Online Synchronous Policy Iteration Based on Concurrent Learning to Solve Continuous-time Optimal Control Problem

Haitao Wang, Dongbin Zhao, and Chengdong Li

***Abstract*—In this paper, a novel online Synchronous Policy Iteration algorithm based on Concurrent Learning (CLSPI) is presented to solve the continuous-time optimal control problem. We design this algorithm based on actor-critic architecture. Original Synchronous Policy Iteration (SPI) algorithm just updates parameters of actor-critic simultaneously, which is rather different to the way standard policy iteration updates. In the scheme of SPI, only current estimation error is utilized to update weights while previous information can also contribute to weights update. Concurrent learning is a new parameters estimation method which combines previous information and current estimation error. CLSPI utilizes concurrent learning to train SPI, in order to improve the learning performance. Finally, two comparison experiments including linear and nonlinear systems are presented to demonstrate that CLSPI can obtain the optimal control policy with faster convergence rate, which shows that CLSPI is more efficient to solve continuous-time optimal control problems.**

I. Introduction

THOUGH control theory has developed over one hundred years, optimal control is still a great challenge especially for nonlinear problems. Under the architecture of traditional control theory, solving optimal control problem means obtaining solutions of Hamilton-Jacobi-Bellman (HJB) equation[1, 2]. Nevertheless, under most conditions the HJB equation is intractable to be solved by traditional analytical methods because of the property of nonlinear partial differential equation (PDE)[1]. Only special problems may have solutions, like linear time-invariant system because in those cases HJB equation will become Algebraic Riccati Equation (ARE). Due to such reasons, many algorithms have been proposed to solve the PDE[3].

Recently, one method called reinforcement learning[4, 5] (RL) has been applied to optimal control. RL algorithm mainly contains two classes of iterative algorithms-Value Iteration (VI) and Policy Iteration (PI), which are all computational intelligence algorithms. Though VI also has been applied to some optimal control problems, we will focus on the PI algorithm in this paper. Policy iteration represents some algorithms that are based on policy evaluation and policy improvement. Generally, policy iteration will begin to evaluate the value function of initial admissible policy and then utilize the evaluation to improve current control policy, hence avoiding the difficulty of directly solving nonlinear HJB equation. The two steps iteration process will repeat again and again until the control policy converges. Actor-critic structure is a popular architect for implementing PI and it has been widely used in model free optimal control[6], adaptive cruise control[7]. In this frame [8], the actor performs actions by interacting with the environment and the critic takes evaluations of actions. Then feedback information from critic is delivered to the actor, resulting in the improvement of the next output of actor. In actor-critic architecture, two neural networks (NN) are designed to approximate respectively value function and control policy. For continuous-time problem, critic NN is trained to solve nonlinear Lyapunov equation, while actor NN will output control policy. Some offline methods have been given by[3]. Also[9, 10] developed online actor-critic PI method for continuous time optimal problem which updates sequentially parameters of actor and critic. In [11, 12], synchronous policy iteration algorithm was presented to solve continuous time optimal control problem. This method differs from traditional actor-critic PI in that it updates parameters of actor or critic NN simultaneously.

Concurrent Learning (CL) is a method for estimating weights which combines previous information and current information. In ordinary gradient descent method only current estimation error is utilized to update parameters. However, we can see that previous estimation errors might also contribute to current update if those errors can be collected. Based on such view, Chowdhary and Eric[13] propose a method called concurrent learning to substitute traditional gradient descent algorithm. It has been applied to model reference adaptive control[14], optimal control[15], reinforcement learning[16, 17]. Here we hope not only current estimation error but also historical errors can be applied to current update in order to improve the performance of algorithm. In this paper, we propose CLSPI, a new synchronous policy iteration algorithm based on concurrent learning which shows better performance than original SPI in [12]. Also two rules of storing historical data will be detailed in the sequel.

This paper is organized as follows. Section II introduces preliminaries of optimal control problems of nonlinear continuous-time systems. Concurrent learning is provided in sections III and IV gives a view of the proposed CLSPI. Section V presents two experiments to show the efficiency of CLSPI. Finally, discussion and conclusion are drawn.

Haitao Wang and Dongbin Zhao are with The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, P. R. China (email: wanghaitao8118@163.com, dongbin.zhao@ia.ac.cn). Chengdong Li is with School of Information and Electrical Engineering, Shandong Jianzhu University, Jinan 250101, P. R. China (email: chengdong.li@foxmail.com).

This work is supported by National Natural Science Foundation (NNSF) of China under Grants No.61273136, No.61034002, and No. 61473176.

II. Problem Statement

In this paper we study continuous time affine nonlinear system as follows:

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

with $x(0) = x_0$, the state $x(t) \in \mathbb{R}^n$, the control input $u(t)$, functional matrix $g(x(t)) \in \mathbb{R}^{n\times m}$ and the nonlinear function $f(x(t)) \in \mathbb{R}^n$. It can be assumed that $f(x) + g(x)u(x)$ is Lipchitz continuous on a compact set $\Omega \subset \mathbb{R}^n$ containing the origin. Furthermore, this system can be stabilized on Ω. That is to say, there must exist a continuous control input $u(t) \in \mathrm{U}$ which can make the system asymptotically stable on Ω. We assume system dynamics $f(x), g(x)$ are known and bounded over Ω. The value function for system (1) is denoted by

$$V(x(t)) = \int_t^\infty r(x(s), u(s))\,\mathrm{d}s \quad (s \ge t) \tag{2}$$

where $r(x,u) = Q(x) + u^T Ru$, $Q(x)$ is positive define and R is symmetric positive definite matrices.

Definition 1(Admissible control policy): A control policy π is said to be admissible with respect to (1) on Ω, denoted by $\pi \in \psi(\Omega)$, if $\pi(0) = 0$, $\pi(x)$ is continuous and can drive the system to equilibrium point from any initial state with finite $V(x_0) \in \Omega$.

For admissible control policy π, its cost function can be written as

$$V^\pi(x_0) = \int_0^\infty r(x(s), u(s))\,\mathrm{d}s\,. \tag{3}$$

So, the nonlinear Lyapunov equation is given by

$$r(x,u) + \left(\frac{\partial V^\pi}{\partial x}\right)^T (f(x) + g(x)\pi(x)) = 0\,. \tag{4}$$

Now, the optimal control will be presented. Provided with dynamic system (1), the set of admissible control policies and value function (2), locate the optimal control policy which can make the cost function (2) minimum. Hence, the optimal value function can be defined by

$$V^*(x_0) = \min \int_0^\infty r(x(s), u(s))\,\mathrm{d}s\,. \tag{5}$$

Meanwhile, Hamilton function of (1) is denoted as

$$H(x,u,V_x) = r(x,u) + V_x^T (f(x) + g(x)u(t))\,, \tag{6}$$

where $V_x = \partial V^\pi / \partial x$. In order to get optimal value function, HJB function should be satisfied. That is

$$0 = \min_{\pi\in\psi(\Omega)} \left(H(x,\pi,V_x)\right). \tag{7}$$

According to optimal control theory, by minimizing Hamilton function solving stationary condition $\partial H / \partial u = 0$ can lead to the optimal control

$$u^*(x) = -\frac{1}{2} R^{-1} g^T(x) V_x^*(x)\,. \tag{8}$$

Substituting this optimal control policy into (6), we can get the HJB equation and the necessary and sufficient condition for (8) will be

$$\begin{aligned} &Q(x) + V_x^{*T} f(x) - \frac{1}{4} V_x^{*T} g(x) R^{-1} g^T(x) V_x^* = 0 \\ &V^*(0) = 0 \end{aligned}\,. \tag{9}$$

If $f(x(t)) + g(x(t))u = Ax + Bu$ with constant A and B, that means it's a linear problem. Then HJB equation can be written as well-known ARE. For nonlinear system in order to find optimal control policy, (9) should be solved firstly. Nevertheless, under most conditions, locating the solution of HJB equation is not easy. So later, CLSPI will be proposed to obtain the optimal control policy for HJB equation.

III. Concurrent Learning

When it comes to parameter estimation, we often use regression to build the relationship between the input $x(t) \in \mathbb{R}^n$ and the output y. Here, the unknown system dynamics are assumed to be linearly parameterized with regression vectors $\Phi\left(x(t)\right) \in \mathbb{R}^n$.

A. Linear Regression

Assuming regression vectors $\Phi\left(x(\mathrm{t})\right)$ are known, bounded and continuously differentiable. Meanwhile the constant ideal weight vector is denoted by $W^* \in \mathbb{R}^n$. So the unknown system model can be given as:

$$y(t) = W^{*T}\Phi(x(t))\,. \tag{10}$$

If we do the regression online, $W(\mathrm{t}) \in \mathbb{R}^n$ can be employed as the online estimate of W^*. With regression vectors known, online estimate of $y(t)$ will be $\hat{y} = W(t)^T \Phi(x(t))$. So the estimation error $\varepsilon(t)$ can be written as

$$\varepsilon(t) = \hat{y} - y(t) = \tilde{W}^T(t)\Phi(x(t))\,, \tag{11}$$

where $\tilde{W}(t)$ is the parameter error which can be defined as $\tilde{W}(t) = W(t) - W^*$.

In order to estimate $W(t)$ some update laws should be employed to guarantee the convergence of $\varepsilon \to 0$. One of the most popular methods is gradient descent. Given the cost function $J = \varepsilon(t)^T \varepsilon(t)$, we have to minimize J to get the estimation of W. Here, we would employ gradient method as the update law. So, the gradient of J can be denoted as $\nabla J = 2 * \Phi(x(t))\varepsilon(t)$. The online update law of $W(t)$ will be

$$\dot{W}(t) = -a\nabla J\,. \tag{12}$$

In this equation, a represents the update rate which might be constant variable or change with the update process. However, this gradient method will not necessarily guarantee $W(t) \to W*$ which requires that the regression vectors $\Phi(x(t))$ are persistently exciting.

Persistently exciting condition[18]: A function $\Phi(t)$ is persistently exciting if there exist $T, \delta_1, \delta_2 > 0$ such that

$$\delta_1 I_n \le \int_t^{t+T} \Phi(\tau)\Phi(\tau)^T d\tau \le \delta_2 I_n \tag{13}$$

holds for all $t \ge 0$ where I_n is identity matrix of order n.

B. Learning with Data Recording

The update law in linear regression described in III.A only considers the current information. If previous information about the model is reused, it can improve the effects of online estimation. In[13], Girish and Eric propose a new update law of online estimating $W(t)$, called concurrent learning which combines previous and current information about x and ε. Its update law can be written as

$$\dot{W}(t) = -\Gamma\Phi(x(t))\varepsilon(t) - \Gamma\sum_{i=1}^{q}\Phi(x_i)\varepsilon_i\,, \qquad (14)$$

where $i \in \{1,2...,q\}$ denotes the index of stored data points, $\Phi(x_i)$ is the regression data at point x_i and ε_i is the output estimation error at point x_i.

IV. Synchronous Policy Iteration Based On CL

In this section, we will introduce a novel synchronous policy iteration based on concurrent learning. This algorithm applies the thought existing in concurrent learning which records previous information for current parameters update to synchronous policy iteration algorithm.

A. Synchronous PI with two NN Function Approximators

As an iteratively updating algorithm standard policy iteration can be implemented with actor-critic structure. This structure consists of two neural networks which are applied to approximating value function (9) or policy solution (7). In every iterative loop, value function will be updated first, which is policy evaluation step. Then according to the value policy improvement step will be processed in order to obtain better control policy. In standard PI, parameters of actor-critic NN shall update sequentially. By contrast, in[11] SPI algorithm has been proposed to update the parameters of actor and critic neural networks simultaneously. Also, extra theoretical proof has been presented to guarantee the convergence of SPI algorithm to the optimal control policy. The architecture of SPI is depicted in Fig. 1.

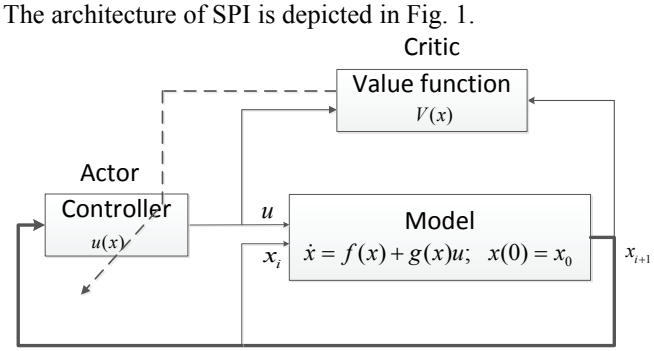

Fig. 1. The architecture of actor/critic

1) Value Function Approximation

Similar to linear regression, we assume there exists W such that value function $V(x)$ can be approximated by NN

$$V(x) = W^T\phi(x) + \varepsilon\,, \qquad (15)$$

where $\phi(x): \mathbb{R}^n \to \mathbb{R}^N$ is the activation functions vector, N is the number of neurons in hidden layer, and ε is the error of approximation. By differentiating (15) we get

$$\frac{\partial V}{\partial x} = \left(\frac{\partial\phi(x)}{\partial x}\right)^T W + \frac{\partial\varepsilon}{\partial x} = \nabla\phi^T W + \nabla\varepsilon\,. \qquad (16)$$

According to[19], as neurons of hidden layer $N \to \infty$, the approximation accuracy can be arbitrary, that is $\varepsilon \to 0$. With NN approximation, Hamilton equation (6) is substituted by

$$H(x,u,W) = r(x,u) + W^T\nabla\phi(f(x) + g(x)u(t)) = \varepsilon_H\,. \qquad (17)$$

Then, once the system dynamics satisfies Lipchitz condition, the approximation error ε_H will converge to zero. Next, the critic and actor updating process of SPI algorithm will be detailed.

2) Critic Neural Network Updating

Assuming the weight of critic NN, W_c which is unknown, provides the optimal approximator for (17). So the estimation of value function will be

$$\hat{V}(x) = \hat{W}_c^T\phi(x)\,, \qquad (18)$$

where $\hat{W}_c$ is the estimation of current weights in critic NN with N neurons in hidden layer. $\phi(x)$ is the activation functions vector . Hence estimation of Lyapuov equation should be

$$H(x,u,\hat{W}_c) = r(x,u) + \hat{W}_c^T\nabla\phi(f(x) + g(x)u(t)) = e_c\,. \qquad (19)$$

Therefore, cost error can be written as

$$E_c = e_c^T e_c / 2\,, \qquad (20)$$

Then minimizing (20) will lead to $\hat{W}_c \to W_c$. One of most popular method to achieve the optimal $\hat{W}_c$ is gradient descent:

$$\dot{\hat{W}}_c = -a_1\frac{\partial E_c}{\partial \hat{W}_c} = -a_1\frac{\sigma_0}{(\sigma_0^T\sigma_0 + 1)^2}[\sigma_0^T\hat{W}_c + Q(x) + u^T Ru]\,, \qquad (21)$$

where $\sigma_0 = \nabla\phi(f + gu)$ and a_1 is the learning rate. This update law will not necessarily assure the convergence of $\hat{W}_c$ because $\phi(x)$ shall satisfy PE condition. Therefore, when implementing SPI algorithm, proper noise should be added to numerical simulations to make sure PE condition is satisfied.

3) Actor neural network updating

The update strategy of critic has been presented in IV.A.2, based on which synchronous policy iteration involving actor NN update will be introduced here. It is known that PI just updates actor-critic networks sequentially. That means, when updating one, the other will hold constant. Comparing with standard PI, SPI can properly update two neural networks synchronously or simultaneously in real-time.

Like critic network, actor network can also be given by similar formation. So output of control solution is

$$u(x) = -R^{-1}g^T(x)\nabla\phi^T\hat{W}_a / 2 \qquad (22)$$

where $\hat{W}_a$ denote the online estimation of actor NN weights W_a and $\nabla\phi$ is the gradient of activation function $\phi(x)$. Then according to[12], by virtue of (21) the actor update law of SPI algorithm can be presented as

$$\dot{\hat{W}}_a = -a_2[(k_2\hat{W}_a - k_1\bar{\sigma}_0^T\hat{W}_c) - \frac{1}{4}F\hat{W}_a M^T\hat{W}_c] \qquad (23)$$

with $\sigma_0 = \nabla\phi(f+gu)$, $\bar{\sigma}_0 = \sigma_0 / (\sigma_0^T \sigma_0 + 1)$, $M = \bar{\sigma}_0^2 / \sigma_0$ and $F = \nabla\phi \, \mathrm{g}(x) R^{-1} \mathrm{g}^T(x) \nabla\phi^T$. In (23), a_2 is the learning rate and k_1, k_2 are constant parameters which can be set to be identity matrix in practical experiments.

B. Parameters Updating Based on CL

In section III, we introduce the thought of concurrent learning which combines previous information and current information to improve the effects of parameters estimation. Though the analysis is based on linear regression, we hope that concurrent learning can be applied to SPI algorithm. The actor-critic structure in section IV.B only uses current estimation error in (20). If we store historical data in stack, then they can be accumulated to update the parameters of two neural networks. So here (20) can be rewritten as

$$E_c = \frac{1}{2} e_{c0}^T e_{c0} + \frac{1}{2} \sum_{i=1}^{q} e_{ci}^T e_{ci} \tag{24}$$

where $i \in \{1, 2..., q\}$ denotes the index of historical data point, e_{c0} denotes current estimation error and e_{ci} represents the historical estimation error at point x_i . So the update law of (23) can be denoted by

$$\dot{\hat{W}}_c = -a_1 \frac{\sigma_0}{(\sigma_0^T \sigma_0 + 1)^2} e_{c0} - a_1 \sum_{i=1}^{q} \frac{\sigma_i}{(\sigma_i^T \sigma_i + 1)^2} e_{ci} . \tag{25}$$

Actually, in order to simplify the updating process of actor-critic neural network, only update law of critic NN will utilize the old information as in (25). And the update law of actor network will remain unchanged as in (23). So the CLSPI algorithm can be schemed in **Algorithm 1**.

Algorithm 1 SPI algorithm based on Concurrent Learning

1: Initialize the parameters like learning rate a_1, a_2 , weights of NN, admissible initial control policy π_0 and so on.
2: **Repeat**
3: Compute the output of actor NN u according to (22) based on current state x_i
4: Calculate the estimation error of critic NN with current $\hat{W}_1$ according to (19)
5: **if** current point should be stored in stack
6: Store the error of critic NN in historical stack
7: **end if**
8: Update weights of critic NN by (25)
9: Update weights of actor NN by (23)
10: Update current state x_{i+1}
11: **Until** $\hat{W}_a$ and $\hat{W}_c$ converge
12: **Return** $\hat{W}_a$ and $\hat{W}_c$

In CLSPI algorithm, some rules should be given to decide whether current point should be stored as in line 5 of **Algorithm 1**. Here two methods of recording historical data will be presented. One just records q points before current data. The other is that new added data will make historical stack vectors full rank.

The first method is the simplest one to implement the new algorithm we proposed. The size of stack should be allocated in advance. Once new data comes, it is added to stack and the oldest point is removed from the stack. As for the second rule, historical stack matrices shall satisfy **full rank condition**. In stimulation experiments, both rules have been realized. Generally, once the size of stack is much larger than n two rules show little difference when being applied to SPI. It should be pointed out that though the size of stack can be set arbitrarily, excessive large size of historical data can increase the computational complexity and result in some problems of convergence of actor-critic NN in practical simulations.

Full rank condition: Let $S = [\bar{\sigma}_1, \cdots, \bar{\sigma}_q]$ be the stack, when every new data is added to stack it should satisfy that $rank(S) = n$. And n is the dimension of $\bar{\sigma}_1$.

V. Experiments

In this section, two experiments will be given to support what we have discussed about CLSPI. One is linear system, and the other is nonlinear system. In both experiments, we can see that the optimal control policy will be achieved and show better performance than SPI algorithm in[12].

A. Linear System

Similar to[11], we choose linear system with quadratic cost function

$$\dot{x} = \begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix} x + \begin{bmatrix} 1 \\ -3 \end{bmatrix} u$$

where Q and R are identity matrix. Actually, in linear system solving HJB equation will be equivalent to acquiring the solution of the algebraic Riccati equation. Accordingly, we choose $\phi(x) = [x_1^2 \ x_1 x_2 \ x_2^2]^T$ as the activation functions vector of critic network. Here, we will present the comparison between SPI and CLSPI algorithm. The learning rates $a_1 = a_2 = 60$ are set in both experiments. $k_1 = k_2 = 5I$ is given in contrast experiments. Also, when CLSPI algorithm is implemented, the number of historical points q is set to be 25. As is presented, when $t < 100$s proper signal noise is mixed into control input in order to make PE condition is satisfied.

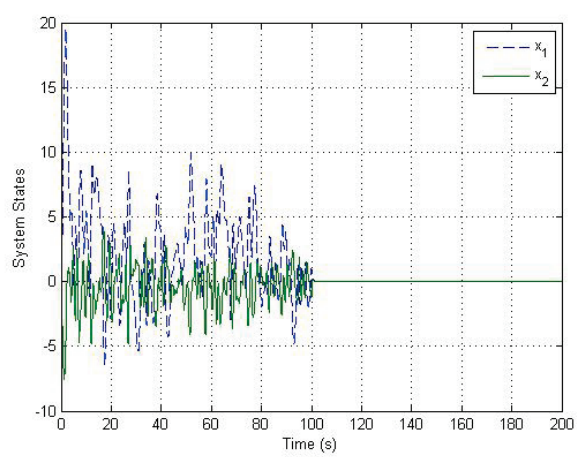

Fig. 2. Evolution of the system states

By solving ARE, the optimal parameters of critic NN can be easily obtained. That is $W_c^* = [0.3199 - 0.1162 \ 0.1292]^T$. Fig. 2 shows the evolution of system states when SPI algorithm is applied to the linear problem. Fig. 3 presents the

trajectories of critic network parameters in CLSPI which converge to the optimal $W_c(t_f) = [0.3162 \; -0.1182 \; 0.1285]$ at 8s. By contrast, in SPI algorithm the weights just converge to the optimal $W_c(t_f) = [0.3155 \; -0.1172 \; 0.1289]$ at about 48s under same settings as in Fig. 4. From the comparison experiment, we can see CLSPI algorithm can converge to optimal control policy much more quickly without affecting the property of convergence in linear case.

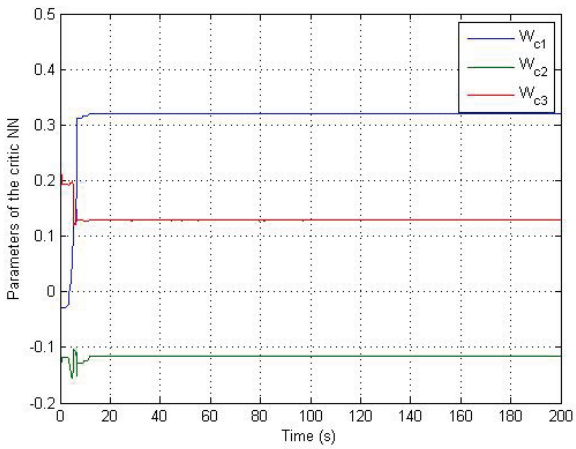

Fig. 3. Evolution of critic parameters with CLSPI

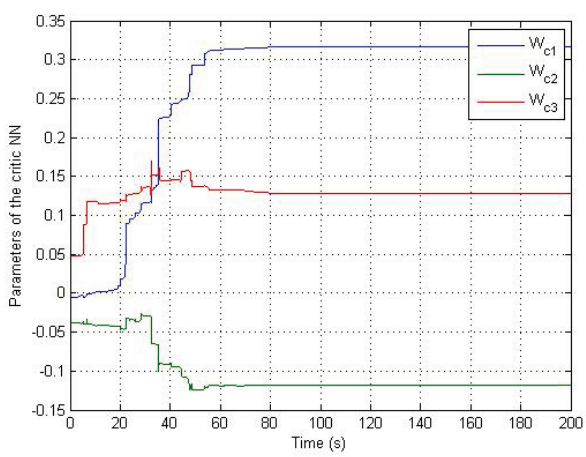

Fig. 4. Evolution of critic parameters with SPI

B. Nonlinear System

A more complex nonlinear will be analyzed in this part. Following[12], we choose affine nonlinear system with a quadratic value function

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

wher $f(x) = \begin{bmatrix} -x_1 + x_2 & -0.5x_1 - 0.5x_2(1-(\cos(2x_1)+2)^2) \end{bmatrix}^T$ $g(x) = \begin{bmatrix} 0 & \cos(2x_1)+2 \end{bmatrix}^T$ .Meanwhile, $R = 1$ and $Q = I$. And the optimal cost function can be $V^* = 0.5x_1^2 + x_2^2$. So, we can get the optimal policy as $u^* = -[x_1 \; x_2]\begin{bmatrix} 0 & \cos(2x_1)+2 \end{bmatrix}^T$. Then the activation function is set to be $\phi(x) = [x_1^2 \; x_1x_2 \; x_2^2]^T$. And the parameters of critic neural network can be represented by $\hat{W}_c = \begin{bmatrix} W_{c1} \; W_{c2} \; W_{c3} \end{bmatrix}^T$.

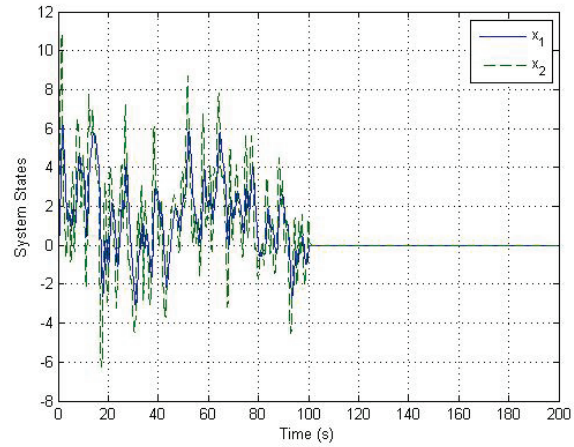

Fig. 5. Evolution of the system states with CLSPI

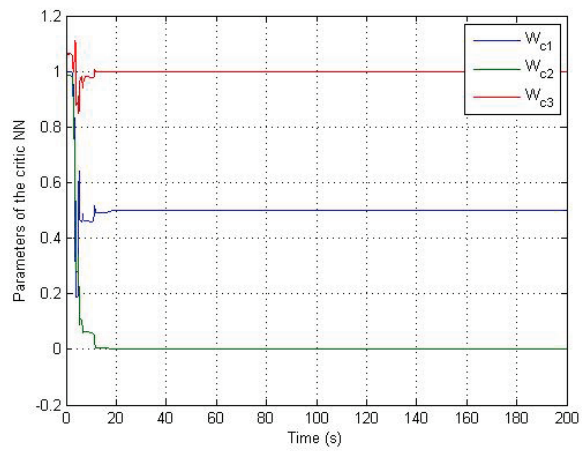

Fig. 6. Evolution of critic parameters with CLSPI

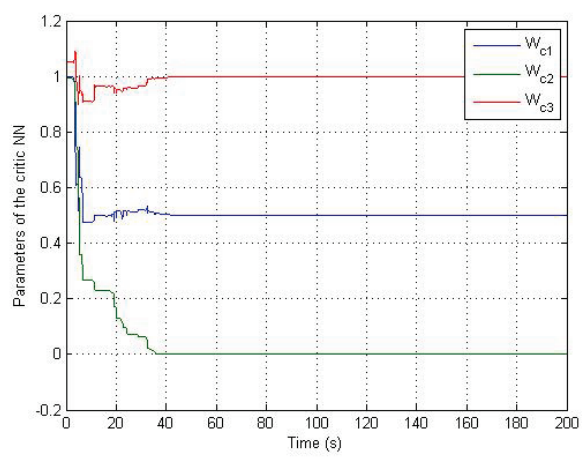

Fig. 7. Evolution of critic parameters with SPI

Here, comparison experiments will also be presented between SPI and CLSPI algorithm. In both experiments the parameters should be $q = 30$, $a_1 = a_2 = 50$, $k_1 = k_2 = 10I$. And q indicates the size of historical data points in stack. Also when $t < 100$s , proper signal noise is mixed into control input in order to identify the model of value function.

The evolution of system states and the trajectories of critic-actor parameters with CLSPI algorithm are presented in Fig. 5, Fig. 6 and Fig. 8. By contrast, Fig. 7 and Fig. 9 display

the trajectories of parameters with SPI algorithm under same conditions. We can see that in CLSPI algorithm, the weights of critic-actor neural network can asymptotically converge to the optimal weight $W_c(t_f) = [0.5000 - 0.00002\ 1.0002]^T$ at about 14s. Nevertheless, SPI algorithm only converges to $W_c(t_f) = [0.5000 - 0.00002\ 1.0002]^T$ at about 40s under the same circumstance. Therefore, CLSPI can show better convergence speed than SPI algorithm.

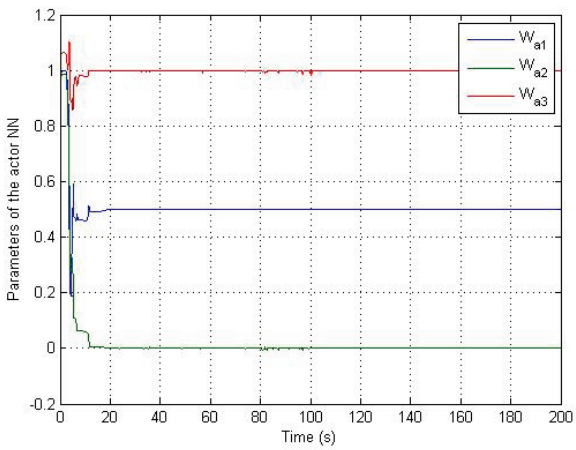

Fig. 8. Evolution of actor parameters with CLSPI

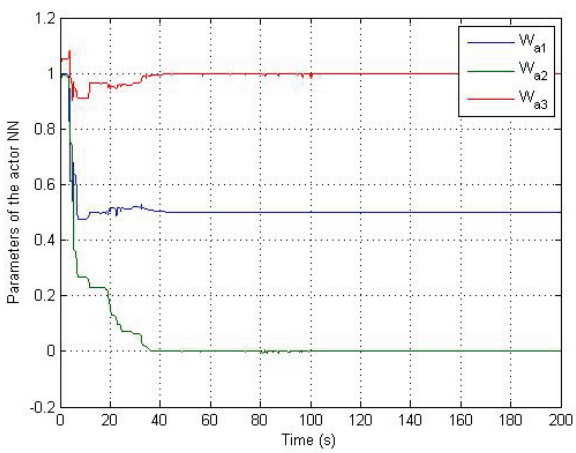

Fig. 9. Evolution of actor parameters with SPI

VI. Discussion And Conclusion

In this paper, CLSPI is introduced and applied to a linear system and a nonlinear system. It is inspired by SPI algorithm which updates the parameters of weights synchronously using current estimation error. Concurrent learning is a new method which combines historical estimation errors and current errors together. We give two different methods of storing historical data. When CL is applied SPI we can see from Fig. 2 and Fig. 5 that the convergence of the control system is not affected. From the two comparison experiments, it is clear that CLSPI converges to optimal control policy much faster than SPI in both linear system and nonlinear system. So CLSPI can show greater advantage over SPI algorithm. Meanwhile, the size of stack represents the computational complexity of CLSPI. If it is too large, great computational sources are required and in practical experiment the convergence property of parameters can be disturbed.

Finally, though CLSPI has been applied to two different systems and presents promising results, theoretical analysis of this algorithm should be looked into furthermore. Also, in this paper only parameters of critic network utilize historical data points; we hope actor network can also be updated by concurrent learning.

References

[1] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. New Jersey: John Wiley & Sons, 2012.

[2] X. Yang, D. Liu, and Q. Wei, "Neuro-optimal control of unknown nonaffine nonlinear systems with saturating actuators," in *Intelligent Control and Automation Science*, 2013, pp. 574-579.

[3] M. Abu-Khalaf and F. L. Lewis, "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach," *Automatica,* vol. 41(5), pp. 779-791, 2005.

[4] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*: MIT Press, 1998.

[5] D. Zhao and Y. Zhu, "MEC---A near-optimal online reinforcement learning algorithm for continuous deterministic systems," *IEEE Trans. Neural Networks and Learning Systems,* DOI.10.1109/ TNNLS.2014.2371046, 2014.

[6] D. Zhao, Z. Xia, and D. Wang, "Model-free optimal control for affine nonlinear systems with convergence analysis," *Automation Science and Engineering, IEEE Transactions on,* vol. PP(99), pp. 1-8, 2014.

[7] D. Zhao, B. Wang, and D. Liu, "A supervised actor–critic approach for adaptive cruise control," *Soft Computing,* vol. 17(11), pp. 2089-2099, 2013.

[8] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," *Handbook of intelligent control: Neural, fuzzy, and adaptive approaches,* vol. 15, pp. 493-525, 1992.

[9] D. Vrabie and F. L. Lewis, "Adaptive optimal control algorithm for continuous-time nonlinear systems based on policy iteration," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, 2008, pp. 73-79.

[10] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica,* vol. 45(2), pp. 477-484, 2009.

[11] K. Vamvoudakis and F. Lewis, "Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem," in *Int. Joint Conf. Neural Networks,* 2009, pp. 3180-3187.

[12] K. Vamvoudakis and F. Lewis, "Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica,* vol. 46(5), pp. 878-888, 2010.

[13] G. Chowdhary and E. Johnson, "Concurrent learning for convergence in adaptive control without persistency of excitation," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, 2010, pp. 3674-3679.

[14] G. Chowdhary and E. Johnson, "Theory and flight-test validation of a concurrent-learning adaptive controller," *Journal of Guidance, Control, and Dynamics,* vol. 34(2), pp. 592-607, 2011.

[15] X. Yang, D. Liu, and Q. Wei, "Near-optimal online control of uncertain nonlinear continuous-time systems based on concurrent learning," in *Int. Joint Conf. Neural Networks (IJCNN)*, 2014, pp. 231-238.

[16] S. Yasini, A. Karimpour, M. B. Naghibi Sistani, and H. Modares, "Online concurrent reinforcement learning algorithm to solve two-player zero-sum games for partially unknown nonlinear continuous-time systems," *International Journal of Adaptive Control and Signal Processing*, 2014.

[17] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems," *Automatica,* vol. 50(1), pp. 193-202, 2014.

[18] S. Sastry and M. Bodson, *Adaptive control: stability, convergence and robustness*: Courier Dover Publications, 2011.

[19] B. A. Finlayson, *The method of weighted residuals and variational principles* vol. 73: SIAM, 2013.