Efficient Online Globalized Dual Heuristic Programming With an Associated Dual Network

Zhou, Ye

DOI
10.1109/TNNLS.2022.3164727

Publication date
2022

Document Version
Final published version

Published in
IEEE Transactions on Neural Networks and Learning Systems

Efficient Online Globalized Dual Heuristic Programming With an Associated Dual Network

Ye Zhou

***Abstract*— Globalized dual heuristic programming (GDHP) is the most comprehensive adaptive critic design, which employs its critic to minimize the error with respect to both the cost-to-go and its derivatives simultaneously. Its implementation, however, confronts a dilemma of either introducing more computational load by explicitly calculating the second partial derivative term or sacrificing the accuracy by loosening the association between the cost-to-go and its derivatives. This article aims at increasing the online learning efficiency of GDHP while retaining its analytical accuracy by introducing a novel GDHP design based on a critic network and an associated dual network. This associated dual network is derived from the critic network explicitly and precisely, and its structure is in the same level of complexity as dual heuristic programming critics. Three simulation experiments are conducted to validate the learning ability, efficiency, and feasibility of the proposed GDHP critic design.**

***Index Terms*— Adaptive critic designs (ACDs), globalized dual heuristic programming, incremental model, neural networks, radial basis functions, reinforcement learning (RL).**

I. Introduction

REINFORCEMENT learning (RL) has obtained arising attention in recent years. It is a framework of intelligent, self-learning methods, in which actions are trained in order to minimize the cost-to-go from interacting with the environment. These self-learning methods link bio-inspired artificial intelligence techniques to the field of optimal and adaptive control to overcome some of the limitations and challenges of traditional model-based control methods [1], [2]. Approximate dynamic programming (ADP) is an RL method aiming to solve optimal control problems with large or continuous state spaces [3]–[6]. They apply an approximation of the cost-to-go of states and/or an approximation toward the optimal control policy so as to tackle the "curse of dimensionality" [7].

As a class of ADP methods, adaptive critic designs (ACDs) have shown great success in optimal adaptive control of nonlinear problems and practical applications [7]–[16]. They are also known as actor-ctirics (ACs) because they separate evaluation (critic) and improvement (actor) using parametric structures. The critic adopts temporal difference (TD) methods to approximate the cost-to-go function, while the actor adapts its parameters toward the optimal policy by applying the principle of optimality [1], [7], [17]. Although they are called ACs, they often need an extra structure to approximate the global system model so as to close the update path of the actor, the critic, or both.

ACDs can generally be categorized into three groups: 1) heuristic dynamic programming (HDP), which is the most basic form and uses the critic to approximate the cost-to-go; 2) dual heuristic programming (DHP), in which the critic approximates the derivatives of the cost-to-go with respect to the critic inputs; and 3) globalized dual heuristic programming (GDHP), which approximates both the cost-to-go and its derivatives. Several studies comparing the before-mentioned ACDs have shown that both DHP and GDHP outperform HDP in success rate and precision [10], [18]. The main reason is that the critic of the DHP and the GDHP directly outputs the derivatives of the cost-to-go, which reduces the error introduced by the derivation backward through the critic of the HDP [19].

Online and efficient learning control with ACDs has been studied for years and is still one of the most active areas in RL today. One of the challenges is that the identification of the global system model is not a trivial task, which needs a certain time and usually an off-line learning phase beforehand [3], [8], [20]–[23]. However, this off-line identification stage needs representative simulation models, which are also difficult to obtain in practice. Several studies [19], [24] have suggested removing the global system model and exploiting previous critic outputs and/or inputs instead. Although this technique has been successfully applied to many ACD methods, it can only relieve the off-line learning phase of the action-dependent forms. Recent attempts [25]–[27] exploited incremental models to replace the global system model in ACDs to relieve the off-line learning stage and to increase the adaptability to uncertainties. Incremental model based ACD methods offered a systematic approach for developing online ACDs, especially HDP and DHP, with simplified structures and algorithms.

However, the major challenge to implement online GDHP is to efficiently while accurately calculate the second-order mixed partial derivatives [10], [28]. GDHP combines the advantages of HDP and DHP by minimizing the error with respect to both the cost-to-go and its derivatives

Manuscript received 3 May 2021; revised 20 December 2021; accepted 1 April 2022. Date of publication 18 April 2022; date of current version 1 December 2023. This work was supported by the Malaysian Ministry of Higher Education for providing the Fundamental Research Grant Scheme (FRGS) under Grant FRGS/1/2020/TK0/USM/03/11.

The author is with the School of Aerospace Engineering, Engineering Campus, Universiti Sains Malaysia, Nibong Tebal 14300, Malaysia, and also with the Faculty of Aerospace Engineering, Delft University of Technology, 2629 Delft, The Netherlands (e-mail: zhouye@usm.my).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3164727.

Digital Object Identifier 10.1109/TNNLS.2022.3164727

simultaneously. From a theoretical point of view, the resulted behavior of GDHP is expected to be superior to HDP and DHP [10]. There are, in general, two designs of GDHP that have been successfully implemented in several applications and are still commonly used today. The first one is to use explicit formulas to calculate the second-order derivatives [10], [28], [29]. This design uses a single critic network to approximate the cost-to-go and relies on analytical calculations to find the derivatives and the mixed second-order derivatives. This method is mathematically accurate but computationally complex and is very hard to extend to networks with multiple-layered features. To lighten the computational load, the mixed-style critic was proposed [10], which outputs the cost-to-go and its derivatives simultaneously. This method is efficient and commonly used in various applications [22], [30]–[34]. However, these two kinds of outputs of the mixed-style critic have independent top-layer weights, due to which their updates are not well associated, and the estimation may not be analytically accurate. These two GDHP designs confront a dilemma of either introducing more computational load by explicitly calculating the second partial derivative term or sacrificing the accuracy by loosening the association between the cost-to-go and its derivatives. This article, therefore, aims at increasing the online learning efficiency and feasibility of GDHP while retaining its analytical accuracy.

The main contribution of this article is that a novel GDHP design was proposed based on an HDP-style critic network and an associated dual network. The HDP-style critic network is designed to be nonlinear in inputs and linear in parameters, and the DHP-style dual network can be derived from the critic network explicitly and precisely. In specific, this article demonstrates this concept with two basic and specific designs, which falls into multilayer perceptron and radial basis function categories. The structure of the proposed GDHP critic design is straightforward, illustrative, and at the same level of complexity as the mixed-style critic network. In addition, because of the nonlinear-in-parameter property, the proposed GDHP critic network is extendable to complex networks, such as multiple-layered features, and other types of approximators.

The remainder of this article is structured as follows. Section II lays the foundation of this research with critic designs and adaptation rules of HDP, DHP, and GDHP. Section III formulates the GDHP design with an associated dual network and presents the framework of the proposed GDHP method. Section IV carries out three simulation experiments to examine the learning ability and efficiency of the proposed critic design and to validate the proposed GDHP algorithm for online learning tasks. Finally, Section V concludes the properties of the proposed GDHP design and addresses the possibilities of future research.

II. Foundations on GDHP

ACDs separate evaluation and improvement using parametric structures: the critic and actor. The critic adopts TD methods to approximate the cost-to-go and/or its derivatives, while the actor adapts its parameters toward the optimal policy

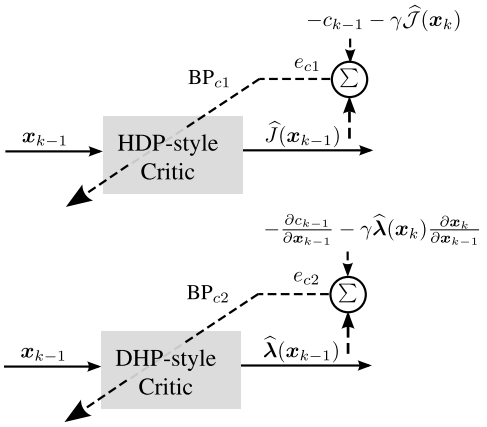

Fig. 1. Critics adaptation in HDP and DHP, where solid lines represent the feedforward flow of signals through their critics, and dashed lines represent the BP pathways for critics, denoted with BP_{c1} for the HDP-style critic and BP_{c2} for the DHP-style critic.

by applying the principle of optimality. The optimization of the actor directly depends on the derivatives of the cost-to-go. This section will illustrate the main difference among HDP, DHP, and GDHP, which lies in their critic designs and adaptation rules.

A. HDP-Style Critic

HDP is the most basic and direct way for policy evaluation. It uses the critic to approximate the true cost-to-go function $\mathcal{J}$, which is the cumulative summation of future cost c from any initial state $\boldsymbol{x}_k$ under current policy μ

$$\mathcal{J}^{\mu}(\boldsymbol{x}_k)=\sum_{l=k}^{\infty}\gamma^{l-k}c_l \tag{1}$$

where $\gamma\in(0,1)$ is a scalar called discount factor or forgetting factor and c_l is the one-step cost at a future time t_l under the current policy.

The error function for the critic $E_{c1}(t_k)$ is defined according to the TD error $\boldsymbol{e}_{c1}(t_k)$ at time t_k as follows:

$$E_{c1}(t_k)=\frac{1}{2}\boldsymbol{e}_{c1}(t_k)^T\boldsymbol{e}_{c1}(t_k) \tag{2}$$

where

$$\boldsymbol{e}_{c1}(t_k)=\widehat{\mathcal{J}}(\boldsymbol{x}_{k-1})-c_{k-1}-\gamma\,\widehat{\mathcal{J}}(\boldsymbol{x}_k). \tag{3}$$

In this equation, $\widehat{\mathcal{J}}(\boldsymbol{x}_{k-1})$ and $\widehat{\mathcal{J}}(\boldsymbol{x}_k)$ are the approximated cost calculated through the critic with weights at the current time $\boldsymbol{w}_{c1}(t_k)$.

The critic weights of HDP can be updated to minimize the error $E_{c1}(t_k)$ with a learning rate η_{c1} along the BP_{c1} error backpropagation (BP) direction as in Fig. 1 as follows [27]:

$$\boldsymbol{w}_{c1}(t_{k+1})=\boldsymbol{w}_{c1}(t_k)+\Delta\boldsymbol{w}_{c1}(t_k) \tag{4}$$

where

$$\Delta\boldsymbol{w}_{c1}(t_k)=-\eta_{c1}\cdot\boldsymbol{e}_{c1}(t_k)\cdot\frac{\partial\widehat{\mathcal{J}}(\boldsymbol{x}_{k-1})}{\partial\boldsymbol{w}_{c1}(t_k)}. \tag{5}$$

B. DHP-Style Critic

DHP, on the other hand, uses its critic network to directly approximate the derivative of the cost-to-go function with respect to the state vector

$$\boldsymbol{\lambda}(\boldsymbol{x}_k) = \frac{\partial \mathcal{J}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_k} \tag{6}$$

which reduces the error introduced by the derivation backward through the critic of the HDP [19], [26]. This is because the adaptation law of the actor relies on the derivatives $\boldsymbol{\lambda}$ instead of the value of $\mathcal{J}$.

Similarly, the TD error to approximate the derivatives can be obtained as [26]

$$\boldsymbol{e}_{c2}(t_k) = \widehat{\boldsymbol{\lambda}}(\boldsymbol{x}_{k-1}) - \frac{\partial c_{k-1}}{\partial \boldsymbol{x}_{k-1}} - \gamma\, \widehat{\boldsymbol{\lambda}}(\boldsymbol{x}_k) \frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{x}_{k-1}}. \tag{7}$$

Also, the critic weights can be updated incrementally along the BP_{c2} error backpropagation direction as in Fig. 1 with

$$\Delta \boldsymbol{w}_{c2}(t_k) = -\eta_{c2} \cdot \boldsymbol{e}_{c2}(t_k)^T \cdot \frac{\partial \widehat{\boldsymbol{\lambda}}(\boldsymbol{x}_{k-1})}{\partial \boldsymbol{w}_{c2}(t_k)}. \tag{8}$$

C. Critic of GDHP

GDHP was proposed to consistently evaluate the absolute level of the cost-to-go as in HDP while also learning about the adaptation slope in fine details as in DHP [35]. GDHP, therefore, minimizes the error with respect to both $\mathcal{J}$ and its derivatives $\boldsymbol{\lambda}$ simultaneously. There are several implementations of GDHP critic network(s), such as dual networks or a single network. We will use $\boldsymbol{w}_c$ to indicate all the neural network weights in GDHP designs. The critic error function can be defined as

$$E_{c3}(t_k) = \frac{\kappa_1}{2} \boldsymbol{e}_{c1}(t_k)^T \boldsymbol{e}_{c1}(t_k) + \frac{\kappa_2}{2} \boldsymbol{e}_{c2}(t_k)^T \boldsymbol{e}_{c2}(t_k) \tag{9}$$

where κ indicates the importance of minimizing the error in the cost function $\mathcal{J}$ or in the derivatives $\boldsymbol{\lambda}$.

The critic weights of GDHP can be updated to minimize the error $E_{c3}(t_k)$ with a learning rate η_{c3} along all the error backpropagation directions incrementally as

$$\begin{aligned}\Delta \boldsymbol{w}_c(t_k) &= -\eta_{c1} \boldsymbol{e}_{c1}(t_k) \frac{\partial \widehat{\mathcal{J}}(\boldsymbol{x}_{k-1})}{\partial \boldsymbol{w}_c(t_k)} - \eta_{c2} \boldsymbol{e}_{c2}(t_k)^T \frac{\partial \widehat{\boldsymbol{\lambda}}(\boldsymbol{x}_{k-1})}{\partial \boldsymbol{w}_c(t_k)} \qquad (10)\\ &= -\eta_{c1} \boldsymbol{e}_{c1}(t_k) \frac{\partial \widehat{\mathcal{J}}(\boldsymbol{x}_{k-1})}{\partial \boldsymbol{w}_c(t_k)} - \eta_{c2} \boldsymbol{e}_{c2}(t_k)^T \frac{\partial^2 \widehat{\mathcal{J}}(\boldsymbol{x}_{k-1})}{\partial \boldsymbol{x}_{k-1} \partial \boldsymbol{w}_c(t_k)} \qquad (11)\end{aligned}$$

where the designated parameters $\eta_{c1} = \eta_{c3}\kappa_1$ and $\eta_{c2} = \eta_{c3}\kappa_2$ can be used in this equation for simplification and consistence with the training of HDP and DHP. The major source of the complexity of GDHP is the calculation of the term $((\partial \widehat{\boldsymbol{\lambda}}(\boldsymbol{x}_{k-1}))/(\partial \boldsymbol{w}_c(t_k)))$ or $((\partial^2 \widehat{\mathcal{J}}(\boldsymbol{x}_{k-1}))/(\partial \boldsymbol{x}_{k-1} \partial \boldsymbol{w}_c(t_k)))$, which are second-order mixed partial derivatives.

There are, in general, three ways to design GDHP [10]. In the first design, which is also the one originally proposed by Werbos [35], an additional network dual to the critic network was created. It inputs the output and the states of all hidden neurons of the critic network and outputs $\boldsymbol{\lambda}$. The mixed second-order derivatives can be calculated by finding derivatives of the dual network outputs $\boldsymbol{\lambda}$ with respect to the weights of the critic network carefully back through the dual network and the critic network. This method, which may increase the length of the backpropagation chain, is straightforward, illustrative, but need very careful design and calculation [10], [35], [36]. The detailed design or implementation was scarcely reported.

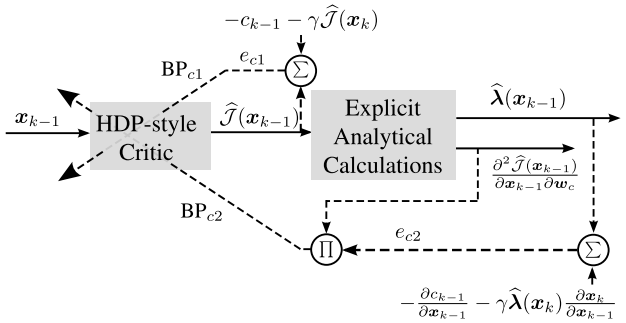

Fig. 2. Critic adaptation in GDHP, which uses explicit formulas to calculate the second-order derivatives. Solid lines represent the feedforward flow of signals through the HDP-style critic and the calculation with explicit formulas, and dashed lines represent the BP pathways for the critic, denoted with BP_{c1} for the cost-to-go error backpropagation pathway and BP_{c2} for the derivatives error backpropagation pathway.

The other two designs, which were proposed by Prokhorov and Wunsch [10], have been successfully implemented in many applications and are still commonly used today.

1) Explicit Formulas to Calculate the Second-Order Derivatives: Instead of creating an illustrative and complex dual network, the derivation of explicit formulas to calculate the second-order derivatives would be an alternative. As shown in Fig. 2, this design of GDHP uses a single HDP-style critic to approximate the cost-to-go $\widehat{\mathcal{J}}(\boldsymbol{x})$, and it relies on mathematical techniques to calculate the derivatives $\widehat{\boldsymbol{\lambda}} = (\partial \widehat{\mathcal{J}}/\partial \boldsymbol{x})$ and the mixed second-order derivatives $(\partial^2 \widehat{\mathcal{J}}/(\partial \boldsymbol{x} \partial \boldsymbol{w}_c))$ elementwisely [10]. The critic, therefore, can be updated by minimizing the error to both $\mathcal{J}$ and its derivatives $\boldsymbol{\lambda}$ incrementally as in (11). These two errors $\boldsymbol{e}_{c1}$ and $\boldsymbol{e}_{c2}$ are used jointly to update the only set of weights in the HDP-style critic, which is mathematically accurate but complex.

Several studies have simplified the formulas by using a forward accumulation of the derivatives or using vectors and matrices to derive the differential operation [28], [29]. However, the computational load is still high and will grow exponentially with the increased width or depth of the network. Also, it is not easy to be applied to high-dimensional problems. Besides, the explicit formulas that calculate the mixed derivatives need to be carefully derived for different activation functions or even a different number of layers.

2) Mixed-Style Critic Network Outputs Cost and Its Derivatives: To lighten the computational load and minimize the structure complexity, the mixed-style critic was proposed [10], which outputs the cost-to-go $\widehat{\mathcal{J}}$ and the derivatives $\widehat{\boldsymbol{\lambda}}$ simultaneously. The critic, therefore, can be updated by minimizing the error to both of these outputs as in (10). This design is most commonly used [30], [33], [34], especially in complex applications. The main reason is that the adaptation of this mixed-style critic, as shown in Fig. 3, is as simple as the DHP method. It is, thus, easy to increase the width or even

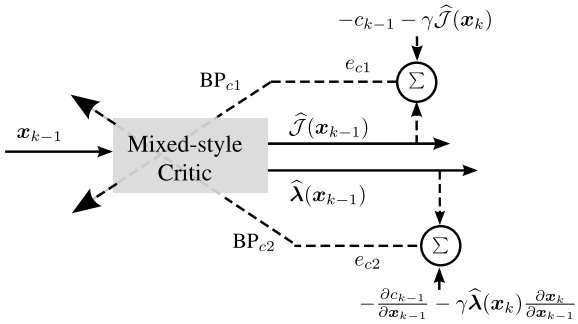

Fig. 3. Mixed-style critic adaptation in GDHP, which outputs the cost-to-go function and its derivatives. Solid lines represent the feedforward flow of signals through the critic, and dashed lines represent the BP pathways for the critic, denoted with BP_{c1} for the cost-to-go error backpropagation pathway and BP_{c2} for the derivatives error backpropagation pathway.

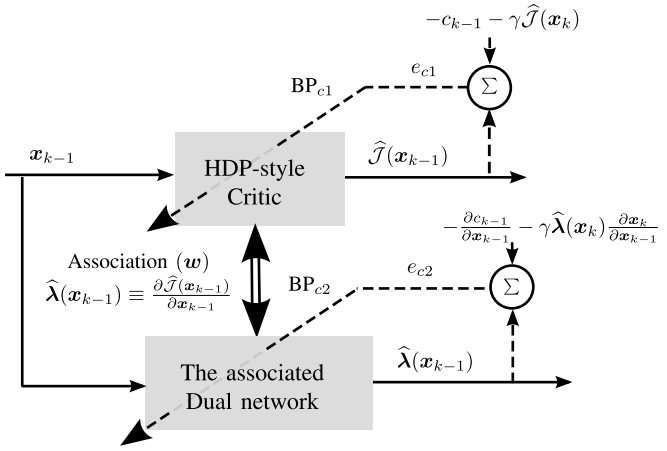

Fig. 4. Adaptation of the critic network and its associated dual network in GDHP. Solid lines represent the feedforward flow of signals through the HDP-style critic and the associated dual network, and dashed lines represent the BP pathways for the HDP-style critic, denoted as BP_{c1}, and the associated dual network, denoted as BP_{c2}.

depth of the network to deal with high-dimensional and highly nonlinear problems.

These two outputs share the weights in the input layer and hidden layers, making them coupled to some extent. However, they have different weights in the top layer, which weaken the connection between them. Analytically, the derivative of the cost function with respect to the input, $\partial\widehat{\mathcal{J}}(\boldsymbol{x})/\partial\boldsymbol{x}$, does not equal to the output $\widehat{\boldsymbol{\lambda}}$ approximated by this critic.

III. GDHP With an Associated Dual Network

This section will propose the GDHP design with an associated dual network.

A. Critic of GDHP With an Associated Dual Network

The concept of this proposed GDHP is based on a specialized design of the HDP-style critic network $\widehat{\mathcal{J}}(\boldsymbol{x},\boldsymbol{w})$, with which we can create a DHP-style dual network $\widehat{\boldsymbol{\lambda}}(\boldsymbol{x},\boldsymbol{w})$ associated with the critic. This associated dual network needs to be derived from the critic network explicitly and precisely as $\widehat{\boldsymbol{\lambda}}(\boldsymbol{x},\boldsymbol{w})=\partial\widehat{\mathcal{J}}(\boldsymbol{x},\boldsymbol{w})/\partial\boldsymbol{x}$, and its structure should be in the same level of complexity as the DHP-style critic network.

The critic model can be generally written as

$$\widehat{\mathcal{J}}(\boldsymbol{x},\boldsymbol{w},\boldsymbol{c})=\sum_j w_j\varphi_j(\boldsymbol{x},\boldsymbol{c})=\boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x},\boldsymbol{c}) \tag{12}$$

where $\boldsymbol{x}\in\mathcal{R}^n$ and $\boldsymbol{w}\in\mathcal{R}^J$ denote the top-layer parameters, j denotes the jth parameter, and $\boldsymbol{c}$ denotes the rest parameters, whose size depends on the width and depth of the network. This equation can be written in the vector form with $\boldsymbol{\varphi}$, which is a nonlinear function of the network input $\boldsymbol{x}$ and the parameters $\boldsymbol{c}$. The derivatives $\widehat{\boldsymbol{\lambda}}$ can, thus, be represented as

$$\widehat{\boldsymbol{\lambda}}(\boldsymbol{x},\boldsymbol{w},\boldsymbol{c})=\frac{\partial\widehat{\mathcal{J}}(\boldsymbol{x},\boldsymbol{w}),\boldsymbol{c}}{\partial\boldsymbol{x}}=\boldsymbol{w}^T\frac{\partial\boldsymbol{\varphi}(\boldsymbol{x},\boldsymbol{c})}{\partial\boldsymbol{x}}. \tag{13}$$

When the activation function $\boldsymbol{\varphi}(\boldsymbol{x},\boldsymbol{c})$ is chosen carefully, the associated network may have the same level of complexity as a DHP-style critic network.

If nonlinear parameters $\boldsymbol{c}$ are fixed, the critic model is linear in the parameters $\boldsymbol{w}$, and the calculation of the second-order mixed derivatives can be further simplified as

$$\frac{\partial^2\widehat{\mathcal{J}}(\boldsymbol{x},\boldsymbol{w},\boldsymbol{c})}{\partial\boldsymbol{x}\partial\boldsymbol{w}}=\frac{\partial\widehat{\boldsymbol{\lambda}}(\boldsymbol{x},\boldsymbol{w},\boldsymbol{c})}{\partial\boldsymbol{w}}=\left(\frac{\partial\boldsymbol{\varphi}(\boldsymbol{x},\boldsymbol{c})}{\partial\boldsymbol{x}}\right)^T. \tag{14}$$

In other words, when the critic model is linear in the parameters, its derivatives with respect to inputs will also be linear in the parameters. Also, the complexity of the second-order mixed derivatives will be reduced to the same as first-order derivatives, which can be obtained through the feedforward calculation with the associated dual network. In addition, using linear-in-parameter critics can help to avoid falling into the local minima trap, which is an intractable problem associated with BP. Note that the critic network and the dual network are still nonlinear in the network input $\boldsymbol{x}$.

As shown in Fig. 4, the critic network and the dual network share the same set of parameters $\boldsymbol{w}$ as association explicitly and accurately. This GDHP design is a special variation of using explicit formulas as in Section II-C1, but the computational load can be reduced to the same level as the mixed-output design in Section II-C2 or even simpler. The choices of the function $\boldsymbol{\varphi}(\boldsymbol{x})$ can be many, but this article will only demonstrate two basic and specific designs, which fall into the multilayer perceptron (MLP) and radial basis function (RBF) categories.

1) Critic With Softplus Activation Functions: The first critic design is based on an MLP NN with a single hidden layer (also known as a two-layer NN), which is the same as most of the current GDHP applications. The difference is that the weights from the input layer to the hidden layer are random constants $C\in\mathcal{R}^{n\times J}$, where J is the number of neurons or the width of the hidden layer. This is inspired by the studies of random features, which proved that, compared to the optimal tuning of the nonlinearities, randomly choosing the nonlinearities in the first layer can produce similar accuracy and faster by one to three orders of magnitude [37].

The activation function can be any commonly used functions in MLP networks, such as sigmoid and hyperbolic functions. To make the associated dual network simple, effective, and easy to demonstrate the association concept, softplus is a good choice because its derivative is a sigmoid function. The output of each softplus function in the hidden layer can be

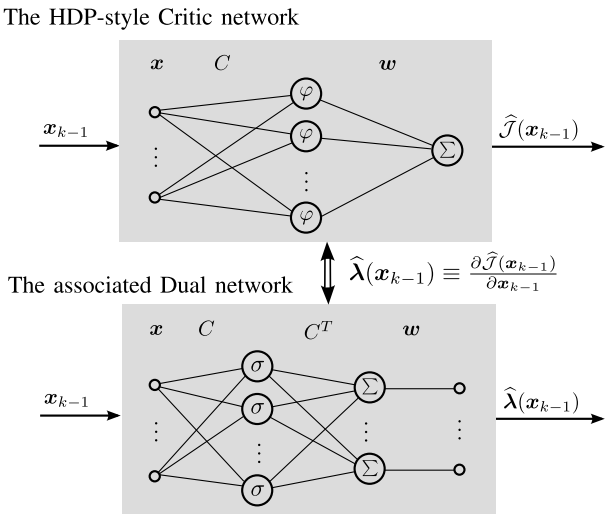

Fig. 5. Critic network with softplus activation functions and its associated dual network with sigmoid activation functions, which shares the to-be-determined parameters $\boldsymbol{w}$. The hidden layer inputs in both networks are the same.

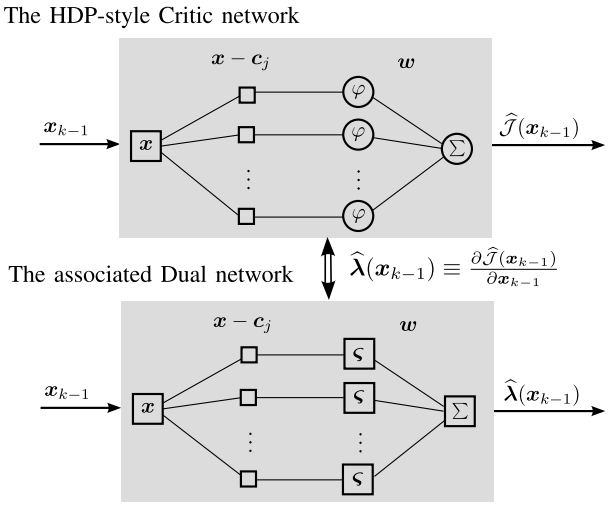

Fig. 6. Critic network with RBF activation functions and its associated dual network, which shares the to-be-determined parameters $\boldsymbol{w}$. The output of each square unit is a vector. These two networks have the same set of centers $\boldsymbol{c}_j$.

written as

$$\varphi_j\left(C_j^T \boldsymbol{x}\right) = \ln\left(1 + e^{C_j^T \boldsymbol{x}}\right) \tag{15}$$

where C_j is the jth column vector of the constant weight matrix C and $C_j^T \boldsymbol{x}$ is the input of the jth hidden neuron.

Therefore, the associated dual network, as in Fig. 5, has J activation functions, each of which is the partial derivative of the jth softplus function output with respect to the critic input

$$\begin{aligned} \boldsymbol{\varsigma}_j(\boldsymbol{x}) &= \frac{\partial \varphi_j\left(C_j^T \boldsymbol{x}\right)}{\partial \boldsymbol{x}} = \frac{1}{1 + e^{C_j^T \boldsymbol{x}}} \cdot e^{C_j^T \boldsymbol{x}} \cdot C_j \\ &= \frac{1}{1 + e^{-C_j^T \boldsymbol{x}}} \cdot C_j = \sigma_j\left(C_j^T \boldsymbol{x}\right) \cdot C_j \end{aligned} \tag{16}$$

where $\boldsymbol{\varsigma}_j : \mathcal{R}^n \to \mathcal{R}^n$ is the jth column vector of the derivatives $\boldsymbol{\varsigma}(\boldsymbol{x}) = \partial \boldsymbol{\varphi}(\boldsymbol{x})/\partial \boldsymbol{x}$ and σ_j is a sigmoid function of $C_j^T \boldsymbol{x}$, which is the input of the hidden layer in the HDP-style critic. Note that if using the sigmoid or hyperbolic tangent activation functions in the HDP-style critic $\varphi_j(C_j^T \boldsymbol{x})$, the hidden layer of the associated dual network can be a function of the output of the hidden layer in the HDP-style critic $\boldsymbol{\varsigma}_j = \varphi_j(1 - \varphi_j) \cdot C_j$ and $\boldsymbol{\varsigma}_j = 1 - \varphi_j^2 \cdot C_j$, respectively. In this case, to reduce the repeated computations, the input of the associated dual network can be the output of each hidden neuron in the HDP-style critic network, where the associated dual network is dependent on the HDP-style critic network.

It may be noticed that this dual network is consistent with the description of the originally proposed GDHP design by Werbos [35], which feeds the output and the states of all hidden neurons of the critic network to the dual network. However, the proposed method in this article, as described, is analytically accurate and more efficient with the association. In addition, because of the linear-in-parameter property, the complex calculation of the backpropagation through the dual network is circumvented. The derivatives can be explicitly represented by a simple equation

$$\widehat{\boldsymbol{\lambda}}(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \cdot \operatorname{diag}(\boldsymbol{\sigma}(C^T \boldsymbol{x})) \cdot C^T \tag{17}$$

where $\boldsymbol{\sigma}(C^T \boldsymbol{x}) = [\sigma_1(C_1^T \boldsymbol{x}), \ldots, \sigma_J(C_J^T \boldsymbol{x})]^T$ and $\operatorname{diag}(\cdot)$ reshapes the vector to a diagonal matrix. Also, the second-order derivatives can be calculated as

$$\frac{\partial^2 \widehat{\mathcal{J}}(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{x} \partial \boldsymbol{w}} = C \cdot \operatorname{diag}(\boldsymbol{\sigma}(C^T \boldsymbol{x})) \tag{18}$$

which is an explicit and straightforward solution.

2) Critic With the Radial Basis Function Network: The radial basis function (RBF) network is often an alternative to MLPs, where the former can be used for both the large-scale global approximation or local fine-tuning [16], [38]. Also, RBFs are often used in (sparse) kernel methods, which may also be used in critic designs [39]. In RBF networks, each hidden neuron is a radial function, whose value depends on the distance between the input to a center point $\|\boldsymbol{x} - \boldsymbol{c}_j\|$ instead of a summation of the weighted input $C_j^T \boldsymbol{x}$.

The second critic implementation is an RBF network with J Gaussian radial functions

$$\varphi_j(\boldsymbol{x}) = e^{-(\|\boldsymbol{x} - \boldsymbol{c}_j\|/r_j)^2} \tag{19}$$

where $\boldsymbol{c}_j$ is the jth center point and r_j denotes its radius. When $\boldsymbol{c}_j$ and r_j are fixed, the critic is linear to the parameters $\boldsymbol{w}$, which are weights connecting the output of each radial functions to the output of the critic network.

The associated network, as in Fig. 6, has J activation functions, each of which is the partial derivative of the jth Gaussian radial function output with respect to the critic input

$$\begin{aligned} \boldsymbol{\varsigma}_j(\boldsymbol{x}) &= \frac{\partial \varphi_j(\boldsymbol{x})}{\partial \boldsymbol{x}} = e^{-(\|\boldsymbol{x} - \boldsymbol{c}_j\|/r_j)^2} \frac{-2}{r_j^2}(\boldsymbol{x} - \boldsymbol{c}_j) \\ &= \varphi_j(\boldsymbol{x}) \frac{-2}{r_j^2}(\boldsymbol{x} - \boldsymbol{c}_j) \end{aligned} \tag{20}$$

where $\boldsymbol{\varsigma}_j$ is the jth row vector of the derivatives $\boldsymbol{\varsigma}(\boldsymbol{x}) = \partial\boldsymbol{\varphi}(\boldsymbol{x})/\partial\boldsymbol{x}$. The derivatives can be explicitly calculated as

$$\widehat{\boldsymbol{\lambda}}(\boldsymbol{x},\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{\varsigma}(\boldsymbol{x}) = \boldsymbol{w}^T\begin{bmatrix}\varphi_1(\boldsymbol{x})\dfrac{-2}{r_1^2}(\boldsymbol{x}-\boldsymbol{c}_1)\\ \vdots \\ \varphi_J(\boldsymbol{x})\dfrac{-2}{r_J^2}(\boldsymbol{x}-\boldsymbol{c}_J)\end{bmatrix}. \quad (21)$$

Also, the second-order derivatives can be represented as

$$\frac{\partial^2\widehat{\mathcal{J}}(\boldsymbol{x},\boldsymbol{w})}{\partial\boldsymbol{x}\partial\boldsymbol{w}} = \boldsymbol{\varsigma}(\boldsymbol{x})^T = \begin{bmatrix}\varphi_1(\boldsymbol{x})\dfrac{-2}{r_1^2}(\boldsymbol{x}-\boldsymbol{c}_1)\\ \vdots \\ \varphi_J(\boldsymbol{x})\dfrac{-2}{r_J^2}(\boldsymbol{x}-\boldsymbol{c}_J)\end{bmatrix}^T \quad (22)$$

which is the output of the hidden layer in the associated dual network.

3) Expanded to Complex Networks: Although increasing the network width can reach a high enough degree of accuracy in most applications, adding the depth of the network would be more efficient in many other cases. The depth of the proposed GDHP critic network can be increased to $L \geq 2$ with different activation functions in each layer, which has never been used in conventional GDHP implementations using explicit formulas because it is too complex to calculate the second-order mixed partial derivatives. However, the linear-in-parameter property of the proposed design ensures that the derivatives $\widehat{\boldsymbol{\lambda}}$ will also be linear in the parameters $\boldsymbol{w}$, as in (13), regardless of the complexity in nonlinear features. Therefore, this design can be easily expanded to more deep and general network designs.

A fully connected MLP network, which has L hidden layers and is linear in parameters $\boldsymbol{w}$, can be represented as

$$\widehat{\mathcal{J}}(\boldsymbol{x},\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{\varphi}_L\big(C_L^T\boldsymbol{\varphi}_{L-1}\big(C_{L-1}^T\ldots\boldsymbol{\varphi}_1\big(C_1^T\boldsymbol{x}\big)\big)\big) \quad (23)$$

where $\boldsymbol{\varphi}_l : \mathcal{R}^{J_l} \to \mathcal{R}^{J_l}$ denotes the activation function in the lth hidden layer with J_l neurons and $C_l \in \mathcal{R}^{J_{l-1}\times J_l}$ are constant parameters from the $(l-1)$th layer to the lth hidden layer. The input of the lth layer neurons is denoted as $\boldsymbol{a}_l$. The derivatives $\widehat{\boldsymbol{\lambda}}$ can, thus, be calculated as

$$\widehat{\boldsymbol{\lambda}}(\boldsymbol{x},\boldsymbol{w}) = \boldsymbol{w}^T\cdot\operatorname{diag}\big(\boldsymbol{\varphi}_L'(\boldsymbol{a}_L)\big)\cdot C_L^T\ldots\operatorname{diag}\big(\boldsymbol{\varphi}_1'(\boldsymbol{a}_1)\big)\cdot C_1^T \quad (24)$$

where $\boldsymbol{\varphi}_l'(\boldsymbol{a}_l) : \mathcal{R}^{J_l} \to \mathcal{R}^{J_l}$ denotes the first-order derivatives of function $\boldsymbol{\varphi}_l$ with respect to $\boldsymbol{a}_l$. Also, the second-order mixed derivatives can be presented explicitly as

$$\frac{\partial^2\widehat{\mathcal{J}}(\boldsymbol{x},\boldsymbol{w})}{\partial\boldsymbol{x}\partial\boldsymbol{w}} = C_1\cdot\operatorname{diag}\big(\boldsymbol{\varphi}_1'(\boldsymbol{a}_1)\big)\ldots C_L\cdot\operatorname{diag}\big(\boldsymbol{\varphi}_L'(\boldsymbol{a}_L)\big). \quad (25)$$

The first-order derivatives of activation functions $\boldsymbol{\varphi}_m'$ can be calculated as functions of the neuron input of the critic network, such as softplus, or as functions of neuron output, such as sigmoid. It is again consistent with the description of the originally proposed GDHP design by Werbos [35] with a straightforward and illustrative structure. However, the associated dual network is an explicit expression, and the second-order mixed derivatives can be obtained with the feedforward calculation of the derivatives $\widehat{\boldsymbol{\lambda}}$.

The applications in this article will use gradient descent to adapt the neural network weights, which follows the conventional routine in ACDs. It is noticeable that the linear-in-parameter property allows the network to be updated using linear optimization methods, such as recursive least square (RLS) with a forgetting factor to reduce the influence of old data. In this sense, polynomials can also be used to approximate the cost-to-go in this design, which will have a simple function of derivatives. Also, this design can be expanded to more powerful but linear-in-parameter approximators, such as multivariate simplex splines [40], which we leave for future work.

B. GDHP Implementation With an Incremental Model

Conventional ACDs use a system model network to approximate the dynamics of the global system, so as to close the adaptation loop for the actor and/or the critic. To make the learning control online, it is suggested [19], [24] to remove the global system model and exploit previous critic outputs and/or inputs instead. In this article, we will use an incremental model, in line with our earlier research [26], [27], to approximate the local linear model varying with time, assuming a sufficiently high sample rate for discretization.

As GDHP uses discrete measurements of system states, a nonlinear system can be written in a general discrete form

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k,\boldsymbol{u}_k) \quad (26)$$

where $\boldsymbol{f} : \mathcal{R}^{n+m} \to \mathcal{R}^n$ provides the system dynamics. When the sample time is sufficiently small, the system dynamics around $\boldsymbol{x}_k$ can be linearized by taking the first-order Taylor expansion as follows:

$$\boldsymbol{x}_{k+1} \approx \boldsymbol{x}_k + F_{k-1}\cdot(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) + G_{k-1}\cdot(\boldsymbol{u}_k - \boldsymbol{u}_{k-1}) \quad (27)$$

where $F_{k-1} = \partial\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})/\partial\boldsymbol{x}|_{\boldsymbol{x}_{k-1},\boldsymbol{u}_{k-1}} \in \mathcal{R}^{n\times n}$ is the system transition matrix and $G_{k-1} = \partial\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})/\partial\boldsymbol{u}|_{\boldsymbol{x}_{k-1},\boldsymbol{u}_{k-1}} \in \mathcal{R}^{n\times m}$ is the input distribution matrix of the linearized system at time step $k-1$. The incremental form of this discrete nonlinear system can be written as

$$\Delta\boldsymbol{x}_{k+1} \approx F_{k-1}\Delta\boldsymbol{x}_k + G_{k-1}\Delta\boldsymbol{u}_k. \quad (28)$$

Instead of using a global model, this time-varying linear model can be used to approximate the local system dynamics at times. The model parameters $\widehat{F}_k$ and $\widehat{G}_k$ can be identified online using the RLS method, which has been elaborated in our previous research [26], [27]. These identified matrices, which approximate the system model derivative terms, can be directly used to close the adaptation loop of the actor and critic

$$\widehat{F}_{k-1} \approx \left.\frac{\partial\boldsymbol{x}_k}{\partial\boldsymbol{x}_{k-1}}\right|_m \quad (29)$$

$$\widehat{G}_{k-1} \approx \left.\frac{\partial\boldsymbol{x}_k}{\partial\boldsymbol{u}_{k-1}}\right|_m \quad (30)$$

where subscript $_m$ denotes the derivatives directly back through the system model.

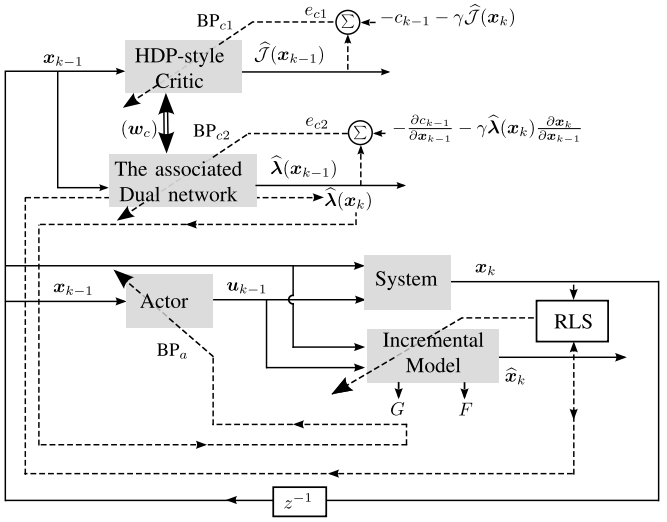

Fig. 7. Schematic of the proposed GDHP with an associated dual network and a time-varying incremental model. Solid lines represent the feedforward flow of signals, and dashed lines represent the adaptation pathways.

Note that the proposed GDHP method also works with global model approximators, especially when the system model is deterministic. If using a nonlinear global model, such as neural networks, it is necessary to calculate these partial derivatives of the nonlinear model output with respect to the inputs as in (29) and (30) back through the global model.

C. GDHP Framework and Adaptation Rules

The design of the critic network and its associated dual network has been described. Also, the online identified incremental model will provide necessary information to approximate the system model derivative terms to close adaptation loops. This section will present the framework of the proposed GDHP method and the adaptation rules for the critic and actor.

The schematic of the proposed GDHP with an associated dual network is shown in Fig. 7. The critic, as described in Section III-A, consists of an HDP-style critic network to approximate the cost-to-go and an associated dual network to approximate the derivatives, which share the same set of weights $\boldsymbol{w}_c$. The actor network with parameters $\boldsymbol{w}_a$ inputs the system state $\boldsymbol{x}$ and outputs the action to take $\boldsymbol{u}$.

1) Critic Adaption Rules: The critic parameters $\boldsymbol{w}_c$ can be updated after each measurement $\boldsymbol{x}_k$ using (10) through two pathways: BP$_{c1}$ to update the HDP-style critic network and BP$_{c2}$ to update the associated dual network, which can be rewritten as follows:

$$\Delta \boldsymbol{w}_c = \underbrace{-\eta_{c1}\boldsymbol{e}_{c1}\frac{\partial \widehat{\mathcal{J}}(\boldsymbol{x}_{k-1})}{\partial \boldsymbol{w}_c}}_{\text{BP}_{c1}\text{ pathway}} \underbrace{-\eta_{c2}\boldsymbol{e}_{c2}^T\frac{\partial \widehat{\boldsymbol{\lambda}}(\boldsymbol{x}_{k-1})}{\partial \boldsymbol{w}_c}}_{\text{BP}_{c2}\text{ pathway}} \tag{31}$$

where $\boldsymbol{e}_{c1}$ and $\boldsymbol{e}_{c2}$ are TD errors to approximate the cost-to-go as in (3) and its derivatives as in (7). The terms $\widehat{\mathcal{J}}(\boldsymbol{x})$ and $\widehat{\boldsymbol{\lambda}}(\boldsymbol{x})$ can be calculated forward through the critic network and the associated network with the current weight $\boldsymbol{w}_c(t_k)$, respectively.

The one-step cost c_{k-1}, for control problems, is often a function of the system state $\boldsymbol{x}_{k-1}$. Thus, the term $\partial c_{k-1}/\partial \boldsymbol{x}_{k-1}$ in (7) is an explicit expression, which is often a function of $\boldsymbol{x}_{k-1}$. The system state $\boldsymbol{x}_k$ can be approximated as a function of the previous state $\boldsymbol{x}_{k-1}$ and the control input $\boldsymbol{u}_{k-1}$. Therefore, the last term in (7), $\partial \boldsymbol{x}_k/\partial \boldsymbol{x}_{k-1}$, needs to be calculated through two pathways: $\boldsymbol{x}_{k-1} \xleftarrow{\text{system}} \boldsymbol{x}_k$ and $\boldsymbol{x}_{k-1} \xleftarrow{\text{actor}} \boldsymbol{u}_{k-1} \xleftarrow{\text{system}} \boldsymbol{x}_k$ [26]

$$\frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{x}_{k-1}} = \left.\frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{x}_{k-1}}\right|_m + \left.\frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{u}_{k-1}}\right|_m \cdot \left.\frac{\partial \boldsymbol{u}_{k-1}}{\partial \boldsymbol{x}_{k-1}}\right|_a. \tag{32}$$

By using the incremental model, the identified matrices, as in (29) and (30), can be applied to approximate these two system model derivative terms as follows:

$$\frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{x}_{k-1}} \approx \widehat{F}_{k-1} + \widehat{G}_{k-1} \cdot \left.\frac{\partial \boldsymbol{u}_{k-1}}{\partial \boldsymbol{x}_{k-1}}\right|_a. \tag{33}$$

Also, the TD error to approximate the derivatives in (7) can be calculated as follows:

$$\begin{aligned}\boldsymbol{e}_{c2}(t_k) = \widehat{\boldsymbol{\lambda}}(\boldsymbol{x}_{k-1}) - \frac{\partial c_{k-1}}{\partial \boldsymbol{x}_{k-1}} \\ - \gamma\,\widehat{\boldsymbol{\lambda}}(\boldsymbol{x}_k)\left[\widehat{F}_{k-1} + \widehat{G}_{k-1} \cdot \left.\frac{\partial \boldsymbol{u}_{k-1}}{\partial \boldsymbol{x}_{k-1}}\right|_a\right].\end{aligned} \tag{34}$$

2) Actor Adaption Rules: The actor weights adaptation in the proposed GDHP is similar to other ACDs; the control policy is improved by updating the actor to minimize the nonnegative cost-to-go, $\mathcal{J}(\boldsymbol{x}_k)$

$$\begin{aligned}\boldsymbol{u}_k^* &= \arg\ \min_{\boldsymbol{u}_k}\ \mathcal{J}(\boldsymbol{x}_k) \\ &= \arg\ \min_{\boldsymbol{u}_k}\ [c_k + \gamma\,\mathcal{J}(\boldsymbol{x}_{k+1})].\end{aligned} \tag{35}$$

By applying the gradient descent method, the actor weights can be incrementally updated as follows:

$$\begin{aligned}\Delta \boldsymbol{w}_a(t_k) &= -\eta_a \cdot \frac{\partial \mathcal{J}(\boldsymbol{x}_k)}{\partial \boldsymbol{u}_k}\frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{w}_a(t_k)} \\ &= -\eta_a \cdot \left[\frac{\partial c_t}{\partial \boldsymbol{u}_k} + \gamma\,\lambda(\boldsymbol{x}_{k+1})\frac{\partial \boldsymbol{x}_{k+1}}{\partial \boldsymbol{u}_k}\right]\frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{w}_a(t_k)}\end{aligned} \tag{36}$$

where η_a is the learning rate to update the actor weights.

In the backpropagation calculation, the derivative of the next state with respect to the control input $\partial \boldsymbol{x}_{k+1}/\partial \boldsymbol{u}_k$, as in (30), can be approximated by the online identified input distribution matrix $\widehat{G}_{k-1}$

$$\Delta \boldsymbol{w}_a(t) \approx -\eta_a \cdot \left[\frac{\partial c_t}{\partial \boldsymbol{u}_k} + \gamma\,\widehat{\lambda}(\widehat{\boldsymbol{x}}_{k+1})\widehat{G}_{k-1}\right]\frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{w}_a(t_k)}. \tag{37}$$

Also, the next state $\widehat{\boldsymbol{x}}_{k+1}$ can be predicted using the identified incremental model as follows:

$$\widehat{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_t + \widehat{F}_{k-1}\Delta \boldsymbol{x}_t + \widehat{G}_{k-1}\Delta \boldsymbol{u}_t. \tag{38}$$

As shown in Fig. 7, the weight update of the actor involves the critic and the system model through BP$_a$ backpropagation direction.

IV. Numerical Experiments

In this section, three simulation experiments are carried out to examine the learning ability and efficiency of the proposed GDHP design with an associated dual network in comparison to the explicit analytical method and the mixed-style critic design. The first experiment investigates the approximation ability of the networks with optimally tuned and random features, which verifies the feasibility of tuning only the top-most weights. Second, the mixed-style critic and the proposed critic designs are applied to approximate a cost function and its derivatives. This experiment examines the associated relation between the cost function and its derivatives of the two critic designs and the robustness in the presence of measurement noise. Finally, the proposed GDHP design is validated in controlling a simplified missile model online to track a reference signal. All these simulations are conducted in the MATLAB environment on a computer with 1.8-GHz CPU and 40 GB of RAM.

A. Feasibility of Critic Approximation With Random Features

The conventional GDHP, especially those that use explicit formulas to calculate the second-order derivatives, adapt all the weights in a fully connected critic network to get the optimal features. To simplify the implementation, the proposed GDHP design, which also explicitly derives the associated dual network, uses random fixed weights in the bottom layers and only updates the top-layer weights. Compared to the optimal tuning of all the weights, the random features, represented by the random fixed weights C, with the same width will produce a less accurate model. In other words, to improve the learning efficiency, the approximation ability in the proposed design will be sacrificed. However, earlier studies [37], [41] showed that random features can reach an equal level of accuracy by increasing the width of a network but still in much less time.

Therefore, this experiment will compare the approximation ability and computational efficiency of neural networks with optimizing features, which are used in conventional GDHP with explicit analytical formulas, and those with random features, which are used in the proposed GDHP design with association. In this numerical experiment, two-layer neural networks with softplus activation functions will be applied to approximate a convex function

$$\vartheta(x) = -\cos(x), \quad x \in [-3, 3]. \tag{39}$$

The first neural network has 10 hidden neurons and random initial weights C and $\boldsymbol{w}$, both of which will be trained using the gradient descent method adaptively. The other three neural networks have random fixed initial weights C, and only their top-layer weights $\boldsymbol{w}$ will be trained using the same gradient descent method. However, the number of hidden neurons will increase from 10 to 30. All the initial weights are randomly chosen from a normal distribution $\mathcal{N}(0, 1)$. The learning rate is small enough for convergence.

Fig. 8 and Table I present the training results of different neural networks in terms of the root mean square (RMS) of the training errors and CPU time. As illustrated, the first

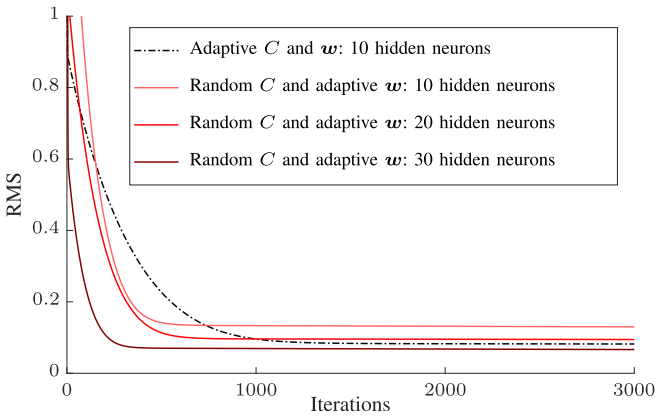

Fig. 8. RMS of training errors using the gradient descent method to train all weights (both C and $\boldsymbol{w}$) and to only train the top-most weights ($\boldsymbol{w}$).

TABLE I
Training Results With Random Features

	Adaptive C and $\boldsymbol{w}$	Random C and adaptive $\boldsymbol{w}$		
Hidden neurons	10	10	20	30
RMS	0.0819	0.1299	0.0945	0.0664
CPU time [s]	2.5239	0.0781	0.1165	0.1761

neural network with 10 hidden neurons and optimal tuning of all weights will have higher training accuracy than the same network with random fixed weights C. However, the CPU time to train all weights, which is 2.5239 s, is much higher than the training of only the top-most weights. When the number of neurons with random fixed weights increases to 20, the number of weights to be tuned is 20, which is the same as the first network. Also, the one with random features is slightly less accurate than the first network with optimally tuned weights. However, when the number of neurons further increases to 30, the network with fixed random weights is capable of a higher degree of accuracy, and the CPU time is 0.01762 s, which is still in much less time. The simulation result indicates that, compared to the conventional GDHP with optimal features and explicit analytical formulas, the proposed GDHP critic design with random fixed features can achieve the same level of accuracy with much less time.

B. Association Between the Cost Function and Its Derivatives

The mixed-style critic that outputs the cost and its derivatives is the most simple-structured and widely used GDHP design. Also, for most cost functions with perfect measurements, such as the near-convex function as in the previous section, it works well. However, the mixed-style critic design does not produce an analytical connection between the approximations of the cost and its derivatives. On the other hand, the proposed critic design has an explicit structure of the dual network, which is straightforward and illustrative while retaining the analytical association.

This experiment will examine the association between a cost function and its derivatives in the mixed-style critic and the proposed critic design and the robustness in the presence of measurement noise. This cost function has a higher degree of

TABLE II
RMS ERROR OF GDHP CRITICS

RMS	$\widehat{\mathcal{J}}(x)$	$\widehat{\lambda}(x)$	$\partial\widehat{\mathcal{J}}(x)/\partial x$
Critic with association	0.382	1.16	1.16
Mixed-style critic	0.212	0.282	10.1

nonlinearity

$$\mathcal{J}(x) = -0.5\ e^{0.5x^2}\cos(8x) + e^x + 0.5\sin(20x) \tag{40}$$

where $x \in [-2, 2]$, and its derivative can be obtained explicitly as

$$\begin{aligned}\lambda(x) &= \partial\mathcal{J}(x)/\partial x \\ &= -0.5e^{0.5x^2}[x\cos(8x) - 8\sin(8x)] + e^x + 10\cos(20x).\end{aligned} \tag{41}$$

In this experiment, both the mixed-style critic and the proposed critic design use RBF networks with the width of 100. The center points are evenly distributed within $[-2, 2]$, and the radius is 0.08. The centers and the radius are fixed, and the parameters $\boldsymbol{w}$ are updated using the gradient descent method with the same learning rate $\eta_{c1} = \eta_{c2}$ as in (10).

1) Approximation Result With Perfect Measurements: Fig. 9 presents the training result of the proposed GDHP critic design with an associated dual network. To minimize the error with respect to both $\mathcal{J}(x)$ and its derivative $\lambda(x)$, the weights adaptation needs to seek a compromise. The RMS of their training errors is 0.382 and 1.16, as highlighted in Table II. Fig. 10 shows the result from the mixed-style critic network, which approximate both the cost function $\widehat{\mathcal{J}}(x)$ and its derivative $\widehat{\lambda}(x)$ with different sets of weights, as described in Section II-C2. The RMS of their training errors is 0.212 and 0.282, both of which are more accurate than the proposed method. However, the RMS error of the derivatives from the approximated cost $\partial\widehat{\mathcal{J}}(x)/\partial x$ is 10.1 as shown in Table II, which is much larger than the critic with association. It also indicates that the analytical calculation of the derivatives from the approximated cost does not match the approximation of the derivative from the mixed-style critic output, i.e., $\widehat{\lambda}(x) \not\equiv \partial\widehat{\mathcal{J}}(x)/\partial x$.

The results reveal that the mixed-style critic in this experiment can approximate the cost function and its derivative very accurately but independently, which means that their updates are not well associated. Thus, the learning ability of GDHP with the mixed-style critic will degenerate into a DHP method because the actor adaptation will only rely on $\widehat{\lambda}(x)$. On the other hand, the proposed method guarantees that $\widehat{\lambda}(x) \equiv \partial\widehat{\mathcal{J}}(x)/\partial x$, because of the association between the HDP-style critic network and its dual network, in compliance with the concept of GDHP.

2) Approximation Result With Noisy Measurements: To further validate the robustness of the critic approximation, high-frequency measurement noise is superimposed to the perfect data samples. The simulated noise is zero-mean normal distributed white noise $\mathcal{N}(0, \sigma)$. Table III provides the RMS of training errors using the two critic designs in the presence of noise with different standard deviations $\sigma = 1$ and $\sigma = 2$. The data from this table can be compared with the data in Table II,

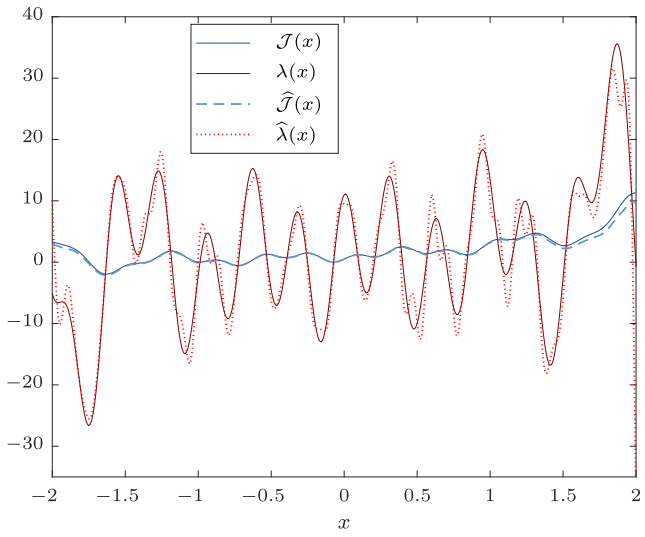

Fig. 9. Outputs from the proposed GDHP critic network $\widehat{\mathcal{J}}(x)$ and its associated dual network $\widehat{\lambda}(x) \equiv \partial\widehat{\mathcal{J}}(x)/\partial x$.

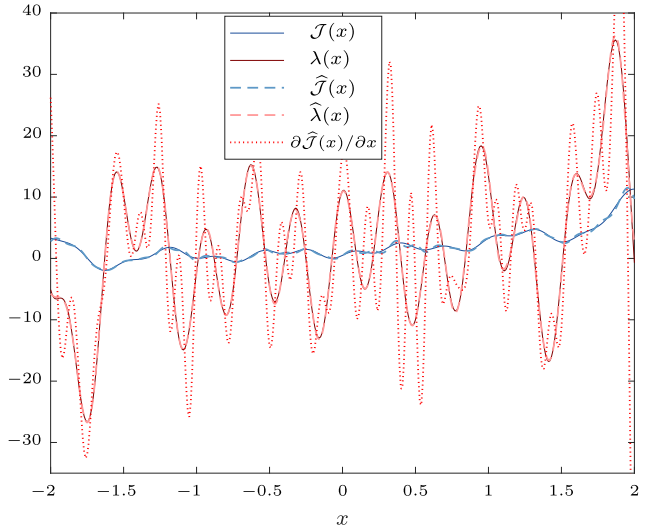

Fig. 10. Outputs from the mixed-style critic network $\widehat{\mathcal{J}}(x)$ and $\widehat{\lambda}(x)$ and the analytical calculation of the derivative $\partial\widehat{\mathcal{J}}(x)/\partial x$.

which indicates that the approximation errors of the mixed-style critic increase considerably in the presence of noise. On the contrary, the approximation errors of the proposed associated critic remain in the same level with the superimposed noise, especially in $\widehat{\lambda}(x)$ or $\partial\widehat{\mathcal{J}}(x)/\partial x$. It is noticeable that, when the standard deviation of the noise increased to $\sigma = 2$, as shown in Fig. 11, the approximation accuracy of the associated critic is higher than the mixed-style critic even in $\widehat{\mathcal{J}}(x)$ and $\widehat{\lambda}(x)$. The results in Table III and Fig. 11 verify that, owning to the strong and analytical association, the proposed critic design is more robust compared to the mixed-style critic in the presence of measurement noise.

The experiment results in this section also reveal that with accurate training data and the same critic structure, DHP will outperform the HDP method. This is because HDP will only approximate $\mathcal{J}(x)$ and calculate the derivative using $\partial\widehat{\mathcal{J}}(x)/\partial x$ as in Figs. 10 and 11(c). Also, the DHP method will directly output the approximated derivative $\widehat{\lambda}(x)$, which fits the true value much more accurately. For the GDHP method, the

TABLE III
RMS ERROR OF GDHP CRITICS WITH RESPECT TO NOISY DATA

RMS	Noise	$\widehat{\mathcal{J}}(x)$	$\widehat{\lambda}(x)$	$\partial\widehat{\mathcal{J}}(x)/\partial x$
Critic with association	$\mathcal{N}(0,1)$	0.394	1.27	1.27
	$\mathcal{N}(0,2)$	0.798	1.50	1.50
Mixed-style critic	$\mathcal{N}(0,1)$	1.12	1.11	25.18
	$\mathcal{N}(0,2)$	2.19	2.18	46.36

proposed critic design has the ability to minimize the error with respect to both $\mathcal{J}$ and its derivative λ and to seek a compromise.

C. GDHP With an Associated Dual Network

In this numerical experiment, the proposed GDHP algorithm will be applied to control a simplified missile model [42], [43] to track a reference signal. The nonlinear model of a short period flight control problem consists of two states: angle of attack α and pitch rate q, and the pitch is controlled using elevator deflection δ_e. The nonlinear model in the pitch axis is simulated around a steady wing-level flight condition

$$\dot{\alpha} = q + \frac{\bar{q}S}{m_a V_T} C_z(\alpha, q, M_a, \delta_e) \tag{42}$$

$$\dot{q} = \frac{\bar{q}Sd_l}{I_{yy}} C_m(\alpha, q, M_a, \delta_e) \tag{43}$$

where $\bar{q}$ is the dynamic pressure, S is the reference area, m_a is the mass, V_T is the speed, d_l is the reference length, I_{yy} is the pitching moment of inertia, C_z is the force coefficient in body Z-direction, and C_m is the pitch moment coefficient. C_z and C_m are nonlinear functions of angle of attack α, pitch rate q, Mach number M_a, and elevator deflection δ_e. The aerodynamic parameters of this model are valid for $-10^\circ < \alpha < 10^\circ$ [42], [43]

$$\begin{aligned}
C_z(\alpha, q, M_a, \delta_e) &= C_{z1}(\alpha, M_a) + B_z\delta_e \\
C_m(\alpha, q, M_a, \delta_e) &= C_{m1}(\alpha, M_a) + B_m\delta_e \\
B_z &= b_1 M_a + b_2 \\
B_m &= b_3 M_a + b_4 \\
C_{z1}(\alpha, M_a) &= \phi_{z1}(\alpha) + \phi_{z2} M_a \\
C_{m1}(\alpha, M_a) &= \phi_{m1}(\alpha) + \phi_{m2} M_a \\
\phi_{z1}(\alpha) &= h_1\alpha^3 + h_2\alpha|\alpha| + h_3\alpha \\
\phi_{m1}(\alpha) &= h_4\alpha^3 + h_5\alpha|\alpha| + h_6\alpha \\
\phi_{z2} &= h_7\alpha|\alpha| + h_8\alpha \\
\phi_{m2} &= h_9\alpha|\alpha| + h_{10}\alpha
\end{aligned} \tag{44}$$

where $b_1, \ldots, b_4$, $h_1, \ldots, h_{10}$ are identified constant coefficients in the flight envelop, and the Mach number M_a is set to be 2.2.

This model has been used to validate incremental model based heuristic dynamic programming (IHDP) and incremental model based dual heuristic programming (IDHP) algorithms in our early studies [26], [27]. In the proposed GDHP algorithm, the incremental model, as described in III-B, will also be used to approximate the local linear model. For a fair comparison to the IDHP algorithm, this GDHP algorithm only changes the DHP-style critic network with the width of 6 to the HDP-style one with an associated dual network with softplus activation functions as in Section III-A1 and the same width. The initial weights of the bottom layer are randomly chosen from a

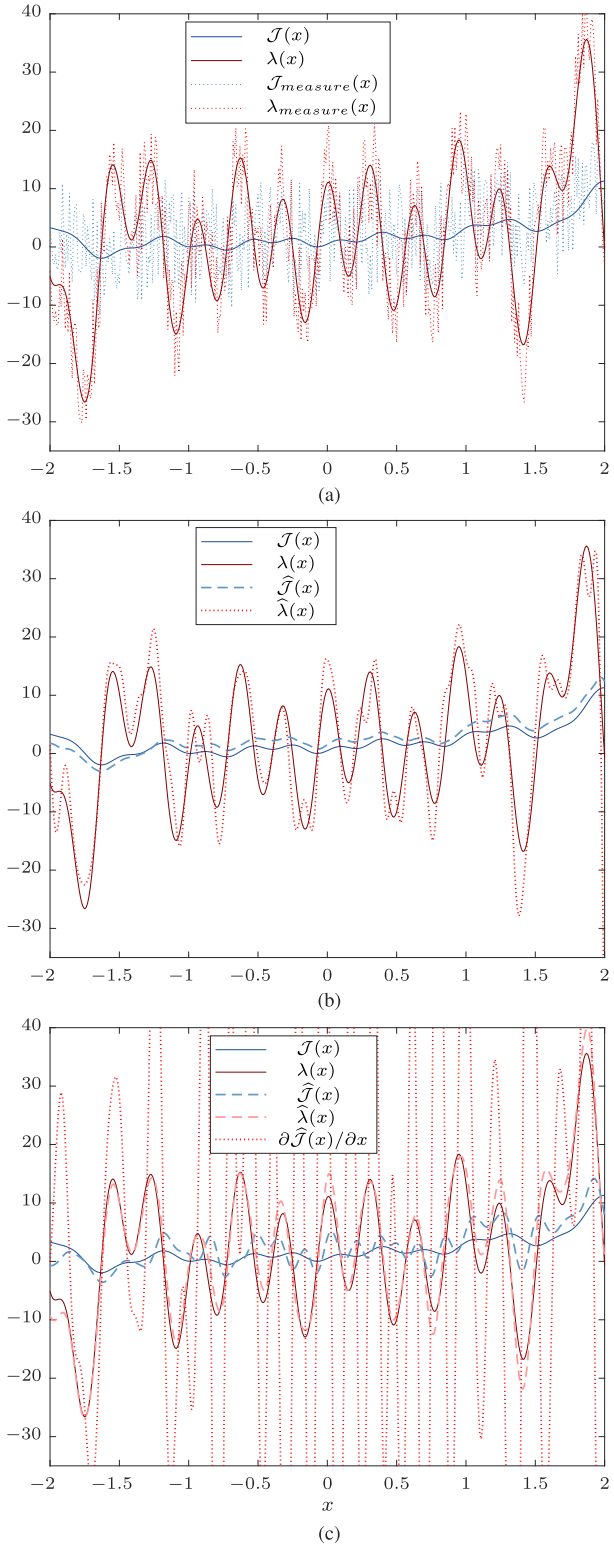

Fig. 11. Approximation with the proposed critic and the mixed-style critic designs in the presence of measurement noise $\mathcal{N}(0, 2)$. (a) Data sample of $\mathcal{J}(x)$ and $\lambda(x)$ with measurement noise. (b) Outputs from the proposed GDHP critic network with association. (c) Outputs from the mixed-style critic network.

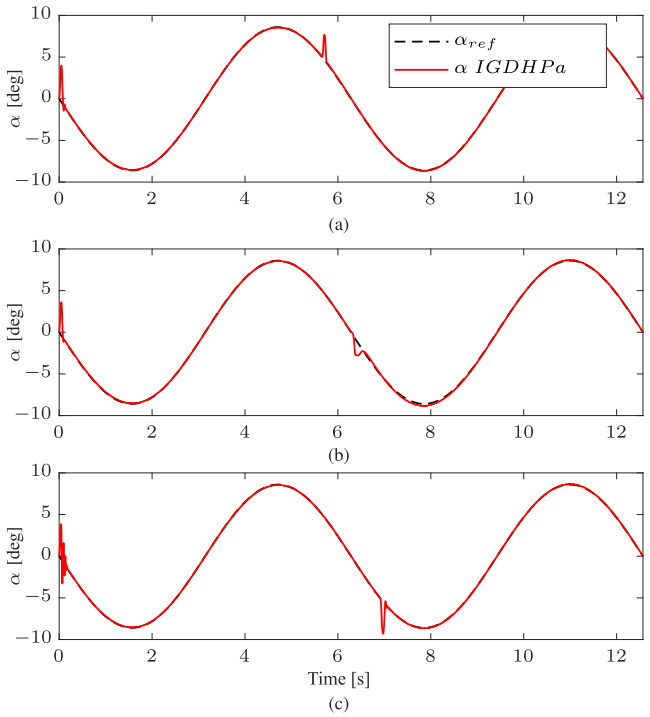

Fig. 12. Online tracking control using the IGDHPa algorithm in the presence of sudden changes in the model at three different angle-of-attack values. (a) Presence of sudden changes at $\alpha = 5°$. (b) Presence of sudden changes at $\alpha = 0°$. (c) Presence of sudden changes at $\alpha = -5°$.

normal distribution $\mathcal{N}(0, 1)$, and those of the top layer are randomly chosen from the range $(-0.01, 0.01)$. In this section, we used IGDHPa to denote this algorithm, where "I" indicates the incremental model and "a" indicates the proposed critic design by association.

In this experiment, the IGDHPa algorithm is applied to an online reference tracking task. To be more specific, the controller is required to control the angle of attack α to track the reference signal α_{ref}, which is a sinusoidal function of time within 2 periods of the reference signal (4π seconds). This online learning controller does not have any knowledge of the system model but only the measurements of the system state and input. Another task is fault-tolerant control with sudden changes in the system model: the changes in signs of the C_{z1}, C_{m1}, b_2, and b_4 terms in (44). These sudden changes may lead to an unstable open-loop plant, and the policy trained with the original system may even increase the instability of the closed loop plant. Therefore, the actor weights will be reset to small, random numbers when the fault is detected [26].

Fig. 12 presents the online training control result using the IGDHPa algorithm in the presence of the aforementioned sudden changes in the system dynamical model. These changes are introduced after the convergence of the policy for the original system. This figure showed a successful GDHP simulation result in online control and fault-tolerant control tasks.

In comparison to the IDHP algorithm we proposed [26], the averaged settling time of using the IGDHPa algorithm does not have significant improvement. This is because the settling time is also constrained by the learning efficiency of the actor. However, it is found that the success rate is increased from 91.1% to 94.6%, and the run time of each training episode with IGDHPa is reduced by 10%–20% compared to the IDHP algorithm. The main reason is the random features used in the critic of the proposed GDHP method, which not only increased the learning efficiency but also, to some extent, prevented some intractable problems associated with MLP BP, such as falling into the local minima trap and sudden growth to infinity weights.

V. Conclusion

This article proposed a new GDHP design based on an HDP-style critic and its associated dual network. The critic and dual networks have random fixed features and share the same set of parameters as association explicitly and precisely. This GDHP design can be seen as a special variation of the originally proposed GDHP design, but the dual network is an explicit expression, and the second-order mixed derivatives can be obtained with the feedforward calculation of the dual network. The accuracy of this proposed method is consistent with using the explicit formulas, while the structure and complexity are of the same level as the mixed-style critic. Therefore, this proposed design is able to increase the learning efficiency and feasibility of GDHP while retaining its analytical accuracy.

To examine the learning ability and efficiency of the proposed GDHP design, this article conducted three simulation experiments. The first experiment results illustrated that the neural network with random fixed features, by increasing its width, can have an equal level of accuracy as optimal features. The result of the second experiment revealed that our proposed method guarantees that the analytical calculation of the derivatives from the approximated cost matches the direct approximation of the derivatives, which outperforms the mixed-style critic design, especially in the presence of measurement noise. Also, the last experiment validated the feasibility of our proposed GDHP algorithm with an online reference tracking control task on a simplified missile model.

This article offered an option to efficient online GDHP with method development, theoretical analysis, and some simulation experiments. Further research is recommended to be undertaken in the following areas: 1) this method will be validated on more complex and realistic control problems with wider networks; 2) the complex structure of conventional GDHP designs causes overfitting and heavy computational loads, which prevent its extension in deep reinforcement learning. As the proposed GDHP critic can be expanded to multiple layers with random fixed features, further development of deep GDHP and its applications is recommended; and 3) neural networks with gradient descent often suffer from some intractable problems, such as falling into local minima trap and overfitting. Therefore, further research might explore linear optimization methods to train the critic and investigate more powerful approximators, such as multivariate simplex splines.

References

[1] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1998.
[2] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] R. Enns and J. Si, "Helicopter trimming and tracking control using direct neural dynamic programming," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 929–939, Aug. 2003.

[4] T. Hanselmann, L. Noakes, and A. Zaknich, "Continuous-time adaptive critics," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 631–647, May 2007.

[5] F.-Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *IEEE Comput. Intell. Mag.*, vol. 4, no. 2, pp. 39–47, May 2009.

[6] J. Si, *Handbook of Learning and Approximate Dynamic Programming*, vol. 2. Hoboken, NJ, USA: Wiley, 2004.

[7] S. G. Khan, G. Herrmann, F. L. Lewis, T. Pipe, and C. Melhuish, "Reinforcement learning and optimal adaptive control: An overview and implementation examples," *Annu. Rev. Control*, vol. 36, no. 1, pp. 42–59, Apr. 2012.

[8] S. Ferrari and R. F. Stengel, "Online adaptive critic flight control," *J. Guid., Control, Dyn.*, vol. 27, no. 5, pp. 777–786, Sep./Oct. 2004.

[9] V. Yadav, R. Padhi, and S. N. Balakrishnan, "Robust/optimal temperature profile control of a high-speed aerospace vehicle using neural networks," *IEEE Trans. Neural Netw.*, vol. 18, no. 4, pp. 1115–1128, Jul. 2007.

[10] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 997–1007, Sep. 1997.

[11] D. Wang, H. He, and D. Liu, "Adaptive critic nonlinear robust control: A survey," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3429–3451, Oct. 2017.

[12] X. Yang and H. He, "Adaptive critic learning and experience replay for decentralized event-triggered control of nonlinear interconnected systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 11, pp. 4043–4055, Nov. 2020.

[13] C. Mu, Z. Ni, C. Sun, and H. He, "Air-breathing hypersonic vehicle tracking control based on adaptive dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 584–598, Mar. 2017.

[14] Q. Zhao, J. Si, and J. Sun, "Online reinforcement learning control by direct heuristic dynamic programming: From time-driven to event-driven," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Feb. 3, 2021, doi: 10.1109/TNNLS.2021.3053037.

[15] X. Yang and Q. Wei, "Adaptive critic learning for constrained optimal event-triggered control with discounted cost," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 91–104, Jan. 2021.

[16] W. Bai, T. Li, Y. Long, and C. L. P. Chen, "Event-triggered multigradient recursive reinforcement learning tracking control for multiagent systems," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jul. 16, 2021, doi: 10.1109/TNNLS.2021.3094901.

[17] K. G. Vamvoudakis and F. L. Lewis, "Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[18] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, "Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 764–773, May 2002.

[19] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 264–276, Mar. 2001.

[20] T. J. J. Lombaerts, E. R. Van Oort, Q. P. Chu, J. A. Mulder, and D. A. Joosten, "Online aerodynamic model structure selection and parameter estimation for fault tolerant control," *J. Guid., Control, Dyn.*, vol. 33, no. 3, pp. 707–723, May 2010.

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

[22] G. G. Yen and P. G. DeLima, "Improving the performance of globalized dual heuristic programming for fault tolerant control through an online learning supervisor," *IEEE Trans. Autom. Sci. Eng.*, vol. 2, no. 2, pp. 121–131, Apr. 2005.

[23] S. Al-Dabooni and D. C. Wunsch, "Online model-free n-step HDP with stability analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1255–1269, Apr. 2020.

[24] Z. Ni, H. He, X. Zhong, and D. V. Prokhorov, "Model-free dual heuristic dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 8, pp. 1834–1839, Aug. 2015.

[25] Y. Zhou, E. van Kampen, and Q. P. Chu, "Nonlinear adaptive flight control using incremental approximate dynamic programming and output feedback," *J. Guid., Navigat. Dyn.*, vol. 40, no. 2, pp. 493–500, 2017.

[26] Y. Zhou, E.-J. van Kampen, and Q. P. Chu, "Incremental model based online dual heuristic programming for nonlinear adaptive control," *Control Eng. Pract.*, vol. 73, pp. 13–25, Apr. 2018.

[27] Y. Zhou, E.-J. van Kampen, and Q. Chu, "Incremental model based online heuristic dynamic programming for nonlinear adaptive tracking control with partial observability," *Aerosp. Sci. Technol.*, vol. 105, Oct. 2020, Art. no. 106013.

[28] M. Fairbank, E. Alonso, and D. Prokhorov, "Simple and fast calculation of the second-order gradients for globalized dual heuristic dynamic programming in neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 10, pp. 1671–1676, Oct. 2012.

[29] B. Sun and E.-J. van Kampen, "Incremental model-based global dual heuristic programming with explicit analytical calculations applied to flight control," *Eng. Appl. Artif. Intell.*, vol. 89, Mar. 2020, Art. no. 103425.

[30] D. Liu, D. Wang, D. Zhao, Q. Wei, and N. Jin, "Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 3, pp. 628–634, Jul. 2012.

[31] X. Zhong, Z. Ni, and H. He, "Gr-GDHP: A new architecture for globalized dual heuristic dynamic programming," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3318–3330, Oct. 2017.

[32] C. Mu, C. Sun, A. Song, and H. Yu, "Iterative GDHP-based approximate optimal tracking control for a class of discrete-time nonlinear systems," *Neurocomputing*, vol. 214, pp. 775–784, Nov. 2016.

[33] J. Yi, S. Chen, X. Zhong, H. He, and W. Zhou, "Event-triggered globalized dual heuristic programming and its application to networked control systems," *IEEE Trans. Ind. Informat.*, vol. 15, no. 3, pp. 1383–1392, Mar. 2019.

[34] Q. Wei, L. Zhu, R. Song, P. Zhang, D. Liu, and J. Xiao, "Model-free adaptive optimal control for unknown nonlinear multiplayer nonzero-sum game," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 2, pp. 879–892, Feb. 2022.

[35] P. J. Werbos, "A menu of designs for reinforcement learning over time," in *Neural Networks for Control*, W. T. Miller, R. S. Sutton, and P. J. Werbos, Eds. Cambridge, MA, USA: MIT Press, 1995, pp. 67–96.

[36] P. Werbos, "Backpropagation: Past and future," in *Proc. 2nd Int. Conf. Neural Netw.*, vol. 1, 1988, pp. 343–353.

[37] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Proc. NIPS*, 2008, pp. 1313–1320.

[38] W. W. Bai, T. S. Li, and S. C. Tong, "NN reinforcement learning adaptive control for a class of nonstrict-feedback discrete-time systems," *IEEE Trans. Cybern.*, vol. 50, no. 11, pp. 4573–4584, Nov. 2020.

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

[40] C. C. de Visser, Q. P. Chu, and J. A. Mulder, "A new approach to linear regression with multivariate splines," *Automatica*, vol. 45, no. 12, pp. 2903–2909, Dec. 2009.

[41] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. NIPS*, 2007, vol. 3, no. 4, p. 5.

[42] S.-H. Kim, Y.-S. Kim, and C. Song, "A robust adaptive nonlinear control approach to missile autopilot design," *Control Eng. Pract.*, vol. 12, no. 2, pp. 149–154, Feb. 2004.

[43] L. Sonneveldt, "Adaptive backstepping flight control for modern fighter aircraft," Ph.D. dissertation, Dept. Control Simul., Fac. Aerosp. Eng., Delft Univ. Technol., Delft, The Netherlands, 2010.

Ye Zhou received the B.S. and M.S. degrees (Hons.) from the School of Mechanical and Electrical Engineering, Northwestern Polytechnical University, Xi'an, China, in 2010 and 2013, respectively, and the Ph.D. degree in control and simulation, from the Faculty of Aerospace Engineering, Delft University of Technology, Delft, The Netherlands, in 2018.

Her Ph.D. research topic was online reinforcement learning control for aerospace systems. She was a Lecturer with the Faculty of Aerospace Engineering, Delft University of Technology, from 2017 to 2018. She is currently a Senior Lecturer with the School of Aerospace Engineering, Universiti Sains Malaysia, and a Guest Researcher with the Faculty of Aerospace Engineering, Delft University of Technology. Her research interests lie in reinforcement learning, nonlinear control, adaptive control, intelligent control, guidance, and navigation.