Received May 6, 2019, accepted May 28, 2019, date of publication June 5, 2019, date of current version June 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2920867

Adaptive Dynamic Programming Approach on Optimal Control for Affinely Pseudo-Linearized Nonlinear System

LIN ZHANG[1], YIAN ZHU[1], HAOBIN SHI[1,2], AND KAO-SHING HWANG[3], (Senior Member, IEEE)

[1]School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China
[2]Key Laboratory of Big Data Storage and Management, Ministry of Industry and Information Technology, Northwestern Polytechnical University, Xi'an 710072, China
[3]Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan

Corresponding author: Kao-Shing Hwang (hwang@mail.ee.nsysu.edu.tw)

This work is supported by the Civil Aircraft Project under Grant XJ-2015-D-76, and the major and key project of the Shaanxi Province key research and development plan under Grant 2016MSZD-G-8-1.

ABSTRACT This paper concerns the optimal control problem of a general affinely pseudo-linearized nonlinear system and puts forward the solving scheme based on adaptive dynamic programming and neural network. In detail, a new more general affinely pseudo-linearized nonlinear system model is presented and covers more practical scenarios than existing models in engineering. In consideration of the Bellman optimal principle, the corresponding model particulars are then analyzed and its adaptive dynamic programming solving scheme is derived. That is, the modified policy iteration is used to estimate the value function and optimal control at each sampled state, and then the calculated state-control pairs and state critic values are utilized to train two neural networks (one works as state evaluator and the other works as optimal control generator, respectively). Through the above, the discrete state space is smoothed into a continuous state space, and the notorious dimensional curse problem could be addressed. Moreover, in order to improve the efficiency and precision of the network training, the samples calculating and network training processes are simultaneously carried out in this paper. In the end, a related simulation experiment is applied and the results demonstrate that the proposed method has more effectiveness and validation for the optimal control problem of the newly proposed general system model than the compared methods.

INDEX TERMS Optimal control, adaptive dynamic programming, system identification, neural network.

I. INTRODUCTION

Control Technology, known as philosophy of cybernetics, is used to perfect the system performances by properly changing the nominal system inputs, and plays an extremely important role in present society. With the development of control technology, it is utilized in many fields, such as the large scale systems (for instance, the human population system, the economics system, and so on), the engineering systems (for example, the advanced robots [1], the aerospace vehicles, and so on). In order to improve the system managing efficiency, optimal control policies regarding various classes of systems were put forward in the past. Those policies aim either, both at increasing the system performances or, and at decreasing the system running costs.

In general, to the best knowledge of us, those studies in optimal control community are of mainly three categories, in terms of whether the system dynamics could be analytically obtained. Each of the three categories is detailed in the following subsection.

The associate editor coordinating the review of this manuscript and approving it for publication was Choon Ki Ahn.

A. OPTIMAL CONTROL POLICIES ON PATTERN-DIFFERENT SYSTEM MODELS

The first category concerns systems with absolutely known analytic dynamics. For those systems, canonical technics like inverse system theories [2], [3], linearization methods [2], [3], variational methods [4], [5], Pontryagin minimum principle [4]–[6], and various dynamic programming approaches (the heuristic dynamic programming, the adaptive dynamic programming, et al. [7]–[9]) are introduced to derive the corresponding optimal strategies. However, in practice, those methods could always be problematic, complying with that the inverse system theory, the linearization methods, et al. require the precise system models which seems harsh to some engineering problems to obtain. Besides, variational methods require no constraints in input signal, meaning the input could be arbitrarily valued in real space and it is also limited in practice. As for Pontryagin minimum principle, the solution to this model is a little hardly to get since it requires to solve a partial differential equation constrained by boundaries at two-endpoints [6].

The second category pays much attention to systems with absolutely unknown analytic dynamics. For those problems, researchers always model the systems dynamics (this is also known as system identification) in the beginning with kinds of tricks, for example, the stochastic Markov process (probability state transferring model), the deterministic Markov process (state space model), the function fitting, the Neural Networks, et al. In accordance with them, dynamic programming approaches (the value iteration, the policy iteration, the approximation dynamic programming, the adaptive dynamic programming [6]–[9], the Q-learning [10], deep learning [11] and so on) were utilized to get optimal policies.

The third category, on the other hand, shows its interests on partially known analytic dynamics, especially the system noised by various disturbances as Gaussian (see Gaussian process in signal/imaging processing [12]), Colored, White, specific-pattern dominated (see stochastic demands in inventory control problem [6]) and so on. For these, one of valuable solving scheme is that before designing or applying optimal control policy, system estimation technics are always utilized to estimate the true values of state variables [12]–[14]. With those estimates instead of true system sates, the optimal control policy could be generated (also known as action, signal, et al.). Transparently, with this preceding procedure, the problem could be degraded into the known-analytic dynamics optimal control problem. The second valuable scheme is that the dynamic problem could be modelled directly as Markov process and use stochastic dynamic programming methods to address it afterwards [15]. That is, instead of consider the deterministic utility function, the problem is optimized over the conditional expectation of random variables [6].

In engineering, the relatively large numbers of practical systems, like robots, airplane, could be analytically modelled to some extent. And those models would always show strongly nonlinearity, and are with modelling errors more or less [7]–[9]. Besides, for an actual system, it always runs in a real-time way, meaning it is as a dynamic system and the performances optimization over it is a time series problem rather than a static optimization one. Those two challenges, the complicated nonlinear systems model and the dynamic optimization problem, is what will be discussed in this paper.

By the art-of-the-state given above, it cannot be denied that when it comes to the dynamic optimization problems, no matter deterministic or stochastic, the dynamic programming approach outstands. Consequently, in the following subsection, the current studies on dynamic programming approach are reviewed.

B. ADAPTIVE DYNAMIC PROGRAMMING APPROACH TO NONLINEAR SYSTEM OPTIMAL CONTROL

Dynamic programming (DP) optimization could handle both stochastic process and deterministic process, and it works in accordance to Bellman Optimal Principle [6]. The typical applications of DP are like the inventory control problem in management [6], the linear quadratic optimal control problem in engineering [4], [5], the value iteration and policy iteration methods in Markov decision process [6], [15], the Q-learning and SARSA algorithm in machine learning [6], [10], [16] and so on. Most of them work on the stochastic scenarios and optimization applies over the conditional expectation. However, DP is always cursed by dimensional issues [6], [15], meaning the computation complexity becomes more and more hard, and even impossible with the increase of the scale of state space, the action space, et al. However, this is merely for discrete problems. In reality, the state space or action space could always be continuous, meaning that it would be impossible to enumerate all possible states and actions, and no mention to optimize over them. What's more, the canonical DP approaches always demand the realization in a backward time direction (non-causal direction), that is, it precludes the use of DP in real-time control [6].

On this dimensional curse problem, except the straitforward tricks, like the Myopic (one step greedy), the Lookahead policies (multi-step greedy) [6], researches put forward the tricks of policy approximation using a cluster of policy with distinct embedded parameters to represent all possible policies, and function fitting with sampled information recovering the underlying continuous pattern [6], since those superficial tricks of one/multi-step-greedy could not tell all the story perfectly. The reason is that the one/multi step greedy introduce too much truncation errors in objective value function although the computing power of computers are stronger and stronger. As for policy approximation, it is also sometimes a little relatively hard to apply in practice because modeling a proper prototype of policy is not simple unless we know this prototype in advance.

Towards the function fitting schemes, in the beginning, the basis function methods were focused. In the basis function methods, the continuous pattern is treated as the linear combination of sufficient basis functions and then using the sampled data to train the weights of each basis [6]. The basis function method shows the sufficiency in Hilbert space by the Riese's representative theorem, the Fourier's representation and the Parseval theorem [17]. However, it performs limitedly in engineering due to that it is complicated to choose the appropriate basis functions (from the Gaussian basis, the polynomial basis, the sinusoid basis, et al.) for every specific practical problems and that the truncation error introduced by the high-order basses in a complete basis space is theoretically significant. Subsequently, with the booming of the Neural Networks (NN) technics and the increasing requiring precision in practice, the NN-based approximation schemes are gradually popularized, especially the BP networks thanks to the structural simplicity and mention-worthy nonlinear approximation capacity [7], [18], [19]. Together with BP, the radial basis NN was also reported to be used on this point [20].

Regarding the dynamic programming approach in time-series optimal control problem, the calculated state-action (state-control) pairs are usually used to train an action-generating network, and the state evaluating values are used to train a critic network. Through the above, an optimal control generating method could be gotten in terms of real-time states in a continuous space. Generally, to guarantee the density of the information needed to train two networks, the samples obtaining process and the networks training process could be carried out simultaneously, until the convergence of the network training [7], [18], [19]. This novel idea on NN-based DP was firstly presented by Miller, Sutton, and Werbos in 1990 [21], and further developed in later decades [7]–[9], [18], [19], [22], [23].

Because of different views and different application fields on this point, the conceptually equivalent numbers of terminologies on this technic are proposed, like the approximate dynamic programming, the asymptotic dynamic programming, the reinforcement learning, the adaptive dynamic programming, the Neuro-Dynamic programming and the neural dynamic programming. Fortunately, *2006 NSF Workshop and Outreach Tutorials on Approximate Dynamic Programming* recommended all of the above items should be uniformly represented as *Adaptive/Approximate Dynamic Programming*.

On optimal control problem for nonlinear systems, researchers in the past mainly focus on one specific class of affinely pseudo-linearized nonlinear system model showed in (1).

$$x_{k+1} = f(x_k) + g(x_k)u_k \tag{1}$$

where x stands for the system state; k denotes the discrete time index; $f(\cdot)$ and $g(\cdot)$ are functions with certain properties to build the model; u is the system input [7], [8], [19]; x_k denote the system state at time index k; and u_k denote the system input at time index k. In the following, the notation $x(k)$ and x_k mean the same. Similarly, the notations $u(k)$, $u(x_k)$ and u_k are equivalent.

In order to obtain the optimal control policy of it, scholars proposed various adaptive dynamic programming (ADP) schemes based on the approximate solution of the Hamilton-Jacobi-Bellman (HJB) equation [7]–[9], [19], [21]. Among all of them, typical adaptive critic schemes are the heuristic dynamic programming (HDP), the action-dependent heuristic dynamic programming (ADHDP), the dual heuristic programming (DHP), the iterative ADP, the globalized dual heuristic programming (GDHP), et al. Actually, those different schemes aim at different application cases and different approximation precision. From the viewpoint of author, the existing methods introduced in literatures belong to one of the following three categories.

The first category is represented by heuristic dynamic programming (HDP) [7], [18], [19], [21]. They target to solve the typical scenario modelled by (1). Namely they constructs the sample state-action pairs and state critic value calculating algorithm and then train two networks (Critic NN and Action NN) to refactor the continuous pattern, so that the curse of dimensionality could be handled.

In the second category, the uncertain case of (1) should be mentioned, that is, the case noised by disturbances [18], [23], as shown in (2).

$$x_{k+1} = f(x_k) + g(x_k)u_k + \Delta F_k \tag{2}$$

where ΔF_k stands for the possible noises. For this derivative problem, scholars introduce robust ADP control system by either modifying the cost function adding penalty to new items, or designing new scheme to bound the impacts caused by noises [18], [23].

The third category, cares about that the nonlinear system model could be absolutely unknown and a third NN is applied to identify the dynamics of the system [9], [24], [25]. In the basis of some assumptions on system dynamics, the stability and convergence analysis admit the effectiveness of the proposed method which requires the use of three independent NN.

Although well studied on problem (1), the pseudo affine-linearized may not always make sense, considering of the low-generality of the system model. Hence, the target of this paper is to discuss a more general pseudo-linearized model, and design the corresponding NN based adaptive dynamic programming scheme to obtain the optimal control policy of it.

C. MOTIVATION AND DISCUSSION ON A NEW MORE GENERAL MODEL

The mentioned more general system is modeled as (3).

$$\begin{aligned} x_{k+1} &= F(x_k, u_k) \\ &= f_{00}(x_k) + g_{10}(x_k)u_k + f_{01}(u_k) + g_{11}(u_k)x_k \end{aligned} \tag{3}$$

where $f_{00}(\cdot)$, $g_{10}(\cdot)$, $f_{01}(\cdot)$ and $g_{11}(\cdot)$ are proper functions with certain properties that will be discussed in Section III.

Practically, the model of (3) covers more application scenarios since theoretically it is a more concise approximation model for every given nonlinear dynamics $F(x, u)$ and the detailed explanation on this point is in Section III.

The difficulties to settle this optimal control problem includes: (a) The state space and the action space (the term action is used to denote the control signal) are continuous, meaning the dimensional curse problem is always existing; (b) The newly proposed system model of (3) is more complex than existing model, meaning the algorithm to generate the optimal control is harder to design.

The general motivation of handling this new problem is:

1) **New Optimal Control Generating Algorithm Designing:** The state samples is randomly collected in state space firstly, and the optimal control generating algorithm is designed to obtain the corresponding optimal control (optimal action) at those states. However, in the following, it can be found that the dilemma is that the derived equations are impossible to get the closed-form solutions, compared to the existing methods for the model of (1). Therefore, the numerical methods will be designed to obtain numerical solutions.
2) **Network Training:** Since the actual state space and action space are continuous, it is impossible to enumerate and investigate all of the states to obtain the corresponding optimal controls. Hence, the sampled information is used to train two networks. The one is for state evaluating (critic network) and the other is for optimal control generating (action network). Note that in the first part (New Optimal Control Generating Algorithm Designing), many enough state-action pairs are collected which could be used as samples to train an action network. Besides, during the implementation of the designed algorithm, the state evaluating values could also be calculated, meaning states and their evaluating values are used as pairs to train a critic network.

D. CONTRIBUTIONS IN THIS PAPER

In summary, all the novelties and possible contributions are displayed in this paper.

1) This paper shows the system (3) is more general compared to the popularly existed one. See Section III.
2) This paper derives the corresponding Hamilton-Jacobia-Bellman equation to the new model, and designs the optimal control generating method, as well as the value function approximation (fitting) scheme (See (10), (11) and Section IV).
3) This paper proposes the complete algorithm to obtain the optimal control (See Section IV and Algorithm 1).
4) This paper proves the effectiveness of the presented NN-based methodology in approximating the derivative of a continuously differentiable function (See Theorem 1).

E. PAPER STRUCTURE

This paper is constructed as follows. The first section is Introduction, in which the problem this paper concerns is posted and the arts-of-the-state related are reviewed. As a warming-up, some related background knowledge pertaining to BP Neural Network, system and control are provided in Section II, in order to make the paper more understandable to readers even having no related field basics. After that, the focused problem is mathematically formulated in Section III. Then in Section IV, the detailed methodology in terms of how to solve the formulated problem is presented. In the last, the main results of this paper as well as related discussions, and the future work are concluded in Section VI.

II. BACKGROUND

Before proceeding to the next sections, this section provides some backgrounds to make the paper more readable for readers from different fields.

A. BP NEURAL NETWORK

In engineering, BP Neural Network (NN), an canonical and important unsupervised machine learning method, is widely used to approximate unknown functions with given input-output data pairs [26]–[28]. The overall system structure of a typical two-layer BP NN is showed in Fig. 1.

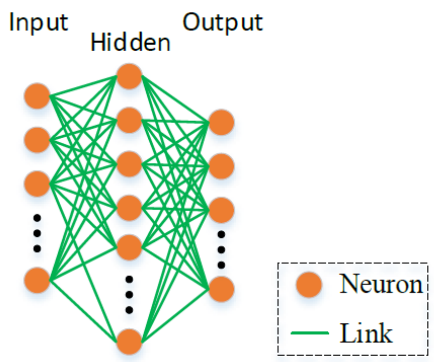

FIGURE 1. The structure of a typical two-layer BP neural network.

It generally consists of three parts called input, hidden and output layer, respectively. For this specific structure, it is also called as two-layer BP NN, indicating one hidden layer and one output layer.

When use a two-layer NN (input-hidden-output) to approximate a function, it has

$$V(x) = \omega^T \cdot \phi(vx + b) + p \tag{4}$$

where x is the input vector with dimension of $n \times 1$; ω is the weight vector from the hidden layer to output with dimension of $l \times 1$; l is the number of neurons in the hidden layer; v is the weight vector from the input to the hidden layer with dimension of $n \times l$; $\phi(\cdot)$ is the fixed activation function. In this paper, the Tan-Sigmoid function $\phi(x) = -1 + 2/(1 + e^{-2x})$ is use as activation function for its nice analytical property (differentiability, its derivative is $\phi'(x) = 4e^{-2x}/(1 + e^{-2x})^2$; and b, p are the constant bias terms.

The power of two-layer BP NN is illustrated in Lemma 1.

Lemma 1: Any smooth function $f(x)$ can be approximated arbitrarily-closely on a compact set using a two-layer neural network (NN) with appropriate weights. The weights can be estimated by gradient descent method. Specifically, the estimate $\hat{f}(x)$ of $f(x)$ is with the form of $\hat{f}(x) = \omega^T \cdot \phi(vx+b)+p$, where the parameters could be found in (4).

Proof: See [27], [28].

B. SYSTEM, STATES AND CONTROL

In control community, the problem of driving the system states or system outputs to specific changing patterns are

$$\begin{aligned}
&F(x,u)\\
&= F_{00}+F_{10}x+F_{11}u+F_{20}x^2+F_{21}xu+F_{22}u^2+F_{30}x^3+F_{31}x^2u+F_{32}xu^2+F_{33}u^3+\ldots\\
&= (F_{00}+F_{10}x+F_{20}x^2+F_{30}x^3+\ldots)+(F_{11}u+F_{22}u^2+F_{33}u^3+\ldots)+(F_{21}x+F_{31}x^2+\ldots)u+(F_{32}u^2+\ldots)x+\ldots\\
&=: f_{00}(x)+f_{01}(u)+(F_{21}x+F_{31}x^2+\ldots)u+(F_{32}u^2+\ldots)x+\ldots\\
&=: f_{00}(x)+f_{01}(u)+g_{10}(x)u+g_{11}(u)x+O(x,u) \qquad (6)
\end{aligned}$$

$$F_{uk} := 2Ru_k+\left[g_0(x_k)+\frac{df_1(u_k)}{du_k}+\frac{dg_1(u_k)}{du_k}x_k\right]^T\frac{\partial V*(x_{k+1})}{\partial x_{k+1}}\bigg|_{u_k^*}=0 \qquad (11)$$

concerned [2], [29]. The system model is usually give as

$$x_{k+1}=f(x_k,u_k) \qquad (5)$$

where x_k, x_{k+1} means the system states at time points k, $k+1$, respectively; u_k is the system inputs and $f(\cdot,\cdot)$ is the system transfer function. Generally, $f(\cdot,\cdot)$ has the linear form so that the corresponding system is called linear system. The system control thus means that the changing laws of system states or system outputs (system behavior) is altered into the desired patterns through the time by giving the proper system input u_k. A typical scenario could be to keep the temperature of a room (the system state x_k) by sensing the real temperature of the room and regulating the working power of the air conditioning (the system input u_k).

III. PROBLEM FORMULATION

Considering the affinely pseudo-linearized system model give in (3), it can be shown that (3) is more general than existing popular model showed in (1).

According to Taylor's expansion theorem, a smooth function $F(x,u)$ could be expressed as (6), as shown at the top of this page. In (6), F_{ij} denotes the corresponding coefficients which could be found in any calculus textbook or online, and $O(x,u)$ is the tail.

Without loss of generality, (6) could be written as (3). Obviously, the model in this paper show stronger universality, compared to the canonical one in (1).

Consuming that without loss of generality [7], $x=0$ is an equilibrium point. That is, $f_0(0)=0$ and $f_1(0)=0$.

Remark 1: Setting zero as an equilibrium point is meaningful in engineering, since many engineering systems concern the state retaining (perturbation rejection) problems. Besides, the property of response on retaining the zero-valued state after disturbance is used to check whether the system is stable or not, to some extent. Therefore, as a result, it can be calculated that $f_0(0)=0$ and $f_1(0)=0$. If in practice the equilibrium point is not zero, it is just needed to give a compensation term to x_k+1 by specific structure of (3).

Remark 2: It is clear that with Remark 1, the ultimate target is driving the system state x_k to zero from any initial value in finite $k<\infty$ (or even acceptable $k<K$, where K is a constant) time steps.

Then an optimal control sequence $u(k)$ is desired to find such that the infinite horizon cost function given in (7) could be minimized for all x_k.

$$V(x_k)=\sum_{n=k}^{\infty}U(x_k,u(x_k))=\sum_{n=k}^{\infty}x_n^TQx_n+u^T(x_n)Ru(x_n) \qquad (7)$$

where $U(x_k,u(x_k))$ is called Utility function and $Q>0, R>0$, meaning the quadratic function is considered as the utility critic in this paper. For simplicity, u_k is used to represent $u(x_k)$ afterwards.

Subsequently, the (7) could be written as

$$\begin{aligned}
V(x_k)&=x_k^TQx_k+u_k^TRu_k+\sum_{n=k+1}^{\infty}(x_n^TQx_n+u_n^TRu_n)\\
&=x_k^TQx_k+u_k^TRu_k+V(x_{k+1}) \qquad (8)
\end{aligned}$$

With Bellman's optimality principle [6], it can be known that for the infinite horizon case, the optimal value function $V*(x_k)$ is time invariant and satisfies the discrete time Hamilton-Jacobia-Bellman equation (DT-HJB)

$$V^*(x_k)=\min_{u_k}[x_k^TQx_k+u_k^TRu_k+V^*(x_{k+1})] \qquad (9)$$

It is should be noted that DT-HJB equation evolves backward in time.

The optimal control sequence $\{u_k^*\}$ should meet the first-order necessary condition, that is

$$\frac{\partial(x_k^TQx_k+u_k^TRu_k)}{\partial u_k}+\frac{\partial x_{k+1}}{\partial u_k}^T\frac{\partial V^*(x_{k+1})}{\partial x_{k+1}}\bigg|_{u_k^*}=0 \quad (10)$$

and therefore the formula of (11), as shown at the top of this page, could be obtained.

If instead considering the model of (1), a degraded result of (11) could be calculated as

$$2Ru_k+g^T(x_k)\frac{\partial V^*(x_{k+1})}{\partial x_{k+1}}\bigg|_{u_k^*}=0 \qquad (12)$$

meaning

$$u_k^*=-\frac{1}{2}R^{-1}g^T(x_k)\frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} \qquad (13)$$

For the degenerated problem (1) and (13), Al-Tamimi et al. [7] present an effective iterative algorithm called heuristic dynamic programming. The main story of it is that it could construct a policy iteration loop to alternately estimate the value function and obtain the optimal solution. That is, the iteration could be evolved between (14) and (15) with the initial value $V_0(x_k) = 0$

$$\begin{aligned} u_{ki} &= \arg\min_u (x_k^T Qx_k + u_k^T Ru_k + V_i(x_{k+1})) \\ &= -\frac{1}{2}R^{-1}g^T(x_k)\frac{\partial V_i(x_{k+1})}{\partial x_{k+1}} \end{aligned} \tag{14}$$

$$\begin{aligned} V_{i+1}(x_k) &= \min(x_k^T Qx_k + u_k^T Ru_k + V_i(x_{k+1})) \\ &= x_k^T Qx_k + u_{ki}^T Ru_{ki} + V_i(x_{k+1}) \end{aligned} \tag{15}$$

where i is the iteration steps at time point k and x_{k+1} is give by (1). Obviously, with this iteration algorithm, the dynamic programming scheme could be applied in forward in time, which is practical in engineering. Further, with the samples got by (14) and (15), two NN could be trained to recognize the underlying continuous value function and optimal control generation function, through which it could handle the dimensional curse issue.

As for the term $\frac{\partial V_i(x_{k+1})}{\partial x_{k+1}}$ in (14), it could use the fitted function to approximate.

If a two-layer NN (input-hidden-output) is used to approximate the value function, that

$$V(x) = \omega^T \cdot \phi(vx + b) + p \tag{16}$$

The meaning of each notation should be found in (4). Additionally, Theorem 1 gives its derivative.

Theorem 1: The derivative function $df(x)/dx$ of any first-order derivative smooth (namely the continuously differentiable) function $f(x)$ could be approximated arbitrarily-closely on a compact set using a two-layer Neural Network (NN). In detail, if $\hat{f}(x) = \omega^T \cdot \phi(vx + b) + p$, then the estimate to the derivative $df(x)/dx$ should be

$$\frac{d\hat{f}(x)}{dx} = \frac{d\hat{f}(x)}{dx} = \omega^T \cdot \nabla\phi(vx + b) \cdot v \tag{17}$$

where ∇ is the gradient operator.

Proof: Let $\hat{f}_n(x)$ be the sequence during executing the gradient descend method to update the weights of NN, namely, $\hat{f}_n(x) = \omega_n^T \cdot \phi(v_n x + b_n) + p_n$, where n is the iteration steps from initial values. Thus, if $n \to \infty$, then $\hat{f}_n(x) \to \hat{f}(x)$. According to Lemma 1, it can be obtained that

$$\limsup_{n\to\infty} \left[f(x) - \hat{f}_n(x)\right] = 0, \quad \forall x \in X \tag{18}$$

where X is the domain of x, and X is a compact set (meaning bounded and closed in real space). In addition, due to that $f(x)$ is continuous, we also have $\hat{f}_n(x)$ is continuous as $n \to \infty$ [30], this could also be concluded by the definition of $\hat{f}_n(x)$.

Since $f(x)$, $\hat{f}_n(x)$ are first-order derivative continuous over X (***Note***: $\hat{f}_n(x)$ is first-order derivative continuous due to its definition.), namely, $f(x), \hat{f}_n(x) \in C^1(X)$, that

$$\begin{aligned} \limsup_{n\to\infty} \left[\frac{df(x)}{dx} - \frac{d\hat{f}_n(x)}{dx}\right] &= \frac{d}{dx} \limsup_{n\to\infty} \left[f(x) - \hat{f}_n(x)\right] \\ &= 0 \end{aligned} \tag{19}$$

The above equality holds due to the property of uniform convergence of $\hat{f}_n(x)$ over a compact set, meaning the differentiation and limitation are interchangeable [30].

Finally, by derivation of $\hat{f}(x)$, (17) can be obtained.

The experimental validation of Theorem 1 could be further found in one numerical experiment in Subsection V-A.

Thus according to Lemma 1, the derivative of $V(x)$ should be

$$\frac{dV(x)}{dx} = \omega^T \cdot \nabla\phi(vx + b) \cdot v \tag{20}$$

where ∇ is the gradient operator.

In order to update weight vectors ω and v during training process, we could just follow the Back-propagation algorithm, illustrated in [7] and [19]–[21].

Coming here, it is theoretically sufficient to execute the algorithm given by (14) and (15) to solve the classic optimal control problem (1). However, compared with (13), it is easily to know that the closed form solution to (11) is hard to take, that is it could not solve the optimal control problem to model (3) directly with iteration method (IM) given by (14) and (15), in general. That is why our story in this paper begins.

In the following section, the effective schemes will be designed to fix the problems this paper faces above.

IV. ADAPTIVE DYNAMIC PROGRAMMING APPROACH TO THE NEW OPTIMAL CONTROL PROBLEM

Since generally it could not get the closed form solution to (11), the numerical solution to it is instead considered. To solve this nonlinear algebra equation, Newton gradient descent should be a first-hand choice. That is

$$u_{ki+1} = u_{ki} - \alpha[\nabla F_{u_{ki}}]^{-1} F_{u_{ki}} \tag{21}$$

where α is a coefficient valued in (0; 1] and $F_{u_{ki}}$ is gradient given by the partial derivative of the left hand of (11) to u_{ki}. Let α_0 be the initial value and α_e be the lower bound. In iteration process, if $\left\|F_{u_{ki+1}}\right\| \leq \left\|F_{u_{ki}}\right\|$, then α should be halved, until it reaches the lower bound α_e. Specifically, see (22).

$$\nabla F_{u_{ki}} = 2R + \left[\frac{d^2 f_1(u_k)}{du_k^2} + \frac{d^2 g_1(u_k)}{du_k^2}x_k\right]^T \frac{\partial V(x_{k+1})}{\partial x_{k+1}}\Big|_{u_{ki}} \tag{22}$$

Unfortunately, Newton gradient method could not always be guaranteed to converge. It strongly depends on the properties of F_{u_k} in (11).

Theorem 2: If $\nabla F_{u_{ki}}$ is continuous and invertible, and $F_{u_{ki}} = 0$ has a unique root in searching interval, then (21) converges to the unique solution that satisfies (11).

Proof: See [31].

In engineering, if the convergence conditions requested by Theorem 2 cannot be satisfied, then it could find the least square solution of $F_{u_k} = 0$. That is

$$u_k^* = \arg\min_u \frac{1}{2}F_{u_k}^T(u)F_{u_k}(u) \tag{23}$$

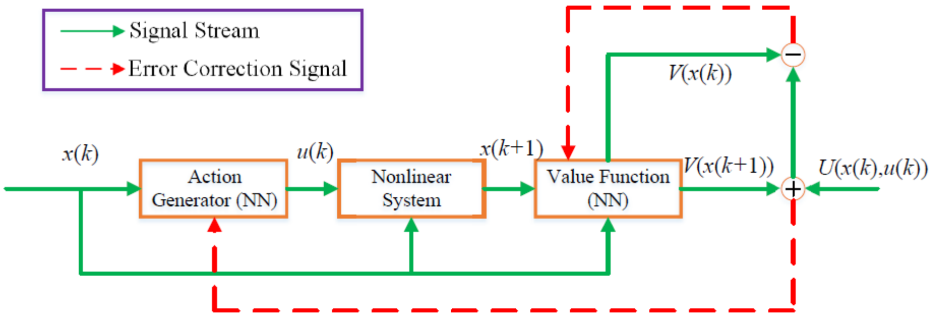

FIGURE 2. **Topology of the training process.**

Let $H(u) := \frac{1}{2}F_{u_k}^T(u)F_{u_k}(u)$, we have $\nabla H(u) = J_G^T F_{u_k}(u)$, where $J_G(u)$ means the Jocabian matrix of $F_{u_k}(u)$.

Then the iteration strategy of u_k could be obtained by Gradient Descent. That is

$$u_{ki+1} = u_{ki} - \alpha_n \nabla H(u_{ki}) \tag{24}$$

Remark 3: The Newton's method looks powerful in convergence rate in practice [32]. This is easy to intuitively understand because this method require the information from the derivative (or Jacobian), meaning the concerned function must be continuous and differentiable, in addition the derivative (or Jacobian) must be invertible. More information means faster convergence speed and higher convergence accuracy. This point is not hard to get. However, those conditions are really restrictive in engineering. Sometimes even the continuity of the function cannot be guaranteed. Thus we rarely consider more on Newton's method practically unless our problems show the perfect enough analytical properties.

Theorem 3: If α_n is given as

$$\alpha_n = \frac{(u_{ki} - u_{ki-1})^T[\nabla H(u_{ki}) - \nabla H(u_{ki-1})]}{(\|\nabla H(u_{ki}) - \nabla H(u_{ki-1})\|)^2} \tag{25}$$

then the iteration (24) could be guaranteed to at least converge to a local solution.

Proof: See [33].

Alternatively, in practice, the fixed point iteration method could also be considered for simplicity, being free from the calculation of gradient and its inverse, if possible.

Let $G_{u_k} := F_{u_k} + u_k$, following iteration algorithm in (26) could be obtained.

$$u_{ki+1} = G_{u_{ki}} \tag{26}$$

where i is the iteration steps.

Theorem 4: If G_{u_k} is a contraction over u_k, then the iteration algorithm (26) could be guaranteed to converge with $u_k = u_k^*$, such that $F_{u_k} = 0$.

Proof: This is by the Banach Fixed Point Theorem. For detail, see [17].

In addition, in order to let x_0 fully cover its definition interval, meaning almost every points (every possible initial state) in this interval could be taken into consideration in training process, and let x_0 uniformly randomly take its value from its definition interval.

In execution, a randomly initialized state x_0 is taken to evolve the iteration between (26) and (15) to get optimal control. If in the following iterations, it reaches the vicinity of a state that have been visited before, then this loop should be terminated and a new cycle is started again with a new random initial state. This is just to avoid a cursed loop from x_k to $x_{k+r} = x_k$, after experiencing $x_{k+1}, x_{k+2}, x_{k+3}, \ldots, x_{k+r}$. This phenomenon is possible since in this paper a deterministic system (3) is concerned.

Overall, the whole topology of the training process is illustrated in Fig. 2.

For detail, Algorithm 1 gives the full procedure of obtaining the optimal solution to the system (3).

Algorithm 1 Algorithm of Obtaining the Optimal Solution to the System (3)

```
Input: $\alpha_0$,$\alpha_e$;
Step 1: Choose the initial state. Randomly value $x_0$ with
    uniform distribution;
Step 2: Get the optimal control. Evolve the iteration
    between (26) [or (24) or (21)] and (15);
Step 3: Train the BP NNs. Use the state-action pairs above
    to sequentially train the action NN, and the states
    and its evaluating values to train the critic NN;
Step 4: Update the system state. Use (3) to update
    the system state;
Step 5: Algorithm Running Control. If two NNs
    converged, then terminate; if the new system state
    reaches the vicinity of visited states, then go to
    Step 1; if the new system state reaches the
    vicinity of zero, then go to Step 1;
Output: The critic NN and the action NN.
```

Remark 4: In Algorithm 1, which strategy of getting the optimal control is preferential, (26), (24) or (21), should depend on the specific formula pattern of model given by (3). The detailed determinant conditions are given by Theorem 4, 3, and 2. That is, firstly, it should be checked whether the (26) is applicable. If not, the feasibility of (21) should then be checked. If not again, (24) is finally used to obtain a least square solution, since the Theorem 3 guarantees its local convergence (in practice the local one is at times sufficient enough). The reason why considering (26) before (21) is that (21) requires an operation of matrix inverse, which is regarded to be computationally hard. Besides, the inverse of a matrix does not always exist.

Remark 5: In Algorithm 1, in order to determine whether the updated system state is within the vicinity of a pre-reached state, an interval that contains all the vicinities obtained before could be constructed. Then for a new coming state, it could be checked that whether it is covered by a sub-interval emerged by vicinities.

V. EXPERIMENT AND ANALYSIS

In this section, an experiment of generating an optimal control of a general affinely pseudo-linearized nonlinear system using the proposed Algorithm 1 is considered, in order to inspect the effectiveness of the methodology in this paper. There are two separate experiments. The first one is to validate the rightness of the Theorem 1, that is, the derivative of a first-order derivative continuous function could indeed be given by (17). The second one is to generate an optimal control of a general affinely pseudo-linearized nonlinear system using the presented Algorithm 1, for inspecting the effectiveness of the methodology in this paper.

A. SIMULATION VALIDATION TO THEOREM 1

In this subsection, the function $f(x) = \frac{1}{10}x \cdot \sin^2(x)$ is considered to approximate its derivative by (17).

The experiment parameters is set as follows:

1) Let the used BP Networks be with the structure of 1-15-1, meaning one input layer containing one neuron, one hidden layer containing fifteen neurons, and one output layer containing one neuron. Its mathematical pattern is given by (16), where $l = 15$.
2) The activation function is Tan-Sigmoid function given in (16).
3) The expected termination control parameters desired in training process is as maximum step $L = 1000$ and convergence determinant (error threshold) $\varepsilon' = 10^{-5}$.
4) The initial learning rate is set as 0.01.

As a result, the approximations of both $f(x)$ and its derivative is shown in Fig. 3 and Fig. 4, respectively.

In Fig. 4, the ground truth, namely the derivative of $f(x)$, could be given from its definition $f(x) = \frac{1}{10}x \cdot \sin^2(x)$, that is

$$\frac{df(x)}{dx} = \frac{1}{5}x \cdot \sin(x) \cdot \cos(x) + \frac{1}{10}x \cdot \sin^2(x) \tag{27}$$

No wonder, both the Fig. 3 and the Fig. 4 support the effectiveness of Theorem 1.

Notably, the proposed method is also powerful to address the noised source data. Suppose the source data generated by $f(x)$ is contaminated by zero-mean normal noise with standard deviation of 0.05. Then the approximations of both $f(x)$ and its derivative given by Theorem 1 are illustrated in Fig. 5 and Fig. 6, respectively.

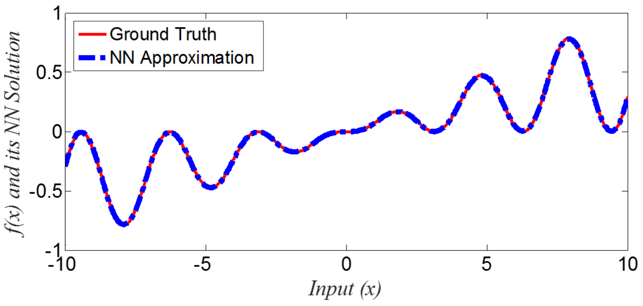

FIGURE 3. The NN approximation, given by (16), to $f(x)$.

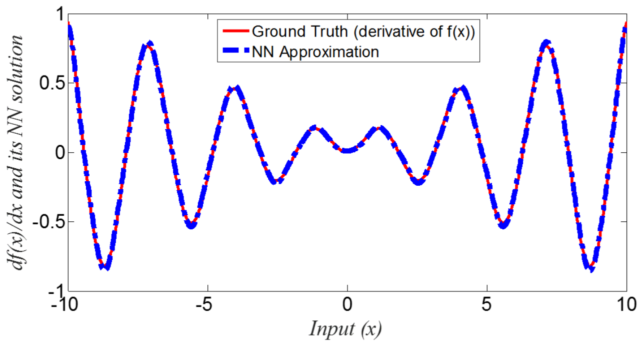

FIGURE 4. The NN approximation, given by (17), to the derivative of $f(x)$.

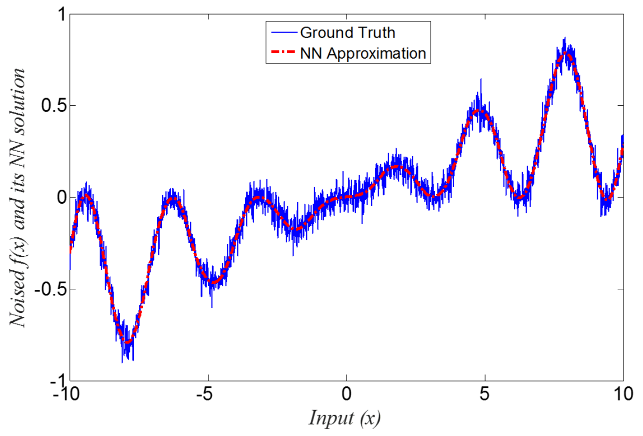

FIGURE 5. The NN approximation, given by (16), to $f(x)$ under noised source data.

Fig. 5 and Fig. 6 show that the presented method is more robust, meaning it could withstands the impact introduced by unexpected noises. However, Fig. 6 still admits the existence of noise could in some degree weaken the ideality of approximation, compared to Fig. 4.

B. THE VALIDATION TO THE OPTIMAL CONTROL GENERATING ALGORITHM

The following system showed in (28) is considered.

$$x_{k+1} = 1 - e^{-\sin(x_k)} + x_k^2 u_k + \sin(u_k) + e^{u_k} x_k \tag{28}$$

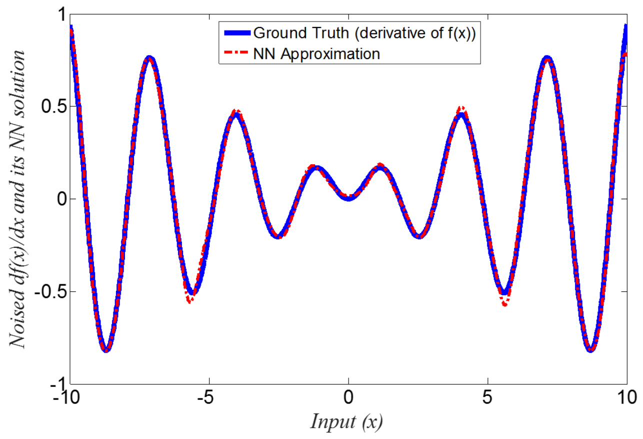

FIGURE 6. The NN approximation, given by (17), to the derivative of $f(x)$ under noised source data.

The reason why setting the system dynamics as this because in engineering it is complex and general enough. The surface of this function is multi-modal (multiple stationary points, which leads to multiple equilibrium points) and really steep in some directions (See also Fig. 7 and Fig. 8).

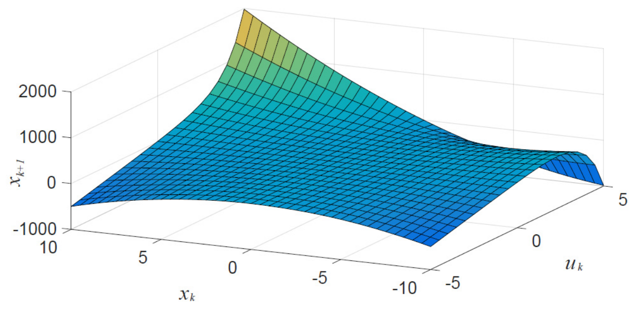

FIGURE 7. The state trajectory of the system (27).

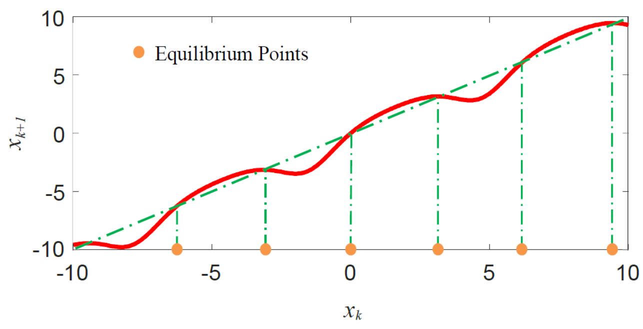

FIGURE 8. The state trajectory of the system (27) without input.

The state trajectory of this system is illustrated in Fig. 7 and if without input it is as in Fig. 8.

Obviously, from Fig. 8 it can be seen that the nominal equilibrium point $x_k = 0$ is not a unique one, meaning this system is not globally stable. That means if exerted an impulse by noise, the system would diverge from $x_k = 0$. In detail, it also has equilibrium points at about $x_k = 3$, $x_k = 6.5$, and $x_k = 9.5$, etc.

As a demonstration, the impulse response of the system is investigated, which is showed in Fig. 9.

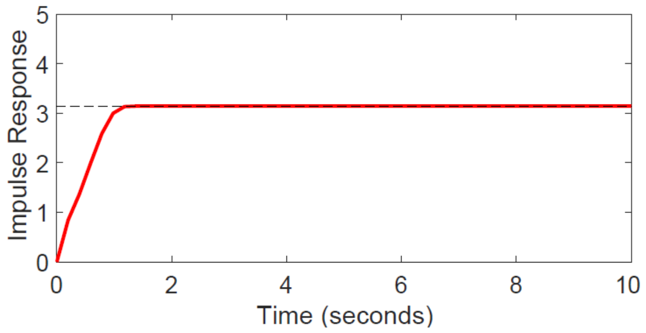

FIGURE 9. The impulse response x_k disturbed by noise (impulse signal here).

It is easy to see that with the impulse input, the system converges to a new equilibrium point $x_k = 3.2$. Thus, the purpose of the control is to drive this system again to its nominal equilibrium point $x_k = 0$.

To get the optimal control, the experiment scenarios is set as follows:

1) Let the simulation time step is l =0.1s.
2) Let $Q = R = \frac{1}{2}$.
3) Let $\alpha_0 = 1.0$ and $\alpha_e = 0.1$.
4) Let the maximum step of the iteration between (24) and
5) (15) is $L = 100$ and the desired accuracy threshold to terminate the iteration process is $\varepsilon = 10^{-2}$.
6) Let all the used BP Networks be with the structure of 1-8-1, meaning one input layer containing one neuron, one hidden layer containing eight neurons, and one output layer containing one neuron; Besides, the activation function is Tan-Sigmoid function; the expected termination control parameters desired in training process is as $L' = 3000$ (maximum step, that is, simulation cycles) and $\varepsilon' = 10^{-2}$ (convergence determinant) above; the initial learning rate is set as 0.2.

Note 1: It should be noted that if ε and ε' is set too small, the algorithm would not converge, or converge in a really long time.

In this case, (11) could have a detailed form

$$F_{u_k} = u_k + (x_k^2 + \cos(u_k) + e^{u_k} \cdot x_k)\frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} = 0 \quad (29)$$

and its Jacobian is

$$J_G^T(u_k) = 1 + (-\sin(u_k) + e^{u_k} \cdot x_k)\frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} = 0 \quad (30)$$

After implementing the Algorithm 1, in which the iteration is used between the (24) and (15), the approximated value function could be obtained and given in Fig. 10, and its derivative approximated by (20) in Fig. 11.

From Fig. 10, it can be seen in this experiment, the minimal value of value function at $x_k = 0$ (nominal equilibrium point)

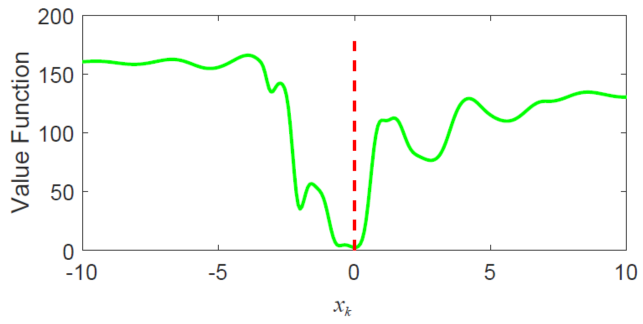

FIGURE 10. The value function against x_k.

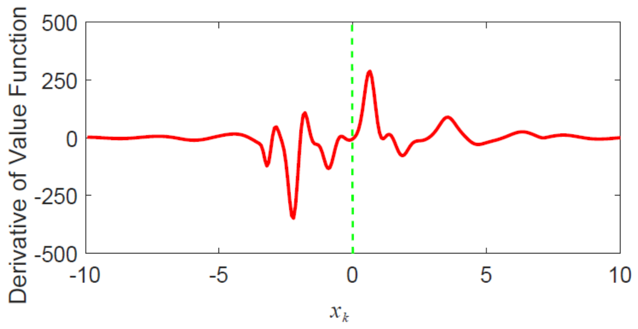

FIGURE 11. The derivative of the value function against x_k.

is indeed gotten, which guarantees the state could be driven to zero with proper input. However, the fluctuation in the curve means this approximation to value function is not ideal. Because in origin expectation, the local minima is not wanted. Yet in practice, it is still applicable since insignificant errors could be tolerable.

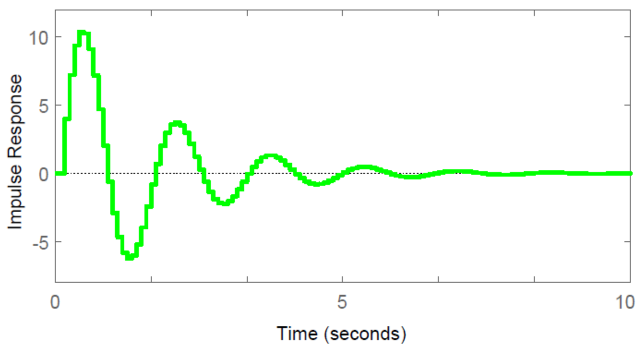

FIGURE 12. The impulse response x_k with optimal control.

The new system impulse response and the optimal control is also obtained and shown in Fig. 12 and Fig. 13, respectively.

It should be mentioned that the reason choosing the (24) to get the optimal control is that it is somewhat hard to verify the conditions required by Theorem 2 and Theorem 4 for this complex system.

From Fig. 12, it can be found that with the optimal control, the system state could be driven to the expected equilibrium point after disturbing by unexpected inputs (generally existed as noises).

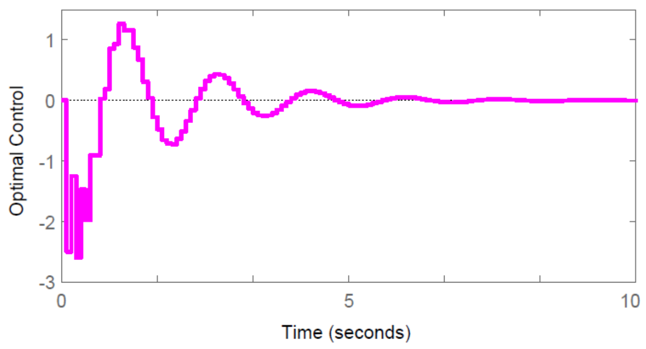

FIGURE 13. The optimal control u_k.

Note 2: In practice, the input signal is expected as continuous as possible, meaning there is no sharp changes in the curve of input. This is a common sense in control theory. Because jerks in input signal require more energy (the instant power is extremely large). Thus, in this experiment, it is restricted that the input signal could change at most $\max|\Delta u_k| = \max|u_k - u_{k-1}| = 0.2$ in each step after 1s (namely when $k \geq 10$). This is why in Fig. 13 there has large changes in the input u_k in the first 1 second, and after the first 1 second, it has a smoother input. However, if the restriction $\max|\Delta u_k| = 0.2$ is set from beginning (namely $k \geq 0$), it may be at times fail to have an efficient input signal. Because the limitation would destroy the convergence of the control process.

VI. CONCLUSION AND DISCUSSION

In this paper the optimal control problem for a more general affinely linearized nonlinear system is studied. Specifically the corresponding discrete time optimal control generating algorithm is derived. And for the fact that the system is evolving in a continuous state space, the Neural Networks schemes are proposed to recover the underlying continuous patterns of value function and optimal control generating function, together with the approximation method of the derivative of one continuously differentiable function using NN. Simulations show that the presented methods could provide an optimal control sequence to the designed optimal control problem so that the disturbed state could be driven to its equilibrium point. Besides, the state critic network and action generating network could be trained online, meaning the dimensional curse problem (introduced by the continuous state space and action space) could be addressed with the proposed scheme. However, it should also be mentioned the solution this paper studies is not a universal one, for some systems it cannot provide an optimal even feasible optimal control. Thus, in future, we will studies the specific application conditions of the proposed algorithm (especially the convergence conditions of the proposed numerical methods), and design more stable and robust algorithms for those systems we expect to handle. Mainly, we should study the stability criteria of the system (3), and the robust control technics for this newly raised problem. The difficulty of robust control is to re-design the optimal condition (7) so that it could take into consideration the disturbances and uncertainties.

REFERENCES

[1] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "BigDog, the rough-Terrain quadruped robot," *IFAC Proc. Volumes*, vol. 41, no. 2, pp. 10822–10825, 2008.
[2] A. Isidori, *Nonlinear Control Systems*. London, U.K.: Springer-Verlag, 2013.
[3] R. Marino and P. Tomei, *Nonlinear Control Design: Geometric, Adaptive and Robust*, vol. 1. London, U.K.: Prentice-Hall, 1995.
[4] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*. Hoboken, NJ, USA: Wiley, 2012.
[5] A. E. Bryson, *Applied Optimal Control: Optimization, Estimation and Control*. Evanston, IL, USA: Routledge, 2018.
[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1, no. 3. Belmont, MA, USA: Athena Scientific, 2005.
[7] 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, Cybern.*, vol. 38, no. 4, pp. 943–949, Aug. 2008.
[8] C. Mu, C. Sun, D. Wang, and A. Song, "Adaptive tracking control for a class of continuous-time uncertain nonlinear systems using the approximate solution of HJB equation," *Neurocomputing*, vol. 260, pp. 432–442, Oct. 2017.
[9] Q. Wei and D. Liu, "Stable iterative adaptive dynamic programming algorithm with approximation errors for discrete-time nonlinear systems," *Neural Comput. Appl.*, vol. 24, no. 6, pp. 1355–1367, May 2014.
[10] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[12] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, vol. 3. New York, NY, USA: Wiley, 1992.
[13] C. K. Chui and G. Chen, *Kalman Filtering*. Berlin, Germany: Springer-Verlag, 2017.
[14] A. Ghoreyshi and T. D. Sanger, "A nonlinear stochastic filter for continuous-time state estimation," *IEEE Trans. Autom. Control*, vol. 60, no. 8, pp. 2161–2165, Aug. 2015.
[15] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2014.
[16] H. Shi, X. Li, K.-S. Hwang, W. Pan, and G. Xu, "Decoupled visual servoing with fuzzyQ-learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 1, pp. 241–252, Jan. 2018.
[17] J. T. Oden and L. Demkowicz, *Applied Functional Analysis*. London, U.K.: Chapman & Hall, 2017.
[18] X. Yang, D. Liu, and D. Wang, "Reinforcement learning for adaptive optimal control of unknown continuous-time nonlinear systems with input constraints," *Int. J. Control*, vol. 87, no. 3, pp. 553–566, Mar. 2014.
[19] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 997–1007, Sep. 1997.
[20] J. J. Murray, C. J. Cox, G. G. Lendaris, and R. Saeks, "Adaptive dynamic programming," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 32, no. 2, pp. 140–153, May 2002.
[21] W. T. Miller, III, P. J. Werbos, and R. S. Sutton, *Neural Networks for Control*. Cambridge, MA, USA: MIT Press, 1995.
[22] 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, no. 5, pp. 779–791, May 2005.
[23] Q. Song, J. C. Spall, Y. C. Soh, and J. Ni, "Robust neural network tracking controller using simultaneous perturbation stochastic approximation," *IEEE Trans. Neural Netw.*, vol. 19, no. 5, pp. 817–835, May 2008.
[24] Q. Wei, R. Song, and P. Yan, "Data-driven zero-sum neuro-optimal control for a class of continuous-time unknown nonlinear systems with disturbance using ADP," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 2, pp. 444–458, Feb. 2016.
[25] D. Liu and Q. Wei, "Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 779–789, Apr. 2013.
[26] J. Li, J.-H. Cheng, J.-Y. Shi, and F. Huang, "Brief introduction of back propagation (BP) neural network algorithm and its improvement," in *Advances in Computer Science and Information Engineering*. Berlin, Germany: Springer, 2012, pp. 553–558.
[27] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.
[28] J. Sarangapani, *Neural Network Control of Nonlinear Discrete-Time Systems*. Boca Raton, FL, USA: CRC Press, 2006.
[29] H. Shi, G. Sun, Y. Wang, and K.-S. Hwang, "Adaptive image-based visual servoing with temporary loss of the visual signal," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 1956–1965, Apr. 2018.
[30] H. L. Royden and P. Fitzpatrick, *Real Analysis*, vol. 32. New York, NY, USA: Macmillan, 1988.
[31] H. Yang, F. Wen, L. Wang, and S. N. Singh, "Newton-Downhill algorithm for distribution power flow analysis," in *Proc. IEEE 2nd Int. Power Energy Conf.*, Dec. 2008, pp. 1628–1632.
[32] A. Gil, J. Segura, and N. M. Temme, *Numerical Methods for Special Functions*, vol. 99. Philadelphia, PA, USA: SIAM, 2007.
[33] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *IMA J. Numer. Anal.*, vol. 8, no. 1, pp. 141–148, 1988.

LIN ZHANG received the master's degree in computer science from Northwestern Polytechnical University, China, in 2013, where he is currently pursuing the Ph.D. degree. His research interests include machine learning, intelligent control, artificial intelligence, decision support systems, and embedded system design.

YIAN ZHU received the Ph.D. degree from Northwestern Polytechnical University, China, in 1991, where he is a Professor and Ph.D. supervisor with the School of Computer Science. His research interests mainly include intelligent control, embedded systems, artificial intelligence, machine learning, and high performance computing.

HAOBIN SHI received the Ph.D. degree from Northwestern Polytechnical University (NPU), China, in 2008, where he is currently an Associate Professor with the School of Computer Science, and the Key Laboratory of Big Data Storage and Management, Ministry of Industry and Information Technology. His research interests include intelligent robots, decision support systems, artificial intelligence, multi-agent systems, and machine learning. He is the Director of the Chinese Association for Artificial Intelligence.

KAO-SHING HWANG (M'93–SM'09) received the M.M.E. and Ph.D. degrees in electrical and computer engineering from Northwestern University, Evanston, IL, USA, in 1989 and 1993, respectively. He was with National Chung Cheng University, Taiwan, from 1993 to 2011. He was the Deputy Director of the Computer Center from 1998 to 1999, the Chairman of the Electrical Engineering Department, from 2003 to 2006, and the Director of the Opti-Mechatronics Institute of the University, from 2010 to 2011. He is a Professor with the Electrical Engineering Department, National Sun Yat-sen University, and an Adjunct Professor with the Department of Healthcare Administration and Medical Informatic, Kaohsiung Medical University, Taiwan. He has been a fellow of the Institution of Engineering and Technology. His research interests include methodologies and analysis for various intelligent robot systems, machine learning, embedded system design, and ASIC design for robotic applications.