2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Policy Iteration Approximate Dynamic Programming Using Volterra Series Based Actor

Wentao Guo, *Student Member, IEEE,* Jennie Si, *Fellow, IEEE,* Feng Liu, *Member, IEEE,* and Shengwei Mei, *Senior Member, IEEE*

***Abstract*—There is an extensive literature on value function approximation for approximate dynamic programming (ADP). Multilayer perceptrons (MLPs) and radial basis functions (RBFs), among others, are typical approximators for value functions in ADP. Similar approaches have been taken for policy approximation. In this paper, we propose a new Volterra series based structure for actor approximation in ADP. The Volterra approximator is linear in parameters with global optima attainable. Given the proposed approximator structures, we further develop a policy iteration framework under which a gradient descent training algorithm for obtaining the optimal Volterra kernels can be obtained. Associated with this ADP design, we provide a sufficient condition based on actor approximation error to guarantee convergence of the value function iterations. A finite bound of the final convergent value function is also given. Finally, by using a simulation example we illustrate the effectiveness of the proposed Volterra actor for optimal control of a nonlinear system.**

I. Introduction

APPROXIMATE dynamic programming (ADP) [1], [2], [3] is a powerful tool to solve for the optimal policy of a multistage decision process problem. As is well known, the "curse of dimensionality" [4] greatly hinders the application of classical dynamic programming in large-scale problems. To circumvent this problem, ADP uses function approximation structures such as neural networks to approximate both the value and the policy functions. In the past decades, several ADP algorithms have been developed [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]. For recent results and reviews on ADP, readers can refer to [1], [2], [3].

In control theoretic terms, ADP opens new opportunities to approximately solve the Hamilton-Jacobi-Bellman (HJB) equation to obtain optimal control. Value iteration and policy iteration are two important ADP approaches. The convergence of a value iteration based heuristic dynamic programming was provided in [6]. The idea of policy iteration [15] is to improve the initial admissible control policy by iterative policy evaluation and policy improvement. Policy iteration is used to obtain the optimal control policy of continuous-time systems and discrete-time systems in [7] and [16], respectively. Furthermore, policy iteration based ADP for continuous-time systems with input constraint is studied in [8]. A novel ADP algorithm for unknown constrained-input systems is proposed in [17] using policy iteration. Some important and unique properties are associated with policy iteration. They include non-increasing and convergent value functions, and stable iterative control policies. These properties have made policy iteration a good candidate for improved control system performance measured by the value function as long as the initial policy is admissible.

Among some early developments of ADP algorithms, an intuitive idea is to place state and control variables into a discretized grid space and thus a look-up table can be used to map states to controls. This approach is apparently not scalable due to its exponentially increasing demand on the storage space and computation burden. Approximate approaches such as actor-critic methods [18] were then proposed that has held great promise. In the actor-critic framework, function approximation structures, such as multi-layer perceptron (MLP) networks [5], [19] and radial basis function (RBF) networks [19] among others, are introduced to approximate the value function and the policy. By learning through examples, the inherent structures of the value and the policy functions can be obtained and approximated. These approaches effectively circumvent the dimension explosion caused by large scale state and control problems.

The problem of value function approximation (VFA), or critic approximation, has been intensively studied in the ADP community. An excellent survey on algorithms of parametric VFA can be found in [20]. Convergence proof as well as approximation error bound of VFA in temporal-difference (TD) learning is presented in [21]. Recently, much attention has been paid to improve the generalization capability and learning efficiency of VFA. Support vector machine (SVM) [22] and sparse kernel machine [23], are introduced into ADP for efficient selection of VFA basis functions.

On the other hand, policy approximation, or actor approximation, has not received the same attention as the value approximation. Usually, artificial neural networks (ANNs) such as MLP networks and RBF networks are routinely used in actor approximation. In [19], it is shown heuristically by simulations that HDP with RBF networks outperforms HDP with MLP networks in the synchronous generator control. Some problems may arise when using a generic approximator such as MLP or RBF. For MLP, one may encounter local minima in network training and thus cannot guarantee solution

This work was supported in part by the National Natural Science Foundation of China (No. 51377092, No. 51321005), the Major Projects on Planning and Operation Control of Large Scale Grid (SGCC-MPLG017-2012) of the State Grid Corporation of China, and the National High Technology Research and Development of China (863 Program, 2012AA050204).

W. Guo, F. Liu, and S. Mei are with State Key Laboratory of Power Systems, Department of Electrical Engineering, Tsinghua University, Beijing 100084, China (e-mail: gwt329@gmail.com, lfeng@tsinghua.edu.cn, meishengwei@tsinghua.edu.cn).

J. Si is with Department of Electrical Engineering, Arizona State University, Tempe, AZ 85287, USA (e-mail: si@asu.edu).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

optimality [21]. According to [20], a learning algorithm may diverge when a nonlinear function approximator is employed. When a RBF is used as generic approximators in ADP, it typically requires a large number of basis to achieve good approximation [24]. This in turn creates a curse of dimensionality of its own. There have been some studies to cleverly place appropriate basis with appropriate parameters. But a cookbook type design approach still is lacking. This motivates our research to seek a new mathematic model for actor approximation.

As a universal approximator, Volterra series [25] is a potential candidate for actor approximation. Volterra series is a powerful tool in nonlinear system analysis. It takes into account historical data, or it is a system with memory, to approximate a nonlinear dynamic mapping between inputs and outputs. Under some mild conditions, a continuous function can be approximated by a Volterra series [26]. Volterra series is a linear summation of a series of terms from the first to high-order mixed product terms of historical inputs. The linear-in-parameter property of Volterra series makes globally optimal parameter search attainable.

This paper focuses on discussing policy approximation or actor approximation, in the framework of policy iteration. Volterra series based actor is developed to approximate a control policy in a multi-input-multi-output (MIMO) setting. Gradient descent training algorithm is provided to search for the optimal Volterra kernel.

Another main contribution of this paper is the convergence analysis of policy iteration algorithm under inevitable but finite actor approximation error. By making use of an error bound idea [27], a sufficient condition on actor approximation error is provided to guarantee the convergence of iterative value function. At the same time, a finite bound of the convergent value function is given.

The rest of this paper is organized as follows. In Section II, the discrete-time nonlinear optimal control problem is formulated in mathematical terms. The classic policy iteration algorithm is also presented. In Section III, we present the proposed Volterra series based actor and the implementation of policy iteration algorithm using Volterra actor. In Section IV, the convergence of policy iteration algorithm with actor approximation error is analyzed. In Section V, simulation studies are made on a nonlinear system. Conclusions are drawn in Section VI.

II. Problem Formulation

Consider a discrete-time nonlinear system of the form,

$$x_{k+1} = F(x_k, u_k), k = 0, 1, 2, \cdots, \tag{1}$$

where $x_k \in \mathbb{R}^n$ is the state vector, $u_k \in \mathbb{R}^m$ is the control vector, $F(x_k, u_k) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the system state transformation function.

For system (1), we define an instantaneous cost function at time k as

$$U(x_k, u_k) = Q(x_k) + u_k^T R u_k, k = 0, 1, 2, \cdots, \tag{2}$$

where $Q(x_k) : \mathbb{R}^n \to \mathbb{R}$ is a positive definite function of x_k and $R \in \mathbb{R}^{m\times m}$ is a positive definite matrix. Total cost function or value function from the state x_k under the control policy $u_k = u(x_k)$ is defined as

$$V^u(x_k) = \sum_{i=k}^{\infty} U(x_i, u(x_i)), \tag{3}$$

where $u(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ represents a control policy.

The objective of optimal control is to find a control policy, which can stabilize system (1) and minimize value function (3). Optimal value function can be represented by

$$V^*(x_k) = \min_u V^u(x_k). \tag{4}$$

According to Bellman's optimality principle [28], we have the following discrete-time HJB equation,

$$V^*(x_k) = \min_{u_k}\{U(x_k, u_k) + V^*(x_{k+1})\}. \tag{5}$$

The optimal control policy is computed from

$$u^*(x_k) = \arg\min_{u_k}\{U(x_k, u_k) + V^*(x_{k+1})\}. \tag{6}$$

Generally speaking, the HJB equation (5) cannot be analytically solved. As an online learning approach, reinforcement learning has been extensively used to solve the aforementioned optimal control problem while bypassing directly solving the HJB equation [1], [2]. Usually, an actor-critic structure is employed for the reinforcement learning implementation, which is illustrated in Fig. 1. The actor provides control input for the controlled system. The critic evaluates the control input.

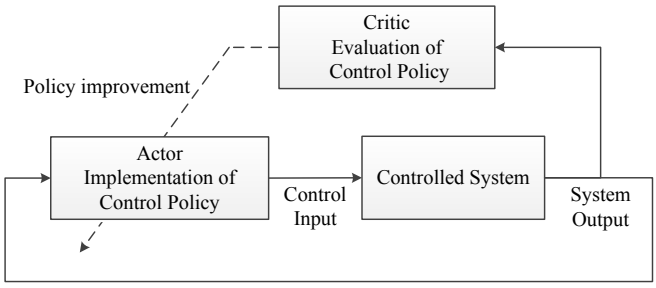

Fig. 1. Actor-critic structure for reinforcement learning

In this paper, policy iteration [15] is employed as the learning algorithm for the actor-critic structure in Fig. 1. The algorithm starts from an admissible control policy [29], i.e. the actor is initialized to be admissible.

Definition 1. (Admissible Control) A control policy $u_k = u(x_k)$ is admissible with respect to value function (3) on $\mathbb{R}^n$, if $u_k = u(x_k)$ is continuous on $\mathbb{R}^n$, $u(0) = 0$, $u_k = u(x_k)$ stabilizes system (1) and the corresponding value function (3) is finite for $\forall x_k \in \mathbb{R}^n$.

The policy iteration algorithm is an iterative process between two interleaving steps [30]:

Policy evaluation:

$$\bar{V}^{(i)}(x_k) = U(x_k, \bar{u}^{(i)}(x_k)) + \bar{V}^{(i)}(x_{k+1}), i = 0, 1, 2, \cdots. \tag{7}$$

In the above equation, i is the iteration number; $\bar{u}^{(i)}(x_k)$ is the control policy in the ith iteration; $\bar{V}^{(i)}(x_k)$ is a value function of the form (3) corresponding to the control policy $u_k = \bar{u}^{(i)}(x_k)$, and $\bar{V}^{(i)}(x_k)$ is a positive definite function of

x_k. Note that in the first iteration step $i = 0$, the control policy $u_k = \bar{u}^{(0)}(x_k)$ is initialized to be an admissible control policy of system (1) under value function (3).

Policy improvement:

$$\bar{u}^{(i+1)}(x_k) = \arg\min_{u_k} \{U(x_k, u_k) + \bar{V}^{(i)}(x_{k+1})\}. \quad (8)$$

Based on the necessary condition of optimal control from policy improvement (8), we have the following,

$$\frac{\partial[U(x_k, u_k) + \bar{V}^{(i)}(x_{k+1})]}{\partial u_k} = 0. \quad (9)$$

By substituting (1) and (2) into (9), a control policy can be obtained as

$$u_k = -\frac{1}{2} R^{-1} (\frac{\partial F(x_k, u_k)}{\partial u_k})^T \nabla \bar{V}^{(i)}(F(x_k, u_k)). \quad (10)$$

Note that (10) is implicit in the control u_k. Hence, a function approximation structure is introduced to represent the control policy as shown in (10),

$$u_k = \bar{u}^{(i+1)}(x_k) = S(K, x_k, x_{k-1}, x_{k-2}, \cdots), \quad (11)$$

where S is a function approximation structure and K is the parameter vector of S. For linear systems, actor (11) can be easily selected as linear state feedback. But for general nonlinear systems, how to construct actor (11) may not be straightforward. MLP network and RBF network have been proposed to serve as an actor approximator [5], [9], [19]. Their limitations were discussed in the previous section. Different from those ideas, Volterra series is employed for actor approximation in this paper.

III. Policy Iteration With Volterra Series Based Actor

A. Volterra series based actor

Under some mild conditions, a scalar function with multiple inputs can be approximated by a Volterra series within any desired accuracy [26].

The nonlinear controller approximation for system (1) is a multi-input-multi-output (MIMO) problem. However, the Volterra series is a multi-input-single-output (MISO) approximator. As was in [31], we first decompose the original MIMO problem into several MISO problems. Then, we employ one Volterra series for each component of the control vector.

Taking a scalar-output controller for example, the Volterra series based actor is of the following infinite series form

$$\begin{aligned} u_k = & \sum_{i=1}^{n} \sum_{\tau=0}^{M} k_1^{(i)}(\tau) x_{k-\tau}(i) \\ & + \sum_{i_1=1}^{n} \sum_{i_2=1}^{i_1} \sum_{\tau_1=0}^{M} \sum_{\tau_2=0}^{M} k_2^{(i_1,i_2)}(\tau_1, \tau_2) x_{k-\tau_1}(i_1) x_{k-\tau_2}(i_2) \\ & + \sum_{i_1=1}^{n} \sum_{i_2=1}^{i_1} \sum_{i_3=1}^{i_2} \sum_{\tau_1=0}^{M} \sum_{\tau_2=0}^{M} \sum_{\tau_3=0}^{M} k_3^{(i_1,i_2,i_3)}(\tau_1, \tau_2, \tau_3) \\ & x_{k-\tau_1}(i_1) x_{k-\tau_2}(i_2) x_{k-\tau_3}(i_3) + \cdots, \end{aligned} \quad (12)$$

where $x_k \in \mathbb{R}^n$, M is memory length, $x_{k-\tau}(i)$ is the ith component of x at time $k - \tau$, $k_1^{(i)}(\tau)$, $k_2^{(i_1,i_2)}(\tau_1, \tau_2)$, and $k_3^{(i_1,i_2,i_3)}(\tau_1, \tau_2, \tau_3)$ are the 1st, 2nd, and 3rd order Volterra coefficients or Volterra kernels, respectively. Note that state feedback control is a special case of the 1st order Volterra actor and can be directly represented by Volterra actor (12).

B. Policy iteration using Volterra series based actor

By using Volterra series (12) as actor, the policy iteration algorithm in (7) and (8) becomes

$$V^{(i)}(x_k) = U(x_k, u^{(i)}(x_k)) + V^{(i)}(x_{k+1}), i = 0, 1, 2, \cdots, \quad (13)$$

$$u^{(i+1)}(x_k) = \arg\min_{u_k \in \mathscr{V}} [U(x_k, u_k) + V^{(i)}(x_{k+1})], \quad (14)$$

where $V^{(i)}(x_k)$ is the value function of the form (3) corresponding to the control policy $u_k = u^{(i)}(x_k)$, and it is positive definite, $\mathscr{V}$ is the set of all controllers represented by Volterra series in (12). The difference between the policy improvement formula (14) and (8) is that (14) is an optimization constrained on $\mathscr{V}$, while (8) is an unconstrained optimization.

C. Implementation of policy evaluation

The discussion of this paper is focused on actor approximation. Hence, without any loss of generality, the following linear value function approximation structure (15) is used for implementation of policy evaluation (13),

$$V(x_k) = \sum_{i=1}^{L} w_i \varphi_i(x_k) = W^T \phi(x_k), \quad (15)$$

where $W \in \mathbb{R}^L$ is the weight vector and $\phi(x_k) \in \mathbb{R}^L$ is the basis function vector, which can be polynomial function, sigmoid function, radial basis function, to name a few. For linear systems, $\phi(x_k)$ can be selected as quadratic functions of state variables.

Based on value function approximation (15), policy evaluation (13) becomes

$$W^{(i)T}(\phi(x_k) - \phi(x_{k+1})) = U(x_k, u^{(i)}(x_k)), \quad (16)$$

where $W^{(i)}$ is the parameter of value function $V^{(i)}(x_k)$. (16) can be solved by batch least square. For that purpose, we first collect N sample data pairs, $\phi(x_k) - \phi(x_{k+1})$ and $U(x_k, u^{(i)}(x_k))$, for k, $k+1$, $\cdots$, $k+N$, where typically $N \geq L$, align them in rows and denote as $\psi(x_k) \in \mathbb{R}^{L \times N}$ and $\mu(x_k, u^{(i)}(x_k)) \in \mathbb{R}^{1 \times N}$, respectively. Then $W^{(i)}$ can be obtained as

$$W^{(i)} = [\psi(x_k)^\dagger]^T \mu^T(x_k, u^{(i)}(x_k)), \quad (17)$$

where $\psi(x_k)^\dagger$ is the pseudo inverse or Moore-Penrose inverse [32] of $\psi(x_k)$. (16) can also be solved by recursive least square (RLS) method [20].

D. Implementation of policy improvement

Volterra series based control policy (12) can be rewritten as

$$u_k = u(x_k) = K^T \sigma(x_k, x_{k-1}, \cdots, x_{k-M}), \quad (18)$$

where K and $\sigma(x_k, x_{k-1}, \cdots, x_{k-M})$ are the kernel vector and the basis function vector of Volterra series, respectively. Since the structure of the Volterra control policy, i.e.

$\sigma(x_k, x_{k-1}, \cdots, x_{k-M})$ is fixed, a Volterra control policy is determined by its kernels. The goal of policy improvement is essentially to solve the optimal Volterra kernels.

In this paper, the optimal Volterra kernels are obtained by gradient decent method

$$\begin{aligned} K_{j+1}^{(i+1)} &= K_j^{(i+1)} + \Delta K \\ &= K_j^{(i+1)} - l\frac{\partial[U(x_k,u_k)+V^{(i)}(x_{k+1})]}{\partial K} \\ &= K_j^{(i+1)} - l\sigma(x_k, x_{k-1}, \cdots, x_{k-M}) \\ &\quad [2Ru_k + [\frac{\partial F(x_k,u_k)}{\partial u_k}]^T \nabla\phi^T(x_{k+1})W^{(i)}]^T \end{aligned} \tag{19}$$

where K and $\sigma(x_k, x_{k-1}, \cdots, x_{k-M})$ are the kernel vector and the basis function vector in (18), l is the learning rate for K, and index j means the jth policy update in the $(i+1)$th policy improvement.

Remark 1. If we consider a nonlinear affine system, i.e., $x_{k+1} = f(x_k) + g(x_k)u_k$, then $\frac{\partial F(x_k,u_k)}{\partial u_k} = g(x_k)$. Then, similar to [2], policy improvement formula (19) only requires partial dynamics of the system, i.e. $g(x_k)$. The policy iteration algorithm using Volterra actor can be partially model-free.

Remark 2. In the above content, the instantaneous cost function (2) is considered, which is quadratic in the control u_k. If we consider a general instantaneous cost function defined as

$$U(x_k, u_k) = Q(x_k) + R(u_k), k = 0, 1, 2, \cdots, \tag{20}$$

where $R(u_k) : \mathbb{R}^m \to \mathbb{R}$ is a positive definite function. The proposed Volterra series based actor is still applicable for such general instantaneous cost function. The Volterra kernels update formula in (19) is modified to

$$\begin{aligned} K_{j+1}^{(i+1)} &= K_j^{(i+1)} - l\sigma(x_k, x_{k-1}, \cdots, x_{k-M}) \\ &\quad [\frac{\partial R(u_k)}{\partial u_k} + [\frac{\partial F(x_k,u_k)}{\partial u_k}]^T \nabla\phi^T(x_{k+1})W^{(i)}]^T \end{aligned} \tag{21}$$

In the $(i+1)$th policy improvement step, updates of (19) continue until $max(|\Delta K|)$ is less than a specified tolerance or the maximum update number is reached.

IV. Convergence Analysis

As discussed above, the difference between the policy iteration with Volterra actor (13) and (14) and the regular policy iteration of (7) and (8) is that the policy improvement (14) is an optimization constrained on all the Volterra series represented control policies, while the policy improvement (8) is an unconstrained optimization on all control policies. In practical implementation, Volterra series is usually of finite order, and training error is inevitable. Hence, there is an approximation error for Volterra actor in each policy improvement step even though Volterra series is a universal approximator. It is well known that the policy iteration converges to the optimal value function. However, when conducting the policy improvement on $\mathscr{V}$, there is an approximation error compared to the unconstrained optimization. Such approximation error exists in every iteration step and may be accumulated and amplified through iterations. Taking this into account, convergence of the policy iteration with actor approximation is questionable. In this section, we study the convergence of policy iteration under actor approximation error by using an error bound approach [27].

Before we proceed, the following assumptions are necessary.

Assumption 1. System (1) is controllable and system dynamics $F(x_k, u_k)$ is Lipschitz continuous for $\forall x_k, u_k$.

Assumption 2. Control policy $u_k = u(x_k)$ satisfies that $u_k = 0$ for $x_k = 0$. State $x_k = 0$ is an equilibrium of system (1) under the control $u_k = 0$, i.e. $F(0, 0) = 0$.

Assumption 3. There exists a compact set of Volterra kernels, Ω_K, s.t., for $\forall K \in \Omega_K$, the corresponding Volterra controller (12) is a stable controller for system (1).

For the ease of discussion, we define a new iterative control policy as

$$\upsilon^{(i+1)}(x_k) = \arg\min_{u_k}[U(x_k, u_k) + V^{(i)}(x_{k+1})], \tag{22}$$

where $V^{(i)}(x_k)$ is the value function of the Volterra control policy $u^{(i)}$ defined in (13). Compare (22) to the Volterra actor based policy improvement (14), it can be noted that $\upsilon^{(i+1)}(x_k)$ is the unconstrained optimal control policy in the $(i+1)$th iteration while $u^{(i+1)}(x_k)$ is the optimal Volterra control policy. Define the value function of $\upsilon^{(i+1)}(x_k)$ as

$$\Lambda^{(i+1)}(x_k) = U(x_k, \upsilon^{(i+1)}(x_k)) + \Lambda^{(i+1)}(x_{k+1}), i = 0, 1, 2, \cdots, \tag{23}$$

where $\Lambda^{(i+1)}(x_k)$ is a positive definite function of x_k. Since $\upsilon^{(i+1)}(x_k)$ is better than $u^{(i+1)}(x_k)$, one has

$$V^{(i+1)}(x_k) \geq \Lambda^{(i+1)}(x_k) \tag{24}$$

where $V^{(i+1)}(x_k)$ is defined in (13). Assume that the approximation error of Volterra actor in a single iteration is finite, i.e.

$$\Lambda^{(i+1)}(x_k) \leq V^{(i+1)}(x_k) \leq \eta\Lambda^{(i+1)}(x_k) \tag{25}$$

holds uniformly, where $\eta \geq 1$.

The relationship between the iterative value function and the optimal value function is presented in the following theorem.

Theorem 1. Let Assumptions 1-3 and (25) hold. Let $V^{(i)}(x_k)$ be defined by (13) and $\bar{V}^{(i)}(x_k)$ be defined by (7). If there exist $0 < \gamma < \infty$ and $1 \leq \rho < \infty$ that make

$$V^*(F(x_k, u_k)) \leq \gamma U(x_k, u_k), \tag{26}$$

$$V^*(x_k) \leq \bar{V}^{(0)}(x_k) \leq \rho V^*(x_k), \tag{27}$$

hold uniformly for $\forall k$, where $V^*(x_k)$ is the optimal value function satisfying the discrete-time HJB equation (4), then

$$V^{(i)}(x_k) \leq \{\rho(\frac{\eta\gamma}{\gamma+1})^i + (1-(\frac{\eta\gamma}{\gamma+1})^i)\frac{\eta}{1-\gamma(\eta-1)}\}V^*(x_k). \tag{28}$$

Proof. The theorem is proved by mathematical induction. Due to space limitations, we only sketch the proof of the theorem.

First, for $i = 0$, $u^{(0)}(x_k)$ and $\bar{u}^{(0)}(x_k)$ are initialized as the same admissible control policy. According to the condition

(27), there is $V^{(0)}(x_k) = \bar{V}^{(0)}(x_k) \leq \rho V^*(x_k)$. Theorem 1 holds for $i = 0$.

Then, by assuming that (28) holds for i and using equations (22), (23), (25), (26), we can obtain that (28) holds for $i + 1$. □

Based on Theorem 1, a sufficient condition for convergence, as well as a bound for final convergent value function, is given in the following theorem.

Theorem 2. Let Assumptions 1-3 and (25) hold, and there exist $0 < \gamma < \infty$ and $1 \leq \rho < \infty$ that make (26) and (27) hold uniformly, respectively. Let the iterative value function $V^{(i)}(x_k)$ be defined by (13). If the following condition is satisfied,

$$1 \leq \eta < \frac{\gamma + 1}{\gamma}, \tag{29}$$

then as $i \to \infty$, $V^{(i)}(x_k)$ converges to a bounded neighborhood of the optimal value function $V^*(x_k)$, i.e.,

$$\lim_{i \to \infty} V^{(i)}(x_k) = V^{(\infty)}(x_k) \leq \frac{\eta}{1 - \gamma(\eta - 1)} V^*(x_k). \tag{30}$$

Proof. For (28), if the condition (29) holds, (30) can be easily obtained as $i \to \infty$. □

Next, we are ready to present value function convergence result under Volterra actor approximation error.

Theorem 3. Let Assumptions 1-3 and (25) hold, and there exist $0 < \gamma < \infty$ and $1 \leq \rho < \infty$ that make (26) and (27) hold uniformly, respectively. Let the iterative value function $V^{(i)}(x_k)$ be defined by (13). Let $\delta u^{(i+1)}(x_k)$ denote the function approximation error of Volterra control policy $u^{(i+1)}(x_k)$ with respect to $v^{(i+1)}(x_k)$, i.e.,

$$u^{(i+1)}(x_k) = v^{(i+1)}(x_k) + \delta u^{(i+1)}(x_k). \tag{31}$$

If the following condition holds,

$$\|\delta u^{(i+1)}(x_k)\| < \frac{\Lambda^{(i+1)}(x_k)}{\gamma \| \frac{\delta \Lambda^{(i+1)}(x_k)}{\delta v^{(i+1)}(x_k)} \|}, \tag{32}$$

where $\frac{\delta \Lambda^{(i+1)}(x_k)}{\delta v^{(i+1)}(x_k)}$ is the functional derivative of the value function $\Lambda^{(i+1)}(x_k)$ defined in (23) with respect to the control policy $v^{(i+1)}(x_k)$, $\|\cdot\|$ is the function norm, then as $i \to \infty$, $V^{(i)}(x_k)$ converges to a bounded neighborhood of the unconstrained optimal value function $V^*(x_k)$, i.e.

$$\lim_{i \to \infty} V^{(i)}(x_k) = V^{(\infty)}(x_k) \leq \frac{\eta}{1 - \gamma(\eta - 1)} V^*(x_k). \tag{33}$$

Proof. By the proposition of functional derivative, we have

$$\begin{aligned} V^{(i+1)}(x_k) &= \Lambda^{(i+1)}(x_k) + [\frac{\delta \Lambda^{(i+1)}(x_k)}{\delta v^{(i+1)}(x_k)}]^T \delta u^{(i+1)}(x_k) \\ &\leq \Lambda^{(i+1)}(x_k) + \| \frac{\delta \Lambda^{(i+1)}(x_k)}{\delta v^{(i+1)}(x_k)} \| \| \delta u^{(i+1)}(x_k) \|. \end{aligned} \tag{34}$$

Then by substituting (32) into (34) and using Theorem 2, (33) can be obtained. □

V. Simulation Study

In practical engineering systems such as chemical engineering processes, a 2nd order Volterra series has been frequently used and usually suffices for control purposes [31], [33], [34], [35]. Given such considerations, a 2nd order Volterra series is used as the actor of policy iteration algorithm in the following simulations.

The simulation study is carried out on system (35) from [27], [36], a nonaffine nonlinear system with single state and single control input,

$$x_{k+1} = F(x_k, u_k) = x_k + \sin(0.1x_k^2 + u_k). \tag{35}$$

Since $\frac{\partial F(x_k, u_k)}{\partial x_k}|_{(0,0)} = 1$, system (35) is marginally stable at the equilibrium $x_k = 0$ and this equilibrium is not an attractive one.

Policy iteration with 2nd order Volterra actor is used to solve the optimal control policy of system (35). The memory length and the learning rate of Volterra series are $M = 5$ and $l = 0.01$, respectively. The numbers of the 1st and the 2nd order Volterra kernels are 6 and 21, respectively. The instantaneous cost is $U(x_k, u_k) = x_k^2 + u_k^2$. The learning process is initialized with an admissible control policy $u^{(0)}(x_k) = -0.3x_k$. By initializing the kernel of x_k as -0.3 and other kernels as 0, we can incorporate the existing control policy without training the Volterra actor. Note that if we use an MLP network or an RBF network as the actor approximator, we have to train the network to approximate the initial control policy $u^{(0)}(x_k) = -0.3x_k$, which is inconvenient and may cause an approximation error. The basis function for value function approximation is selected as $\phi(x_k) = [x_k^2, x_k^4, x_k^6]^T \in \mathbb{R}^3$. The learning process lasts for $1,000$ discrete time steps. Policy evaluation is carried out on data samples collected in 20 time steps. Policy improvement is terminated if $max(|\Delta K|) < 10^{-6}$ or the maximum policy updating step 200 is reached.

The weights in the value function during learning are illustrated in Fig. 2. The Volterra kernels are illustrated in Fig. 3. During $1,000$ discrete time steps, 5 iterations of policy evaluation and policy improvement take place. It can be clearly seen in Fig. 2 and Fig. 3 that the weights of the value function and the Volterra kernels converge.

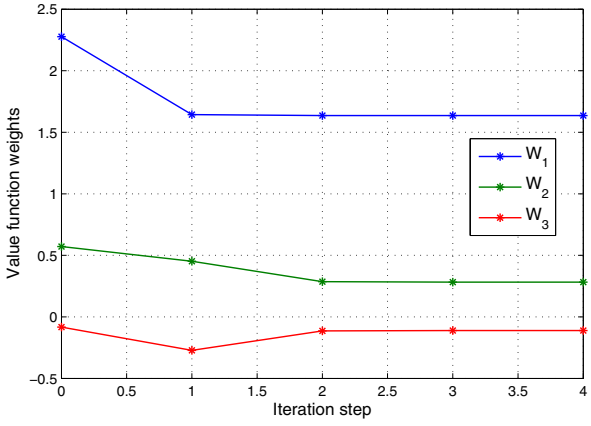

Fig. 2. Value function weights

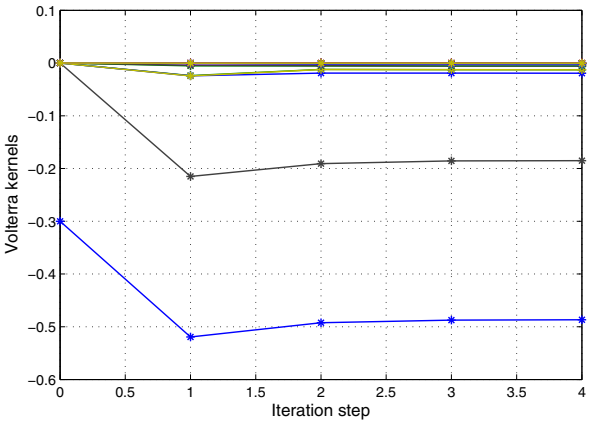

Fig. 3. Volterra kernels of nonlinear controller

To further analyze the final Volterra kernels, histogram of the final Volterra kernels is illustrated in Fig. 4. We can see that most of the Volterra kernel values (25 of 27) are within $-0.1 \sim 0$, and only 2 Volterra kernels have absolute values greater than 0.1. The final Volterra controller with absolute kernel value greater than 0.01 is

$$\begin{aligned} u_k = &- 0.4869x_k - 0.185x_k^2 - 0.0193x_k x_{k-1} \\ &- 0.0128x_{k-1} - 0.0128x_{k-1}^2. \end{aligned} \tag{36}$$

From (36), we can see that the basis with absolute kernel value greater than 0.1 are all based on the current state x_k. In those basis with absolute kernel values in $0.01 \sim 0.1$, x_{k-2}, x_{k-3}, x_{k-4} and x_{k-5} do not appear, and only x_k and x_{k-1} contribute to the Volterra controller (36). It implies that recent historical information is more important for actor approximation.

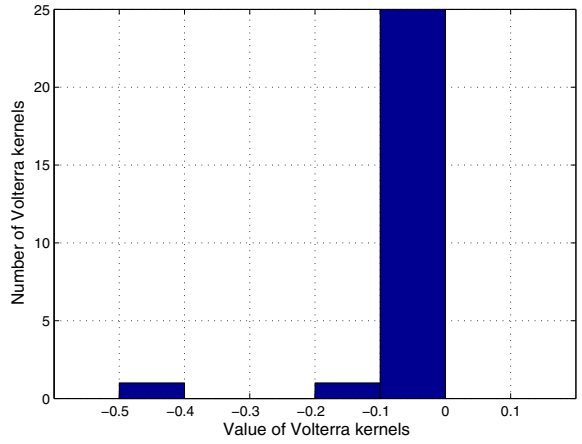

Fig. 4. Histogram of final Volterra kernels

By implementing the final Volterra controller in each policy improvement step into system (35) and simulating it for 20 discrete time steps, we can obtain the approximated total cost of the Volterra controller in each iteration, which is illustrated in Fig. 5. According to Fig. 5, the approximated total cost is a non-increasing sequence. Actually the total cost is reduced in each iteration and gradually approaches the minimal cost specified by the optimal Volterra controller.

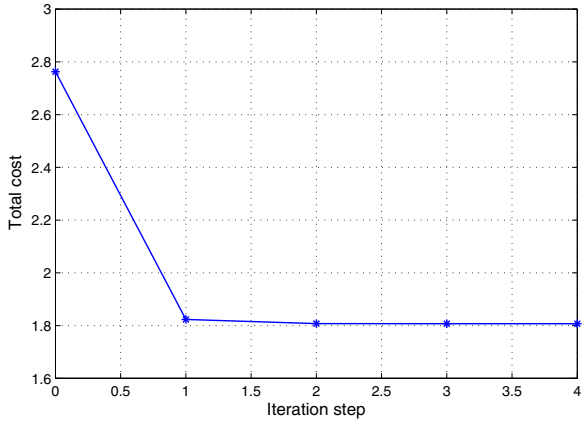

Fig. 5. Total cost

Time domain responses under the original controller and the optimal 2nd order Volterra controller are illustrated in Fig. 6. It can be seen that the time domain response speed becomes faster by using the policy iteration algorithm with Volterra actor.

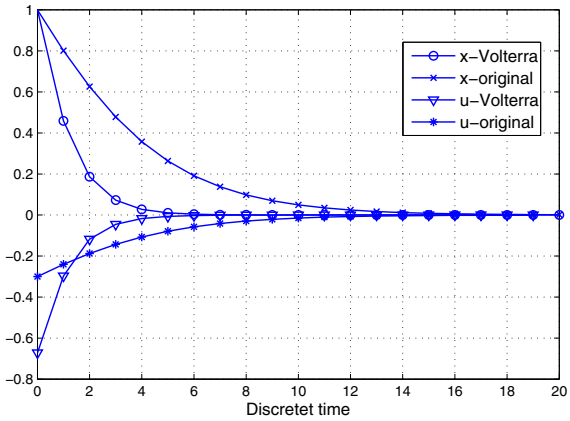

Fig. 6. Time domain responses

VI. Conclusion and Discussion

This paper focuses on developing a Volterra series based actor in an approximate dynamic programming framework. Specifically, we make use of Volterra series for actor approximation developed from policy iteration. Such Volterra actor is linear-in-parameter, which is beneficial for obtaining globally optimal parameters. We provide a convergence proof for the policy iteration algorithm under Volterra actor approximation error, a sufficient condition on the actor approximation error to guarantee convergence of the iterative value function. Furthermore, a bound on the final convergent value function is given. Even though our results are focused on approximating the actor, it is expected that the same concept of Volterra series approximation can be applied in critic or value function

approximation. Results along those lines will be developed in future studies.

REFERENCES

[1] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits Syst. Mag.*, vol. 9, no. 3, pp. 32–50, 2009.

[2] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Syst. Mag.*, vol. 32, pp. 76–105, Dec. 2012.

[3] F.-Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: an introduction," *IEEE Comput. Intell. Mag.*, vol. 4, pp. 39–47, May 2009.

[4] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, *Handbook of learning and approximate dynamic programming: scaling up to the real world*. Piscataway, NJ: Wiley-IEEE Press, 2004.

[5] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, pp. 264–276, Mar. 2001.

[6] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof," *IEEE Trans. Syst., Man, Cybern. B*, vol. 38, pp. 943–949, Aug. 2008.

[7] J. J. Murray, C. J. Cox, G. G. Lendaris, and R. Saeks, "Adaptive dynamic programming," *IEEE Trans. Syst., Man, Cybern. C*, vol. 32, pp. 140–153, May 2002.

[8] 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, pp. 779–791, May 2005.

[9] H. Zhang, Q. Wei, and Y. Luo, "A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm," *IEEE Trans. Syst., Man, Cybern. B*, vol. 38, pp. 937–942, Aug. 2008.

[10] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Netw.*, vol. 8, pp. 997–1007, Sept. 1997.

[11] H. He, Z. Ni, and J. Fu, "A three-network architecture for on-line learning and optimization based on adaptive dynamic programming," *Neurocomputing*, vol. 78, pp. 3–13, Feb. 2012.

[12] T. Dierks and S. Jagannathan, "Online optimal control of affine nonlinear discrete-time systems with unknown internal dynamics by using time-based policy update," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, pp. 1118–1129, July 2012.

[13] D. M. C. Cervellera, M. Gaggero, "Low-discrepancy sampling for approximate dynamic programming with local approximators," *Computers and Operations Research*, vol. 43, pp. 108–115, 2014.

[14] M. Gaggero, G. Gnecco, and M. Sanguineti, "Dynamic programming and value-function approximation in sequential decision problems: error analysis and numerical results," *Journal of Optimization Theory and Applications*, vol. 156, no. 2, pp. 380–416, 2013.

[15] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press, 1960.

[16] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, pp. 621–634, Mar. 2014.

[17] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, pp. 1513–1525, Oct. 2013.

[18] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuška, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. Syst., Man, Cybern. C*, vol. 42, pp. 1291–1307, Nov. 2012.

[19] J.-W. Park, R. G. Harley, and G. K. Venayagamoorthy, "Adaptive-critic-based optimal neurocontrol for synchronous generators in a power system using MLP/RBF neural networks," *IEEE Trans. Ind. Appl.*, vol. 39, pp. 1529–1540, Sept./Oct. 2003.

[20] M. Geist and O. Pietquin, "Algorithmic survey of parametric value function approximation," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 24, pp. 845–867, June 2013.

[21] J. N. Tsitsiklis and B. V. Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Trans. Autom. Control*, vol. 42, pp. 674–690, May 1997.

[22] A. K. Deb, Jayadeva, M. Gopal, and S. Chandra, "SVM-based tree-type neural networks as a critic in adaptive critic designs for control," *IEEE Trans. Neural Netw.*, vol. 18, pp. 1016–1030, July 2007.

[23] X. Xu, Z. Hou, C. Lian, and H. He, "Online learning control using adaptive critic designs with sparse kernel machines," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 24, pp. 762–775, May 2013.

[24] B. Liu and J. Si, "The best approximation to C^2 functions and its error bounds using regular-center Gaussian networks," *IEEE Trans. Neural Netw.*, vol. 5, pp. 845–847, Sept. 1994.

[25] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. New York, NY: John Wiley and Sons, 1980.

[26] C. Lesiak and A. J. Krener, "The existence and uniqueness of Volterra series for nonlinear systems," *IEEE Trans. Autom. Control*, vol. 23, pp. 1090–1095, Dec. 1978.

[27] D. Liu and Q. Wei, "Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems," *IEEE Trans. Cybern.*, vol. 43, pp. 779–789, Apr. 2013.

[28] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.

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

[30] D. P. Bertsekas, *Abstract Dynamic Programming*. Belmont, MA: Athena Scientific, 2013.

[31] D. Song, R. H. M. Chan, V. Z. Marmarelis, R. E. Hampson, S. A. Deadwyler, and T. W. Berger, "Nonlinear dynamic modeling of spike train transformations for hippocampal-cortical prostheses," *IEEE Trans. Biomed. Eng.*, vol. 54, pp. 1053–1066, June 2007.

[32] V. N. Katsikis, D. Pappas, and A. Petralias, "An improved method for the computation of the Moore - Penrose inverse matrix," *Applied Mathematics and Computation*, vol. 217, pp. 9828–9834, Aug. 2011.

[33] P. Koukoulas and N. Kalouptsidis, "Second-order Volterra system identification," *IEEE Trans. Signal Process.*, vol. 48, pp. 3574–3577, Dec. 2000.

[34] R. K. Pearson, B. A. Ogunnaike, and I. F. J. Doyle, "Identification of structurally constrained second-order Volterra models," *IEEE Trans. Signal Process.*, vol. 44, pp. 2837–2846, Nov. 1996.

[35] K. Kim, S. B. Kim, E. J. Powers, R. W. Miksad, and F. J. Fischer, "Adaptive second-order Volterra filtering and its application to second-order drift phenomena," *IEEE J. Ocean. Eng.*, vol. 19, pp. 183–192, Apr. 1994.

[36] F.-Y. Wang, N. Jin, D. Liu, and Q. Wei, "Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound," *IEEE Trans. Neural Netw.*, vol. 22, pp. 24–36, Jan. 2011.