrics and the Lie brackets of a nonlinear system may be constructed with neural networks.

Inspired by the work of approximate linearization in [3,4], we propose a new nonlinear system approximation strategy in this paper. With the capability of neural network approximation of a function and its derivatives, the proposed network approximator can be automati-

* The project is supported by National Natural Foundation of China (69604001) and Guangdong Provincial Natural Science Foundation of China (960234).
Manuscript received Apr. 15, 1996, revised Apr. 4, 1997.

cally trained to approximately satisfy the integrability condition. This means that, no matter whether the linearization condition is satisfied or not, the given nonlinear system can by approximately linearizied with the geometric method.

The rest of this paper is organized as follows: First, the approximation capability of neural networks is discussed in Section 2. In Section 3, after an introduction of nonlinear system linearizability condition, the approximate linearization method with neural networks as well as a convergence analysis are proposed. In Section 4, a typical example of a beam and ball system control is given to show the feasibility of this method. In Section 5, we give the conclusion and discussions.

2 Neural Networks and Their Approximation Capabilities

The capability of sufficiently complex multilayer feedforward networks to approximate an unknown mapping $f: \mathbb{R}^r \to \mathbb{R}$ arbitrarily well has been investigated in detail in[10]. Kurt Hornik. et. al., extended this result into Sobolev space in their recent work[11], and rigorously pointed out that multilayer feedforward networks with a single hidden layer and an appropriately smooth hidden layer activation function are capable of approaching an arbitrary function and its derivatives.

An example of the application of this result is to approximately integrate partial differential equations which will show its importance in the next section. We note that any network which is suitably trained to approximate a mapping satisfying some nonlinear partial differential equations will have an output function that itself approximately satisfies this partial differential equation by virtue of its approximation of the mapping's derivatives. To construct these neural networks, an extended backpropagation training algorithm is proposed in the Appendix, where, for the reason of simplicity, the errors of only one degree derivatives are considered.

3 Approximate Linearizatin with Neural Networks

Consider a SISO nonlinear system of the form

$$\dot{x} = f(x) + g(x)u, \tag{1}$$

where f and g are smooth ($\mathbb{C}^r$ with r sufficiently large) vector fields defined on an open subset of $\mathbb{R}^n$. Let x^o be an equilibrium point of the undriven system, i. e. $f(x^o) = 0$. The conditions for input-to-state linearization are well known:

Theorem 3. 1[2] The nonlinear system (2) can be locally (around x^o) transformed into a controllable linear system by state feedback and a change of coordinates if and only if

$$\dim \operatorname{span}\{g \quad ad_f g \quad \cdots \quad ad_f^{n-1} g\} = n \tag{2}$$

and

$$\Delta(f,g) := \operatorname{span}\{g \quad ad_f g \quad \cdots \quad ad_f^{n-2} g\} \tag{3}$$

is involutive, both in a neighborhood of x^o.

The first condition (3) can be thought of as a local controllability (or linear controllability) condition and is usually satisfied by the type of systems that we encounter (or design) in engineering. The second condition(4) is an integrability condition that is violated by many practical systems[6,7].

When these conditions are satisfied, one can define a nutural output for the system(2).

Indeed, by the Frobenius theorem, conditions (3) and (4) imply that the partial differential equation

$$\frac{\partial h}{\partial x}[g \quad ad_f g \quad \cdots \quad ad_f^{n-2} g] = 0 \tag{4}$$

has a nontrivial solution $h(x)$.

While the involutivity condition of Δ of (4) is not always satisfied in engineering practice, one possible way to linearize system (2) in engineering is to take an approximate approach to integrate Δ. The results discussed in last section inspire us to engage a neural network $\Phi(x,w) \in \Sigma(G), w^T = (\beta^T, \gamma^T)$ to learn to approximately integrate the partial differential equation (5).

The involutiveness of span $\{g, ad_f g, \cdots, ad_f^{n-2} g\}$ is equivalent to the existence of a nonzero scale function $h: M \to R$ satisfying $< dh, ad_f^i g >= 0, i = 0,1,\cdots,n-2$. Hence, feedback linearizability of the system of (2) reduces to the existence of function $h: M \to R$ such that

$$\begin{cases} < dh, ad_f^i g >= 0, & i = 0,1,\cdots,n-2, \\ < dh, ad_f^{n-1} g > \neq 0. \end{cases} \tag{5}$$

For a system of (2), if a neural network $\hat{h}(x) = \Phi(x,w)$ can approximately satisfy the involutivity condition (5), or equivalently

$$\begin{cases} \| < d\hat{h}, ad_f^i g > \| \leqslant \varepsilon, & i = 0,1,\cdots,n-2, \\ < d\hat{h}, ad_f^{n-1} g > \neq 0 \end{cases} \tag{6}$$

the system (2) is said to be approximately linearizable with neural networks.

The learning process of this neural network can be implemented by the simple backpropagation method with the difference that the neural network has to approach not only a function but the compositions of its derivatives with other functions as well.

According to (7), the learning error function is defined as

$$E = \sum_{i=1}^{n-2} \| < d\phi, ad_f^i g > \|^2. \tag{7}$$

Usually the function h in (6) is chosen as the naturel output function of the system (2), so an another term should be added into the error function

$$E = \| \phi - y \|^2 + \sum_{i=1}^{n-2} \| < d\phi, ad_f^i g > \|^2. \tag{8}$$

To train the engaged neural network $\Phi(x,w)$, backpropagation algorithm is used in this paper. The learning rule

$$\frac{dw}{dt} = -\eta \frac{\partial E}{\partial w} \tag{9}$$

is a little more complicated one compared with the existing backpropagation method. An examplary one with $i = 1$ is given in the appendix.

With the trained neural network in hand, one then defines a change of coordinates $\mu(x) = (\mu_1(x), \cdots, \mu_n(x))$ by

$$\begin{cases} \mu_1(x) = \phi(x), \\ \mu_i(x) = L_f \mu_{i-1}(x), & 2 \leqslant i \leqslant n, \end{cases} \tag{10}$$

then we have the new system

$$\begin{cases} \dot{\mu}_1 = \mu_2 + \delta_1 u, \\ \cdots \\ \dot{\mu}_{n-1} = \mu_n + \delta_{n-1} u, \\ \dot{\mu}_n = d(x) + a(x)u, \end{cases} \tag{11}$$

where $\delta_i = 1 < d\mu, ad_f^i g >, i = 1, \cdots, n-2$, and $\| \delta_i \| \leqslant \varepsilon$, Neglect the small value terms, and choose

$$u = \frac{1}{a(x)}[-d(x) + v], \tag{12}$$

the original system can be transformed approximately into a linear system in Brunovsky canonical form by the coordinate change $\mu = \mu(x)$ and the state feedback (13).

For the tracking problem, the error bounds are analyzed as follows:

Choose

$$v = y_d^{(n)} + k_1(\mu_{n-1} - y_d^{(n-1)}) + \cdots + k_{n-1}(\mu_1 - y_d), \tag{13}$$

where $k_1, \cdots, k_{n-1}$ is selected such that $s_n + k_1 s_{n-1} + \cdots + k_{n-1}s$ is a stable polynomial. Define $e = Y_d - \mu$, the closed loop system can be written as

$$\dot{e} = Ae + \delta u, \tag{14}$$

where A is a Hurwitz matrix defined by v of (14). Suppose that $Y_d = (y_d, \dot{y}_d^{(1)}, \cdots, y_d^{(n)})$ is bounded, i.e., $\| Y_d \| \leqslant k_0, k_0 > 0$, it is easy to show that

$$\| \mu \| = \| (\mu - Y_d) + Y_d \| \leqslant \| e \| + k_0 \tag{15}$$

and

$$\| d(\phi^{-1}(\mu)) \| \leqslant k_1 \| \mu \|, \quad \left| \frac{1}{a(\phi^{-1}(\mu))} \right| < k_2, \quad k_1, k_2 > 0, \tag{16}$$

then

$$\| \delta u \| \leqslant k_3 \varepsilon (\| e \| + k_0), \quad k_3 = k_1 \cdot k_2. \tag{17}$$

Choose a Lyapunov function

$$v = e^{\mathrm{T}} P e, \tag{18}$$

where $P > 0$ is chosen such that $A^{\mathrm{T}}P + PA = -I$. Differentiating V along the trajectories of the closed loop system of (2), we have

$$\begin{aligned} \dot{V} &= - \| e \|^2 + 2e^{\mathrm{T}} P \cdot (\delta \cdot u) \\ &\leqslant - \| e \|^2 + k_4 \| e \| \varepsilon (\| e \| + k_0) \\ &= - (1 - k_4 \varepsilon) \| e \|^2 + k_0 k_4 \| e \| \varepsilon \\ &= - (1 - k_4 \varepsilon)(\| e \|^2 - 2k_0 k_5 \| e \| \varepsilon + k_0^2 k_5^2 \varepsilon^2) + (1 - k_4 \varepsilon) k_0^2 k_5^2 \varepsilon^2 \\ &= - (1 - k_4 \varepsilon) \cdot \| e \| \cdot (\| e \| - 2k_0 k_5 \varepsilon). \end{aligned} \tag{19}$$

When $\| e \| > 2k_0 k_5 \varepsilon$, there exists

$$\dot{V} \leqslant - (1 - k_4 \varepsilon) \cdot \| e \| \cdot (\| e \| - 2k_0 k_5 \varepsilon) \leqslant - (1 - k_4 \varepsilon) \| e \|^2. \tag{20}$$

Which implies that, $e(t)$ will exponentially converge to a ball of order ε.

Further more, if we choose $\varepsilon_0 = \max\{k_0, \varepsilon\}$, from (20), it is easy to find that as long as $\| e \| > 2k_5 \varepsilon_0^2$, the convergence property also exits. This means $e(t)$ will exponentially con-

verge to a ball of order ε_0^2.

The above discussion can be concluded as:

Thorem 3.2 Suppose system (2) is controllable at x^o, and there exist a neural network which satisfies (7) for a given $\varepsilon > 0$ around x^o. Then, if the desired trajectory Y_d is bounded, the coordinate change $\mu(x)$ of (11) and the control law (13)(14) result in local exponential stable approximate tracking around x^o. The tracking error bound will be of order ε. Furthermore, if $\| Y_d \| \leqslant \varepsilon$, the tracking error bound will be order of ε^2.

4 A Ball and Beam Example

To demonstrate the feasibility of the proposed method, a typical simulation experiment, the ball and beam system as shown in Fig. 1, is engaged in.

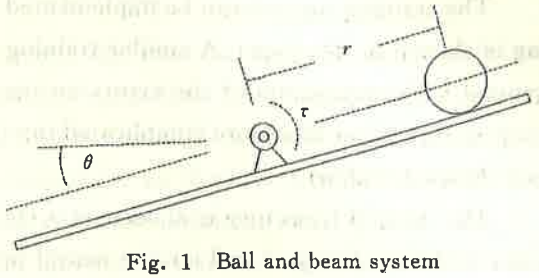

Fig. 1 Ball and beam system

The dynamic equations of this system are given by

$$(\frac{J_b}{R^2} + M)\ddot{r} + MG\sin\theta - Mr\dot{\theta}^2 = 0,$$

$$(Mr^2 + J + J_b)\ddot{\theta} + 2Mr\dot{r}\dot{\theta} + MGr\cos\theta = \tau, \tag{21}$$

where τ is the torque applied to the beam, J is the moment of inertia of the beam, M and J_b represent the mass and moment of inertia of the ball respectively, R is the radius of the ball, G stands for the acceleration of gravity.

Define

$$x = (x_1, x_2, x_3, x_4)^{\mathrm{T}} = (r, \dot{r}, \theta, \dot{\theta})^{\mathrm{T}},$$

$$B = \frac{M}{J_b/R^2 + M}, \quad u = \frac{\tau - 2Mr\dot{r}\dot{\theta} - MGr\cos\theta}{Mr^2 + J + J_b}$$

the system can be written in state-space form as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ B(x_1 x_4^2 - G\sin x_3) \\ x_4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u.$$

As discussed in [7], this system does not satisfy the involutivity condition (4). So it can not be linearized by feedback linearization. We use a feedforward neural network to learn to approximately integrate the partial differential equation (5) of this system in a specific domain, so that an approximate nonlinear transformation can be constructed in this domain and the system can then be linearized in an approximate sense.

The engaged feedforward neural network consists of four inputs (x_1, x_2, x_3, x_4), one output (u) and one single hidden layer containing 15 nodes with the activation functions shown as below

$$G(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

The training process is designed as:

i) Move the beam manually or with a simple controller (the linear controller for example) near the desired trajectory to collect the input-output data, then train the network with these data so that the neural network constructed controller can operate the system (though with a very bad performance).

ii) Retrain the network with the newly collected input-output data of the system operated by the constructed controller with the trained network.

iii) Repeat step ii) till a satisfied performance is approved.

The training process can be implemented on-line or off-line. For simplicity, off-line training is chosen in this paper. A similar training algorithm is described in the appendix, where a general backpropagation of the errors on the derivatives of only one degree is shown. What used in this paper is a more complicated one (on the derivatives up to third degree) which is not shown for short.

The desired trajectory is chosen as $y_d(t) = 3\cos(\pi t/5)$. Fig. 2 and Fig. 3 show that after some training of step i) and ii), the neural network constructed feedback has gained some ability to operate the beam-ball system with a poor performance. With the training going on the tracking errors decrease. Fig. 4, Fig. 5 and Fig. 6 show that after about 8000 times training, the tracking error has been tightly bounded which means the system has been approximately linearized. Fig. 7 describes the seperate learning errors of (8) at this time, where the bold, dashed and doted lines represent δ_1, δ_2 and δ_3 separately.

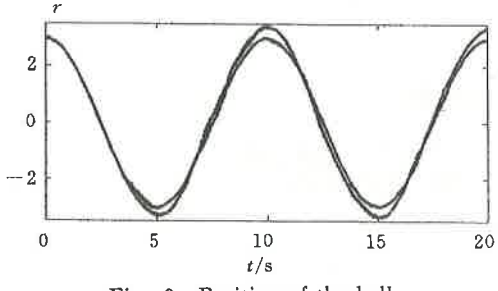

Fig. 2 Position of the ball

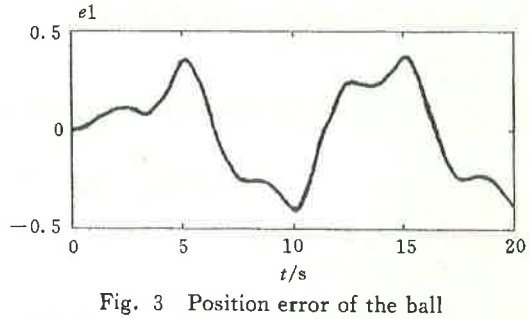

Fig. 3 Position error of the ball

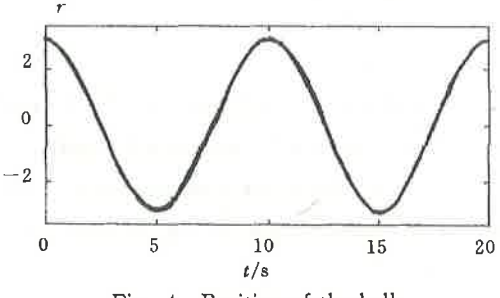

Fig. 4 Position of the ball

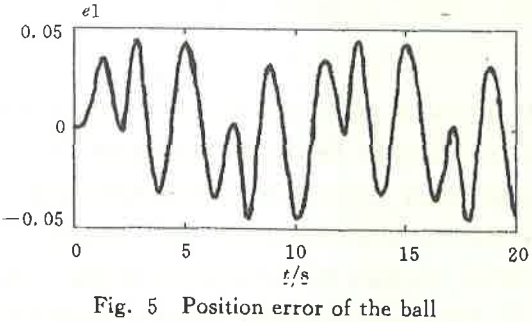

Fig. 5 Position error of the ball

5 Conclusion and Discussion

Neural network approximation of a function and its derivatives offers a very promising tool to introduce neural network theory to control system design[12]. This paper is a preliminary application of this method to nonlinear system linearization. Being different from other

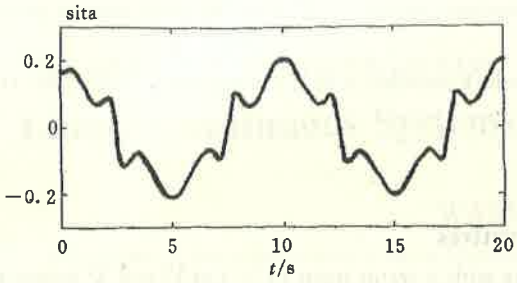

Fig. 6 Angle of the beam

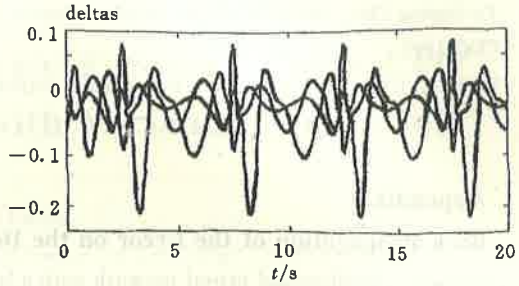

Fig. 7 Learning errors

neural network linearization methods[13], this proposed algorithm has less constraints and can be much more widely used, especially to nonlinearizable systems. Furthermore, it is easy to implement this control, because only off-line learning is needed to train the neural network. The feasibility shown in simulation experiment encourages us to pursuit further work along these directions: 1) extend this result to nonlinear systems containing zero-dynamics; 2) introduce sliding control to compensate the effect of uncertainties in the successive differentiations of the neural network constructed output (12). This idea has been discussed for robust input-output feedback linearization of nonlinear systems with "unmatching uncertainties"[14]; 3) look for other neural network models and training algorithms which will be more efficient for linearization feedback design.

References

1 Byrnes, C. I. and Isidori, A.. Local stabilization of minimum-phase nonlinear systems. Systems and Control Letters, 1988, 11: 9—17

2 Isidori, A.. Nonlinear Control Systems: An Introduction, 2nd ed. New York: Springer-Verlag, 1989

3 Krener, A. J.. Approximate linearization by state feedback and coordinate change. Systems and Control Letters, 1984, 5: 181—185

4 Hauser, J.. Nonlinear control via uniform system approximation. Systems and Control Letters, 1991, 17: 145—154

5 Krener, A. J. Karahan, S., Hubbard, M. and Frezza, R.. Higher order linear approximations to nonlinear control systems. Proc. of 26th Conference on Decision and Control, 1987, 519—523

6 Hauser, J. and Murry, R. M.. Nonlinear controllers for non-integrable systems: the acrobot example. Proc. of 1990 Automatic Control Conference, San Diego, CA, 1990

7 Hauser, J., Sastry, S. and Kokotoviic, P.. Nonlinear control via approximation input-output linearization: the ball and beam example. IEEE Trans. Automat. Contr., 1992, AC-37(3): 392—398

8 Hunt, L. R. and Janos Turi. A new algorithm for constructing approximate transformations for nonlinear systems. IEEE Trans. Automat. Contr., 1993, AC-38(10): 1553—1556

9 Nam, K. et al.. Some numerical aspects of approximate linearization of single input non-linear systems. Int. J. Control, 1993, 57(2): 463—472

10 Hornik, K., Stinchcombe, M. and White, H.. Multilayer feedforward networks are universal approximators. Neural Networks, 1989, 2(3): 359—366

11 Hornik, K., Stinchcombe, M. and White, H.. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. Neural Networks, 1990, 3(5): 551—560

12 Piere Cardaliaguet and Guillaume Euvrard. Approximation of a function and its derivative with a neural network. Neural Networks, 1992, 5(2): 207—220

13 Fu-Chuang Chen and Khalil, H. K.. Adaptive Control of Nonlinear System Using Neural Networks. Proc. of the 29th CDC, 1990

14 Slotine, J. J. E. and Hedrick, J. K.. Robust input-output feedback linearization. Int. J. Control, 1993, 57(5): 1133—1139

Appendix

Back Propagation of the Error on the Derivatives

Consider a feedforward neural network with n layers with a vector input (x_i^1). Let V_j^k and X_j^k denote the potential and the state of neuron j from layer k, having activation function $G()$, y_{jq}^k denote the partial derivative of hte state X_j^k with respect to the q'th input X_j^1:

$$V_i^{k+1} = \sum_j w_{ij}^k x_j^k, \quad x_i^{k+1} = G(V_i^{k+1}),$$

$$y_{jq}^k = \frac{\partial x_j^k}{\partial x_q^1}, \quad Y_{iq}^{k+1} = \sum_j w_{ij}^k y_{jq}^k, \quad y_{iq}^{k+1} = G'(V_i^{k+1}) Y_{iq}^{k+1},$$

and consider the cost function on the n-th layer $E(x_i^n, y_{iq}^n)$ as well as the equalities:

the gradient of E with respect to w_{ij}^k is proposed as

$$\frac{\partial E}{\partial x_i^k} = \sum_j w_{ji}^k \frac{\partial E}{\partial x_j^{k+1} G'(V_j^{k+1})} \sum_{j,q} \frac{\partial E}{\partial y_{jq}^{k+1}} G''(V_j^{k+1}) Y_{jq}^{k+1},$$

$$\frac{\partial E}{\partial y_{iq}^k} = \sum_j w_{ji}^k \frac{\partial E}{\partial y_{jq}^{k+1}} G'(V_j^{k+1}),$$

$$\frac{\partial E}{\partial w_{ij}^{k+1}} = \frac{\partial E}{\partial x_i^k} G'(V_i^k) x_j^{k+1} + \sum_q \frac{\partial E}{\partial y_{iq}^k} [G''(V_i^k) Y_{iq}^k x \backslash + k + 1_j + G'(V_i^k) y_{jq}^{k+1}.$$

It is easy to find that the standard back-propagation formulas can be deduced if we replace y_{iq}^k and Y_{iq}^k by zero, and $\frac{\partial E}{\partial x_i^k} G'(V_i^k)$ by δ_i^k.

基于神经网络的非线性系统近似线性化

裴海龙 周其节

(华南理工大学自动控制工程系·广州,510641)

摘要：神经网络具有同时逼近某一函数及其高阶导数的功能,这一结果为神经网络在非线性系统中的应用提供了可行的工具．本文提出了一种利用网络近似功能的非线性系统的近似线性化方法,无论系统是否满足可积条件,神经网络都可实现其对合条件的近似积分,从而构造满足系统近似线性化的反馈控制．对球-杆系统的仿真结果显示了这种方法的有效性．

关键词：神经网络；近似线性化

本文作者简介

裴海龙 1965 年生．分别于 1986 年和 1989 年在西北工业大学获工学学士和工学硕士学位．1993 年于华南理工大学获工学博士学位．现为华南理工大学自动控制工程系副教授．研究兴趣为非线性控制,神经网络控制和机器人控制．

周其节 1930 年生．1955 年哈尔滨工业大学研究生毕业．现为华南理工大学自动控制系教授,博士生导师．主要研究领域为非线性系统理论,自适应控制,变结构控制,小波分析,机器人控制等．