Reinforcement Learning NN-based Controller Design for Aero-engine

Hong-Mei ZHANG[1,a], Yu-Ling LIANG[2,b,*] and Guang-Yan XU[3,c]

[1,3]School of Aerospace Engineering, Shenyang Aerospace University, Shenyang, 110136, China

[2]School of Automation, Shenyang Aerospace University, Shenyang, 110136, China

[a]zhang.hongmei@163.com,[b]18804038702@163.com,[c]guangyan_xu@163.com

*Corresponding author

Keywords: Aero-engine, Neural Networks, Reinforcement Learning, Cost function.

Abstract. A reinforcement learning NN-based controller for aero-engine is presented, which is suitable for tracking problems at a steady-state operating point. The proposed controller design has two entities:an action NN that is designed to produce optimal signal for aero-engine and a critic NN that approximates certain strategic utility function which evaluates the performance of the action network . In accordance with a turbofan engine, the control system is designed at the selected operating point. The simulation results show that the perfect performance of the controller，and the controller has strong anti-interference ability.

Introduction

With the development of aero-engine technology and the improvement of its performance, the control system is simple to complex，one of the main factors to be considered in designing the control systems for modern and advance aero-engines is the factor of uncertainty [1]. An intelligent control system is such one which forms the knowledge about unknown characteristics of control object and its operational environment in the process of self-learning, and the information obtained can used then in the process of automatic decision making so that the performance of control processes is being constantly improved. Because Neural Networks (NNs) can approximate any nonlinear function with arbitrary precision, and have the very strong learning ability and adaptive ability for complex uncertainty problems, they are widely used in system modeling and controller designing [2,3]. A self-learning PID controller based on NNs was developed [4]. The parameters of PID controller are tuned online with the NNs to make the output of the controlled aero-engine follow the desired output of a reference model. A Genetic Algorithms was applied to learn the connection weights of a NN, which construct a NN controller [5]. The controller has mapped capabilities of neural networks, fast global convergence and reinforcement learning properties of genetic algorithms. A back-stepping control strategy based on neural networks was presented in view of nonlinearity and uncertainty of aero-engine [6], but the control law is limited by the form of model. Reinforcement learning in discrete time is proposed for the state-feedback controller to solve the tracking problem of steady-state operating points in this paper.

Reinforcement Learning NN-based Controller Design

In this paper, with the information of the system states and dynamics, an approximate optimization is accomplished using the proposed controller [7,8], which is to use the optimal control to approach cost function of the system. The control system approximates the Bellman equation

$$J(x(k)) = \min_{u(k)} \{J(x(k+1)) + U(x(k), x(k+1))\} . \quad (1)$$

Where $x(k)$ is the state, $u(k)$ is the control at time step k , while the strategic utility function $J(x(k))$

represents the minimum cost or performance measure associated with going from k to final step N, $U(x(k),x(k+1))$ is the utility function denoting the cost incurred in going from k to $k+1$ using control $u(k)$, and $J(x(k+1))$ is the minimum cost or performance measure associated in going from state $k+1$ to the final step N.

In this paper, considering the following affine nonlinear system in the form as

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ &\vdots \\ x_n(k+1) &= f(x(k)) + h(x(k))u(k) + d(k) \\ y(k) &= x_1(k) \end{aligned} \tag{2}$$

where the state vector $x(k) = [x_1(k), x_2(k), \cdots, x_n(k)]^T \in R^{nm}$ at time instant k with each $x_i(k) \in R^m, i = 1, \cdots, n$. $f(x(k)) \in R^m$ is a smooth unknown nonlinear vector, $h(x(k)) \in R^{m \times m}$ is a diagonal matrix of unknown nonlinear vector, $u(k) \in R^m$ is the control input vector. $y(k) \in R^m$ is the output vector, and $d(k) \in R^m$ is the unknown but bounded disturbance vector, whose bound is assumed to be a known constant such that $\|d(k)\| \le d_m$.

Assumption 1: Without loss of generality, let the matrix $h(x(k)) \in R^{m \times m}$ be a positive definite diagonal matrix for each $x(k) \in R^{nm}$ with $h_{\min} \in R^+$ and $h_{\max} \in R^+$ representing the minimum and maximum eigenvalues of the matrix $h(x(k))$, respectively, such that $0 < h_{\min} \le h_{\max}$.

In the controller architecture, the NN has two layers as shown in Fig.1. The output of the NNs can be given by

$$Y = W^T \phi(V^T x). \tag{3}$$

Where V and W are the hidden layer and output layer weights, respectively. The number of hidden layer nodes is denoted as N_2, ϕ is the activation function vector of the hidden layer.

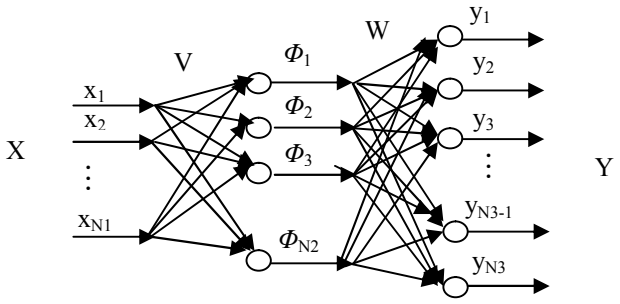


Fig. 1: Two-layer NN structure

The block diagram of reinforcement learning NN-based controller is shown in Fig. 2 where an action NN is providing the control signal to the aero-engine while a critic NN approximates the long-term cost function.

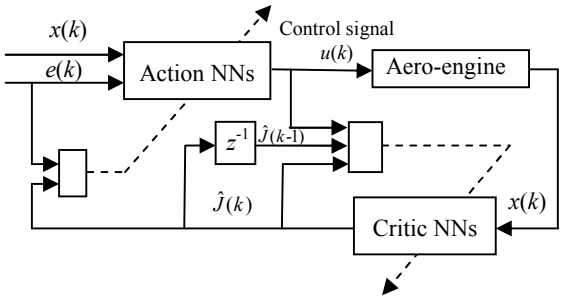


Fig. 2: Reinforcement learning NN-based controller structure

Assumption 2: The desired trajectory of the system state $x_d(k) = [x_{1d}(k), x_{2d}(k), \cdots, x_{nd}(k)]^T \in R^{nm}$

is arbitrarily selected, which satisfies $x_{id}(k+1) = x_{(i+1)d}(k)$, $i = 1, \cdots, n-1$, $y_d(k)$ is a known smooth bounded function.

Note that, from Assumption 2 and 1, one can derive that $x_{id}(k) = y_d(k+i-1), i = 1, \cdots, n-1$. Meanwhile, to introduce optimality into our design, the long-term cost function is defined as

$$J(k) = J(x(k), u) = \sum_{i=t_0}^{\infty} \gamma^i r(k+i) = \sum_{i=t_0}^{\infty} \gamma^i \left[q(x(k+i)) + u^T(k+i) F u(k+i) \right] . \tag{4}$$

where $J(k)$ stands for $J(x(k), u)$ for simplicity, u is a control policy, F is a positive definite matrix, and $q(x(k))$ is a positive semidefinite function of the states, while $\gamma (0 \le \gamma \le 1)$ is the discount factor for the infinite-horizon problem. As observed from (4), the long-term cost is defined in terms of the discounted sum of the immediate cost or Lagrangian $r(k)$, which is given by

$$\begin{aligned} r(k) &= q(x(k)) + u^T(k) R u(k) = (x_1(k) - x_{1d}(k))^T R (x_1(k) - x_{1d}(k)) + u^T(k) F u(k) \\ &= (y(k) - y_d(k))^T R (y(k) - y_d(k)) + u^T(k) F u(k) \end{aligned} . \tag{5}$$

where R is a positive semidefinite matrix. The immediate cost function $r(k)$ can be viewed as a system performance index for the current step. This standard quadratic cost function is based on the output tracking error $e(k) = y(k) - y_d(k)$.

Our objective is to design a reinforcement learning NN-based controller for the system (2) so that the long-term cost function $J(k)$ is minimized and an optimal control input can be generated.

Action Network Design

The tracking error at instant k is defined as

$$e_i(k) = x_i(k) - x_{id}(k) = x_i(k) - y_d(k+i-1), \ i = 1, \cdots, n. \tag{6}$$

Then, the future value of the tracking error using system dynamics from (2) can be rewritten as

$$e_n(k+1) = f(x(k)) + h(x(k)) u(k) + d(k) - y_d(k+n) . \tag{7}$$

A desired control signal can be given by

$$u_d(k) = h^{-1}(x(k))(-f(x(k)) + y_d(k+n) + l_1 e(k)) . \tag{8}$$

where $l_1 \in R^{m \times m}$ is a design matrix selected such that the tracking error $e_n(k)$ converges to zero.

From (8) and considering Assumption 2, the desired control signal can be approximated as

$$u_d(k) = \omega_a^T \phi_a(\nu_a^T s(k)) + \varepsilon_a(s(k)) = \omega_a^T \phi_a(s(k)) + \varepsilon_a(s(k)) . \tag{9}$$

where $s(k) = [x^T(k), y_d^T(k), y_d^T(k+n)]^T \in R^{(n+2)m}$ is the action NN input vector. The action NNs consists of two layers, and $\omega_a \in R^{n_a \times m}$ and $\nu_a \in R^{(n+2)m \times n_a}$ denote the desired weights of the output and hidden layers respectively, with $\varepsilon_a(s(k))$ being the action NNs approximation error and n_a being the number of the neurons in the hidden layer. Since ν_a is fixed, for simplicity purposes, the hidden layer activation function vector $\phi_a(\nu_a^T s(k)) \in R^{n_2}$ is denoted as $\phi_a(s(k))$.

Considering the fact that the target weights of the action NNs are unknown, the actual NNs weights have to be trained online, and its actual output can be expressed as

$$u(k) = \hat{\omega}_a^T(k) \phi_a(\nu_a^T s(k)) = \hat{\omega}_a^T(k) \phi_a(s(k)) . \tag{10}$$

where $\hat{\omega}_a(k) \in R^{n_a \times m}$ is the actual weight matrix of the output layer at instant k.

Using the action NNs output as the control signal and substituting (9) and (10) into (7), it yields

$$\begin{aligned} e_n(k+1) &= f(x(k)) + h(x(k))u(k) + d(k) - y_d(k+n) = l_1 e_n(k) + h(x(k))(u(k) - u_d(k)) + d(k) \\ &= l_1 e_n(k) + h(x(k)) \times (\tilde{\omega}_a{}^T \phi_a(s(k)) - \varepsilon_a(s(k)) + d(k)) = l_1 e_n(k) + h(x(k))\varsigma_a(k) + d_a(k) \end{aligned} \tag{11}$$

where

$$\tilde{\omega}_a(k) = \hat{\omega}_a(k) - \omega_a . \tag{12}$$

$$\varsigma_a(k) = \tilde{\omega}_a{}^T(k)\phi_a(s(k)) . \tag{13}$$

$$d_a(k) = -h(x(k))\varepsilon_a(s(k)) + d(k) . \tag{14}$$

In the meantime, we can notice that

$$e_i(k) = x_i(k) - x_{id}(k) = e_n(k-n+i), \quad i = 1, \cdots, n. \tag{15}$$

Critic Network Design

As stated earlier, an optimal controller should be able to stabilize the closed-loop system while minimizing the cost function at the same time. In this paper, a critic NNs are employed to approximate the target long-term cost function $J(k)$. Since $J(k)$ is unavailable at the kth time instant in an online learning framework, the critic NNs are tuned online in order to ensure that its output converges close to $J(k)$. The prediction error for the critic is defined as

$$e_c(k) = \gamma \hat{J}(k) - \hat{J}(k-1) + r(k) . \tag{16}$$

$$\hat{J}(k) = \hat{\omega}_c^T(k)\phi_c(v_c^T x(k)) = \hat{\omega}_c^T(k)\phi_c(x(k)) . \tag{17}$$

where $\hat{J}(k) \in R$ representing the critic NNs output which is an approximation of $J(k)$. In our design, the critic NN is also two-layer NNs, while $\hat{\omega}_c(k) \in R^{n_c \times 1}$ and $v_c \in R^{nm \times n_c}$ represent its actual weight matrix of the output and hidden layers, respectively. The term n_c denotes the number of the neurons in the hidden layer. The system states $x(k) \in R^n$ are selected as the critic network input. The activation function vector of the hidden layer $\phi_c(v_c^T x(k)) \in R^{n_c}$ is denoted as $\phi_c(x(k))$ for simplicity. Provided that enough number of the neurons is in the hidden layer, the optimal long-term cost function $J^*(k)$ can be approximated by the critic network with arbitrarily small approximation error $\varepsilon_c(k)$ as

$$J^*(k) = \omega_c^T \phi_c(v_c^T x(k)) + \varepsilon_c(x(k)) = \omega_c^T \phi_c(x(k)) + \varepsilon_c(x(k)) . \tag{18}$$

similarly, the critic network NNs weight estimation error can be defined as

$$\tilde{\omega}_c(k) = \hat{\omega}_c(k) - \omega_c . \tag{19}$$

where the approximation error is given by

$$\varsigma_c(k) = \tilde{\omega}_c^T(k)\phi_c(x(k)) . \tag{20}$$

$$\begin{aligned} e_c(k) &= \gamma \hat{J}(k) - \hat{J}(k-1) + r(k) \\ &= \gamma \varsigma_c(k) + \gamma J^*(k) - \varsigma_c(k-1) - J^*(k-1) + r(k) - \gamma \varepsilon_c(k) + \varepsilon_c(k-1) \end{aligned} . \tag{21}$$

Weight Update for the Critic Network

Following the discussion from the above section, the objective function to be minimized by the critic networks can be defined as a quadratic function of tracking errors as

$$E_c(k) = \frac{1}{2} e_c^T(k) e_c(k) = \frac{1}{2} e_c^2(k) . \tag{22}$$

Using a standard gradient-based adaptation method, the weight updating algorithm for the critic network is given by

$$\hat{\omega}_c(k+1) = \hat{\omega}_c(k) + \Delta\hat{\omega}_c(k) \ . \tag{23}$$

$$\Delta\hat{\omega}_c(k) = \alpha_c \left[-\frac{\partial E_c(k)}{\partial \hat{\omega}_c(k)} \right] = -\alpha_c \frac{\partial E_c(k)}{\partial e_c(k)} \frac{\partial e_c(k)}{\partial \hat{J}(k)} \frac{\partial \hat{J}(k)}{\partial \hat{\omega}_c(k)} = -\alpha_c \gamma \phi_c(x(k)) e_c(k), \alpha_c \in R \ . \tag{24}$$

$$\hat{\omega}_c(k+1) = \hat{\omega}_c(k) - \alpha_c \gamma \phi_c(x(k)) \times \left(\gamma \hat{J}(k) - \hat{J}(k-1) + r(k) \right) . \tag{25}$$

Weight Update for the Action Network

The basis for adapting the action NNs is to track the desired trajectory and to lower the cost function. Therefore, the error for the action NNs can be formed by using the functional estimation error $\varsigma_a(k)$ and the error between the nominal desired long-term cost function $J_d(k) \in R$ and the critic signal $\hat{J}(k)$. Now, the cost function vector is $\overline{J}(k) = [\hat{J}(k) \ \ \hat{J}(k) \ \ \cdots \ \ \hat{J}(k)]^T \in R^{m\times 1}$. Let

$$e_a(k) = \sqrt{h(x(k))}\varsigma_a(k) + \left(\sqrt{h(x(k))} \right)^{-1} (\overline{J}(k) - J_d(k)) = \sqrt{h(x(k))}\varsigma_a(k) + \left(\sqrt{h(x(k))} \right)^{-1} \overline{J}(k) . \tag{26}$$

Where $\varsigma_a(k)$ is defined in (13). The desired long-term cost function $J_d(k)$ is nominally defined and is considered to be zero, which means as low as possible.

Hence, the weights of the action NNs $\hat{\omega}_a(k)$ are tuned to minimize the error

$$E_a(k) = \frac{1}{2} e_a^T(k) e_a(k) \ . \tag{27}$$

$$\begin{aligned} \Delta\hat{\omega}_a(k) &= -\alpha_a \frac{\partial E_a(k)}{\partial \hat{\omega}_a(k)} = -\alpha_a \frac{\partial E_a(k)}{\partial e_a(k)} \frac{\partial e_a(k)}{\partial \varsigma_a(k)} \frac{\partial \varsigma_a(k)}{\partial \hat{\omega}_a(k)} = -\alpha_a \phi_a(s(k)) \left(h(x(k))\varsigma_a(k) + \overline{J}(k) \right)^T \\ &= -\alpha_a \phi_a(s(k)) \times \left(e_n(k+1) - l_1 e_n(k) - d_a(k) + \overline{J}(k) \right)^T \end{aligned} \ . \tag{28}$$

where $\alpha_a \in R^+$ is the adaptation gain of the action NNs. However, since $d_a(k)$ is unavailable, in the ideal case, we take it as zero and obtain the weight updating algorithm for the action NNs as

$$\hat{\omega}_a(k+1) = \hat{\omega}_a(k) - \alpha_a \phi_a(s(k)) \times (e_n(k+1) - l_1 e_n(k) + \overline{J}(k))^T \ . \tag{29}$$

Simulation and Analysis of Result

A certain turbofan engine is considered to be our control object. The system states includes the low-pressure rotor speed n_1 and the high-pressure rotor speed n_2. The control variables are the fuel flow m_f of the combustion chamber and the nozzle area A_8. The output variables are the a low-pressure rotor speed n_1 and turbine nozzle pressure ratio p_{36}. The state variables model of small perturbation about a steady-state operating point of an engine is established by fitting method [9] as follows:

$$\begin{aligned} \dot{x}_p &= A_p x_p + B_p u_p \\ y_p &= C_p x_p + D_p u_p \end{aligned} \ . \tag{30}$$

Where $x = (\Delta n_1 \ \ \Delta n_2)^T, u = (\Delta m_f \ \ \Delta A_8)^T, y = (\Delta n_1 \ \ \Delta P_{36})^T$. Here, Δ stands for deviations from the corresponding equilibrium position. It is extremely disadvantageous for design calculation because

the model varies greatly in magnitude. In this paper, we use the normalization method to deal the state variable model. The normalized matrixes are defined as follows:

$$S_x = \begin{bmatrix} (n_1)_0 & \\ & (n_2)_0 \end{bmatrix}, S_u = \begin{bmatrix} (m_f)_0 & \\ & (A_8)_0 \end{bmatrix}, \quad S_y = \begin{bmatrix} (n_1)_0 & \\ & (P_{36})_0 \end{bmatrix}. \tag{31}$$

where diagonal matrix elements are stable working points data of the state. The normalize variables x, u, y can be defined as $x = S_x^{-1} x_p$, $u = S_u^{-1} u_p$, $y = S_y^{-1} u_p$.

Thus we have the small perturbation state normalized variable model as follows：

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{32}$$

where, $A = S_x^{-1} A_p S_x$, $B = S_x^{-1} B_p S_u$, $C = S_y^{-1} C_p S_x$, $D = S_y^{-1} D_p S_u$. (33)

the engine variables normalized state space model at a steady-state operating point as follows:

$$\begin{aligned} \dot{x} &= \begin{bmatrix} -2.250 & -0.722 \\ -0.870 & -1.423 \end{bmatrix} x + \begin{bmatrix} 0.780 & 0.602 \\ 0.532 & 0.416 \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 \\ -3.743 & 6.826 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ -0.088 & 0.5038 \end{bmatrix} u \end{aligned} \tag{34}$$

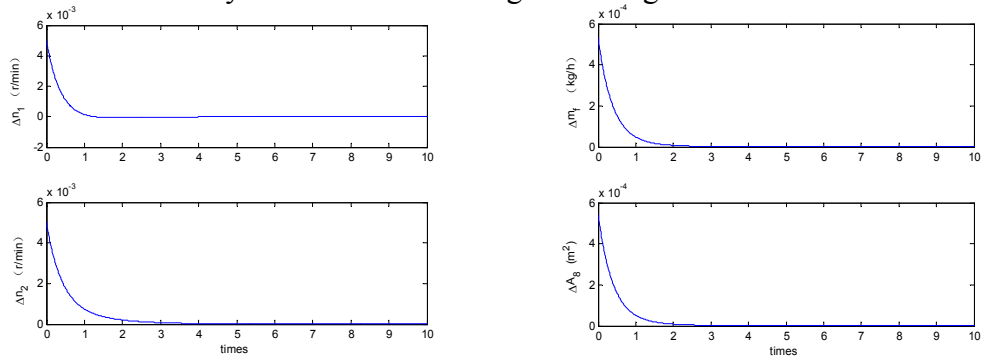
A reinforcement learning NN-based controller is designed for the system such that the following holds: 1) the state $x(k)$ follows a desired trajectory $x_d(k) = [x_{1d}(k), \ldots, x_{nd}(k)]^T \in R^{nm}$; 2) the long-term cost function (4) is minimized so that an optimal (or near optimal) control input can be generated. When the engine is working at a non-design point, our objective is to make the rotor speeds tracking the rotor speeds of the steady-state operating point, in other words, the rotor speeds variation relative to the steady-state operating point should converge to zero [10].

In the simulation, the initial state $x_{pre}(k) = [0.005, 0.005]^T$, the desired state $x_d(k) = [0,0]^T$, and the step=0.01, the parameter s of the system as follows is given as follows:

Table 1 The simulation parameters

Parameter	R	F	γ	l_1	α_a
Value	Diag{0.1,0.1}	[0.1 0;0 0.1]	0.1	0.1	0.1
Parameter	α_c	n_a	n_c	d_m	
Value	0.01	20	20	0.0001	

The proposed controller in this paper has been applied in the control of the aero-engine model and simulation results of the system are shown in Fig. 3 and Fig. 4.

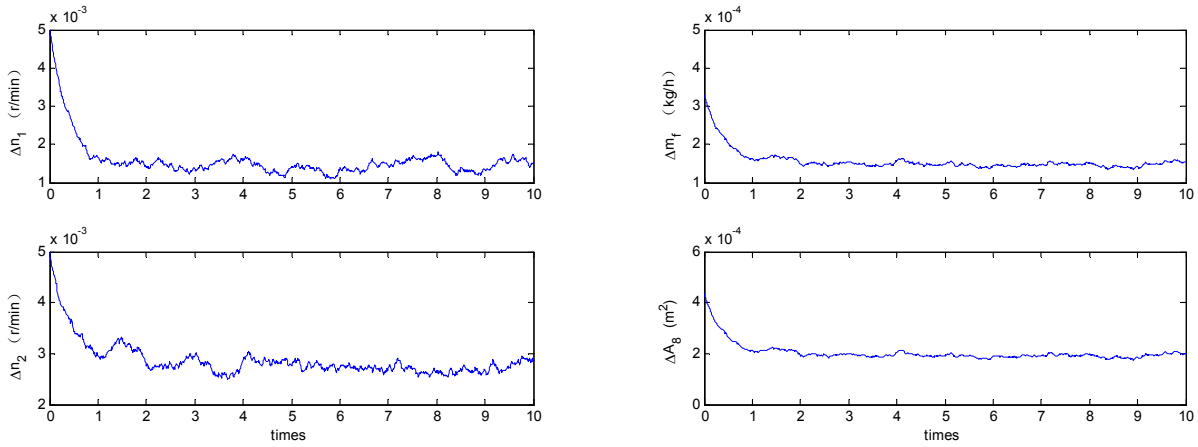


(a) The responses of the engine model (b) The inputs of the engine model

Fig. 3 Performance of the reinforcement learning NN-based controller

Fig. 3 shows the responses and the inputs of the engine model intuitively. The system features the rapid responses, the excellent tracking performance and the smooth transition process.

To test the stability of the system, a small bounded disturbance is input to the system. In Fig. 4 show the response of the changed system.



(a) The responses of the system (b) The inputs of the system

Fig. 4 Performance of the reinforcement learning NN-based controller with a small bounded disturbance

Fig. 4 shows that the responses of the rotor speeds with a small bounded disturbance haven't changed significantly. The simulation shows that the proposed controller has strong anti-interference ability.

A reinforcement learning NN-based controller has been applied for aero-engine to deliver a desired performance under bounded disturbance. Simulation results show that the proposed controller has excellent characteristics and also can overcome the exterior disturbance.

References

[1] J.G. Sun, V. Vasilyev, B. Hyasov, Advanced Multivariable Control Systems of Aeroengines. Beihang University Press, Beijing, 2005, pp. 285-288.

[2] K.S. Narendra, Neural Networks for Control Theory and Practice, Proc. of the IEEE, 84(1996) 1385-1406

[3] K. Funahashi, On the approximate realization of continuous mappings by neural networks, Neur. Networks, 2(1989) 183-192.

[4] H. Yao, Y. Yuan, L.L. Bao, J.G. Sun, Self-learning PID control based on neural networks for aero-engines, Journal of Propulsion Technology, 28(2007) 313-316.

[5] J.A. Fang, S.H. Shao, A Neural Network Controller Using Genetic Algorithms Learning, Control and Decision, 8(1993) 208-212.

[6] M.X. Pan, J.Q. Huang, S. Yin, Backing control strategy for aero-engine using neural networks. Journal of Aerospace Power, 24(2009) 2344-2348.

[7] Q.M. Yang, S. Jagannathan, Reinforcement Learning Controller Design For Affine Nonlinear Discrete-Time Systems using Online Approximates, IEEE Transactions on Cybernetics, 4(2012) 377-390.

[8] P. He, S. Jagannathan, Reinforcement Learning-based Output Feedback Control of Nonlinear Systems with Input Constraints, Proceeding of the 2004 American Control Conference Boston, Massachusetts June 30 -July 2(2004).

[9] Z.P. Feng, J.G. Sun, J.Q. Huang, and H.W. Cai, A new method for establishing a state variable model of aeroengine. Journal of Aerospace Power, 13(1998) 435-438.

[10] Y.Q. Guo, H. Zhang, Y.B. Yang, The Design of Aero-engine Optimal Controller Suitable for All Flight Envelope with Neural Network Approximator, Journal of aerospace power 15(2000) 331-333.