Actor-Critic Learning Control Based on ℓ_2-Regularized Temporal-Difference Prediction With Gradient Correction

Luntong Li, Dazi Li, Tianheng Song, and Xin Xu, *Senior Member, IEEE*

***Abstract*— Actor-critic based on the policy gradient (PG-based AC) methods have been widely studied to solve learning control problems. In order to increase the data efficiency of learning prediction in the critic of PG-based AC, studies on how to use recursive least-squares temporal difference (RLS-TD) algorithms for policy evaluation have been conducted in recent years. In such contexts, the critic RLS-TD evaluates an unknown mixed policy generated by a series of different actors, but not one fixed policy generated by the current actor. Therefore, this AC framework with RLS-TD critic cannot be proved to converge to the optimal fixed point of learning problem. To address the above problem, this paper proposes a new AC framework named critic-iteration PG (CIPG), which learns the state-value function of current policy in an on-policy way and performs gradient ascent in the direction of improving discounted total reward. During each iteration, CIPG keeps the policy parameters fixed and evaluates the resulting fixed policy by ℓ_2-regularized RLS-TD critic. Our convergence analysis extends previous convergence analysis of PG with function approximation to the case of RLS-TD critic. The simulation results demonstrate that the ℓ_2-regularization term in the critic of CIPG is undamped during the learning process, and CIPG has better learning efficiency and faster convergence rate than conventional AC learning control methods.**

***Index Terms*— ℓ_2-regularization, actor-critic (AC), policy gradient (PG), reinforcement learning (RL), value function approximation.**

I. Introduction

LEARNING control is one of the basic reinforcement learning (RL) tasks. The process of learning control is to obtain an optimal policy during the interactions with the environment to maximize some numerical values associated with a long-term objective for a system that is usually formulated as a Markov decision process (MDP) [1]. In learning control problems with continuous state and action space, actor-critic (AC) methods, including conventional policy-gradient (PG)-based algorithms [2]–[6] and approximate dynamic programming [7]–[14], have been widely studied. The basic idea in AC learning control is to evaluate the value function by temporal difference learning (TD learning) and update the policy parameters in the direction of the PG. In the actor, the PG can help to generate continuous actions in the space of policy parameters. In the critic, TD learning can help to reduce the variance of the estimation of expected returns and thus accelerate the learning [15]. These advantages promote AC methods into a class of high-efficiency RL methods. These methods have been applied to various fields such as robotics [16]–[19], power control [20]–[22], traffic control [23], [24], and finance [25].

In AC, all kinds of policy evaluation methods, including first-order TD learning [15], [26], residual gradient TD learning [27], TD learning with gradient correction [28], and least-squares TD (LSTD) [29]–[33], can be used in the critic. Some researchers have extended the above policy evaluation algorithms with ℓ_2-regularization to prevent solutions from overfitting. The basic idea of ℓ_2-regularization is to solve the ℓ_2 penalized least-squares problem, also known as ridge regression [35]–[37]. The ℓ_2-regularization problem has a simple closed-form solution. However, when ℓ_2-regularization is adopted in RLS-TD learning, the regularization term decreases during the learning process [38]. Thus, the parameters of value function approximation keep increasing, although the objective function could converge.

In a policy evaluation task, RLS-TD learning has a higher data efficiency and faster convergence rate than first-order TD learning. Thus, RLS-TD learning has been successfully applied in the critic learning of AC methods [4], [39]–[41]. Since the policy is changing with the actor, the resulting policy is termed time-varying policy, and the RLS-TD algorithms in the AC framework do not work in the same way as on-policy learning. Thus, it is hard to deal with convergence analysis for these algorithms. Bhatnagar *et al*. [15] also point out that it remains unclear how to incorporate LSTD methods into a context in which the policy keeps changing.

To solve the above problems in the AC methods, this paper proposes an AC framework named critic-iteration PG (CIPG) based on the RLS-TD critic. Two versions of RLS-TD algorithms, the ℓ_2-regularized recursive LSTD with gradient correction (termed RRC) and a fast ℓ_2-regularized recursive LSTD with gradient correction (termed FRRC) form [38], are used as CIPG's critic to make the effect of ℓ_2-regularization sustainable during the learning process. Convergence analysis

Manuscript received May 3, 2017; revised October 24, 2017 and January 25, 2018; accepted February 10, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61573052, Grant 61751311, and Grant U1564214, in part by the Beijing Natural Science Foundation under Grant 4182045, and in part by the Fundamental Research Funds for the Central Universities of China under Grant ZY1839. *(Corresponding author: Dazi Li.)*

L. Li, D. Li, and T. Song are with the Department of Automation, Beijing University of Chemical Technology, Beijing 100029, China (e-mail: lidz@mail.buct.edu.cn).

X. Xu is with the Institute of Unmanned Systems, College of Mechatronics and Automation, National University of Defense Technology, Changsha 410073, China (e-mail: xuxin_mail@263.net).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2808203

of the CIPG is also provided. In empirical experiments, the proposed algorithm is applied to solve three benchmarks in RL, namely, the 20-state Markov chain walk, the inverted pendulum, and the mountain car. The simulation results show that the critic in CIPG can sustain the effect of ℓ_2-regularization during the whole learning process. Furthermore, CIPG has a better learning efficiency and faster convergence rate than conventional AC learning control framework.

The rest of this paper is organized as follows. In Section II, the background of MDP and stochastic PG theory are presented. We present our CIPG algorithm in Section III and introduce RRC and FRRC as critics in our AC algorithm. In Section IV, the corresponding convergence analysis is given. The simulation results are shown in Section V. The conclusions are drawn in Section VI.

II. Problem Formulation

A. Background of the Markov Decision Process

The task of learning control is to find an optimal policy through interactions with the stochastic environment. In order to simplify the formulation, we model the problem as a discrete-time MDP that is defined as a tuple (S, A, P_0, R, γ), where S is the state space, A is the action space, P_0 is the state transition mapping: $P_0(s_{t+1}|s_t, a_t.) : S \times A \times S \to [0, 1]$, which specifies the conditional density of moving from state s_t to s_{t+1} when the action a_t is given; $R : S \to R$ is a reward function that gives the immediate reward r_t at each time step; and $\gamma \in [0, 1]$ is a discount factor that has been discussed how to choose in [42]. A stochastic policy $\pi_u : \mathcal{S} \to P(\mathcal{A})$ maps state space to a set of probability measures over A, where $u \in \mathbb{R}^n$ is the parameter vector and $\pi_u(a_t|s_t)$ denotes the conditional probability density of a_t in state s_t. The state-value function $V^\pi(s)$ related to a given policy π is defined as the expected discounted sum of rewards obtained during the interaction starting from s_0. Our aim is to find a policy π that maximizes the total discounted reward starting from state s_0

$$J(\pi) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t | \pi, s_0\right] = \sum_{\mathcal{S}} d^\pi(s) \sum_{\mathcal{A}} \pi_u(a_t|s_t) Q^{\pi_u}(s, a), \tag{1}$$

where $d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t P(s_t = s|s_0, \pi)$ denotes the discounted state distribution starting from s_0 and then following policy π. We assume that the PG $\partial \pi_u(a_t|s_t)/\partial u$ always exists, as does the given policy $\pi_u(a_t|s_t)$. The state-value function $V^\pi(s)$ and the action-value function $Q^\pi(s, a)$ are defined as

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t | s_0 = s, \pi\right] \tag{2}$$

and

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t | s_0 = s, a_0 = a, \pi\right] \tag{3}$$

respectively.

Given some basis functions $\phi(s) \in \mathbb{R}^n$, a linear function approximation (LFA) can be used to approximate the state-value function: $V^\pi(s) \approx \theta^T \phi(s)$, where $\theta \in \mathbb{R}^n$ is a linear parameter vector.

B. Stochastic Policy Gradient for Start-State Formation

In this section, the stochastic PG is derived for learning control problems with discounted total rewards (also known as start-state formulation) based on the average-reward formulation in the previous works [2], [4], [6], [15].

The gradient of the start-state formulation for parameterized policies is as follows [2]:

$$\nabla_u J(\pi_u(a|s)) = \sum_{\mathcal{S}} d^\pi(s) \sum_{\mathcal{A}} \nabla_u \pi_u(a|s) Q^\pi(s, a). \tag{4}$$

For any baseline function $b(s)$, which can be arbitrary function of state s, the gradient of the discount total rewards in (4) can be rewritten as

$$\nabla_u J(\pi_u(a|s)) = \sum_{\mathcal{S}} d^\pi(s) \times \sum_{\mathcal{A}} \nabla_u \pi_u(a|s)(Q^\pi(s, a) - b(s)). \tag{5}$$

The role of the baseline in (5) is to reduce the variance of the gradient estimate $\nabla_u J(\pi_u(a|s))$, so as to accelerate the rate of convergence of the PG algorithm [1]. If $\theta_1 \in \mathbb{R}^n$ and $\phi(s, a) \in \mathbb{R}^n$ are, respectively, the linear parameter vector and the basis function, then the action-value function in (5) is usually replaced by the approximation $\tilde{Q}^\pi(s, a) = \theta_1^T \phi(s, a)$, on the condition that 1) $\tilde{Q}^\pi(s, a)$ is compatible with the policy parameterization, i.e., $\nabla_{\theta_1} \tilde{Q}^\pi(s, a) = \nabla_u \log \pi_u(a|s)$, and 2) when the parameter update of θ_1 converges, $\tilde{Q}^\pi(s, a)$ satisfies

$$\sum_{\mathcal{S}} d^\pi(s) \sum_{\mathcal{A}} \pi_u(a|s)(Q^\pi(s, a) - \tilde{Q}^\pi(s, a)) \nabla_{\theta_1} \tilde{Q}^\pi(s, a) = 0. \tag{6}$$

According to [23, Lemma 2], $b(s) = V^\pi(s)$ is the optimal baseline that minimizes the variance in the action-value function estimator. By substituting $b(s) = V^\pi(s)$ into (5), we have

$$\nabla_u J(\pi_u(a|s)) = \sum_{\mathcal{S}} d^\pi(s) \sum_{\mathcal{A}} \nabla_u \pi_u(a|s) A^\pi(s, a). \tag{7}$$

where $A^\pi(s, a) \equiv Q^\pi(s, a) - V^\pi(s)$ is the advantage function.

Bhatnagar *et al.* [15] derived four different AC algorithms to update the policy parameters in the direction of the average reward gradient. We extend some of their results to the start-state formulation, and we have Lemmas 1 and 2, which are the analogs in [15, Lemmas 3 and 4].

Lemma 1: Under a given policy π, the TD error $\delta_t = r_{t+1} + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)$ is an unbiased estimate of the advantage function $A^\pi(s, a)$ under the start-state formulation

$$\mathbb{E}[\delta_t | s_t, a_t, \pi] = A^\pi(s_t, a_t). \tag{8}$$

Lemma 2: If TD recursion converges with probability 1, and $\delta_t^\pi = r_{t+1}^{t \to \infty} + \gamma \theta^{\pi T} \phi(s_{t+1}) - \theta^{\pi T} \phi(s_t)$ denotes a stationary estimate of the TD error, then the bias in the estimate of the gradient of the discount reward can be formulated as

$$\mathbb{E}\left[\delta_t^\pi \psi(s_t, a_t) | u\right] = \nabla_u J(\pi_u(a|s)) + \sum_{\mathcal{S}} d^\pi(s) \nabla_u \overline{V^\pi}(s). \tag{9}$$

where $\overline{V^\pi}(s)$ denotes the following quantity: $\sum_a \pi(a|s)[R(s, a) + \gamma \sum_{s_{t+1}} P_0(s_{t+1}|s_t, a_t) \theta^{\pi T} \phi(s_{t+1})] - J(\pi_u)$.

III. Actor-Critic Learning Control Algorithm Based on RLS-TD Learning

In the conventional AC framework, first-order algorithms, commonly TD and its variants, are used as the adaptive policy evaluation methods for the critic. In the AC framework, the policy is time-variant because of the policy improvement update of the actor. TD-like algorithms, as the stochastic gradient descent methods can gradually ignore the old samples generated under the previous policy and put more emphasis on the new samples under the current policy. Therefore, TD-like algorithms can evaluate the value function of the current policy without bias, in spite of having a low convergence rate. In the task of policy evaluation, RLS-TD algorithms have much higher sample efficiency than TD-like algorithms. However, when RLS-TD algorithms take the place of TD like algorithms and are used as the critic, this unbiased evaluation cannot be achieved. The reason lies in the fact that RLS-TD algorithms are batch algorithms but not adaptive algorithms, e.g., each single sample is equally effective to the critic update. Due to the time-variant policy mentioned earlier in AC, the samples in the trajectory are generated by different policies at each time step. Thus, the value function evaluated by RLS-TD algorithms cannot be an accurate evaluation either for the previous or the current policy. Essentially, RLS-TD does not work in the way of on-policy learning. To retain the high sample efficiency of RLS-TD and to make it work in the way of on-policy learning, a new AC framework is presented based on a sustainable ℓ_2-regularized RLS-based critic in the direction of the discounted reward gradient.

A. AC Framework Based on RLS-TD Learning

We use $\widehat{V}^{\pi}(s) = \theta^T \phi(s)$ to denote the approximation of the state-value function in state s and assume that the basis functions $\{\phi(s), s \in \mathcal{S}\}$ are linearly independent. Since the critic uses an LSTD learning method, which can exclude the step-size schedule, the step-size schedule is considered only for the actor. Let $\{\beta_t\}$ be the step-size sequence that satisfies

$$\sum_t \beta_t = \infty, \quad \sum_t \beta_t^2 < \infty. \tag{10}$$

Another crucial assumption is that the parameter of the actor should change more slowly than that of the critic. Thus, the step size sequence $\{\beta_t\}$ should be set as a sufficiently small positive value. Hence, the critic converges faster than the actor.

In Algorithm 1, the behavior policy u_m^b is used to generate samples at mth iteration. Since ℓ_2-regularized recursive LSTD with gradient correction (RRC) is a batch learning algorithm, the behavior policy u_m^b remains unchanged until the end of each mth iteration. All RRC parameters are reset to zero at the beginning of each iteration. In this way, RRC works in the way of on-policy learning at each iteration, because the policy in each iteration is invariant and thus the samples are generated by this identical policy. Therefore, RRC can evaluate this current policy without bias. This AC framework is a policy iteration-like method using PG as the actor. Furthermore, on-policy learning can improve the convergence performance of RRC and other RLS-TD algorithms in the AC framework. This is the main difference between the proposed AC framework and the previous ones. Peters and Schaal [4] proposed a series of natural gradient-based AC algorithms. One of their algorithms used an least-squares TD Q-learning (LSTDQ) method in a critic update. However, in this paper, they did not treat LSTDQ as a batch leaning algorithm in their settings. In this case, since the policy is changing, the LSTDQ evaluates the expectation of the sum of the policy at all time steps, but not the policy at the current time step. Therefore, convergence analysis is hard to achieve for their LSTDQ-based AC algorithm. Although a forgetting factor is used in their LSTDQ settings, to forget part of its accumulated sufficient statistics, we found that the forgetting rate will change the stationary distribution of state s under the policy π. This causes an extra approximate error in policy evaluation. Hence, it is more reasonable to use least squares methods as batch leaning algorithms in critic update of the AC algorithm. In Algorithm 1, we use RRC and fast RRC (FRRC) [38], two versions of LSTD learning with gradient correction [32], as our critic.

Algorithm 1 CIPG Based on RRC

Initialization:
Parameters:
$\pi_u(a|s)$: parameterized policy with initial parameters $u = u_0$;
$u_0^b = u_0$: behavior policy;
$\phi(s)$: value function basis function;
$\bar{\varepsilon} \in \mathbb{R}^+$: regularization parameter;
$\gamma \in [0, 1]$: discount factor.
1:**Repeat**
2: Reset critic parameters: $\rho_0 = 0$, $h_0 = 0$.
3: Set $P_0 = 1/\bar{\varepsilon} I$ and $G_0 = \bar{\varepsilon} I$;
4: **Repeat**
6: Initial state s_0 and step size β;
7: Draw action $a_t \sim \pi_{u_m^b}(a_t|s_t)$, observe next state $s_{t+1} \sim P(s_t, a_t, s_{t+1})$ and reward r_{t+1}.
8: Critic update: according to RRC or FRRC.
9: Compute TD error: $\delta_{t+1} = r_{t+1} - \gamma \theta_{t+1}^T \phi(s_{t+1}) - \theta_t^T \phi(s_t)$.
10: Actor update: $u_{t+1} = u_t + \beta_t \delta_t \psi_{s_t a_t}$.
11: **Until reach desired steps**.
11: Behavior policy update: $u_m^b = u_{t+1}$.
12:**Until reach desired iteration**.
Return $\pi_u(a|s)$.

B. Sustainable ℓ_2-Regularized RLS-TD With Gradient Correction

The state matrix in an RLS-TD algorithm is often initialized as a diagonal matrix with a small positive coefficient to make the state matrix invertible. In the perspective of regularization, this initial coefficient can realize ℓ_2-regularization (also mentioned as ridge regression). Thus, the objective function is combined by a mean-square-projected Bellman error (MSPBE) and a ℓ_2-regularization term. As this initial coefficient is constant, the weight of the ℓ_2-regularization term

is also constant. However, the number of samples increases in the process of learning, and the percentage of MSPBE as error loss also gradually increases. Accordingly, the percentage of ℓ_2-regularization term is squashed. This compression means that the effect of ℓ_2-regularization decreases during learning and the ℓ_2-norm of the parameters increases. This weaken regularization is not good for convergence. To solve this problem, RRC [38] adds a regularization step at each time step to make the effect of ℓ_2-regularization sustainable but not weaken during learning. Then, RRC's objective is to find the solution of the following objective function:

$$f(\theta^{**}) = \arg\min_{\mu} \frac{1}{2}\|\tilde{\Phi}\mu - (\tilde{R} + \gamma\,\tilde{\Phi}')\theta\|^2 + \bar{\varepsilon}\|\mu\|_2^2, \quad (11)$$

where the sample matrices $\tilde{\Phi}$, $\tilde{\Phi}'$, and $\tilde{R}$ are defined as

$$\tilde{\Phi} \equiv \begin{bmatrix} \phi(s_1)^T \\ \vdots \\ \phi(s_m)^T \end{bmatrix}, \quad \tilde{\Phi}' \equiv \begin{bmatrix} \phi(s_1')^T \\ \vdots \\ \phi(s_m')^T \end{bmatrix}, \quad \text{and} \quad \tilde{R} \equiv \begin{bmatrix} r_1 \\ \vdots \\ r_m \end{bmatrix}, \quad (12)$$

respectively, and $\bar{\varepsilon} \in \mathbb{R}$ is a ℓ_2-regularization parameter. Other ℓ_2-regularized LSTD algorithms find the solution of

$$f(\theta^{*}) = \arg\min_{\mu} \frac{1}{2}\|\tilde{\Phi}\mu - (\tilde{R} + \gamma\,\tilde{\Phi}')\theta\|^2 + \lim_{k\to m}\frac{1}{k}\bar{\varepsilon}\|\mu\|_2^2. \quad (13)$$

Both the right-hand sides of (11) and (13) consist of an MSPBE and a ℓ_2-regularization term. Using an auxiliary LFA with parameters $\omega \in \mathbb{R}^n$ to approximate the TD error, the linear system solved by RRC is

$$\begin{bmatrix} \tilde{\Phi}^T\tilde{\Phi} - \gamma\,\tilde{\Phi}^T\tilde{\Phi}' & \gamma\,\tilde{\Phi}'^T\tilde{\Phi} \\ \tilde{\Phi}^T\tilde{\Phi} - \gamma\,\tilde{\Phi}^T\tilde{\Phi}' & \tilde{\Phi}^T\tilde{\Phi} \end{bmatrix}\rho + \sum_{i=1}^{k}\bar{\varepsilon} I\rho = \begin{bmatrix} \tilde{\Phi}^T\tilde{R} \\ \tilde{\Phi}^T\tilde{R} \end{bmatrix}, \quad (14)$$

where $\rho = [\theta; \omega]$.

The state matrix in (14) can be rewritten in the following incremental form:

$$\tilde{G}_k \equiv \sum_{i=1}^{k}\left(\begin{bmatrix} \phi_i(\phi_i - \gamma\,\phi_i')^T & \gamma\,\phi_i'\phi_i^T \\ \phi_i(\phi_i - \gamma\,\phi_i')^T & \phi_i\phi_i^T \end{bmatrix} + \bar{\varepsilon} I\right), \quad \tilde{h}_k \equiv \sum_{i=1}^{k}\begin{bmatrix} r_i\phi_i \\ r_i\phi_i \end{bmatrix}. \quad (15)$$

If we denote P_k as the inverse of state matrix $\tilde{G}_k$, then the incremental form of P_k is

$$P_k = \left\{ P_{k-1} + \underbrace{\begin{bmatrix} \phi_k \\ \phi_k \end{bmatrix}}_{u_{1,k}} \underbrace{\begin{bmatrix} (\phi_k - \gamma\,\phi_k')^T & \mathbf{0} \end{bmatrix}}_{v_{1,k}^T} + \underbrace{\begin{bmatrix} \gamma\,\phi' \\ \phi_k \end{bmatrix}}_{u_{1,k}} \underbrace{\begin{bmatrix} \mathbf{0} & \phi_k^T \end{bmatrix}}_{v_{1,k}^T} + \underbrace{\bar{\varepsilon}\mathcal{I}_{(1)}\mathcal{I}_{(1)}^T + \bar{\varepsilon}\mathcal{I}_{(2)}\mathcal{I}_{(2)}^T + \cdots + \bar{\varepsilon}\mathcal{I}_{(2n)}\mathcal{I}_{(2n)}^T}_{\bar{\varepsilon} I} \right\}^{-1}, \quad (16)$$

where $\mathcal{I}_{(i)}$ is an element vector whose elements are zeros, except for the ith one, which is 1, P_k can be solved using the Sherman–Morrison formula $2(n+1)+2n$ times. The recursive steps for the vector product terms $u_{1,k}v_{1,k}^T + u_{2,k}v_{2,k}^T$ in (16) are shown as follows:

$$\begin{cases} \Psi_{1,k} = P_{k-1}u_{1,k} \\ K_{1,k} = \Psi_{1,k}\left(v_{1,k}^T P_{k-1}\right)/\left(1 + v_{1,k}^T\Psi_{1,k}\right) \\ D_k = P_{k-1} - K_{1,k} \end{cases} \quad (17)$$

$$\begin{cases} \Psi_{2,k} = D_k u_{2,k} \\ K_{2,k} = \Psi_{2,k}\left(v_{2,k}^T D_k\right)/\left(1 + v_{2,k}^T\Psi_{2,k}\right) \\ P_k = D_k - K_{2,k}. \end{cases} \quad (18)$$

By defining $P_{k,(j)}$ as the inverse of $\tilde{G}_k$ with the first j terms of $\bar{\varepsilon} I$, the recursive steps for the last $2n$ terms in $\tilde{G}_k$ can be updated by

$$\begin{aligned} P_{k,(j)} &= P_{k,(j-1)} - P_{k,(j-1)}\bar{\varepsilon}\mathcal{I}_{(j)} \\ &\quad \times\left(1 + \mathcal{I}_{(j)}^T P_{k,(j-1)}\bar{\varepsilon}\mathcal{I}_{(j)}\right)^{-1}\mathcal{I}_{(j)}^T P_{k,(j-1)} \\ &= P_{k,(j-1)} - \bar{\varepsilon}\,P_{k,(j-1)}(\cdot, j) \\ &\quad \times(1 + \bar{\varepsilon}\,P_{k,(j-1)}(j, j))^{-1}P_{k,(j-1)}(j, \cdot). \end{aligned} \quad (19)$$

Then, RRC updates the parameters by $\rho_k = P_{k,(2n)}\tilde{h}_k$. Text I shows the pseudocode of RRC.

Text 1 RRC

Initialization:

Parameters and samples:

Regularization parameter $\bar{\varepsilon} \in \mathbb{R}_+$,
Discounted factor $\gamma \in [0, 1]$,
Value function basis function $\phi(s)$,
State transition and reward samples from time-step 1 to $m\{s_i, r_i, s_i'\}_{i=1}^m$.

Variables:

$\rho_k = \mathbf{0}$, $\tilde{h}_0 = \mathbf{0}$, $\tilde{G}_0 = \bar{\varepsilon} I$.

1: **Repeat**
2: Compute $u_{1,k}$, $u_{2,k}$, $v_{1,k}^T$, and $v_{2,k}^T$ as in (14),
3: Compute P_k as in (15) and (16),
4: **Repeat**
5: Compute $P_{k,(j)}$ as in (17),
6: $j \leftarrow j + 1$.
7: **Until** $j = 2n$.
8: Compute $\rho_k = P_{k,(2n)}\tilde{h}_k$,
9: $k \leftarrow k + 1$,
10: **Until** $k = m$.
Return ρ_m.

The computational complexity of RRC is $O(n^3)$. We can reduce it to $O(n^2)$ by assuming that the number of samples is much larger than the dimension of basis function. When a new sample comes, only one element $\bar{\varepsilon}\mathcal{I}_{(p)}\mathcal{I}_{(p)}^T$ of $\bar{\varepsilon} I$ in (19) is added, and then P_k can be recursively updated by

$$P_k = P_{k-1} - \bar{\varepsilon}\,P_{k-1}(\cdot, p)(1 + \bar{\varepsilon}\,P_{k-1}(p, p))^{-1}P_{k-1}(p, \cdot). \quad (20)$$

This variant of RC is termed fast sustainable ℓ_2-regularized RC (FRRC), and the pseudocode of FRRC is the same as RRC expect that we use the following steps to replace steps 1–10 in Text 1.

Text 2

Repeat
  Compute $u_{1,k}$, $u_{2,k}$, $v_{1,k}^T$, and $v_{2,k}^T$ as in (16),
  Compute P_k as in (20),
  $p \leftarrow p+1$,
  If $(p > 2n)$, $p \leftarrow 1$.
  Compute $\rho_k = P_k \tilde{h}_k$,
  $k \leftarrow k+1$,
Until $k = m$.

IV. Convergence Analysis

We now present the convergence analysis of the proposed AC framework. We prove that CIPG converges to a local optimal policy under some conditions and assumptions.

Since our algorithm's actor update is the same as [2] and [23], with the exception that our algorithm is under the start-state formulation, the convergence analysis of our actor update should be similar to that of [2] and [15]. Furthermore, in the critic update, the least squares method is used to find the TD fixed point of value function parameters in place of the first-order algorithms in [15]. As mentioned earlier, our critic update works in the way of on-policy learning during each iteration. Hence, the convergence analysis of the proposed algorithm can be divided into two steps: the critic's convergence analysis and the actor's convergence analysis.

We recall the convergence analysis result of the critic from [38]. The corresponding assumptions are summarized as follows.

Assumption 1: The underlying policy is stationary; the MDP is finite, ergodic, and stationary; and the expectation of the reward function is bounded.

Assumption 2: The basis function matrix is full rank, and is bounded for each state.

Assumption 3: All of the recursive inversing equations satisfy $1 + v^T A^{-1} u \neq 0$ in the denominators.

Under these assumptions, RRC and FRRC algorithms converge to the unique TD fixed point θ^{**} of (11) with a probability of one [38]. That is

$$\lim_{k\to\infty} \rho_k = \lim_{k\to\infty} \begin{bmatrix} \theta_k & \omega_k \end{bmatrix}^T \approx \begin{bmatrix} \theta^{**} & \mathbf{0} \end{bmatrix}^T. \tag{21}$$

Note that the fixed point in (11) is different from other ℓ_2-regularized least-squares-based algorithms, such as RC and LSTD.

Fig. 1 shows the different learning processes of (11) and (13). In RC and LSTD, with the increasing number of samples, the radii of the ℓ_2-norm ball of μ will also increase and the TD fixed point is thus changing. This accounts for why the effect of ℓ_2-regularization of RC and LSTD decreases. However, the radii of the ℓ_2-norn ball of RRC and FRRC are constant; therefore, they can sustain the ℓ_2-regularization effect in the learning process.

Now, we present the convergence analysis of the actor, which is similar to that in [2]. From Conditions 1–3, we can obtain the convergence theorem as follows.

Condition 1: $\hat{\delta}_t^{\pi}$ is an unbiased estimate of δ_t^{π} :$\mathbb{E}[\hat{\delta}_t^{\pi}|u] = \delta_t^{\pi}$, in which $\hat{\delta}_t^{\pi} = r_{t+1} + \gamma \theta^{\pi T} \phi(s_{t+1}) - \theta^{\pi T} \phi(s_t)$ denotes the estimation of TD error at time step t.

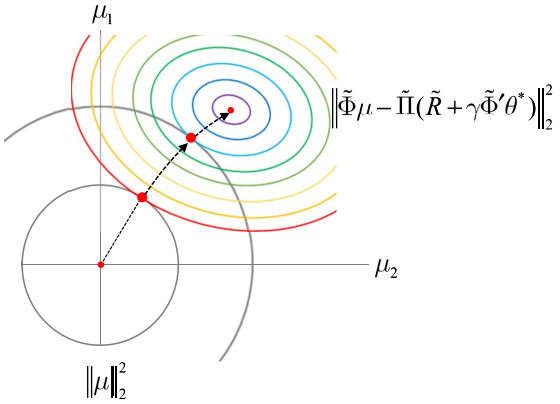

Fig. 1. Trajectory of a fixed point throughout the learning process.

Condition 2: where $\widehat{\nabla_u J(\pi_u)} = \delta_t \psi(s,a)$ is an unbiased estimate of $\nabla_u J(\pi_u)$.

Condition 3: where $\partial^2 \pi_u(a|s)/\partial u_i \partial u_j$ and $\partial^2 J(\pi_u)/\partial u_i \partial u_j$ are bounded.

Theorem 1 (Convergence of Critic-Iteration Policy Gradient With Function Approximation): Let $\pi_u(a|s)$ and $\widehat{V}^{\pi}(s)$ be any differentiable function approximation for the policy and value function, respectively. Let $\{\beta_t\}$ and $\hat{\delta}_t = r_{t+1} + \gamma \theta^T \phi(s_{t+1}) - \theta^T \phi(s_t)$ be the step-size schedule that satisfies the condition (10) and the estimation of TD error at time step t, respectively. Then, for an MDP with bounded rewards $\{r_t\}$, following the parameter iterations in Algorithm 1, we have $\lim_{t\to\infty} \nabla_u J(\pi_u) = 0$ with a probability of 1, where $\nabla_u J(\pi_u) = \mathbb{E}[\delta_t^{\pi} \psi(s_t, a_t)|u] = \mathbb{E}[\hat{\delta}_t^{\pi} \psi(s_t, a_t)|u]$.

Proof: Let θ^{**} be the optimum weight parameter that minimizes the MSPBE $\text{MSPBE}(\theta) = \|V_\theta - \Pi T V_\theta\|_D^2$, where Π is a projection operation, T is the Bellman operator, and $D \in \mathbb{R}^{|S|\times|S|}$ is a state-distribution diagonal matrix [2]. The error in the estimation of $\delta_t^{\pi} = r_{t+1} + \gamma \theta^{**T} \phi(s_{t+1}) - \theta^{**T} \phi(s_t)$ caused by our AC framework is $e_t^{\pi} = \hat{\delta}_t^{\pi_{m-1}} - \delta_t^{\pi_m}$, where π_m and π_{m-1} are the policy in the mth and $(m-1)$th iteration. The resulting PG is $\widehat{\nabla_u J}(\pi_u) = (\delta_t^{\pi_m} - e_t^{\pi}) \psi(s_t, a_t)$. Since $\beta_t \to 0$, we have for all m greater than some index $\bar{m}$, the distance $\|\pi_{m-1} - \pi_m\|^2$ is small enough, then the RLS-TD critic (RRC or FRRC) in our AC framework evaluates the value function of π_m. As mentioned earlier, RRC and FRRC algorithms converge to the unique TD fixed point θ^{**} under the policy π_m. Thus, we have $\mathbb{E}[\delta_t^{\pi_m}|u_m] = \hat{\delta}_t^{\pi_{m-1}}$ and $e_t^{\pi} \to 0$. Then, we can obtain $\widehat{\nabla_u J}(\pi_u) = \delta_t^{\pi_m} \psi(s_t, a_t)$. From Lemma 1, and following the definition of the gradient of the parameterized policy in formula (4), we have $\widehat{\nabla_u J}(\pi_u) = \delta_t \psi(s,a)$'C which is a unbiased estimate of $\nabla_u J(\pi_u)$, and causes the actor to be updated along with the direction of the PG. Hence, Conditions 1 and 2 are verified.

For Condition 3: a simple calculation shows that

$$\partial^2 \pi_u(a|s)/\partial u_i \partial u_j = \pi_u(a|s)[\psi'(s,a)\psi(s,a) - \sum_{a' \in \mathcal{A}} \pi_u(a'|s)\psi'(s,a)\phi(s,a')]. \tag{22}$$

Thus, $\partial^2\pi_u(a|s)/\partial u_i\partial u_j$ exists and is bounded. As we assume that rewards $\{r_t\}$ are bounded, we can directly conclude that $\partial^2 J(\pi_u)/\partial u_i\partial u_j$ is also bounded. Thus, $\nabla_u J(\pi_u)$ is a Lipschitz continuous function.

With the above-mentioned conditions and the step size defined in (10), [43, Proposition 3.5] holds, and the actor then converges to a local optimum. The claim follows.

Next, we turn to discuss how the error in using function approximation affects the convergence and performance of the proposed method. In order to make discussion more clearly, we replace $J(\pi_u)$ with the notation $J(u)$ without changing the meaning. Furthermore, we consider the MDPs with cost defined as a negative reward $-r_t$ at time step t, and the corresponding objective is to minimize the long-term-discounted total negative rewards. This minimization problem is equivalent to the maximization problem as we considered in the original policy accent settings. Thus, a change occurs only in our actor update

$$u_{t+1} = u_t - \beta_t\delta_t\psi_{s_t a_t}. \tag{23}$$

The first-order Taylor expansion of $J(u_{t+1})$ shows that

$$J(u_{t+1}) = J(u_t) + \beta_t\nabla_u J(u_t)'d_t + o(\beta_t) \tag{24}$$

where d_t denotes the decent direction. In lieu of Lemma 2, the decent direction in our method becomes

$$d_t = -(\nabla_u J(u_t) + e^{\pi_{u_t}}), \tag{25}$$

where $e^{\pi_{u_t}} = \sum_{\mathcal{S}} d^{\pi_{u_t}}(s)[\nabla_u \overline{V^{\pi_{u_t}}}(s) - \gamma\,\nabla_u\theta^{\pi_{u_t}T}\phi(s)]$. To ensure that the directions d_t will not become asymptotically orthogonal to the gradient direction $\nabla_u J(u_t)$ near the nonstationary point, we enforce the following condition [43] on d_t for all t:

$$\begin{aligned}\nabla_u J(u_t)'(\nabla_u J(u_t) + e^{\pi_{u_t}}) &\geq c_1\|\nabla_u J(u_t)\|^2,\\ \|\nabla_u J(u_t) + e^{\pi_{u_t}}\| &\leq c_2\|\nabla_u J(u_t)\|,\end{aligned} \tag{26}$$

where c_1 and c_2 are some positive scalars. In view of the Taylor expansion (24), the above condition implies that

$$\begin{aligned}J(u_{t+1}) - J(u_t) &= \beta_t\nabla_u J(u_t)'d_t\\ &\leq -\beta_t c_1\|\nabla_u J(u_t)\|^2\\ &\leq 0,\end{aligned} \tag{27}$$

which means $J(u_t)$ is monotonically nonincreasing. Thus, if the error in using function approximation satisfies condition (27), then the policy is monotonically improved by our actor update.

V. Simulation and Experiments

This section compares the performances of the proposed two AC algorithms CIPG-RRC and CIPG-FRRC with two other algorithms: 1) RLSTD-PG [4] (RLSTD in the AC framework that uses RLSTD as TD-like critic) and 2) c-PG [15] (conventional PG-based AC algorithm). Three learning control benchmarks are considered: a discrete 20-state chain walk problem is chosen to test the ℓ_2-regularization effect for the parameters in the critic and the mountain car (a goal-reaching task) and inverted pendulum (an avoidance control task) problems are chosen to test the convergence performances. All the simulation results are averaged over 20 runs. Because all of these experiments are learning control problems with finite action space, Gibbs distribution is used to represent the policy. We believe that our algorithms can also be extended to continuous action space cases using normal distribution [44].

A. 20-State Chain Walk Problem

A simple discrete 20-state chain walk problem [31] is first considered to show that CIPG converges to the optimal policy. This MDP consists of 20 states (numbered from 1 to 20), 2 actions ("left" and "right"), and 1 reward at each boundary of the chain (states 1 and 20). Each action has a successful possibility of 0.9 to be executed, and a fail possibility of 0.1, which leads to the execution of the opposite action. The discount factor for this MDP is set to 0.9. The optimal policy is to go left in states from 1 to 10 and to go right in states from 11 to 20.

LFA is constructed to represent state-value functions, and a parameterized Gibbs distribution is used to represent policy. The state feature vector $\phi_s \in \mathbb{R}^d$ for the first experiment contains 11 radial basis function (RBF) basis functions plus a constant term

$$\phi_s = \left[1, \exp\left(\frac{\|s-k\|_2^2}{-2\sigma^2}\right)\right]^T, \quad k = 1, 3, 5, \ldots, 21, \sigma = 7. \tag{28}$$

The state-action feature vector $\phi_{\text{sa}} \in \mathbb{R}^{d\times m}$ is constructed using ϕ_s

$$\phi_{sa_i} = [\underbrace{0,\ldots,0}_{d\times(i-1)}, \phi_s, \underbrace{0,\ldots,0}_{d\times(m-i)},]^T. \tag{29}$$

The initial policy parameters u_0 and critic parameters ρ_0 are set at 0. Since the critic excludes step-size schedules in both RRC and FRRC, we only design step-size sequences for the actor, according to the condition in (10)

$$\beta_t = \beta_0\frac{\beta_c}{\beta_c + t^{2/3}} \tag{30}$$

where $\beta_0 = 0.1$ and $\beta_c = 100$.

Training data in our experiments consist of independent, identically distributed (i.i.d.) samples [45]. In the i.i.d. formulation, the states s_t are i.i.d. generated independently and identically distributed, according to uniform distribution. From each s_t, the next state s_{t+1} is generated according to current policy, and a corresponding reward r_t is generated according to reward function. The final i.i.d. data sequence is (s_k, r_k, s_{k+1}), for $k = 1, 2, \ldots, m$, where m is the length of the sequence. The lengths of the data sequences in three experiments are 1000, 3000, and 1000, respectively. Computational times of the 20-state chain work problem, the inverted pendulum, and the mountain car problem are listed in Table I.

Fig. 2 shows different convergence trends of parameters of two kinds of ℓ_2-regularization schemes: conventional ℓ_2-regularization scheme (corresponding to RLSTD and RC shown in Fig. 2) and sustainable ℓ_2-regularization scheme

TABLE I
COMPUTATIONAL TIMES OF DIFFERENT ALGORITHMS

Problem \ Algorithm	c-PG	RLSTD-PG	CIPG-RRC	CIPG-FRRC
20-stste Chain walk problem	2.28s	2.85s	5.27s	3.68s
Inverted Pendulum	30.39s	40.34s	127.08s	47.89s
Mountain Car	4.29s	5.16s	10.67s	6.53s

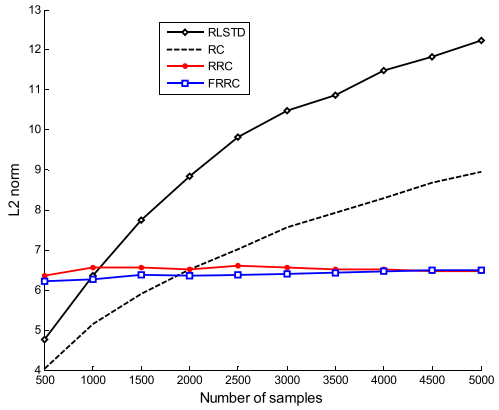

Fig. 2. ℓ_2-norm of RLSTD's, RC's, RRC's, and FRRC's parameters, averaged over 20 runs.

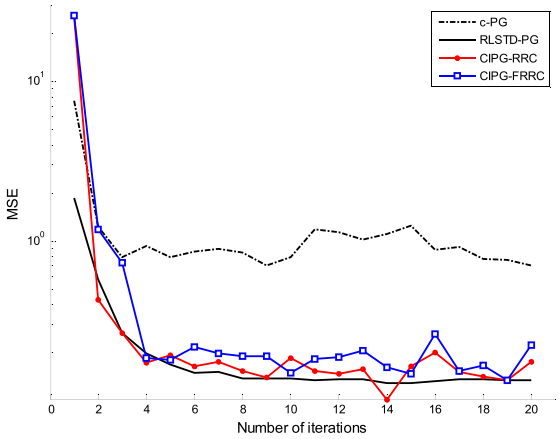

Fig. 3. MSEs of RLSTD-PG, CIPG-RRC, CIPG-FRRC, and c-PG. The results are averaged over 20 runs.

(corresponding to RRC and FRRC shown in Fig. 2). The ℓ_2 regularization parameters of RLSTD, RC, RRC, and FRRC were initialized as 0.2, 0.2, 0.0001, and 0.0001, respectively. In Fig. 2, the ℓ_2-norms of RLSTD's and RC's parameters increase with the increasing number of samples, whereas those of the RRCs and FRRCs keep almost invariant. Particularly, at the time step of 2000, the curves of ℓ_2-norms of RC, RRC, and FRRC are intersected, because at this time step these three algorithms have the equivalent regularization weight in objective function.

In the experiments of Figs. 3 and 4, we initialize the ℓ_2-regularization parameters of RLSTD, RRC, and FRRC, as $\varepsilon = 1$, $\bar{\varepsilon} = 0.0001$, and $\bar{\varepsilon} = 0.0001$, respectively. The mean-squared error (MSE) $\|\hat{V}^\pi - V^{\pi *}\|^2$ is used to measure the performances of these algorithms and c-PG, where $\hat{V}^\pi$ and $V^{\pi *}$ are the evaluated value function of current policy and true value function of optimal policy, respectively. In order to compare the convergence of the three algorithms (RLSTD-PG, CIPG-RRC, and CIPG-FRRC) with c-PG, we tested the policy improvement of these algorithms after the equivalent samples.

Fig. 3 shows the MSEs of c-PG, RLSTD-PG, CIPG-RRC, and CIPG-FRRC on a chain walk problem. All algorithms displayed fast converge. The least-squares-based AC algorithms had smaller MSEs than c-PG. The MSEs of all algorithms decreased with the increasing number of iterations. After four iterations (4000 samples), the MSE of least-squares-based AC algorithms was about 0.2. After 20 iterations (20 000 samples), c-PG's MSE was about 0.7.

Fig. 4 shows the improved policies of CIPG-RRC, RLSTD-PG, and c-PG after two iterations (1000 samples for each iteration). The figure shows the probability of taking the left action according to the improved policy. The optimal policy in this problem is to go left at states from 1 to 10 and to go right otherwise with a probability of 1. CIPG-RRC performs best among these algorithms. After six iterations (6000 samples), CIPG-RRC converges to a near optimal policy, which represents better performance than the policies learned by RLSTD-PG and c-PG.

B. Inverted Pendulum

In the second experiment, the inverted pendulum problem noted in [31] is considered. As shown in Fig. 5, the task is to balance a pendulum with unknown length and mass by applying one of three candidate actions a at each time step: [−50N, 0N, +50N]. All the actions are noisy by adding a uniform noise in [−10N, +10N]. The continuous state space includes the vertical angle ξ and the angular velocity $\dot{\xi}$ of the pendulum. The dynamics of the system is given by the following nonlinear differential equation:

$$\ddot{\xi} = \frac{g\sin(\xi) - \alpha(ml(\dot{\xi})^2\sin(2\xi)/2 + \cos(\xi)a)}{4l/3 - \alpha ml\cos^2(\xi)} \tag{31}$$

where $g = 9.84$ m/s^2, $m = 2.0$ kg, $M = 8.0$ kg, $l = 0.5$ m, $\alpha = 1/(M+m)$ stand for the gravity constant, the mass of the pendulum, the mass of the car, the length of the pendulum, and a constant, respectively. The reward is 0 when $\xi \in [-\pi/2, \pi/2]$ and is -1 otherwise. The simulation time step and the discount factor are set at 0.1 s and 0.96, respectively.

We used identical LFA as in the first experiment for the state feature vectors and state-action feature vectors except that state feature vectors ϕ_s have nine RBF basis functions plus a constant term

$$\phi_s = \left[1, \exp\left(\frac{\|s - k_i\|_2^2}{-2\sigma^2}\right)\right]^T \tag{32}$$

where $s = (\xi, \dot{\xi})$, the centers k_i are nine points of the grid $\{-0.5, 0, 0.5\}\times\{-0.5, 0, 0.5\}$, and $\sigma = 0.5$ (we normalize the states ξ and $\dot{\xi}$ to $[-1, 1]$). We set the step-size schedule parameters for the second experiment at $\beta_0 = 0.1$ and $\beta_c = 1000$.

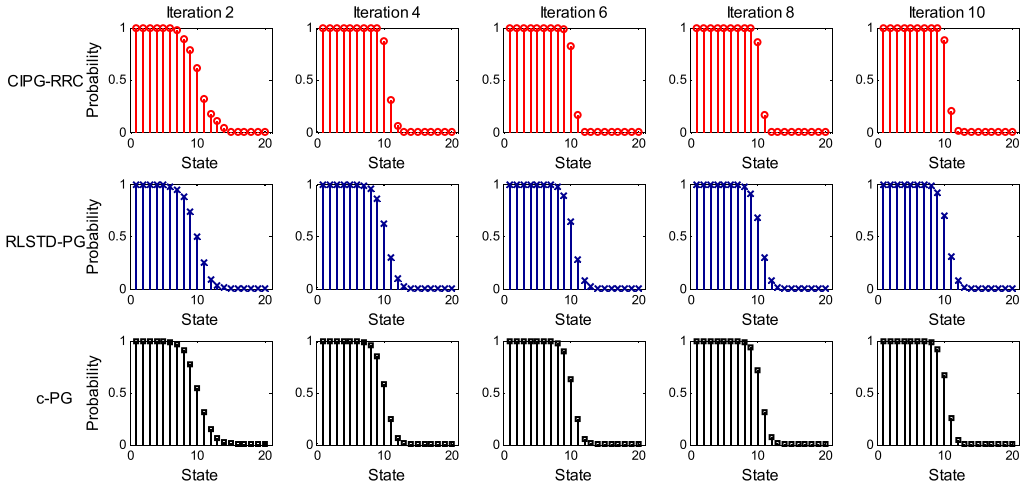

Fig. 4. Improved policy after each iteration (CIPG-RRC: red shade; RLSTD-PG: blue shade; c-PG: black shade). The results are averaged over 20 runs.

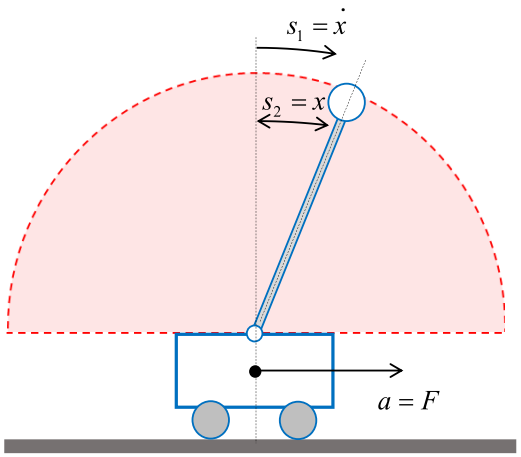

Fig. 5. Inverted pendulum problem.

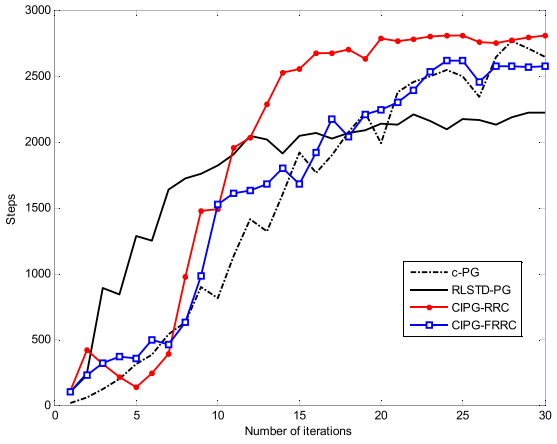

Fig. 6. Inverted pendulum: average steps.

The performances of the policies learned by the least-squares-based AC algorithms and c-PG are shown in Figs. 6 and 7. For each iteration, the improved policies by these algorithms are executed 100 times to evaluate the average numbers of balancing steps and the average probabilities

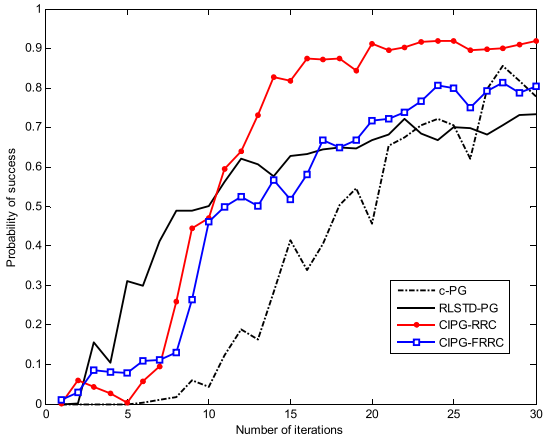

Fig. 7. Inverted pendulum: average probability of success.

of success. If a policy balances the pendulum over 3000 steps in a run, we call it a successful run. CIPG-RRC performs best among all the AC algorithms and returns very good policy, given 15 training iterations. With 30 training iterations, the expected number of balancing steps of CIPG-RRC is about 2800 steps. The success rate of CIPG-RRC increases fastest among all algorithms at the first 15 iterations and converges to 0.9 probability of success, which is higher than any other algorithm's success rate. Besides, algorithms CIPG-FRRC and c-PG converge to a good policy and perform better than RLSTD-PG after 15 training iterations.

C. Mountain Car

We now consider the mountain car problem suggested in [46]. The task in this problem is to drive an underpowered car to the top of a mountain (shown in Fig. 8). At each time step, three actions a are allowed: full throttle forward ($+1$), full throttle reverse (-1), and zero throttle (0). The car's position x_t and velocity $\dot{x}_t$ are updated by the following dynamic:

$$\begin{cases} x_{t+1} = x_t + \dot{x}_{t+1} \\ \dot{x}_{t+1} = \dot{x}_t + 0.001u - 0.0025\cos(3x_t) \end{cases} \tag{33}$$

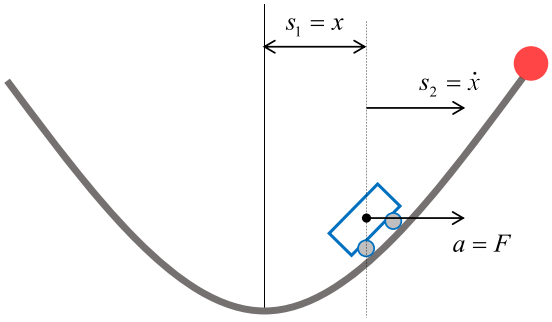

Fig. 8. Mountain car problem.

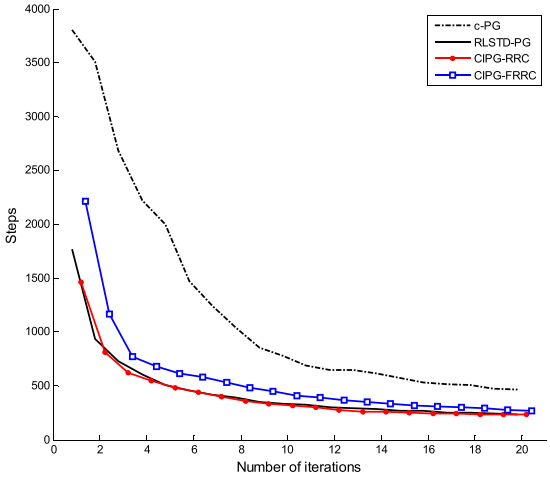

Fig. 9. Mountain car: average steps.

where $x_{t+1} \in [-1.2, 0.5]$ and $\dot{x}_{t+1} \in [-0.07, 0.07]$. $\dot{x}_{t+1}$ resets to 0 when the car reaches the left bound of x_{t+1}.

Reward 1 is given as long as the car reaches the goal $x = 0.5$; otherwise, the reward is 0 at each time step. For the state $s = (x, \dot{x})$, the state feature vectors and the state-action feature vectors have the same form as the second experiment except that the centers k_i have 12 points of the grid $\{-1.1, -0.6, -0.1, 0.4\} \times \{-0.7, 0, 0.7\}$ and $\sigma = 0.5$ (we normalize the state $\dot{x}$ to $[-1, 1]$) and the step-size schedule parameters for the this problem are set at $\beta_0 = 0.1$ and $\beta_c = 1000$.

The performances of the policies learned by the proposed two algorithms, RLSTD-PG and c-PG, are shown in Figs. 9 and 10. For each iteration, the learned policies by these algorithms are executed 100 times to evaluate the average number of steps and the probability of success. If a policy drives the car to the goal within 5000 steps in a run, then we call it a successful run. RLSTD-PG, CIPG-RRC, and CIPG-FRRC return good policies after a few number of iterations. With three iterations (3000 samples), the average number of steps of these three algorithms is less than 800 steps and the probability of success is 100%. The c-PG performs worse than least-squares-based AC methods in this task. c-PG needs more than seven iterations (7000 samples) to achieve a good policy: the expected number of steps is about 1200 and the probability of success is 93.8%. After only two iterations (2000 samples),

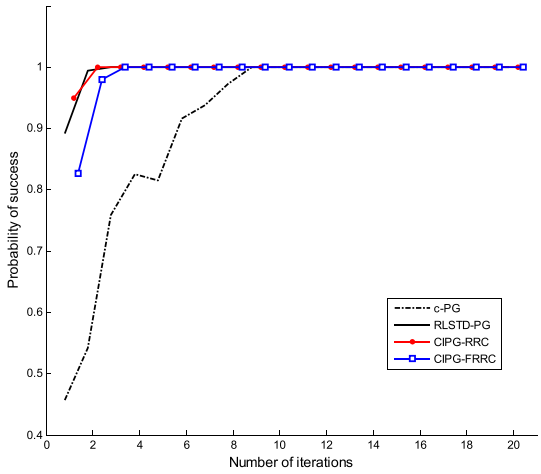

Fig. 10. Mountain car: average probability of success.

CIPG-RRC and RLSTD-PG converge to an optimal policy with the probability of 1.

VI. Conclusion

In this paper, the problem of how to use RLS-based algorithm in the AC framework is studied. Two RLS-based AC algorithms were proposed, namely, CIPG with a sustainable ℓ_2-regularized RLS-TD learning with gradient correction (CIPG-RRC). Furthermore, we also use a fast sustainable ℓ_2-regularized RC to reduce the $O(n^3)$ computational complexity of CIPG-RRC to $O(n^2)$, a low computational complexity counterpart termed CIPG-FRRC was derived. These AC algorithms can avoid the difficulty of tuning the step size in the critic on the contrary to conventional first-order PG-based AC algorithms, and sustain the ℓ_2-regularization effect during the learning process. From convergence analysis, it was found that RRC and FRRC converged to a fixed point that sustained the effect of ℓ_2-regularization throughout the learning processes and the actor converged to the locally optimal policy under the critic and value function approximation. The simulation results showed that CIPG-RRC and CIPG-FRRC are efficient methods for learning control problems with ℓ_2-regularization. The ℓ_2-norm of the sustainable ℓ_2-regularized algorithms remained more stable during the learning process, as compared with the conventional RLS-TD learning algorithms. CIPG-RRC and CIPG-FRRC achieved high convergence rates on the 20-state chain walk, the inverted pendulum, and the mountain car tasks, using the samples generated by the i.i.d. sampling method. These two algorithms have more deterministic parametric probability distributions than conventional PG-based AC algorithms. Hence, these two AC algorithms are efficient options for learning control tasks.

Based on the CIPG-RRC and CIPG-FRRC algorithms proposed in this paper, further works may include the following aspects: 1) the computational complexity of the proposed algorithms could be further reduced for online learning problems; 2) some recent advanced PG algorithms can be incorporated to improve the performance of the proposed AC framework; and 3) the real-world applications of our algorithms should be studied.

REFERENCES

[1] C. Szepesvári, *Algorithms for Reinforcement Learning*. San Rafael, CA, USA: Morgan and Claypool Publishers, Jun. 2010.

[2] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems 12*. Cambridge, MA, USA: MIT Press, Nov. 2000, pp. 1057–1063.

[3] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM J. Control Optim.*, vol. 42, no. 4, pp. 1143–1166, 2003.

[4] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, nos. 7–9, pp. 1180–1190, Mar. 2008.

[5] B. Luo, D. Liu, H.-N. Wu, D. Wang, and F. L. Lewis, "Policy gradient adaptive dynamic programming for data-based optimal control," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3341–3354, Oct. 2017.

[6] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. (2015). "High-dimensional continuous control using generalized advantage estimation." [Online]. Available: https://arxiv.org/abs/1506.02438

[7] R. Song, F. Lewis, Q. Wei, H.-G. Zhang, Z.-P. Jiang, and D. Levine, "Multiple actor-critic structures for continuous-time optimal control using input-output data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 4, pp. 851–865, Apr. 2015.

[8] D. Wang, D. Liu, Q. Wei, D. Zhao, and N. Jin, "Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming," *Automatica*, vol. 48, no. 8, pp. 1825–1832, Aug. 2012.

[9] Z. Ni, H. He, D. Zhao, X. Xu, and D. V. Prokhorov, "GrDHP: A general utility function representation for dual heuristic dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 614–627, Mar. 2015.

[10] X. Xu, Z. Hou, C. Lian, and H. He, "Online learning control using adaptive critic designs with sparse kernel machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 5, pp. 762–775, May 2013.

[11] H. Zhang, L. Cui, X. Zhang, and Y. Luo, "Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2226–2236, Dec. 2011.

[12] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 3, pp. 621–634, Mar. 2014.

[13] H. Zhang, H. Liang, Z. Wang, and T. Feng, "Optimal output regulation for heterogeneous multiagent systems via adaptive dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 1, pp. 18–29, Jan. 2017.

[14] B. Luo *et al.*, "Output tracking control based on adaptive dynamic programming with multistep policy evaluation," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

[15] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor–critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, Nov. 2009.

[16] B. Kim, J. Park, S. Park, and S. Kang, "Impedance learning for robotic contact tasks using natural actor-critic algorithm," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 2, pp. 433–443, Apr. 2010.

[17] B. Luo, H.-N. Wu, and T. Huang, "Optimal output regulation for model-free Quanser helicopter with multi-step Q-learning," *IEEE Trans. Ind. Electron.*, vol. 65, no. 6, pp. 4953–4961, Jun. 2018.

[18] O. Tutsoy, D. E. Barkana, and S. Colak, "Learning to balance an NAO robot using reinforcement learning with symbolic inverse kinematic," *Trans. Inst. Meas. Control*, vol. 39, no. 11, pp. 1735–1748, Apr. 2016.

[19] O. Tutsoy, "CPG based RL algorithm learns to control of a humanoid robot leg," *Int. J. Robot. Autom.*, vol. 30, no. 2, pp. 1–7, 2015.

[20] H. R. Berenji and D. Vengerov, "A convergent actor-critic-based FRL algorithm with application to power management of wireless transmitters," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 4, pp. 478–485, Aug. 2003.

[21] D. Vengerov, N. Bambos, and H. R. Berenji, "A fuzzy reinforcement learning approach to power control in wireless transmitters," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 4, pp. 768–778, Aug. 2005.

[22] S. Xie, W. Zhong, K. Xie, R. Yu, and Y. Zhang, "Fair energy scheduling for vehicle-to-grid networks using adaptive dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 8, pp. 1697–1707, Aug. 2016.

[23] S. Richter, D. Aberdeen, and J. Yu, "Natural actor-critic for road traffic optimisation," in *Proc. 20th Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2007, pp. 1169–1176.

[24] L. Chun-Gui, W. Meng, S. Zi-Gaung, L. Fei-Ying, and Z. Zeng-Fang, "Urban traffic signal learning control using fuzzy actor-critic methods," in *Proc. 5th Int. Conf. Natural Comput.*, Tianjin, China, Aug. 2009, pp. 368–372.

[25] C. V. L. Raju, Y. Narahari, and K. Ravikumar, "Reinforcement learning applications in dynamic pricing of retail markets," in *Proc. IEEE Int. Conf. E-Commerce*, Newport Beach, CA, USA, Jun. 2003, pp. 339–346.

[26] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, Aug. 1988.

[27] M. Geist and B. Scherrer, "Off-policy learning with eligibility traces: A survey," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 289–333, Jan. 2014.

[28] R. S. Sutton *et al.*, "Fast gradient-descent methods for temporal-difference learning with linear function approximation," in *Proc. 26th Int. Conf. Mach. Learn.*, Montreal, QC, Canada, Jul. 2009, pp. 993–1000.

[29] S. J. Bradtke and A. G. Barto, "Linear least-squares algorithms for temporal difference learning," *Mach. Learn.*, vol. 22, nos. 1–3, pp. 33–57, Jan. 1996.

[30] J. A. Boyan, "Technical update: Least-squares temporal difference learning," *Mach. Learn.*, vol. 49, nos. 2–3, pp. 233–246, Nov. 2002.

[31] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Mach. Learn. Res.*, vol. 4, no. 6, pp. 1107–1149, Dec. 2004.

[32] T. Song, D. Li, L. Cao, and K. Hirasawa, "Kernel-based least squares temporal difference with gradient correction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 771–782, Apr. 2016.

[33] X. Xu, D. Hu, and X. Lu, "Kernel-based least squares policy iteration for reinforcement learning," *IEEE Trans. Neural Netw.*, vol. 18, no. 4, pp. 973–992, Jul. 2007.

[34] A. M. Farahmand, M. Ghavamzadeh, C. Szepesvári, and S. Mannor, "Regularized policy iteration," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 2008, pp. 441–448.

[35] A. M. Farahmand, M. Ghavamzadeh, C. Szepesvari, and S. Mannor, "Regularized fitted Q-iteration for planning in continuous-space Markovian decision problems," in *Proc. Amer. Control Conf.*, St. Louis, MO, USA, Jun. 2009, pp. 725–730.

[36] M. W. Hoffman, A. Lazaric, M. Ghavamzadeh, and R. Munos, "Regularized least squares temporal difference learning with nested ℓ_2 and ℓ_1 penalization," in *Proc. 9th Eur. Conf. Recent Adv. Reinforcement Learn.*, Athens, Greece, 2012, pp. 102–114.

[37] D.-R. Liu, H.-L. Li, and D. Wang, "Feature selection and feature learning for high-dimensional batch reinforcement learning: A survey," *Int. J. Autom. Comput.*, vol. 12, no. 3, pp. 229–242, Jun. 2015.

[38] T. Song and D. Li, "Online ℓ_2-regularized reinforcement learning via RBF neural network," in *Proc. 28th Chin. Control Decision Conf.*, Yinchuan, China, May 2016, pp. 6627–6632.

[39] J. Park, J. Kim, and D. Kang, "An RLS-based natural actor-critic algorithm for locomotion of a two-linked robot arm," in *Computational Intelligence and Security* (Lecture Notes in Computer Science), vol. 3801. Berlin, Germany: Springer-Verlag, 2005, pp. 65–72.

[40] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," presented at the 3rd IEEE-RAS Int. Conf. Human Robots, Karlsruhe, Germany, Sep. 2003, pp. 1–20.

[41] X. Xu, H.-G. He, and D. Hu, "Efficient reinforcement learning using recursive least-squares methods," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 259–292, Jan. 2002.

[42] O. Tutsoy and S. Colak, "Adaptive estimator design for unstable output error systems: A test problem and traditional system identification based analysis," in *Proc. Inst. Mech. Eng. I, J. Syst. Control Eng.*, vol. 229, no. 10, pp. 902–916, Nov. 2015.

[43] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA, USA: Athena Scientific, 1996.

[44] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *Proc. Amer. Control Conf. (ACC)*, Montreal, QC, Canada, Jun. 2012, pp. 2177–2182.

[45] R. S. Sutton, C. Szepesvári, and H. R. Maei, "A convergent O(n) temporal-difference algorithm for off-policy learning with linear function approximation," in *Proc. Conf. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2009, pp. 1609–1616.

[46] R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

Luntong Li received the B.S. degree in automation and the M.S. degree in control science and engineering from the Department of Automation, Beijing University of Chemical Technology, Beijing, China, in 2013 and 2016, respectively, where he is currently pursuing the Ph.D. degree with the College of Information Science and Technology.

His current research interests include reinforcement learning and approximate dynamic programming.

Dazi Li received the Ph.D. degree in engineering from the Department of Electrical and Electronic Systems, Kyushu University, Fukuoka, Japan, in 2004.

She is currently a Full Professor of automatic control and the Chair of the Department of Automation, College of Information Science and Technology, Beijing University of Chemical Technology, Beijing, China. Her research interests include machine learning and artificial intelligence, advanced process control, complex system modeling and optimization, and fractional calculus system. She is currently an Associate Editor of *ISA Transactions*.

Tianheng Song received the M.S. and Ph.D. degrees in control science and engineering from the Department of Automation, Beijing University of Chemical Technology, Beijing, China, in 2011 and 2016, respectively.

Since 2017, he has been holding a post-doctoral position at the College of Information Science and Technology, Beijing University of Chemical Technology. His current research interests include reinforcement learning and neural networks.

Xin Xu (M'07–SM'12) received the B.S. degree in electrical engineering from the Department of Automatic Control, National University of Defense Technology (NUDT), Changsha, China, in 1996, and the Ph.D. degree in control science and engineering from the College of Mechatronics and Automation, NUDT, in 2002.

He is currently a Full Professor with the College of Mechatronics and Automation, NUDT. He has authored or co-authored over 150 papers in international journals and conferences, and co-authored four books. His research interests include reinforcement learning, approximate dynamic programming, machine learning, robotics, and autonomous vehicles.

Dr. Xu was a recipient of the Second Class National Natural Science Award of China in 2012. He is a Committee Member of the IEEE RAS Technical Committee on Approximate Dynamic Programming and Reinforcement Learning and the IEEE CIS Technical Committee on Robot Learning. He is currently an Associate Editor of *Information Sciences* and a Guest Editor of the IEEE TRANSACTIONS ON SYSTEM, MAN, AND CYBERNETICS: SYSTEMS.