Adaptive Optimal Control for A Class of Continuous-Time Unknown Nonlinear Systems via Generative Adversarial Networks

Chenglong Wang, Haiyang Fang, Shuping He*

Abstract: In this paper, we study the adaptive optimal controller design problems for a class of continuous-time nonlinear systems by using the generative adversarial networks (GANs). Combining the Q-learning algorithm and GANs scheme, we successfully design a new adaptive optimal control algorithm for continuous-time unknown nonlinear systems. We adopt the latest GANs training strategy to stabilize the nonlinear systems and prove the convergence of the designed adaptive optimal control algorithm. Finally, the effectiveness of the proposed method is verified by a simulation example and the superiority of the algorithm is illustrated by comparing the traditional actor-critic algorithm.

Keywords: Generative adversarial networks (GANs), Adaptive optimal control, Nonlinear systems, Q-learning, Reinforcement learning.

I. Introduction

When we search for the optimal control scheme of nonlinear systems, we always need to solve the Hamilton-Jacobi-Behrman (HJB) equations. In general, it is awalys difficult to solve the HJB equations directly. As a favorable tool, dynamic programming (DP) might be helpful and a series of meaningful results have been published. But there exists some problems when applying DP to tackle the high dimensional complex systems. For this reason, a great deal of effort has been devoted to develop algorithms that approximate the solution of this equation [1]–[4]. In fact, the adaptive dynamic programming (ADP) [5]–[7]method can be introduced to the adaptive optimal controller design for nonlinear systems. It involves a computational intelligence technique called PI [8], [9]. In [10], the authors presented a new online PI algorithm to solve the optimal controller of linear systems without knowing the internal dynamics. Then the relevant method was introduced to linear systems with completely unknown dynamics [11]. Liu and Wei [12] extended the PI algorithm to the optimal control problem of discrete nonlinear systems. Luo and Huang [13] came up with a data-based approximate PI scheme for affine nonlinear continuous-time optimal control problem. Song et. al. [14] studied a data-driven PI-based optimal control algorithm for a class of continuous-time stochastic systems with Markovian jumps. For other applications of PI schemes, the readers can refer to [15]–[17].

In the other methods, the agents learn optimal value functions and follow the value iteration strategy. For example, Q-learning tries to directly find the action value of the optimal strategy, which is not necessarily applicable to the data generation. In other words, the obtained Q-learning strategy is usually different from the sample generation strategy. However, while solving the problem of high-dimensional observation space, Q-learning can only deal with discrete low-dimensional action space. Many interesting tasks, especially the physical control tasks, have continuous and higher-dimensional operational space. Q-learning cannot be directly applied to the continuous domain, because it relies on finding the action function that maximizes the action value; while in the continuous case, each step needs to be an iterative optimization process.

One of the main goals in the field of artificial intelligence is to solve complex tasks through high dimensional sensor input. In recent years, some significant progress has been made in combining deep learning with reinforcement learning (RL) method. The actor-critic algorithm was introduced and further developed by Werbos to solve the optimal control problem online.

The actor-critic (AC) method and generative adversarial networks (GANs) scheme are two kinds of multilevel optimization structures with independent attributes. In both cases, the information flow is a simple feedforward from a model that takes action or generates samples to a second model, which evaluates the output of the first model. The second model is the only one that has direct access to the special information in the environment, either the reward information or the real sample from the relevant distribution and the first model must learn only from the error signals in the second model. In the AC method, the neural network function approximation matrix is used to estimate the action value function. It is difficult and unstable to use large nonlinear function approximators to learn value functions. The innovation of GANs lie in its ability to learn value functions in a stable and robust way by using generative adversarial networks.

In this paper, we propose a new adaptive optimal control method inspired by the analogy of AC and GANs scheme. We use GANs to distribute Bellman target updates through the generator/discriminator architecture. Our algorithm can effectively solve the adaptive optimal control problem of continuous-time unknown nonlinear systems in a stable and robust way. We will demonstrate the effectiveness of our

C. Wang, H. Fang and S. He are with the Key Laboratory of Intelligent Computing and Signal Processing (Ministry of Education), School of Electrical Engineering and Automation, Anhui University, Hefei 230601, China.

*Tel./fax: +86 551 63861413. E-mail address: shuping.he@ahu.edu.cn

proposed algorithm as a viable alternative scheme.

The rest of this paper is organized as follows. In Section II, we give the system description and discuss the optimal control problem for continuous-time unknown nonlinear systems. Then the core idea and the structure of GANs are introduced. In Section III, the GANs-based adaptive optimal control method is proposed and the stabilizing strategies are given. We also show the convergence of the GANs-based RL algorithm. In Section IV, a simulation example is given to verify the effectiveness of the proposed algorithm.

II. Preliminaries

A. Problem Description

Consider a class of continuous-time nonlinear systems described by:

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

with the state $x(t) \subseteq R^n$, the control input $u(x(t)) \subseteq R^m$ and the unknown nonlinear function $f(x(t), u(x(t))) \subseteq R^{m\times n}$. We assume that $f(x(t), u(x(t)))$ is Lipschitz continuous on $\Omega \subseteq R^n$ which contains the origin and nonlinear system (1) is stabilizable, i.e, there exists a continuous-time control function $u(x(t))$ such that it is asymptotically stable on Ω.

For convenience, we rewrite $x(t)$ as x_t. The infinite horizon integral cost is defined as:

$$V(x_t) = \int_0^\infty r(x_t, u(x_t))d\tau, \tag{2}$$

where $r(x_t, u(x_t))$ is a known cost function described by the systematic observations.

To solve the optimal control problem of nonlinear system (1), we need to find an admissible optimal control policy $u^*(x_t)$ to minimize the cost index (2). The control policy $u(x_t)$ is called to be admissible if it is continuous, has a finite associated cost and can stabilize nonlinear system (1).

Then, we define the following Hamiltonian equation:

$$H(x_t, u(x_t), V_x^*(x_t)) = r(x_t, u(x_t)) + V_x^{*T}(f(x_t, u(x_t))), \tag{3}$$

where $V_x^*(x_t)$ is the derivative of $V^*(x_t)$ with respect to x_t and the optimal cost function $V^*(x_t)$ satisfies the following HJB equation:

$$0 = \min[H(x_t, u(x_t), V_x^*(x_t)]. \tag{4}$$

In order to find the optimal control solution for nonlinear system (1), we can formulate the following policy iteration.

B. Continuous-Time Policy Iteration Algorithm

For nonlinear system (1), if we know the internal system dynamic function $f(x_t, u(x_t))$, i.e., the internal systems dynamic is completely known, we can use the following PI Algorithm 1 to design the optimal control policy $u(x_t)$.

Algorithm 1 PI Algorithm

Initialization:

Select an admissible control policy $u^0(x_t)$.

Step 1 Policy Evaluation:

Solve for $V^i(x_t)$ by using

$$V^i(x_t) = \int_t^{t+T} r(x_t, u^i(x_t))d\tau + V^i(x_{t+T}), \tag{5}$$

where $V^i(x_t)$ represents the cost function generated by the ith iteration.

Step 2 Policy Improvement:

Determine a control policy by:

$$u^{(i+1)}(x_t) = \arg\min_{u(x_t)}[H(x_t, u(x_t), V_x^i(x_t)], \tag{6}$$

where $u^i(x_t)$ represents the control policy generated by the ith iteration.

Loop Iteration:

Iterate Step 1 and Step 2; until $u^i(x_t)$ converges to the optimal control policy $u^*(x_t)$.

Note that the PI Algorithm 1 requires the knowledge of the system dynamics. In order to avoid using the system internal dynamics, we introduce the Q-learning algorithm.

C. Q-Learning

Let us define the Q function associated with the control policy $u(x_k)$ as:

$$Q(x_k, u(x_k)) = r(x_k, u(x_k)) + V(x_{k+1}) \tag{7}$$

where $Q(x_k, u(x_k))$ is a function with respect to the state x_k and the control policy $u(x_k)$ at the time step k. Define the optimal Q function as:

$$Q^*(x_k, u(x_k)) = r(x_k, u(x_k)) + V^*(x_{k+1}). \tag{8}$$

In terms of $Q^*(x_k, u(x_k))$, one can write the Bellman equation as:

$$V^*(x_k) = \min(Q^*(x_k, u(x_k))) \tag{9}$$

Thus the optimal control policy can be given as:

$$u^*(x_k) = \arg\min_{u(x_k)}(Q^*(x_k, u(x_k))). \tag{10}$$

In contrast to PI Algorithm 1, it does not depend on the dynamics of nonlinear system (1). Owing to the fact that it is difficult to find an optimal control policy of $u(x_t)$ at each time-step for nonlinear system (1). Thus, it is impossible to solve the adaptive optimal control problem by directly applying the Q-learning methods.

D. GANs

The GANs formulates an unsupervised learning problem as a game between two opponents: a generator G which samples from a distribution and a discriminator D which classifies the samples as real or false, shown in Fig. 1. The GANs game is then formulated as a zero-sum game where the value is

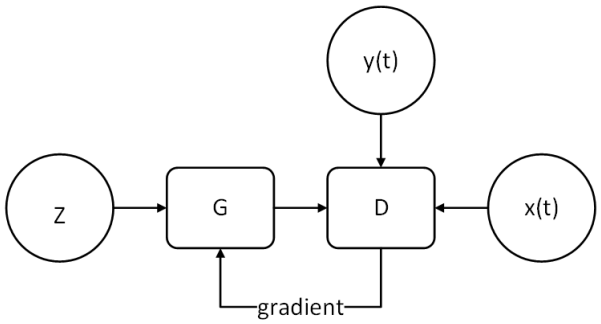

Fig. 1. Information structure and communication method of GAN. The $G(Z)$ and $D(y, x_t)$ represent models with different loss functions. The x_t represents the generation from $G(Z)$, The $y(t)$ is the standard information as the target and the Z represents the random noise signal. The solid line represents the information flow and the gradient represents the gradient flow used by the other model.

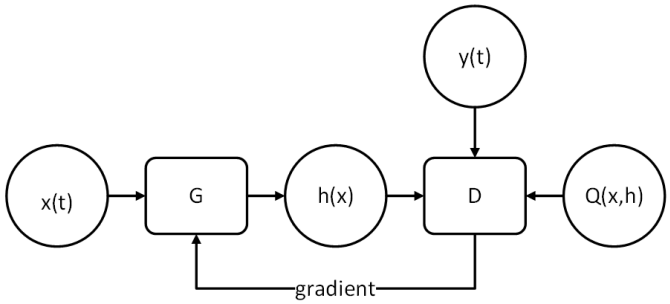

Fig. 2. Structure diagram of reinforcement learning method based on GAN. The G and D represent the neural network model, the x_t represents the state of the system, the $h(x_t)$ represents the adaptive optimal controller of the system simulated by the generator function $G(x_t)$ and $Q(x_t, h(x_t))$ is the Q value in Q-learning algorithm.

the cross-entropy loss between the discriminator's prediction $D(G(Z))$ and the authentically generated $y = D(x_t)$. In other words, D and G play the following two-player minimax game with the value function $V(D, G)$ given by:

$$\min_G \max_D V(D,G) = E_x[\log D(x_t)] + E_z[\log(1 - D(G(Z)))] \tag{11}$$

where E_x and E_z are the expectation of $\log D(x_t)$ and the $\log(1 - D(G(Z)))$ on the probability distribution x_t and Z.

The adversarial GANs demonstrates that it is a reasonable model to build an iterative optimization model. We will illustrate the adaptive control algorithm related to GANs in the following Section III.

III. Main Results

A. GANs Based RL

In order to solve the adaptive optimal control problem for nonlinear system (1), we present a GANs-based RL method, shown in Fig. 2.

The GANs-based RL method is to learn a generator function $G(x_t)$ which specifies the current policy by mapping states x_t to a specific control policy $u(x_t) = h(x_t)$ and a discriminator function $D(x_t, h(x_t))$ which can be learned using the Bellman equation with Q-learning. The discriminator function $D(x_t, h(x_t))$ that predicts the expected discounted reward after taking an control action $h(x_t)$ in state x_t is defined as:

$$D(x_t, h(x_t)) = R_t \tag{12}$$

where $R_t = \int_{j=t}^{T} Y^{(j-t)} r(x_j, h(x_j)) d\tau$ is the sum of the discounted future reward with a discounting factor $Y \in [0, 1]$. The recursive Bellman equation is defined as:

$$D(x_t, h(x_t)) = \int_{j=t}^{t+1} Y^{(j-t)} r(x_j, h(x_j) d\tau + D(x_{t+1}, h(x_{t+1})). \tag{13}$$

We consider the $D(x_t, h(x_t))$ is parameterized by θ^Q. It can be optimized by minimizing the following loss:

$$L(\theta^Q) = E(D(x_t, h(x_t)) - y_t)^2 \tag{14}$$

where $y_t = \int_{j=t}^{t+1} Y^{(j-t)} r(x_j, h(x_j) d\tau + D(x_{t+1}, h(x_{t+1}))$. E is the expectation on the probability distribution of the system states and it satisfies the Markov process.

The control policy $h(x_t)$ of the generation is updated by applying the chain rule to equation (12) with respect to the generator network parameters θ^u as:

$$\nabla_{\theta^u} h(x_t) = \nabla_G D(x_t, G(x_t)) \nabla_{\theta^u} G(x_t). \tag{15}$$

Now the optimal controller $u^*(x_t)$ can be obtained through the following policy iteration:

$$\theta^{Q^*} = \arg\min_{\theta^Q} E(y_t - D(x_t, h(x_t)))^2 \tag{16}$$

$$h^*(x_t) = \arg\min_{h(x_t)} E(D(x_t, h(x_t)|\theta^{Q^*}) \tag{17}$$

where $D(x_t, h(x_t))|\theta^{Q^*})$ is $D(x_t, h(x_t)$ under the optimal discriminator function parameters θ^{Q^*} in each iteration and $u^*(x_t) \leftarrow h^*(x_t)$ as iterations increase.

Analogy with GANs, we built the following two-player minimax game with the value function $V(D, G)$ given by:

$$\min_G \max_D V(D,G) = -L(\theta^Q) + E(D(x_t, G(x_t))|\theta^{Q^*}). \tag{18}$$

We regard y_t as the score of target data and $D(x_t, h(x_t))$ as the score of data generated by $G(x_t)$ and update the classifier by maximizing $-L(\theta^Q)$. At the same time, we update $G(x_t)$ by minimizing the classifier's score $D(x_t, G(x_t)|\theta^{Q^*})$ when the discriminator parameters remain the unchanged optimal parameters θ^{Q^*} at each iteration.

In practice, $h(x_t)$ represents a limited control action of the control policy $u(x_t)$ via the generator function $G(x_t)$ and we optimize θ^u instead of $u(x_t)$. The generator function $G(x_t)$ generates a control action function $h(x_t)$ acting on the system environment and the system generates the result of multiple control actions to produce an offline data set Buffer C. The discriminator function $D(x_t, h(x_t))$ approaches the target model by learning and updating the weight parameters θ^Q in the data set Buffer C. The function $G(x_t)$ through the chain reaction to update the weight parameters θ^u and to generate

Algorithm 2 RL-GANs Algorithm

Step 1 Initialization:

Randomly initialize the discriminator function $D(x_t, h(x_t))$ and the generator function $G(x_t)$ with the weights θ^Q and θ^u. Initialize the replay buffer C.

Step 2 Iteration:

Initialize a random process Z for the action exploration.
for $episode = 1, N$ **do**
  Select the control action function $h(x_t) = G(x_t) + Z$.
  Receive the initial observation state x_0.
  Initialize the target functions $D^*(x_t, h(x_t))$ and $G^*(x_t)$ with the weights $\theta^{Q^*} \leftarrow \theta^Q$ and $\theta^{u^*} \leftarrow \theta^u$.
  for $t = 1, T = step$ **do**
    Explore the environment until the system reaches the equilibrium state.
    Execute h_t and observe the reward r_t and the new state x_{t+1}.
    Store the transition $(x_t, h(x_t), r_t, x_{t+1})$ in C.
    Sample a random mini-batch n of $(x_t, h(x_t), r_t, x_{t+1})$ from C.
    Set

$$y_t = r_t + D^*(x_{t+1}, G^*(x_{t+1})). \tag{19}$$

    Update the weights θ^Q by maximizing $-L(\theta^Q)$ as:

$$-L(\theta^Q) = \frac{1}{n}\sum\nolimits_n (y_t - D(x_t, h(x_t)))^2. \tag{20}$$

    Update the weights θ^u using the sampled gradient:

$$\nabla_{\theta^u} G(x_t) = \frac{1}{n}\sum\nolimits_n \nabla_G D(x_t, G(x_t)) \nabla_{\theta^u} G(x_t). \tag{21}$$

    Update the target networks weight: $\theta^{Q^*} \leftarrow \theta^Q$ $\theta^{u^*} \leftarrow \theta^u$
  end for
end for

a more optimized controller $h^*(x_t)$. We can implement our method online by using Algorithm 2, shown in Fig. 3.

B. Stabilizing Strategies

We adopt some stability methods in GANs field to stabilize the weight parameters. We have listed three important methods as follows:

1. We use a trick to prevent gradients from disappearing, that is, label smoothing replaces $0/1$ labels with $\xi/1-\xi$, which guarantees the generator will always have informative gradients.
2. Inspired by GANs, the historical average method adds a drag term to the gradient descent method to punish steps that deviate too far from the parameters of the old average method. This method can effectively prevent the oscillation caused by different targets of the two models.
3. To prevent the generator from collapsing onto a single sample, the mini-batch discrimination extends the role of the discriminator from calculating the value of a single sample to calculating entire mini-batch.

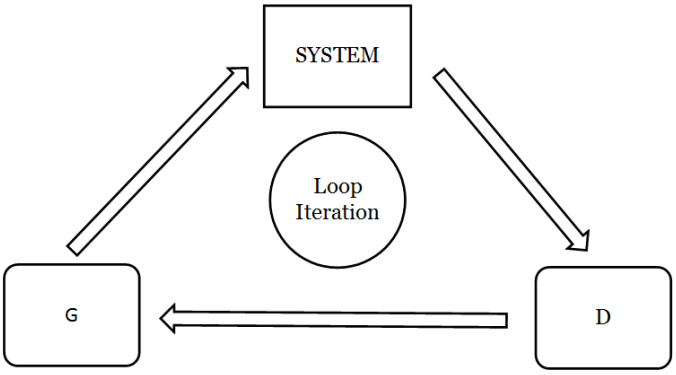

Fig. 3. The training flow chart of the Algorithm 2.

By using the training method in GANs field, we can well make the generator function $G(x_t)$ and the discriminator function $D(x_t, h(x_t))$ to reach stable weight parameters. On the basis of the network parameters stability, we can establish the following convergence proof.

C. Convergence of the GANs-based RL Algorithm

We demonstrate the convergence of our Algorithm 2 by two steps. Firstly, the generator function $G(x_t)$ and the discriminator function $D(x_t)$ of Algorithm 2 can converge to the optimal weight parameters θ^{u^*} and θ^{Q^*} by enough network training in each iteration. Then, the GANs-based RL method can iterate to the optimal control policy $u^*(x_t)$ in the case of the optimal weight parameters θ^{u^*} and θ^{Q^*} of each iteration.

Theorem 1: Suppose the functions $G(x_t)$ and $D(x_t, h(x_t)$ have enough capacity at each iteration of the Algorithm 2. The discriminator function $D(x_t, h(x_t)$ is allowed to reach its optimal $D^*(x_t, h(x_t))$ and the generate function $G(x_t)$ is updated to improve the following criterion:

$$\theta^{u^*} = \arg\min_{\theta^u} D^*(x_t, h(x_t)). \tag{22}$$

Then the policy iteration of Algorithm 2 converges to the optimal weight parameters θ^{u^*} and θ^{Q^*} in each iteration by Algorithm 2.

Proof: Note that $D(x_t, h(x_t))$ is convex in $h(x_t)$. In other words, if $f(x_t) = D_x(x_t, h(x_t))$ and $D_x(x_t, h(x_t))$ is convex in x_t for every $h(x_t)$, then we have $f_h(x_t) = \partial D(x_t, h(x_t))$. It is equivalent to computing a gradient descent update for $h(x_t)$ at the optimal function $D^*(x_t, h(x_t))$ when $f(x_t) = D^*(x_t, h(x_t))$. Therefore, with suffificiently small updates of network weight, we have $\theta^{Q^*} \leftarrow \theta^Q$ and $\theta^{u^*} \leftarrow \theta^u$ in each iteration. This completes the proof.

Theorem 2: $h(x_t)$ in Algorithm 2 converges to the optimal control policy $u^*(x_t)$ with the initial admissible control policy $h^0(x_t)$ when $\theta^{Q^*} \leftarrow \theta^Q$ and $\theta^{u^*} \leftarrow \theta^u$ in each iteration of Algorithm 2.

Proof: According to Theorem 1, we can conclude that in each iteration of Algorithm 2, it will converge to the optimal weight parameters θ^{u^*} that minimize the $D(x_t, h(x_t)|\theta^{Q^*})$ without using knowledge on the internal dynamics of system (1) in each iteration. In [18], it was shown that using the

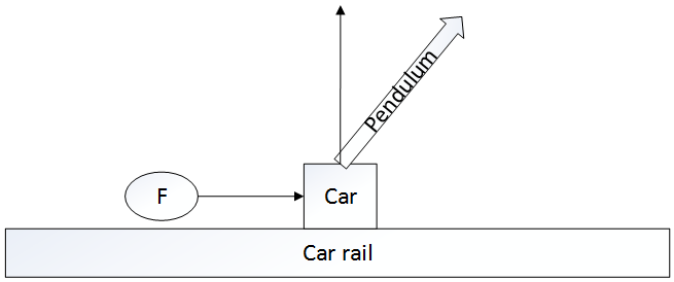

Fig. 4. Model diagram of inverted pendulum device.

policy iteration conditioned by an initial admissible control policy $h^0(x_t)$, all the subsequent control policies $h^i(x_t)$ will be admissible. Consider that $D(x_t, h(x_t)|\theta^{Q^*})$ and $h(x_t)$ in Algorithm 2 which stand for $Q^*(x_t, u(x_t))$ and $u(x_t)$ in Q-learing Algorithm. Thus, the policy iteration of (16) and (17) will converge to the solution of the HJB equation (4) without using knowledge on the internal dynamics of system (1) when $\theta^{Q^*} \leftarrow \theta^Q$ and $\theta^{u^*} \leftarrow \theta^u$ in each iteration. Ultimately, it follows that $u(x_t) = h(x_t)$ converges to the optimal control policy $u^*(x_t)$ by policy iteration and it concludes the proof.

In order to demonstrate the effectiveness of Algorithm 2, we illustrate by the following section IV. Meanwhile, we also compare the existing AC algorithm to illustrate the superiority of our designed algorithm.

IV. Simulation Experiments

To show the feasibilty and applicability of our designed methods, we give a simulation example related to the inverted pendulum, shown in Fig 4. The nonlinear model of a inverted pendulum device [19] is expressed as:

$$\begin{cases} (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}\sin\theta = F \\ (I+ml^2)\ddot{\theta} = mgl\sin\theta - ml\ddot{x}\cos\theta \end{cases} . \quad (23)$$

where the variables of the inverted pendulum device are described in Table 1.

TABLE I
MODEL PARAMETERS

Parameters	Description
M	Car's Mass
m	Mass of the pendulum rod
l	Length to the center of rod mass
b	Friction of the cart
I	Inertia of the rod
F	Force applied to the cart
x	Car position
g	Acceleration due to garvity

We construct our actual nonlinear system by giving some concrete values. For simplicity and repeatability of the simulation experiment, we give the relevant parameter settings.

The angle θ is between the inverted pendulum and the vertical direction. The random angle θ is between $-\pi$ and π. The system observations are $\dot{\theta}$, $\sin(\theta)$ and $\cos(\theta)$.

The action represents the effect of the controller on the inverted pendulum which is equivalent to F in the Fig 4. The size of F ranges from -1.0 to 1.0.

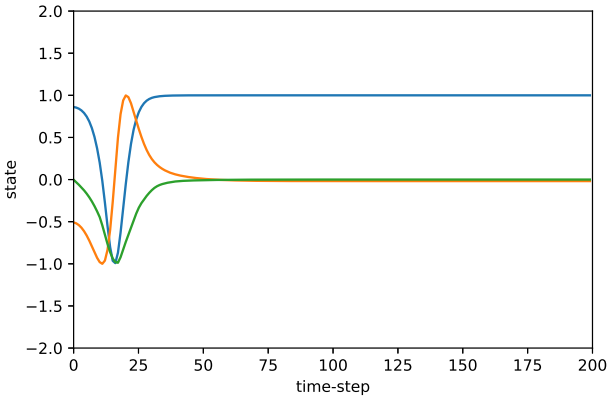

Fig. 5. The figure shows the state curve of the inverted pendulum. The blue line is $\cos(\theta)$, the orange line is $\sin(\theta)$ and the green line is $\dot{\theta}$. The system states converge to the target states under the action of the controller.

The exact cost equation is expressed as:

$$r(t) = \theta^2 + 0.1 * \dot{\theta}^2 + 0.001 * F^2 \quad (24)$$

where θ is normalized between $-\pi$ and π. Thus the maximum cost is $(\pi^2 + 0.1 * 1^2 + 0.001 * 1^2)$ and the minimum cost is 0. Essentially, the goal is to maintain zero angle, minimize rotation speed and minimize force.

The number of the training iterations is set as 1000. The maximum value of the controller exploration steps is set as 10,000. The controller keeps taking actions in the process of each iteration until the system reaches a balance point.

The discriminator network has an input layer, two hidden layers and an output layer. The generator network has the same network structure as the discriminator network. The dimensions of the two hidden layers are respectively 400 and 300.

The mass of the car is 1.0, the length of the inverted pendulum is 1.0 and the friction of the car is 0.0.

Fig. 5 shows that the designed RL-GANs algorithm can achieve a stable target state. We can observe from Fig. 6 that the system states tend to be stabilizable by the designed optimal controller after 75 time steps. Fig. 7 shows that the RL-GANs algorithm can converge to an optimal cost and reach the optimal control after 100 iterations. We compare our results with the general AC algorithm in Fig. 8 and find that our RL-GANs algorithm converges faster and shows more stable. In addtion, the test cost of the AC algorithm has multiple crests and is instability in the training iterations.

V. Conclusion

Inspired by the RL techniques, the adaptive optimal control problem is studied in this paper. We propose a new framework to successfully learn the optimal controller design of continuous-time unknown nonlinear system via GANs. Our experiments show that the designed RL-GANs algorithm can improve the traditional RL algorithm and show good performance features. The simulation example related to the

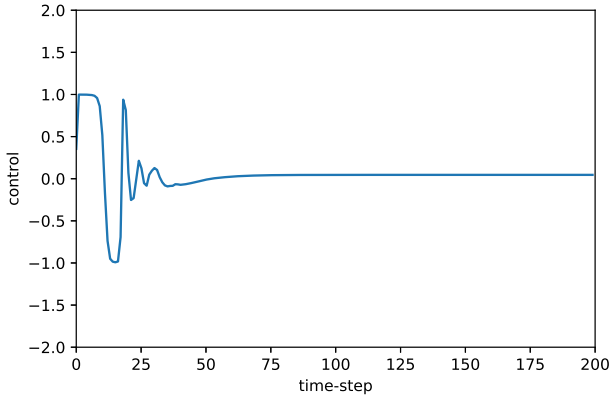

Fig. 6. The figure shows the control process of the inverted pendulum.

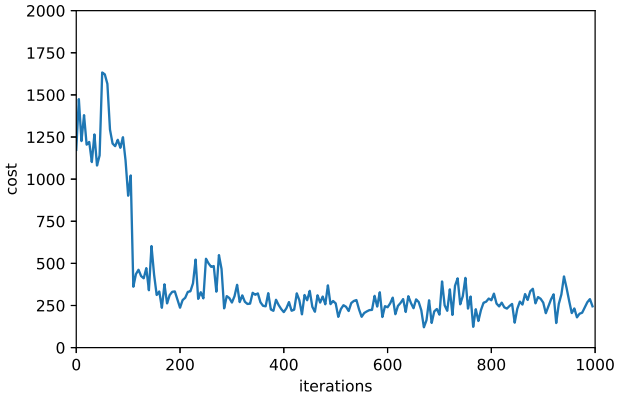

Fig. 7. The figure shows that the test cost value of Algorithm 2. There is fluctuation behind the cost value. its volatility could be offset in the environment of extremely sparse returns.

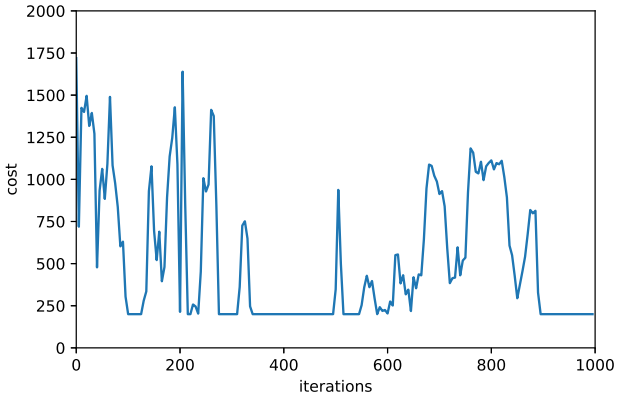

Fig. 8. The figure shows that the test cost value of AC algorithm. The convergence rate is slow and the graph fluctuates greatly.

inverted pendulum show the feasibility and applicability of our designed methods.

ACKNOWLEDGEMENT

This work was supported in part by the National Natural Science Foundation of P. R. China under Grant 61203051, the Key Support Program for University Outstanding Youth Talent of Anhui Province under Grant gxydZD2017001 and the open fund for Discipline Construction, Institute of Physical Science and Information Technology, Anhui University.

REFERENCES

[1] M. Abu-Khalaf, F. Lewis, "Nearly Optimal Control Laws for Nonlinear Systems with Saturating Actuators Using a Neural Network HJB Approac" , *Automatica*, vol. 41, no. 5, pp. 779-791, 2005.

[2] R. Beard, G. Saridis, j. Wen, "Galerkin approximations of the generalized Hamilton-Jacobi-Bellman equation", *Automatica*, vol.33, no. 12, pp.2159-2177, 1997.

[3] J. Murray, C. Cox, G. Lendaris, and R. Saeks, "Adaptive Dynamic Programming", IEEE Trans. on Systems, Man and Cybernetics, vol. 32, no. 2, pp 140-153, 2002.

[4] R. A. Howard, "Dynamic Programming and Markov Processes", *MIT Press*, Cambridge, Massachusetts, 1960.

[5] X. Yang, H. He, D. Liu and Y. Zhu,"Adaptive dynamic programming for robust neural control of unknown continuous-time non-linear systems", *IET Control Theory and Applications*, vol. 11, no. 14, pp. 2307–2316, Sep. 2017.

[6] X. Zhong, H. He, H. Zhang and Z. Wang, "Optimal control for unknown discrete-time nonlinear Markov jump systems using adaptive dynamic programming", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 12, pp. 2141–2155, Dec. 2014.

[7] F. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control", *IEEE Circuits and Systems Magazine*, vol. 9, no. 3, pp. 32–50, Aug. 2009.

[8] W. Powell, "Approximate Dynamic Programming: Solving The Curses of Dimensionality", *Optimization Methods and Software*, vol. 24, no. 1, Feb. 2007.

[9] D. Vrabie, O. Pastravanu, M. Khalaf and F. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration", *Automatica*, vol. 45, no. 2, pp. 477-484, Feb. 2009.

[10] Y. Jiang and Z. Jiang, "Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics", *Automatica*, vol. 48, no. 10, pp. 2699–2704, Oct. 2012.

[11] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 621–634, Mar. 2014.

[12] B. Luo, T. Huang and D. Liu, "Data-based approximate policy iteration for affine nonlinear continuous-time optimal control design", *Automatica*, vol. 50, no. 12, pp. 3281–3290, Dec. 2014.

[13] J. Song, S. He, F. Liu, Y. Niu and Z. Ding, "Data-driven policy iteration algorithm for optimal control of continuous-time to stochastic systems with Markovian jumps", *IET Control Theory and Applications*, vol. 10, no. 12, pp. 1431–1439, Aug. 2016.

[14] H. Zhang, H. Liang, Z. Wang and T. Feng, "Optimal output regulation for heterogeneous multiagent systems via adaptive dynamic programming", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 1, pp. 18–29, Oct. 2017.

[15] D. Wang, J. Qiao, "Approximate neural optimal control with reinforcement learning for a torsional pendulum device ", *NEURAL NETWORKS* , vol. 117, pp. 1–7, Sep. 2019.

[16] B. Luo, H. Wu, T .Huang, "Reinforcement learning solution for HJB equation arising in constrained optimal control problem ", *NEURAL NETWORKS*, vol. 71, pp. 150-158, Nov. 2015.

[17] H. Hachiya, T. Akiyama, "Adaptive importance sampling for value function approximation in off-policy reinforcement learning ", *NEURAL NETWORKS*, vol. 22, no. 10, pp. 1399-1410, Dec. 2009.

[18] R. Beard, G. Saridis, and J. Wen, "Galerkin approximations of the generalized Hamilton–Jacobi–Bellman equation", *Automatica*, vol. 33, pp. 2159–2177, 1997.

[19] Y. Zhou and Z. Wang, "Optimal Feedback Control for Linear Systems with Input Delays Revisited", *Journal of Optimization Theory and Applications*, vol. 163, no. 3, pp. 989-1017, 2014.