Abstract—This paper addresses reinforcement learning based, direct signal tracking control with an objective of developing mathematically suitable and practically useful design approaches. Specifically, we aim to provide reliable and easy to implement designs in order to reach reproducible neural network-based solutions. Our proposed new design takes advantage of two control design frameworks: a reinforcement learning based, data-driven approach to provide the needed adaptation and (sub)optimality, and a backstepping based approach to provide closed-loop system stability framework. We develop this work based on an established direct heuristic dynamic programming (dHDP) learning paradigm to perform online learning and adaptation and a backstepping design for a class of important nonlinear dynamics described as Euler-Lagrange systems. We provide a theoretical guarantee for the stability of the overall dynamic system, weight convergence of the approximating nonlinear neural networks, and the Bellman (sub)optimality of the resulted control policy. We use simulations to demonstrate significantly improved design performance of the proposed approach over the original dHDP.

Index Terms—Reinforcement learning, tracking control, direct heuristic dynamic programming (dHDP), backstepping.

I. INTRODUCTION

We consider the problem of data-driven optimal tracking control for discrete-time nonlinear Euler-Lagrange systems which are represented in a wide range of applications, such as robotic manipulators [1], wearable robots [2], aircraft systems [3], ground vehicles [4], and many more applications in mechanical, electrical and electromechanical systems [5]. Unlike its counter-part problem for linear systems, or the regulation control problems, there are only a handful of results on data-driven nonlinear optimal tracking control problems. Common and natural approaches to the existing designs are the use of machine learning or neural networks techniques as they can learn directly from data by means of universally approximating properties.

The model based nonlinear tracking control problem has been studied extensively. Well-established approaches include backstepping control [6], observer-based control [7] and nonlinear adaptive/robust control [8], [9], some of which have been demonstrated for applicability in specialized applications [10]. The recently emerged data-driven design approaches have also been extensively addressed [11]. However, these results focus on the stabilization of nonlinear dynamic systems. As is well-known, real engineering applications require considerations of optimal control performance, not just stability to account for factors such as energy consumption, tracking error rate, and more. Therefore, optimal tracking control solutions of nonlinear systems are sought after. A classical formulation of the problem is to obtain nonlinear optimal tracking control solutions from solving the Hamilton-Jacobi-Bellman (HJB) equation. But solving the HJB equation poses great challenges for general nonlinear systems. One such challenge is a lack of closed-form analytical solution even if a mathematical description of the nonlinear dynamics is available. Additionally, traditional approaches to solving the HJB equations are backward in time and therefore, can only be solved offline. Data-driven nonlinear optimal tracking control designs provide new promises to address these challenges, yet, they face new obstacles.

Currently, there is only a handful of results concerning the theory, algorithm design and implementation of data-driven optimal tracking control. Central to these results [12]–[29] are the use of reinforcement learning to establish an approximate solution to the HJB equation. Yet, few results have demonstrated that these methods are not only mathematically suitable but also practically useful in order to address real engineering problems in terms of providing reliable, easy to implement designs that lead to reproducible neural network-based solutions.

In addressing the design of reinforcement learning based optimal tracking of coal gasification problem, the authors of [15] first applied offline neural network identifications to establish the necessary mathematical descriptions of the nonlinear system dynamics and the desired tracking trajectory in order to carry on the control design. As such, it is questionable if approaches based on similar ideas can potentially be useful for other applications as obtaining reproducible models will be the first barrier to overcome. This is not a trivial problem, as it requires great expertise and the subject is still under investigation because current neural network modeling results usually introduce large variances which depend on the designer and the hyperparameters used in learning the models.

Most existing reinforcement learning based tracking control
design relies on a reference model from which a (continuously differential) desired tracking signal $x_d$ can be obtained [12–26]. While it may be feasible and useful for certain applications such as flight validation [30], a directly specified reference signal $x_d$ is conceivably helpful to broaden the impact of tracking control for many practical applications. When a reference model is required to define a desired trajectory, it is almost surely to introduce additional uncertainties to the system under control. Some reported results [12–26] also require generating a corresponding reference control, which can also be challenging and make some of these approaches less applicable. Another common feature in many existing learning based tracking control is the assumption that the nonlinear dynamics are partially known (or specifically the input dynamics are known). Such an assumption is needed to directly formulate an optimal control policy in closed from. This has simplified the problem but in the meantime, it limits the applicability of these design techniques to some extent.

In this paper, we aim at developing a new nonlinear dHDP-based tracking control framework with its applicability in mind, that is to say we aim for a design framework that is mathematically suitable and practically useful. In a previous study [31], we have shown that well initialized actor-critic neural networks in dHDP can significantly improve the quality of the optimal value function approximation and the optimal control policy. From the same study we realized the importance of finding a good quality (locally) optimal solution to the Bellman equation. In this study, our development aims at providing reliable design approaches to nonlinear tracking control by taking advantage of the best of the two worlds: a backstepping control strategy to provide a feedback system stability framework and a dHDP online adaptation control strategy to provide a feed-forward compensation in order to obtain near optimal tracking control solution. In the feedback control performance objective, we take into account the feed-forward control input. We therefore can provide an overall system stability guarantee. Because of our innovative solution construct, we avoid the use of a reference model for the desired trajectory to remove one source of approximation error. Additionally, we address the issue of unknown nonlinear dynamics via learning from data, i.e., our design is data-driven, not model based.

The authors of [27] proposed a tracking control solution based on dHDP in [32], which makes use of an additional neural network to provide an internal goal signal. This is theoretically suitable but it complicates the problem as discussed previously as it almost surely introduces additional errors to the solution. These errors are on top of the uncertainties introduced by variances due to dHDP tracking solutions as pointed out in [31]. As such, the reproducibility of the approach is yet to be demonstrated systematically.

To directly use a reference trajectory in place of a model reference structure, backstepping idea can be employed as it allows for the construction of both feedback control laws and associated Lyapunov functions in a systematic way [6]. The backstepping idea in this context was examined in [28], [29] with critic-only reinforcement learning control. However, their results are limited to partially known system dynamics. Alternatively, the dHDP construct allows for direct use of reference trajectory as well in the design of tracking control. The idea was demonstrated via simulations in [13], [27] where a dual critic network design was proposed based upon dHDP. The results are promising, yet, they lack a theoretical support for (sub)optimality and stability analysis. Even though tracking control results in [13] were obtained for dHDP as well, our current approach is fundamentally different as we propose a new control strategy and we use an informative tracking error based stage cost instead of a binary cost. Specifically, the current study is motivated to circumvent some long-standing issues in reinforcement learning, including Q-learning and dHDP, that is to reduce the variances in the resulted action and/or critic networks after training and thus to improve the reproductivity and consistency of the trained actor/critic network outcomes even if they are trained by different designers.

Our contributions of this work are as follows.

1) We introduce a new reinforcement learning control design framework that is within a backstepping feedback construct. Specifically, the dHDP is to provide the feed-forward compensation for unknown system dynamics. This also removes our dependence on a reference model for the desired tracking trajectory.

2) We provide a theoretical guarantee for the stability of the overall dynamic system, weight convergence of the approximating nonlinear neural networks, and the Bellman (sub)optimality of the resulted control policy.

3) We provide systematic simulations to not only demonstrate how the proposed design method works but also to show how the proposed algorithm can significantly improve reproducibility of the results under dHDP integrated with backstepping feedback stabilizing control.

The rest of the paper is organized as follows. Section II provides the problem formulation. Section III presents the backstepping design. Section IV develops the reinforcement learning control. Section V provides theoretical analyses of the proposed algorithm. Simulation and comparison results are presented in Section VI and the concluding remarks are given in Section VII.

II. PROBLEM FORMULATION

We consider a class of nonlinear dynamics described as Euler-Lagrange systems, the behavior of which can be used to describe several important engineering systems, such as robotic manipulators, wearable robots, aircraft systems, ground vehicles, to name just a few [5]:

$$M(q)\ddot{q} + V_m(q, \dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d(t) = \tau(t).$$  \hspace{1cm} (1)

In (1), $q(t), \dot{q}(t), \ddot{q}(t) \in \mathbb{R}^n$ denote the link position, velocity, and acceleration vectors, respectively; $M(q) \in \mathbb{R}^{n \times n}$ denotes the inertia matrix; $V_m(q, \dot{q}) \in \mathbb{R}^{n \times n}$ the centripetal-coriolis matrix; $G(q) \in \mathbb{R}^n$ the gravity vector; $F(\dot{q}) \in \mathbb{R}^n$ friction; $\tau_d(t) \in \mathbb{R}^n$ a general nonlinear disturbance; and $\tau(t) \in \mathbb{R}^n$ represents the torque control input. The subsequent development is based on the assumption that $q(t)$ and $\dot{q}(t)$ are measurable; and $M(q), V_m(q, \dot{q}), G(q), F(\dot{q})$ and $\tau_d(t)$ are unknown.
To carry on the development of tracking control of the dynamics in (1), we use the following general discrete-time state space representations where \( x_1(k) \in \mathbb{R}^n \) and \( x_2(k) \in \mathbb{R}^n \) denote link position and velocity, respectively. Applying the Euler discretization as in [33], systems (1) can be rewritten as

\[
x_1(k+1) = h x_2(k) + x_1(k),
\]

\[
M^x x_2(k+1) = u(k) - g(k) - \tau_d(k),
\]

where \( h = t_{k+1} - t_k, u(k) = \tau(t_k), g(k) = v_n(k)x_2(k) + G(k) + F(k) - M^x x_2(k), M^x \) denotes \( M(t_{k+1})/h \) and \( M^- \) denotes \( M(t_k)/h \). The control objective is for the output \( x_1(k) \) to track a desired time-varying trajectory \( x_{1d}(k) \) as closely as possible.

**Assumption 1.** The inertia matrix \( M(q) \) is symmetric, positive definite, and the following inequality holds:

\[
M_{\min} \leq (M^-)^T (M^-),
\]

where \( M_{\min} \) is a known positive constant, and \( \| \cdot \| \) denotes the standard Euclidean norm.

**Assumption 2.** The nonlinear disturbance term \( \tau_d(k) \) is bounded, i.e., \( \tau_d(k) \in \mathcal{L}_\infty \).

**Remark 1.** In subsequent development, the desired trajectory \( x_{1d}(k) \) is not defined by any reference model, but rather, \( x_{1d}(k) \) is simply and directly provided. Our proposed new closed-loop, data-driven nonlinear tracking control solution is as shown in Fig. 1. There are two major components in the design. The backstepping design provides a feedback control structure, the feed-forward signal from the backstepping framework is then accounted for by using a dHDP online learning scheme. Even though the dHDP alone can theoretically be used for tracking control while preserving some qualitative properties such as those in [13], our current approach focus on improving reliability of the design and reproducibility of the results while maintaining theoretical suitability. Next, we provide a comprehensive introduction of the two constituent control design blocks.

### III. Backstepping To Provide Baseline Tracking Control

The control goal is to find \( u(k) \) (Fig.1) so that the output \( x_1(k) \) of the system (2) tracks the desired time-varying trajectory \( x_{1d}(k) \). The discrete-time backstepping scheme for the tracking problem of (2) with unknown system dynamics \( M^+(k), g(k) \) and the disturbance \( \tau_d(k) \) are unknown) is derived step by step as follows. Refer to Fig.1, the two blocks play complementary roles in constructing the final control signal \( u(k) \). The backstepping control block is designed as follows.

**Step 1.** To develop the backstepping design, a virtual control function is synthesized [6]. Let \( e_1(k) \in \mathbb{R}^n \) be the deviation of \( x_1(k) \) from the target \( x_{1d}(k) \in \mathbb{R}^n \), i.e.,

\[
e_1(k) = x_1(k) - x_{1d}(k).
\]

From (2) and (4), we have

\[
e_1(k+1) = h x_2(k) + x_1(k) - x_{1d}(k+1).
\]

Then, we view \( x_2(k) \) as a virtual control in (5) and introduce the error variable

\[
e_2(k) = x_2(k) - \alpha(k),
\]

where \( \alpha(k) \in \mathbb{R}^n \) is a stabilizing function for \( x_2(k) \) to be chosen as

\[
\alpha(k) = \frac{1}{h}(c_1 e_1(k) - x_1(k) + x_{1d}(k+1)),
\]

where \( c_1 \) is a design constant. Then equation (5) becomes

\[
e_1(k+1) = c_1 e_1(k) + h e_2(k).
\]

**Step 2.** The final control law is synthesized to force \( e_2(k) \) to tend towards zero or a small value. The error variable \( e_2(k) \) as the forward signal is written as

\[
M^+ e_2(k+1) = M^+ x_2(k+1) - M^+ \alpha(k+1) + \tau_d(k),
\]

where \( \alpha(k+1) \) is written as:

\[
\alpha(k+1) = \frac{1}{h}(c_1 (e_1(k) + h e_2(k)) - h x_2(k) - x_1(k) + x_{1d}(k+2)).
\]

The control input \( u(k) \) is selected as

\[
u(k) = \hat{f}(k) + c_2 e_2(k)
\]

where \( c_2 \) is a design constant, \( \hat{f}(k) \) is an estimate of the combined unknown dynamic \( f(k) \) by neural networks, and \( f(k) \) is written as

\[
f(k) = g(k) + M^+ \alpha(k+1).
\]

Substituting (11) and (12) into (9) yields

\[
M^+ e_2(k+1) = c_2 e_2(k) + \hat{f}(k) - \tau_d(k),
\]

where

\[
\hat{f}(k) = \hat{f}(k) - f(k).
\]

**Fig. 1.** Schematic diagram of the proposed tracking control framework.

### IV. Reinforcement Learning To Provide Data-driven Feed-forward Input Control

The dHDP block is used to provide control input for unaddressed dynamics due to lack of a mathematical description of system (1). Here we use dHDP as a basic structural framework of adaptive critic designs to approximate the cost-to-go function and the optimal control policy at the same time [32], [34]. The dHDP takes advantage of the potential scalability of the adaptive critic designs and the intuitiveness of Q-learning [35]. Moreover, comparing with other reinforcement learning approaches, the dHDP has been shown a feasible tool for
solving complex and realistic problems including the stabilization, tracking, reconfiguring control of Apache helicopters [36]–[38], damping low frequency oscillations in large power systems [39], and wearable robots with human in the loop [40]–[42]. We therefore consider the dHDP can potentially provide the necessary application in the backstepping design and together, we provide a new learning control method that improves the reproducibility of results when applied to meaningful real applications.

A. Basic Definitions

The dHDP based reinforcement learning supplemental control is as shown in Fig. 1. The stage cost \( r(k) \) is defined as

\[
r(k) = e_t^T(k)Qe_1(k) + u^T(k)Ru(k),
\]

where \( Q \in \mathbb{R}^{n \times n} \), \( R \in \mathbb{R}^{n \times n} \) are positive semi-definite matrices. Then, the cost-to-go \( J(k) \) is written as

\[
J(k) = \sum_{k=0}^{\infty} \gamma^k r(k + 1),
\]

where \( 0 < \gamma < 1 \) is a discount factor for the infinite-horizon tracking problem. We require \( r(k) \) to be a semi-definite function of the output error \( e_1(k) \) and control \( u(k) \), so the cost function is well-defined. Based on (16), we formulate the following Bellman equation:

\[
J(k - 1) = \gamma J(k) + r(k).
\]

B. Actor-Critic Networks

The dHDP design follows that in [32] with an actor neural network and a critic neural network. Hyperbolic tangent is used as the transfer function in the actor-critic networks to approximate the control policy and the cost-to-go function.

1) Critic Neural Network: The critic neural network (CNN) consists of three layers of neurons, namely the input layer, the hidden layer and the output layer. The input of CNN is

\[
x_c(k) = \begin{bmatrix} x_a(k) \\ u(k) \end{bmatrix},
\]

where

\[
x_a(k) = [x_1(k), x_2(k), e_1(k), e_2(k), x_{1d}(k+2)]^T.
\]

The output of CNN is defined as

\[
f(k) = \hat{w}_{c2}(k) \ast \phi(\hat{w}_{c1}(k) \ast x_c(k)) = \hat{w}_{c2} \phi_c(k),
\]

where \( \hat{w}_{c1} \) is the estimated weight matrix between the input layer and the hidden layer, and \( \hat{w}_{c2} \) is the estimated weight matrix between the hidden layer and the output layer, and \( \phi(\cdot) \) is the hyperbolic tangent activation function,

\[
\phi(v) = \frac{1 - \exp(-v)}{1 + \exp(-v)}.
\]

From (17), the prediction error \( e_c(k) \) is used to estimate weights

\[
e_c(k) = \gamma \hat{f}(k) - [\hat{f}(k-1) - r(k)].
\]

Specifically, the weights of CNN are updated to minimize the following approximation error

\[
E_c(k) = \frac{1}{2} e_c^T(k)e_c(k).
\]

Gradient descent is used to adaptive the critic network weights are given by applying chain rule and back-propagation [43]. For the input-to hidden layer,

\[
\Delta \hat{w}_{c1}(k) = l_c \left[ -\frac{\partial E_c(k)}{\partial \hat{w}_{c1}(k)} \right]
\]

\[
\frac{\partial E_c(k)}{\partial \hat{w}_{c1}(k)} = \frac{\partial E_c(k)}{\partial \hat{J}(k)} \frac{\partial \hat{J}(k)}{\partial \phi_c(k)} \frac{\partial \phi_c(k)}{\partial \hat{w}_{c1}(k)}
\]

\[
= \gamma e_c(k) \hat{w}_{c2}(k) \left[ \frac{1}{2} \left( 1 - \phi_c^2(k) \right) \right] x_c(k),
\]

Similarly for the hidden-to-output layer,

\[
\Delta \hat{w}_{c2}(k) = l_c \left[ -\frac{\partial E_c(k)}{\partial \hat{w}_{c2}(k)} \right]
\]

\[
\frac{\partial E_c(k)}{\partial \hat{w}_{c2}(k)} = \frac{\partial E_c(k)}{\partial \hat{f}(k)} \frac{\partial \hat{f}(k)}{\partial \phi_c(k)} \frac{\partial \phi_c(k)}{\partial \hat{w}_{c1}(k)} \frac{\partial \hat{w}_{c1}(k)}{\partial \hat{w}_{c2}(k)}
\]

\[
= \gamma e_c(k) \hat{w}_{c2}(k) \hat{w}_{c1}(k) \left[ \frac{1}{2} \left( 1 - \phi_c^2(k) \right) \right] x_c(k).
\]

In the above, \( l_c > 0 \) is the learning rate.

2) Action Neural Network: In this algorithm, the action neural network (ANN) is to approximate the unknown dynamics \( f(k) \) in (12). The input to the ANN is \( x_a(k) \), the respective output is given as follows:

\[
f(k) = \hat{w}_{a2}(k) \ast \phi(\hat{w}_{a1}(k) \ast x_a(k)) = \hat{w}_{a2} \phi_a(k),
\]

where \( \hat{w}_{a1} \) and \( \hat{w}_{a2} \) are the estimated weight matrices.

The training of the ANN can be done by adapting the weights so as to minimize \( E_a(k) \), and \( E_a(k) \) is defined as

\[
E_a(k) = \frac{1}{2} e_a^T(k)e_a(k),
\]

where the prediction error \( E_a(k) \) for the action element is defined as

\[
e_a(k) = \hat{f}(k) + \tilde{f}(k) - U_c,
\]

In the above, \( U_c \) is the ultimate performance objective in the tracking control design paradigm, which is defined as \( U_c = 0 \); \( \tilde{f}(k) \) is the function approximation error shown in (14) and it can be rewritten as

\[
\tilde{f}(k) = M^* e_2(k+1) - c_2 e_2(k) + \tau_e(k).
\]

The desired tracking performance will be achieved if the function approximation error \( \tilde{f}(k) \) approaches 0.

The update rule is then similar to that in CNN, for the input-to hidden layer,

\[
\Delta \hat{w}_{a1}(k) = l_a \left[ -\frac{\partial E_a(k)}{\partial \hat{w}_{a1}(k)} \right]
\]

\[
\frac{\partial E_a(k)}{\partial \hat{w}_{a1}(k)} = \frac{\partial E_a(k)}{\partial \hat{f}(k)} \frac{\partial \hat{f}(k)}{\partial \phi_a(k)} \frac{\partial \phi_a(k)}{\partial \hat{w}_{a1}(k)}
\]

\[
= e_a(k) \left[ \hat{w}_{c2}(k) \frac{1}{2} \left( 1 - \phi_a^2(k) \right) \right] x_a(k),
\]
and for the hidden-to-output layer,
\[ \Delta \hat{w}_{a2}(k) = l_a \left[ - \frac{\partial E_a(k)}{\partial \hat{w}_{a2}(k)} \right] \]
\[ \frac{\partial E_a(k)}{\partial \hat{w}_{a2}(k)} = \frac{\partial E_a(k)}{\partial \hat{J}(k)} \frac{\partial \hat{J}(k)}{\partial u(k)} \frac{\partial \hat{f}(k)}{\partial J(k)} \frac{\partial J(k)}{\partial u(k)} \frac{\partial J(k)}{\partial \hat{w}_{a2}(k)} \]
\[ = e_a(k) \left[ \hat{w}_{c2}(k) \left( \frac{1}{2} (1 - \phi_2^2(k)) \right) \right] \phi_a(k), \]
where \( w_{ca}(k) \) is the weight associated with the input element from ANN, i.e., the part of \( w_{c1} \) which connect with \( u(k) \), and \( l_a > 0 \) is the learning rate.

**Remark 3.** As \( e_a(k) \) is formulated in (28), the term \( \hat{f}(k) \) contains the unknown disturbance \( \tau_e(k) \) and parameter \( M^+ \).

To implement the update rule, \( e_a(k) \) can be taken as \( e_a(k) = J + M^+ e_2(k) - c_2 e_2(k) \), where \( M^+ \) is unknown but can be initialized with a rough estimate in (29).

Algorithm 1 summarizes the implementation procedure of the dHDP-based tracking control.

**Algorithm 1.** Direct signal tracking control based on dHDP

Specify desired trajectory \( x_{id} \):
Initialization: \( w_{a}(0), w_{c}(0), x_{1}(0), x_{2}(0) \);
Set hyperparameters: \( l_a, l_c, c_1, c_2 \);
Calculate virtual control \( \alpha(0) \) according to (7);
Calculate \( e_2(0) \) according to (6);
Calculate \( r(0) \) according to (15);
**Backstepping design:**
Calculate \( \hat{f}(k) \) according to (26);
Calculate \( u(k) \) according to (11);
Take control input \( u(k) \) into (2);
Produce \( x(k+1), r(k+1) \) according to (2), (15);
**dHDP design:**
Obtain \( \alpha(k+1) \) by (7);
Calculate \( e_2(k+1) \) according to (6);
Calculate \( \hat{J}(k) \) according to (20);
Calculate \( e_a(k) \) according to equation (22);
\( w_{c}(k+1) = w_{c}(k) + \Delta w_{c}(k) \);
Calculate \( e_a(k) \) (Remark 3) ;
\( w_{a}(k+1) = w_{a}(k) + \Delta w_{a}(k) \);
Iterate until converge.

V. LYAPUNOV STABILITY ANALYSIS

In this section, we provide a theoretical analysis for the stability of the overall dynamic system, weight convergence of the actor and critic neural networks, and the Bellman (sub)optimality of the control policy.

A. Preliminaries

Let \( w^*_c, w^*_a \) denote the optimal weights, that is,
\[ w^*_a = \arg \min_{\hat{w}_a} \| \hat{J}(k) + \hat{f}(k) - U_c \|, \]
\[ w^*_c = \arg \min_{\hat{w}_c} \| \gamma \hat{J}(k) + r(k) - \hat{J}(k-1) \|. \]
(32)

Then, the optimal cost-to-go and control law can be expressed as
\[ u^*(k) = w^*_a \phi_a(k) + e_a(k), \]
\[ J^*(k) = w^*_c \phi_c(k) + e_c(k) \]
(33)

where \( e_a(k) \) and \( e_c(k) \) are the reconstruction errors of the actor and critic neural networks, respectively.

**Assumption 3.** The optimal weights for the actor-critic networks exist and they are bounded by two positive constants \( w_{am} \) and \( w_{cm} \), respectively.

\[ \| w^*_a \| \leq w_{am}, \| w^*_c \| \leq w_{cm}. \]
(34)

Then, the weight estimation errors of the actor and critic neural networks are described respectively
\[ \hat{w}_a(k) := \hat{w}_a(k) - w^*_a, \hat{w}_c(k) := \hat{w}_c(k) - w^*_c. \]
(35)

**Remark 4.** A weight parameter convergence result was obtained for the dHDP in [44] under the condition that the weights between the input and hidden layers remain unchanged during learning. The result was later extended to allowing for all the weights in the actor and critic networks to adapt during learning [45]. Another study [13] addressed tracking control using dHDP for a Brunovsky canonical system. Such a system may be mathematically interesting but practically limiting. Additionally, the design in [13] requires reference models. In this study, we take reference of [44] and [45] to prove our new results on weight convergence for tracking control. Note that [44] and [45] are about regulation control, not tracking control. Notice also that both works lack a system stability result.

**Lemma 1.** Under Assumption 3, consider the weight vector of the hidden-to-output layer in CNN. Let
\[ L_1(k) = \frac{1}{L_c} tr \left( (\hat{w}_{c2}(k))^T \hat{w}_{c2}(k) \right). \]
(36)

Then its first difference is given by
\[ \Delta L_1(k) = -\gamma^2 \| \zeta_c(k) \|^2 - (1 - \gamma^2 L_c \| \phi_c(k) \|^2) \times \| \gamma \hat{w}_{c2}(k) \phi_c(k) + r(k) - \hat{w}_{c2}(k-1) \phi_c(k-1) \|^2 + \| \gamma w^*_c \phi_c(k) + r(k) - \hat{w}_{c2}(k-1) \phi_c(k-1) \|^2, \]
(37)
where \( \zeta_c(k) = \hat{w}_{c2}(k) \phi_c(k) \) is an approximation error of the critic output.

**Proof of Lemma 1.** The first difference of \( L_1(k) \) can be written as
\[ \Delta L_1(k) = \frac{1}{L_c} tr \left[ (\hat{w}_{c2}(k+1))^T \hat{w}_{c2}(k+1) \right. \]
\[ - (\hat{w}_{c2}(k))^T \hat{w}_{c2}(k). \]
(38)

With the updating rule in (25), \( \hat{w}_{c2}(k+1) \) can be rewritten as
\[ \hat{w}_{c2}(k+1) = \hat{w}_{c2}(k) - w^*_c \]
\[ = \hat{w}_{c2}(k) - \gamma \left( \frac{1}{\gamma} \phi_c(k) \gamma \hat{w}_{c2}(k) \phi_c(k) \phi_c(k) \right) + r(k) - \hat{w}_{c2}(k-1) \phi_c(k-1) \]
(39)

Then, the first term in the brackets of (38) can be given as
\[ tr \left[ (\hat{w}_{c2}(k+1))^T \hat{w}_{c2}(k+1) \right. \]
\[ = (\hat{w}_{c2}(k))^T \hat{w}_{c2}(k) - 2 \gamma \hat{w}_{c2}(k) \phi_c(k) \]
\[ \times \| \gamma \hat{w}_{c2}(k) \phi_c(k) + r(k) - \hat{w}_{c2}(k-1) \phi_c(k-1) \|^2 + \gamma^2 L_c^2 \| \phi_c(k) \|^2 \times \| \gamma \hat{w}_{c2}(k) \phi_c(k) + r(k) - \hat{w}_{c2}(k-1) \phi_c(k-1) \|^2. \]
(40)
As \( \hat{w}_{c2}(k) \phi_c(k) \) is a scalar, by using (35), we can rewrite the middle term in the above formula as follows:

\[
\begin{align*}
&-2\gamma l_c \hat{w}_{c2}(k) \phi_c(k) [\gamma \hat{w}_{c2}(k) \phi_c(k) + r(k) \\
&- \hat{w}_{c2}(k-1) \phi_c(k-1)] \\
= & l_c (||\gamma \hat{w}_{c2}(k) \phi_c(k) + r(k) - \hat{w}_{c2}(k-1) \phi_c(k-1) \\
&- \gamma \hat{w}_{c2}(k) \phi_c(k) ||^2 - ||\gamma \hat{w}_{c2}(k) \phi_c(k) ||^2 \\
&- ||\gamma \hat{w}_{c2}(k) \phi_c(k) + r(k) - \hat{w}_{c2}(k-1) \phi_c(k-1)||^2) \
&= -2\gamma l_c ||\hat{w}_{c2}(k) \phi_c(k) + r(k) - \hat{w}_{c2}(k-1) \phi_c(k-1)||^2 \\
&- \gamma \hat{w}_{c2}(k) \phi_c(k) + r(k) - \hat{w}_{c2}(k-1) \phi_c(k-1) \\
&- \hat{w}_{c2}(k-1) \phi_c(k-1) ||^2.
\end{align*}
\]

Substituting (40) and (41) into (38), we obtain Lemma 1.

**Lemma 2.** Under Assumption 3, consider the weight vector of the hidden-to-output layer in ANN. Let

\[
L_2(k) = \frac{1}{l_c} \beta_1 \Gamma (|w_{a2}(k)||^2 (|w_{a2}(k)|^2) \\
\times ||w_{a2}(k) \phi_c(k) + \zeta_r(k) - d(k)||^2 + 8||\zeta_r(k)||^2 \\
+ 8l_c ||w_{a2}(k) \phi_c(k) + \zeta_r(k) - d(k)||^2 \\
+ ||\hat{w}_{c2}(k) \phi_C(k) + \zeta_r(k) - d(k)||^2),
\]

then its first difference is bounded by

\[
\Delta L_2(k) \leq \frac{1}{\beta_1} \left( (1 - l_c ||\phi_c(k)||^2 ||w_{a2}(k)||^2) \\
\times ||w_{a2}(k) \phi_c(k) + \zeta_r(k) - d(k)||^2 + 8||\zeta_r(k)||^2 \\
+ 8l_c ||w_{a2}(k) \phi_c(k) + \zeta_r(k) - d(k)||^2 \\
+ ||\hat{w}_{c2}(k) \phi_C(k) + \zeta_r(k) - d(k)||^2 \right),
\]

where \( \zeta_r(k) = \hat{w}_{a2}(k) \phi_c(k) \) is an approximation error of the action network output; \( C(k) = \frac{1}{2} (1 - \phi_c^2(k)) w_{a2}(k) \); \( \beta_1 > 0 \) is a weighting factor; and the lumped disturbance \( d(k) = (M^+ - M) \epsilon_{c2}(k) + \tau_r(k) + \epsilon_r(k). \)

**Proof of Lemma 2.** The first difference of \( L_2(k) \) can be written as

\[
\Delta L_2(k) = \frac{1}{l_c} \beta_1 \Gamma (|w_{a2}(k+1)||^2 (|w_{a2}(k+1)|^2) \\
- (|w_{a2}(k)||^2 (|w_{a2}(k)||^2)),
\]

with the updating rule in (31), \( \hat{w}_{a2}(k+1) \) can be rewritten as

\[
\hat{w}_{a2}(k+1) = \hat{w}_{a2}(k) - l_c \phi_c(k) \hat{w}_{c2}(k) \phi_c(k) \\
+ \zeta_r(k) - d(k)) ||^2 - \hat{w}_{a2}(k),
\]

based on this expression, it is easy to obtain that

\[
\begin{align*}
&= (\hat{w}_{a2}(k)||^2 \hat{w}_{a2}(k) + l_c ||\phi_c(k)||^2 ||\hat{w}_{c2}(k) \phi_c(k)||^2 \\
&\times ||\hat{w}_{c2}(k) \phi_c(k) + \zeta_r(k) - d(k)||^2 \\
&- 2l_c \hat{w}_{c2}(k) \phi_c(k) \hat{w}_{c2}(k) \phi_c(k) + \zeta_r(k) - d(k)) ||^2 \zeta_r(k). \\
\end{align*}
\]

Substituting (46) into (44), we have

\[
\Delta L_2(k) = \frac{1}{\beta_1} \left( (l_c ||\phi_c(k)||^2 ||w_{a2}(k)||^2) \\
\times ||w_{a2}(k) \phi_c(k) + \zeta_r(k) - d(k)||^2 \\
- 2\hat{w}_{c2}(k) \phi_c(k) \hat{w}_{c2}(k) \phi_c(k) + \zeta_r(k) \\
- d(k)||^2 \zeta_r(k)).
\]
Then, we obtain
\[
\begin{align*}
    \text{tr } [(\hat{w}_c(k+1))^T \hat{w}_c(k+1)] \\
    = (\hat{w}_c(k))^T \hat{w}_c(k) + \gamma^2 \ell^2 \|\gamma \hat{w}_c(k) \phi_c(k) + r(k) - \hat{w}_c(k-1) \phi_c(k-1)\|^2 B^T(k) B(k) \\
    - 2\gamma \ell (\gamma \hat{w}_c(k) \phi_c(k) + r(k)) \\
    - \hat{w}_c(k-1) \phi_c(k-1))^2 B^T(k) \hat{w}_c(k).
\end{align*}
\]

By introducing the property of trace function,
\[
\text{tr } (x_c(k) A^T(k) \hat{w}_c(k)) = \text{tr } (\hat{w}_c(k) x_c(k) A^T(k)) = \text{tr } (\hat{w}_c(k) x_c(k) A^T(k)),
\]
the last term in (55) can be expressed as
\[
- 2\gamma \ell (\gamma \hat{w}_c(k) \phi_c(k) + r(k) - \hat{w}_c(k-1) \phi_c(k-1)) \\
\times x_c(k) A^T(k) \hat{w}_c(k)
\]
\[
= \gamma \ell (\|\gamma \hat{w}_c(k) \phi_c(k) + r(k) - \hat{w}_c(k-1) \phi_c(k-1)\|^2 \\
- \hat{w}_c(k) x_c(k) A^T(k) \|x_c(k) | x_c(k) A^T(k)\|^2 \\
- \|\hat{w}_c(k) \phi_c(k) + r(k) - \hat{w}_c(k-1) \phi_c(k-1)\|^2).
\]

Therefore, substituting (56) and (57) into (52), we have Lemma 3.

**Lemma 4.** Under Assumption 3, consider the weight vector of the input-to-hidden layer in ANN. Let
\[
L_4(k) = \frac{1}{b_3} \text{tr } (\hat{w}_a(k))^T \hat{w}_a(k),
\]
then its first difference is bounded by
\[
\Delta L_4(k) \leq \frac{1}{b_3} (\|\hat{w}_c(k) \phi_c(k) + \zeta(k) - d(k)\|^2 \|x_o(k)\|^2 \\
\times \|\hat{w}_c(k) C(k) D^T(k)\|^2 \\
+ \|\hat{w}_c(k) \phi_c(k) + \zeta(k) - d(k)\|^2 \\
+ \|\hat{w}_a(k) x_o(k)\|^2 \|\hat{w}_c(k) C(k) D^T(k)\|^2)
\]
where \(D(k) = \frac{1}{2} (1 - \phi_o^2(k)) \hat{w}_a(k)\), and \(b_3 > 0\) is a weighting factor.

**Proof of Lemma 4.** The first difference of \(L_4(k)\) can be written as
\[
\Delta L_4(k) = \frac{1}{b_3} \text{tr } (\hat{w}_a(k+1))^T \hat{w}_a(k+1) \\
- (\hat{w}_a(k))^T \hat{w}_a(k).
\]

With the updating rule in (30), \(\hat{w}_a(k+1)\) can be rewritten as
\[
\begin{align*}
\hat{w}_a(k+1) &= \hat{w}_a(k) - b_o \hat{w}_c(k) \phi_c(k) + \zeta(k) - d(k) \\
&= \hat{w}_a(k) - b_o (\hat{w}_c(k) \phi_c(k) + \zeta(k) - d(k)) \\
&\times D(k) C^T(k) (\hat{w}_c(k))^T x_o^T(k).
\end{align*}
\]

Let us consider
\[
\begin{align*}
\text{tr } [(\hat{w}_a(k+1))^T \hat{w}_a(k+1)] \\
= (\hat{w}_a(k))^T \hat{w}_a(k) + b_o^2 \|\hat{w}_c(k) \phi_c(k) + \zeta(k) - d(k)\|^2 \\
\times \|\hat{w}_c(k) C(k) D^T(k)\|^2 \|x_o(k)\|^2 \\
- 2b_o \|\hat{w}_c(k) D^T(k) (\hat{w}_c(k) \phi_c(k) + \zeta(k) - d(k))^T \\
\times \hat{w}_a(k) x_o(k).
\end{align*}
\]

Then, by using the property of trace function, \(\text{tr } (X^T Y + Y^T X) = \text{tr } (X^T Y) + \text{tr } (X^T Y)^T = 2\text{tr } (X^T Y)\) and \(\text{tr } (XY) = \text{tr } (YX)\), the last term in (62) is bounded by
\[
-2b_o \|\hat{w}_c(k) D^T(k) (\hat{w}_c(k) \phi_c(k) + \zeta(k) - d(k))^T \\
\times \hat{w}_a(k) x_o(k) \leq b_o \|\hat{w}_c(k) \phi_c(k) + \zeta(k) - d(k)\|^2 + \|\hat{w}_a(k) x_o(k)\|^2 \\
\times \|\hat{w}_c(k) C(k) D^T(k)\|^2).
\]

We have the statement of Lemma 4 by substituting (61) and (62) into (60).

**B. Stability convergence and (sub)optimality results**

With Lemmas 1-4 in place, we are now in a position to provide results on closed-loop stability of the system, convergences of the neural networks in dHDP, and the Bellman (sub)optimality of the resulted control policy.

**Definition 1.** (Uniformly Ultimately Boundedness of a Discrete Dynamical System [46], [47]) A dynamical system is said to be uniformly ultimately bounded with ultimate bound \(b > 0\), if for any \(a > 0\) and \(k_0 > 0\), there exists a positive number \(N = N(a, b)\) independent of \(k_0\), such that \(\|\hat{\xi}(k)\| \leq b\) for all \(k \geq N + k_0\) whenever \(\|\hat{\xi}(k_0)\| \leq a\).

In the following, \(\hat{\xi}(k)\) can represent tracking errors \(e_1(k)\), \(e_2(k)\) or weight approximation errors \(\hat{w}_a(k), \hat{w}_c(k)\), which are related to the stable system or the weight convergence of the actor and critic neural networks.

**Theorem 1.** (Stability Result) Let Assumption 2 and Assumption 3 hold and the stage cost as given in (15). The tracking errors \(e_1(k)\) and \(e_2(k)\) defined in equation (4) and (6), respectively, of the considered system are uniformly ultimately bounded with the following chosen gains \(c_1\) and \(c_2\) (in equation (8) and (11), respectively).

\[
|c_1| < \frac{\sqrt{2}}{\zeta}, \quad |c_2| < \frac{1}{\sqrt{2}} \sqrt{2\zeta - 4\zeta^2}
\]

**Proof of Theorem 1.** We introduce a candidate Lyapunov function:
\[
L_o(k) = e_1^T(k) e_1(k) + (M^- e_2(k))^T (M^- e_2(k)).
\]

The first difference of \(L_o(k)\) is given as
\[
\Delta L_o(k) = e_1^T(k+1) e_1(k+1) \\
+ (M^+ e_2(k+1))^T (M^+ e_2(k+1)) \\
- e_1^T(k) e_1(k) - (M^- e_2(k))^T (M^- e_2(k)),
\]
by substituting \(e_1(k+1)\) and \(M^+ e_2(k+1)\) from (8) and (13), we have
\[
\Delta L_o(k) = \frac{1}{2} (|c_1|^2 - 1) e_1^T(k) e_1(k) \\
+ (2|c_2|^2 - (M^-)^T (M^-) + 2\zeta^2) e_2^T(k) e_2(k) \\
+ 8 \|e_o(k)\|^2 + 8 \|e_o(k)\|^2 + 4 \|\tau_t(k)\|^2.
\]

Based on (64), and also let \(H_1 = 8 \|e_o(k)\|^2 + 8 \|e_o(k)\|^2 + 4 \|\tau_t(k)\|^2\), we obtain
\[
\Delta L_o(k) < - (1 - 2(c_1^T) e_1^2(k) \\
- (M^- - 2(c_2^T + \zeta^2)) e_2^2(k) + H_1.
\]
where \( H_{1m} = 8\|\zeta_m\|^2 + 8\|\epsilon_m\|^2 + 4\|\tau_m\|^2 \), \( \zeta_m \), \( \epsilon_m \), and \( d_m \) are the upper bounds of \( \zeta_m^k(k) \), \( \epsilon_m^k(k) \), and \( \tau_m^k(k) \), respectively. As in (68), the first difference \( \Delta L_w(k) < 0 \).

According to Definition 1 and the bounded initial states and weights, this demonstrates that the errors \( e_1(k) \) and \( e_2(k) \) are uniformly ultimately bounded from time step \( k \) to \( k+1 \), and the boundedness of control input can also be ensured according to (11).

**Theorem 2.** (Weight Convergence) Under Assumption 3 and based on Theorem 1, let the weights of the actor and critic neural networks be updated according to (24), (25), (30) and (31). Then \( \hat{w}_c \) and \( \hat{w}_a \) are uniformly ultimately bounded provided that the following conditions are met:

\[
\begin{align*}
\beta_1 < \beta_2 - \gamma, \\
\beta_1 < \min_k \left\{ l_c - \gamma^2 \frac{\beta_2 - \gamma}{\beta_1} \left( \frac{\|\Phi(k)\|^2}{\beta_1} + \frac{1}{\beta_1^2} \| x_c(k) \|^2 \right) \right\}, \\
\beta_1 < \min_k \left\{ l_a - \gamma^2 \frac{\beta_2 - \gamma}{\beta_1} \left( \frac{\|\Phi(k)\|^2}{\beta_1} + \frac{1}{\beta_1^2} \| x_a(k) \|^2 \right) \right\}^{-1}.
\end{align*}
\]

**Remark 4.** With \( \gamma, \beta_1, \beta_2, \beta_3 \) provided in (16), (42), (49) and (58), respectively, we can choose \( l_c, l_a \) to satisfy (70) by setting \( \beta_2 > \gamma > 0 \) and \( \beta_3 > \beta_1 > 0 \).

**Proof of Theorem 2.** We introduce a candidate of Lyapunov function:

\[ L_w(k) = L_1(k) + L_2(k) + L_3(k) + L_4(k), \]

where \( L_1(k), L_2(k), L_3(k) \) and \( L_4(k) \) are shown in (36), (42), (49) and (58). Then the first difference of \( L_w(k) \) is given as

\[
\begin{align*}
\Delta L_w(k) &\leq -\left( \gamma^2 - \frac{8}{\beta_1} \right) \|\zeta_c(k)\|^2 - \left( 1 - \gamma^2 l_c \right) \|\Phi_c(k)\|^2 \\
&\quad - \frac{\gamma^2 l_c}{\beta_2} \| A(k) \|^2 \| x_c(k) \|^2 \frac{\|\Phi(k)\|^2}{\beta_1} \| x_c(k) \|^2 \\
&\quad + \| r(k) - \hat{w}_c(k-1)^T \Phi_c(k-1) \|^2 - \| \hat{w}_c(k) \|^2 \| \Phi(k) \|^2 \\
&\quad + \| \zeta_a(k) \|^2 - d(k)^2 \left( \frac{1}{\beta_1} - \frac{l_a}{\beta_1^2} \| \hat{w}_a(k) \|^2 \| C(k) \|^2 \right) \\
&\quad \| \Phi(k) \|^2 \| \hat{w}_a(k)^T \| \| C(k) \| A(k) \| x_a(k) \|^2 \\
&\quad - \frac{1}{\beta_1} H_2,
\end{align*}
\]

where \( H_2 \) is defined as

\[
H_2 = \frac{\gamma}{\beta_1} \frac{l_a}{\beta_1} \| \hat{w}_a(k) \|^2 \| C(k) \|^2 \| x_a(k) \|^2.
\]

for \( l_c \) and \( l_a \) satisfying (70) and also by selecting \( \beta_2 > \gamma \) and \( \beta_3 > \beta_1 \), we obtain

\[
\begin{align*}
\Delta L_w(k) &\leq \left( 8 - \frac{8}{\beta_1} \right) \|\zeta_c(k)\|^2 + H_2. \tag{74}
\end{align*}
\]

Applying the Cauchy–Schwarz inequality, we have

\[
\begin{align*}
H_2 &\leq \left( \frac{8}{\beta_1} \right) \| w_{e_2} \|^2 + \| \phi_e \|^2 \\
&\quad + \frac{1}{\beta_1} \left( \frac{8}{\beta_2} \| w_{e_2} \|^2 + 4\| \phi_e \|^2 \right) \\
&\quad + \frac{\gamma}{\beta_2} \left( \| w_{e_2} \|^2 + \| \phi_e \|^2 \right) \| A(k) \|^2 \| x_a(k) \|^2 \\
&\quad + \frac{1}{\beta_3} \left( \| w_{e_2} \|^2 + \| \phi_e \|^2 \right) \| D(k) \|^2 \| x_a(k) \|^2.
\end{align*}
\]

where \( \phi_E, \epsilon_m, w_{e_2}, A_m, C_m, D_m, \phi_a, \epsilon_a, \) and \( x_{e_2} \) are the upper bounds of \( \phi_k, \epsilon_k, w_{e_2}, A_k, C_k, D_k, \phi_a, \epsilon_a, \) and \( x_k \), respectively.

Therefore, if \( \gamma^2 - \frac{8}{\beta_1} > 0 \), that is, \( \beta_1 > \frac{8}{\gamma} \) and \( \gamma \in (0, 1) \), then for \( l_a, l_c \) with constraints from (70), and

\[
\|\zeta_c(k)\|^2 > \sqrt{\frac{H_2}{\gamma^2 - \frac{8}{\beta_1}}}, \tag{76}
\]

the first difference \( \Delta L_w(k) \) < 0 holds.

From Definition 1, this result means that the estimation errors \( \hat{w}_c \) and \( \hat{w}_a \) are uniformly ultimately bounded from the time step \( k \) to \( k+1 \), respectively.

**Remark 5.** Results of Theorem 1 and Theorem 2 hold under less restrictive conditions than those in [44], [45] that require bounded stage cost. We require an initially bounded system state and actor-critic network weights only.

**Theorem 3.** ((Sub)optimality Result) Under the conditions of Theorem 2, the Bellman optimality is achieved within finite approximation error. Meanwhile, the error between the obtained control law \( u(k) \) and optimal control \( u^*(k) \) is uniformly ultimately bounded.

**Proof of Theorem 3.** From the approximate cost-to-go (20) and the cost-to-go expressed in (33), we have

\[
\begin{align*}
\| J(k) - J^*(k) \| &\leq \| \hat{w}_c(k)^T \Phi(k) - w_{e_2} \Phi(k) - \epsilon(k) \| - \| \hat{w}_c(k) \Phi_m + \epsilon_m \|
\end{align*}
\]

where \( \phi_m, \epsilon_m, \phi_a, \epsilon_a \) are the upper bound of \( \phi_k, \epsilon_k, \phi_a, \epsilon_a \), respectively. This comes directly as \( \| \hat{w}_c \| \) and \( \| \hat{w}_a \| \) are both uniformly ultimately bounded as the time step \( k \) increases as shown in Theorem 2. It demonstrates that the Bellman optimality is achieved within finite approximation errors.

VI. SIMULATION STUDY

In this section, a single-link robot manipulator is used to demonstrate how the proposed algorithm works and how it improves reproducibility of results over the original dHDP. The considered single-link robot manipulator in this section is shown in Fig. 2 with the following motion equation:

\[
M \ddot{q} + G(q) + \tau_d (t) = \tau(t), \tag{79}
\]

where \( H \) is defined as

\[
\begin{align*}
\|\epsilon_1(k)\| > \sqrt{\frac{H_{1m}}{1 - 2\|\epsilon_1(k)\|^2}} \tag{69}
\end{align*}
\]
where \( G(q) = \frac{1}{2} \times 9.8 \times m \times l \times \sin(q) \) and \( M = 5 \); \( m \) and \( l \) are the mass and the half length of the manipulator in Fig. 2, respectively. The values of \( m, l \) and \( \tau_D \) are different in different simulation cases below. Note that, simulations of the model (79) is to provide a simulated environment in place of a real physical environment where data can be directly measured from and used in controller design, that is to say that the proposed approach is data-driven.

The feedback gain parameters \( c_1 \) and \( c_2 \) in (64) are chosen as \( c_1 = 0.7 \) and \( c_2 = -5 \). The CNN and ANN in Fig. 1 respectively has six hidden nodes. The discount factor \( \gamma \) in (16) is chosen as 0.95. As the considered single-link robot manipulator system (79) is in continuous-time, the Runge-Kutta discretization method is used to discretize system (79). We use a sampling time period of \( h = 0.02s \), i.e., we obtain 50 samples per second.

The mean square error (MSE) below in (80) is used for adapting the learning rates \( l_a \) and \( l_c \) in the actor and critic neural networks, respectively. The initial \( l_a \) and \( l_c \) are 0.1.

\[
\text{MSE} = \frac{1}{n^+ - n^-} \Sigma_{k=n^-}^{n^+} ||e_1(k)||^2, \quad (80)
\]

where \((n^+ - n^-)\) is the number of data samples between time stamps \( n^- \) and \( n^+ \). After the 500th sample, the MSE of the previous 100 samples is calculated, denoted as MSE_{100}. Let \( \eta_1 = 0.1 \times 10^{-3} \) and \( \eta_2 = 0.3 \times 10^{-3} \). The learning rate \( l_a \), \( l_c \) will reduce to half if MSE_{100} < \( \eta_1 \), or they will be reset as the initial value if MSE_{100} > \( \eta_2 \), or they will not change if \( \eta_2 \leq \text{MSE}_{100} \leq \eta_2 \).

In the following, we demonstrate the effectiveness of the proposed algorithm by comparing it with dHDP in [32] and feedback stabilizing control [6], where the feedback stabilizing control law is \( u(k) = c_2 e_2(k) \). From (11), which is to demonstrate the effectiveness of the output \( \hat{f}(k) \) of dHDP as shown in Fig. 1.

\[ \text{A. Tracking by dHDP with and without backstepping} \]

In this comparison study, we set \( x_1(0) = -0.1, x_2(0) = 0.1, m = 1, l = 1, \) and \( \tau_D = 0 \). An episode consists of 6000 consecutive samples and a trial has one episode in this case study. We define a trial as a success if the MSE of the last 3000 samples (denoted as MSE_{3000}) is less than the MSE of first 3000 samples (denoted as MSE_{3000}).

A total of 50 trials were conducted to obtain results reported here. Each trial used randomly initialized actor and critic weights uniformly distributed between \([-1, 1]\).

The dHDP reached 14% success rate. The MSE_{3000} of the successful cases is \( 1.454e^{-1} \). A typical success trial of tracking is shown in Fig. 3. In comparison, the success rate of the proposed algorithm is 100% and the MSE_{3000} is \( 2.135e^{-5} \). A typical tracking trial is shown in Fig. 4. It is shown that the tracking performance is improved by the proposed algorithm.

\[ \text{B. Reproducibility of the proposed scheme} \]

While the previous evaluation has demonstrated the effectiveness of the proposed tracking control design, we are now in a position to perform a comprehensive evaluation. To do so, eight different scenarios represented by different model parameters as shown in Table I are tested to quantitatively evaluate the reproducibility of results using the proposed algorithm. In Table I, “Pulse” denotes pulse disturbance, i.e.,

\[
\tau_D(t) = \begin{cases} 
2 & t = 20 \\
0 & t \neq 20
\end{cases}
\quad (81)
\]

| case | initial state | m, l | d | trial | success rate | # reset |
|------|--------------|------|---|-------|--------------|--------|
| 1    | (-0.1, 0.1)  | (1,1)| N/A | 50    | 96%          | 2      |
| 2    | (0.1, -0.1)  | (1,1)| N/A | 50    | 82%          | 16     |
| 3    | (0.2, -0.2)  | (1,1)| N/A | 50    | 80%          | 24     |
| 4    | (-0.2, 0.2)  | (1,1)| N/A | 50    | 94%          | 5      |
| 5    | (-0.1, 0.1)  | (1,1)| Pulse | 50 | 76%          | 28     |
| 6    | (-0.1, 0.1)  | (1,1)| Gaussian | 50 | 84%          | 12     |
| 7    | (-0.1, 0.1)  | (2,2)| N/A | 50    | 60%          | 26     |
| 8    | (-0.1, 0.1)  | (2,2)| N/A | 50    | 50%          | 35     |

Fig. 2. Single-link robot manipulator.

Fig. 3. Tracking performance for dHDP without backstepping.

Fig. 4. Tracking performance for dHDP without backstepping.
and “Gaussian” denotes Gaussian noise, i.e., the noise whose mean and standard deviation are 0 and 8.25, respectively. We define a success as when the \( \text{MSE}_{3000} \) is less than that of feedback stabilizing control method in one episode. The success rate of 50 trials is shown in Table I. The robustness of the proposed algorithm is verified by different cases.

C. Reproducibility with reset after failure

To test if the proposed method can lead to eventual success, we used a “reset” mechanism. We reset the weights of the actor and critic neural networks after a failed episode and observe the next episode. Thus in this subsection, a trial contains more than one episode. The accumulated reset numbers of 50-trial episodes are shown in Table I and labeled as “# reset”. The average \( \text{MSE}_{3000} \) of the last successful episode is shown in Fig. 8, compared to that of feedback stabilizing control. It can be seen that the proposed algorithm outperforms feedback stabilizing control. The effectiveness of dHDP with a backstepping feedback loop is verified, that is, the feedback compensation term produced by dHDP can help feedback stabilizing control improve the tracking performance.

VII. CONCLUSION

This study aims at developing a mathematically suitable and practically useful, data-driven tracking control solution. Toward this goal, we introduce a new algorithm that takes advantage of the potential closed-loop system stability framework based on backstepping design for Euler-Lagrange systems, and the adaptivity and (sub)optimality of dHDP to account for the unknown dynamics. Based on the proposed algorithm, we have shown stability of the overall dynamic system, weight convergence of the actor-critic neural networks, and (sub)optimality of the Bellman solution. Our simulations provide a quantitative comparison that highlights the effectiveness of this algorithm, that is, the convergence speed, and tracking accuracy have all improved significantly. As the dHDP has been shown feasible to solve complex engineering application problems, it is expected that this algorithm also has the potential for applications of tracking control of nonlinear dynamic systems in practice.
