Hybrid Reinforcement Learning for Optimal Control of Non-Linear Switching System

Xiaofeng Li, Lu Dong, *Member, IEEE*, Lei Xue, *Member, IEEE*, and Changyin Sun, *Senior Member, IEEE*

***Abstract*— Based on the reinforcement learning mechanism, a data-based scheme is proposed to address the optimal control problem of discrete-time non-linear switching systems. In contrast to conventional systems, in the switching systems, the control signal consists of the active mode (discrete) and the control inputs (continuous). First, the Hamilton–Jacobi–Bellman equation of the hybrid action space is derived, and a two-stage value iteration method is proposed to learn the optimal solution. In addition, a neural network structure is designed by decomposing the Q-function into the value function and the normalized advantage value function, which is quadratic with respect to the continuous control of subsystems. In this way, the Q-function and the continuous policy can be simultaneously updated at each iteration step so that the training of hybrid policies is simplified to a one-step manner. Moreover, the convergence analysis of the proposed algorithm with consideration of approximation error is provided. Finally, the algorithm is applied evaluated on three different simulation examples. Compared to the related work, the results demonstrate the potential of our method.**

***Index Terms*— Adaptive dynamic programming (DP), hybrid action space, normalized advantage value function (NAF), reinforcement learning (RL), switching system.**

I. Introduction

IN THIS article, the optimal control problem of the discrete-time (DT) non-linear switching system is studied with the consideration of hybrid action space. The switching system contains several subsystems so that the control signal includes a discrete action (active mode) and continuous parameters (control input) [1]. Many real-world applications can be modeled as switching problems, including aerospace systems, robotics, mechanical engineering, and traffic signal control systems [2]–[4]. The properties of stability, controllability, and observability are studied in the first place [5], [6]. Afterward, the goal of optimizing the performance function captures the researchers' attention [7].

The existing literature of optimal switching control problems can be classified into two groups according to the nature of subsystems: the optimal scheduling problem of autonomous switching systems and the optimal control problem of controlled switching systems. As for the switching system with autonomous subsystems, optimizing switching instants with fixed mode sequence is one of the widely studied issue [8]–[10]. The non-linear programming-based methods are utilized to find the open-loop local optimal solution [9], [10]. Once the initial state is changed, the solution should be recomputed from the scratch. In [11], the restriction of the fixed mode sequence is removed, and the optimal sequence is determined by alternating between two stages: 1) optimize the switching instants with a given mode sequence and 2) update the mode sequence. Note that the aforementioned approaches are designed for continuous-time subsystems. The optimal control problem of orbit transfer with ON–OFF actuators is solved by formulating it as a general switching system with two DT autonomous subsystems [12]. Then, a closed-form solution is obtained based on the dynamic programming (DP) method with value function approximators. Recently, the joint optimal control of switching instants and control inputs of subsystems has attracted attention from researchers [13]–[15]. Several approaches have been proposed to deal with the switching linear system with the quadratic cost function [16]–[19]. In [16], a novel two-stage approach is proposed to codesign the optimal switching instants and control inputs of general switching linear quadratic regulation (SLQR) problems with the prespecified sequence of the active mode. A DP-based method is proposed to address the SLQR problem with the infinite-horizon performance function that can get the analytical expression of both optimal switching and control policies [17]. Moreover, a relaxed value iteration (VI) approach is designed to reduce the computational complexity of the aforementioned DP method by searching for the near-optimal solution within prespecified bound [18]. In addition, the receding-horizon instead of infinite-horizon SLQR problem is solved by a relaxed DP algorithm with theoretical analysis [19]. The stability of the policy is guaranteed with *a posteriori* stability criteria, while a region-reachability criterion is derived in terms of linear matrix inequalities.

In recent years, reinforcement learning (RL) methods have achieved striking performance in dealing with sequential

Manuscript received 12 January 2021; revised 13 August 2021 and 23 December 2021; accepted 24 February 2022. Date of publication 13 April 2022; date of current version 30 October 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0101400, in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20202006, and in part by the National Natural Science Foundation of China Grant 62073225, Grant 61921004, and Grant 62173251. *(Corresponding author: Changyin Sun.)*

Xiaofeng Li is with the School of Artificial Intelligence, Anhui University, Hefei 230601, China (e-mail: 230169413@seu.edu.cn).

Lu Dong is with the School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: ldong90@seu.edu.cn).

Lei Xue and Changyin Sun are with the School of Automation, Southeast University, Nanjing 210096, China, and also with the Key Laboratory of Measurement and Control of Complex Systems of Engineering, Ministry of Education, Southeast University, Nanjing 210096, China (e-mail: rayseu@seu.edu.cn; cysun@seu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3156287.

Digital Object Identifier 10.1109/TNNLS.2022.3156287

decision-making problems [20]–[22]. In the control engineering field, it is referred to as adaptive dynamics programming (ADP) that solves the optimal control problem by using the function approximator for the value function [23]. Generally, ADP is constituted by the actor–critic (AC) structure, where the critic network takes the system state as input and outputs the approximation of value function, while the actor network approximates the mapping between states and control inputs [24]. Li *et al.* [25] classify the family of ADP schemes into three categories: model-based ADP [26], model-free ADP [27], and data-based ADP [28]. Specifically, the model-based methods require knowing the system dynamics as *a priori*. The model-free algorithms need to employ a model network to identify the system function before the training process, while the data-based scheme is completely based on the transitions by interacting with the system. Recently, ADP techniques have been applied to deal with various optimal control problems, including time-delayed systems [29], control constrained systems [30], and event-triggered mechanisms [31].

However, to the best of our knowledge, there exist few studies concerning the optimal control of non-linear switching systems, while most of the literature focuses on autonomous systems [32]–[34]. The existing ADP methods are designed to deal with either discrete or continuous action space solely and are formidable to handle the hybrid discrete action and continuous parameters. Zhang *et al.* [35] proposed a two-stage model-based ADP algorithm to find the optimal hybrid policy by approximating a costate value function. In addition, a model-free iterative algorithm is designed to solve the Hamilton–Jacobi–Bellman (HJB) equation with the globalized dual-heuristic programming (GDHP) structure, which employs a model network to identify the system dynamics in the first place [36]. However, on the one hand, it is arduous to obtain the complete dynamics of complex non-linear systems. On the other hand, the identification error may degrade the control performance.

In order to deal with the hybrid nature of action space, an RL-based algorithm is proposed to codesign the continuous control of subsystems and the discrete action of the active mode. Specifically, a novel neural network (NN) structure is designed to implement the algorithm by using a restrict formulation of the Q-function. The main contributions of this article are summarized as follows.

1) Considering the hybrid action space of switching non-linear systems, a transformed HJB equation is first derived in this article, and a two-stage data-based VI method is proposed to deal with the hybrid action space.
2) Inspired by the normalized advantage value function (NAF) method [37], a special NN structure is designed by decomposing the Q-function into a value function and an NAF. The advantage function is a quadratic function of continuous control so that we can obtain and update an analytical continuous policy and the Q-function simultaneously. Using this special NN structure instead of the AC structure can cut the number of NNs to train at each step in half.
3) A convergence analysis of the proposed algorithm is provided with the consideration of approximation error.

The rest of this article is organized as follows. Section II gives the formulation of the problem and derives the transformed HJB equation of the switching system. In Section III, we first propose a two-stage data-based algorithm by alternatively updating the continuous policy and the Q-function. Then, a hybrid RL method is presented with a novel NN design. The convergence analysis of the approximated iteration method is given in Section IV. In Section V, the simulation results are provided to demonstrate the performance of the designed algorithm. Finally, conclusion remarks are provided in Section VII.

II. Problem Formulation

Consider a general non-linear non-affine DT switching system with the following dynamics:

$$x_{k+1} = f_{\upsilon_k}(x_k, u_k) \tag{1}$$

where $x_k \in \mathbb{R}^n$ and $u_k \in \mathbb{R}^m$ denote the system state and the control parameter, respectively. The subscript k denotes the index of the time step, while the superscripts n and m denote the dimensions of state space and parameter space, respectively. Note that the discrete action $\upsilon_k \in \mathcal{P} = \{1, 2 \ldots, P\}$ determines the active subsystem, where P denotes the number of subsystems. For any subsystem, we assume that the dynamics $f_\upsilon : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is Lipschitz continuous with $f_\upsilon(0, 0) = 0$.

In contrast to conventional non-linear systems, the action space of the switching system consists of two terms, i.e., the discrete action υ and the continuous parameters u. The set of discrete actions $\mathcal{P}$ is finite, while each action υ is associated with a set of control parameters $u \in \mathbb{R}^m$ so that the action space can be described by

$$\mathcal{A} = \Big\{(\upsilon, u) | \upsilon \in \mathcal{P}, u \in \mathbb{R}^m\Big\}. \tag{2}$$

For each subsystem, there exists a corresponding continuous policy. Let $\pi_\upsilon(x)$ and $\pi_u(x, \upsilon)$ denote the policies of discrete action and its corresponding continuous parameters, respectively. In addition, let $\pi(x)$ denote the hybrid control policy such that $\pi(x) = (\pi_\upsilon(x), \pi_u(x, \pi_\upsilon(x)))$. The objective is to codesign the hybrid policy to minimize the infinite-horizon performance function

$$J = \sum_{k=0}^{\infty} U(x_k, \upsilon_k, u_k). \tag{3}$$

In this article, the cost function is defined to be a quadratic function with $U(x, \upsilon, u) = x^T Q(\upsilon) x + u^T R(\upsilon) u$, where $Q(\upsilon) \in \mathbb{R}^{n\times n}$ and $R(\upsilon) \in \mathbb{R}^{m\times m}$ are positive definite matrices.

Given any hybrid policy $\pi(x)$, the Q-function is defined as

$$Q^\pi(x_k, \upsilon_k, u_k) = U(x_k, \upsilon_k, u_k) + \sum_{l=k+1}^{\infty} U\Big(x_l, \pi_\upsilon(x_l), \pi_u(x_l, \pi_\upsilon(x_l))\Big). \tag{4}$$

In addition, the value function is defined as

$$V^{\pi}(x_k, \upsilon_k) = U\Big(x_k, \upsilon_k, \pi_u(x_k, \upsilon_k)\Big) + \sum_{l=k+1}^{\infty} U\Big(x_l, \pi_\upsilon(x_l), \pi_u(x_l, \pi_\upsilon(x_l))\Big). \quad (5)$$

Based on Bellman's optimality principle [38], the optimal Q-function satisfies the following HJB equation:

$$Q^*(x_k, \upsilon_k, u_k) = U(x_k, \upsilon_k, u_k) + \min_{\upsilon\in\mathcal{P}} \inf_{u\in\mathbb{R}^m} Q^*(x_{k+1}, \upsilon, u) \quad (6)$$

where $x_{k+1} = f_{\upsilon_k}(x_k, u_k)$ and Q^* denotes the Q-function of the optimal hybrid controller $\pi^*(x)$. Note that the right-hand side of (6) contains two kinds of operators, i.e., $\min_{\upsilon\in\mathcal{P}}$ and $\inf_{u\in\mathbb{R}^m}$, so that it requires to codesign the discrete action policy and the continuous control policy. However, most existing RL methods are designed for either discrete or continuous action space, which cannot be directly applied for solving (6).

Remark 1: Considering the hybrid action of the controlled switching system, the above defined Q-function depends not only on the state-action pair (x_k, u_k) but also on the active mode υ_k given that the same x_k and u_k but different υ_k may lead to a different x_{k+1} and even a completely different trajectory. In addition, $V^{\pi}(x_k, \upsilon_k)$ also depends on υ_k so that it outputs the accumulated costs along the trajectory generated by the hybrid policy $\pi(x)$ given the current state x_k and the active mode υ_k. It is worthwhile noting that, once υ_k is given, u_k can be determined by $\pi_u(\upsilon_k)$. Then, for any $\upsilon \in \mathcal{P}$, the advantage item $A^{\pi}(x_k, \upsilon_k, u_k)$ defined in Section III can evaluate the difference incurred by executing u_k instead of $\pi_u(\upsilon_k)$ at the current time step. Similar to the original NAF method, the analytical solution of the continuous policy can be obtained by parameterizing $A^{\pi}(x_k, \upsilon_k, u_k)$ as a quadratic function of continuous control input.

III. Algorithm Design

In this section, we first extend the VI method to solve the derived equation (6) by alternatively updating the continuous policy and the associated Q-function. In the second subsection, with a special NN structure, a hybrid RL algorithm is proposed to simultaneously update the continuous policy and the Q-function. Considering that both $Q(x, \upsilon, u)$ and $V(x, \upsilon)$ depend on υ, the following iterations should be implemented for $\forall \upsilon \in \mathcal{P}$.

A. Two-Stage Value Iteration Method

Considering the hybrid control signal and the coupling between the discrete action and the continuous control, traditional VI-based methods cannot be directly utilized to find the optimal policy. A two-stage VI method is proposed, which divides the learning process into two steps, i.e., searching for the optimal continuous policies and updating the Q-functions.

Specifically, the learning process starts with selecting an initial guess on $Q^*(x, \upsilon, u)$ and let $Q^{(0)}(x, \upsilon, u) = 0, \forall \upsilon \in \mathcal{P}$. Then, for $i = 0, 1, \ldots$, the VI method iterates between solving the continuous policies

$$\pi_u^{(i)}(x_k, \upsilon_k) = \arg\inf_{u_k\in\mathbb{R}^m} Q^{(i)}(x_k, \upsilon_k, u_k) \quad (7)$$

and updating the Q-functions

$$Q^{(i+1)}(x_k, \upsilon_k, u_k) = U(x_k, \upsilon_k, u_k) + \min_{\upsilon\in\mathcal{P}} Q^{(i)}\big(x_{k+1}, \upsilon, \pi_u(x_{k+1}, \upsilon)\big) \quad (8)$$

until the iteration is converged. The notation i denotes the iteration index.

However, taking infinitum over continuous space is computationally intractable so that the actor network is often utilized to approximate the solution of (7). In addition, the critic network is also employed to approximate the Q-function. In contrast to conventional non-linear systems, each subsystem of (1) has its own actor and critic networks so that one needs to train a number of $2P$ NNs at each iteration step, which may lead to a heavy computational burden. In Section III-B, we propose a single-stage iterative learning algorithm by extending the original NAF structure that can cut the NNs needed to train in half. This algorithm is named the hybrid NAF (HNAF) algorithm in the following.

B. HNAF Algorithm

For any fixed $\upsilon \in \mathcal{P}$, its corresponding continuous policy is hard to obtain because $Q^{(i)}(x, \upsilon, u)$ is a non-convex function of u. To address this issue without introducing an extra actor network, we decompose the Q-function into a value function term and an advantage value function term

$$Q^{\pi}(x, \upsilon, u) = V^{\pi}(x, \upsilon) + A^{\pi}(x, \upsilon, u). \quad (9)$$

Specifically, inspired by the NAF method [37], we assume that the advantage function is a quadratic function w.r.t. u with the following expression:

$$A(x, \upsilon, u) = \frac{1}{2}\big(u - \mu(x, \upsilon)\big)^T P(x, \upsilon)\big(u - \mu(x, \upsilon)\big) \quad (10)$$

where $P(x, \upsilon)$ is a positive-definite square matrix that is dependent of state and active mode. Therefore, if υ is fixed, the minimum of the Q-function, $\arg\inf_{u\in\mathbb{R}^m} Q(x, \upsilon, u)$, can easily and analytically be determined by $\mu(x, \upsilon)$, that is to say

$$\pi_u(x, \upsilon) = \mu(x, \upsilon). \quad (11)$$

Furthermore, by substituting (9)–(11) into (8), we can rewrite (8) as

$$\begin{aligned} Q^{(i+1)}(x_k, \upsilon_k, u_k) &= U(x_k, \upsilon_k, u_k) + \min_{\upsilon\in\mathcal{P}} Q\big(x_{k+1}, \upsilon, \mu^{(i)}(x_{k+1}, \upsilon)\big) \\ &= U(x_k, \upsilon_k, u_k) + \min_{\upsilon\in\mathcal{P}} V^{(i)}(x_{k+1}, \upsilon). \end{aligned} \quad (12)$$

The architecture of the HNAF algorithm is illustrated in Fig. 1, while the detailed structure of NN is shown in the dotted box on the left-hand side of this figure. For each $\upsilon \in \mathcal{P}$, there exists a corresponding NN, which takes state and continuous control signal as inputs. The states pass

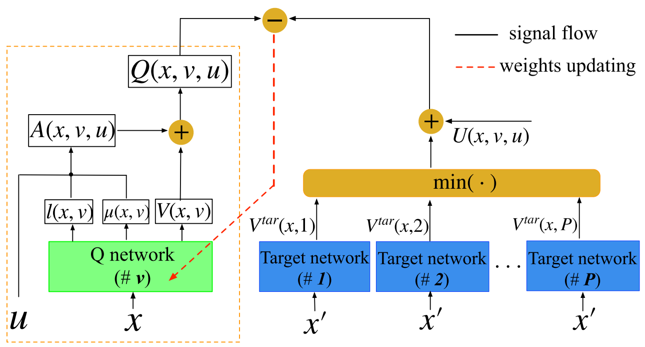

Fig. 1. Architecture of the HNAF algorithm. For notation simplicity, we let x denote the current state and x' denote the state at the next time step. The detailed structure of the designed NN is shown in the dotted box on the left-hand side of this figure. There exists an independent NN for each mode, so a number of P NNs are trained at every step.

through several layers, and the outputs are the value function $V(x,v)$, the expected continuous control signal $\mu(x,v)$, and a state-dependent vector $l(x,v)$, respectively. Then, the advantage function can be computed according to (10) with

$$P(x,v) = L(x,v)L(x,v)^T \tag{13}$$

where $L(x,v)$ is a lower triangular matrix whose entries come from the output vector $l(x,v)$ of the NN with the diagonal terms exponentiated. This exponential operation guarantees that $L(x,v)$ is a column full rank matrix by preventing zero element on the diagonal. Therefore, one knows that $P(x,v)$ is positive definite according to the property of real symmetric matrix [39]. Once $V(x,v), \forall v \in \mathcal{P}$ are known, the discrete action can easily be determined according to

$$v = \arg\min_{v\in\mathcal{P}} V(x,v). \tag{14}$$

The outline of the HNAF algorithm is summarized in Algorithm 1. Analogous to the DQN algorithm [21], the target network is utilized to stabilize the training process of NNs, while the experience replay technique is employed to improve sample efficiency. In contrast to the two-stage VI algorithm, the HNAF algorithm is an on-learning method, and the state and action spaces can be explored by using $\epsilon -$ greedy. Let $\theta^Q(v)$ denote all weights of the vth NN, while let $\theta^V(v)$ denote the weights of $V(x,v)$. In addition, let $\theta^{V'}$ denote the weights of corresponding part of the target network. According to (12), the target value of $Q^{(i+1)}(x,v,u)$ can be computed by

$$\begin{aligned} y^{(i+1)}(x_k,v_k,u_k,x_{k+1}) &= U(x_k,v_k,u_k) \\ &\quad + \min_{v\in\mathcal{P}} V_{\text{tar}}^{(i)}\left(x_{k+1},v;\theta^{V'}(v_k)\right). \end{aligned} \tag{15}$$

Then, the loss function can be obtained by

$$E(v) = \left(y^{(i+1)}(x_k,v_k,u_k,x_{k+1}) - Q^{(i+1)}(x_k,v_k,u_k;\theta^Q)\right)^2. \tag{16}$$

The weights of NNs can be tuned based on the stochastic gradient descent (SGD) method, and this backpropagation process can be automated by using the deep learning libraries, such as Pytorch [40] and Tensorflow [41].

Algorithm 1 HNAF Algorithm

```
1  Initialize the hyper-parameters: batch size $B$, soft update
   parameter $\tau$, learning rate $\alpha$, max steps $M$, max
   iterations $I$;
2  Initialize the NNs $Q^{(0)}(x,v,u;\theta^Q(v)) = 0, \forall v \in \mathcal{P}$;
3  Initialize the target NNs $Q'^{(0)}(x,v,u;\theta^{Q'}(v))$ with
   $\theta^{Q'}(v) \leftarrow \theta^Q(v), \forall v \in \mathcal{P}$;
4  Initialize the replay buffers $\mathcal{D}(v) \leftarrow \emptyset, \forall v \in \mathcal{P}$;
5  for $i = 1,2,\ldots,I$ do
6      Receive initial state $x_0$ according to the uniform
       distribution of state space;
7      for $k = 0,1,\ldots,T$ do
8          Input $x_k$ into the NNs and compute $V(x,v)$ and
           $\mu(x,v)$ for any $v \in \mathcal{P}$. Then, select the hybrid
           action $(v,u_k)$ according to (11) and (14);
9          Execute $(v_k,u_k)$ and observe $U_k$ and $x_{k+1}$;
10         Store transition $(x_k,v_k,u_k,U_k,x_{k+1})$ into $\mathcal{D}(v)$
           according to $v_k$;
11         for $v = 1,2,\ldots,P$ do
12             Randomly sample a batch of $B$ transitions from
               $\mathcal{D}(v)$;
13             Compute the target value $y^{(i+1)}(x_k,v_k,u_k,x_{k+1})$
               according to (15);
14             Update $\theta^Q(v)$ by minimizing the loss
               function (16);
15         end
16         Soft update the weights of target networks:
           $\theta^{Q'}(v) \leftarrow \tau\theta^Q(v) + (1-\tau)\theta^{Q'}(v), \forall v \in \mathcal{P}$
17     end
18 end
```

Remark 2: In recent years, DHP and GDHP algorithms have been proposed to deal with the optimal control problem of non-linear controlled switching systems [35], [36]. The DHP algorithm requires knowing the exact dynamics of all subsystems, while the GDHP algorithm utilizes an extra model network to identify the system model before training the actor and critic NNs. Compared with these algorithms, our algorithm requires no information of system but transitions that are much easier to obtain. Moreover, the number of trained NNs is significantly decreased especially for complicated switching systems with a large number of subsystems.

Remark 3: Both the deep deterministic policy gradient (DDPG) algorithm [42] and the original NAF method are proposed to deal with the MDP environments with continuous action space. The DDPG algorithm is usually implemented with the AC structure, while the original NAF method represents the Q-function in a restrict formulation so that the control policy has an analytical formulation. Consider the hybrid action space of the switching system, these two methods cannot directly be applied to the optimal switching control problem. For each subsystem, the HNAF algorithm utilizes an independent NN to represent its corresponding Q-function, value function, and continuous policy. Thus, it requires training a number of P NNs at every step. Once the learning process is completed, the discrete action is determined by comparing several value function values, $\upsilon = \arg\min_{\upsilon \in \mathcal{P}} V(x, \upsilon)$, while the associated continuous input can be obtained by $u = \mu(x, \upsilon)$.

Remark 4: In [43], the concept of the parameterized action space Markov decision process (PAMDP) is first introduced and studied. The parameterized action space consists of a set of discrete actions and continuous parameters. Intuitively, the parameterized action can be handled either by approximating it with a finite discrete set or relaxing it into a continuous set [44], [45]. The P-DQN algorithm is proposed to directly address the hybrid structure of the action space by combining the DQN and DDPG algorithms [45]. Moreover, the MP-DQN algorithm is proposed to eliminate the "false gradient" problem in the P-DQN framework by using the multipass method and achieving the state-of-the-art performance [46]. However, the existing methods for parameterized RL algorithms are all implemented with the AC structure. In this article, we propose a novel RL algorithm with a single NN for each subsystem. On the other hand, a rigorous convergence analysis with consideration of approximation error is first provided in this article, which is not included in the PAMDP literature.

IV. Convergence Analysis

In this section, the effect of approximation error incurred by the NN approximator is investigated. The boundedness of the approximated HNAF algorithm is provided by constructing auxiliary value functions. Before proceeding, the definition of admissible continuous policy is made.

Definition 1: A hybrid policy $\pi(x) = (\pi_\upsilon(x), \pi_u(x, \upsilon))$ is said to be admissible w.r.t. (3) if $\pi(x)$ can stabilize the system (1) with $\pi_u(0, \upsilon) = 0, \forall \upsilon \in \mathcal{P}$. In addition, for any initial state, its corresponding performance function $J = \sum_{k=0}^{\infty} U(x, \pi_\upsilon(x), \pi_u(x, \pi_\upsilon(x)))$ is finite.

Assumption 1: For system (1), there exists at least one admissible hybrid policy for any $x_0 \in \mathbb{R}^n$.

In general, the approximation error cannot be completely avoided if parametric function approximators are utilized. In addition, the advantage value function of the HNAF algorithm has a more restrict formulation than a general NN function, which may reduce the approximation capability. Let $\varepsilon^{(i)}(x, \upsilon, u)$ denote the approximation error at the ith iteration. We can rewrite (12) as

$$\hat{Q}^{(i+1)}(x_k, \upsilon_k, u_k) = U(x_k, \upsilon_k, u_k) + \min_{\upsilon \in \mathcal{P}} \inf_{u \in \mathbb{R}^m} \hat{Q}^{(i)}(x_{k+1}, \upsilon, u) + \varepsilon^{(i)}(x_k, \upsilon_k, u_k). \quad (17)$$

Before analyzing the effect of the approximation error, we first prove that the Q-function sequence generated by the exact HNAF algorithm will converge to the optimal solution of (6).

Theorem 1: Let $\{Q^{(i)}(x, \upsilon, u)\}_{i=0}^{\infty}$ denote the sequence of Q-functions generated by (7) and (8) with $Q^{(0)}(x, \upsilon, u) = 0, \forall \upsilon \in \mathcal{P}$. If Assumption 1 holds, the generated Q-function sequence is nondecreasing and will converge to the optimal Q-function, $\lim_{i\to\infty} Q^{(i)}(x, \upsilon, u) = Q^*(x, \upsilon, u), \forall \upsilon \in \mathcal{P}$.

Proof: First, the proof that the sequence $\{Q^{(i)}(x, \upsilon, u)\}_{i=0}$ is monotonic nondecreasing can easily be derived by extending the conclusion in [47]. Considering Assumption 1, $Q^{(i)}(x, \upsilon, u), \forall i$ is upper bounded by a finite value function and, therefore, will converge to a limit function, $\lim_{i\to\infty} Q^{(i)}(x, \upsilon, u) = Q^{\infty}(x, \upsilon, u), \forall \upsilon \in \mathcal{P}$. Finally, $Q^{\infty}(x, \upsilon, u) = Q^*(x, \upsilon, u)$ can be proved by the contradiction method. ■

As discussed in the ADP literature [48], the approximation error will accumulate with the increase in iteration steps. Therefore, the monotonicity and convergence of approximated Q-function do not follow from Theorem 1. The following theorem proves the boundedness of the approximated Q-function $\hat{Q}^{(i)}(x, \upsilon, u)$ by constructing auxiliary value functions.

Theorem 2: Let $|\varepsilon^{(i)}(x, \upsilon, u)| \le \beta U(x, \upsilon, u), \forall i$, where $\beta \in [0, 1)$. Let $\{\overline{Q}^{(i)}(x, \upsilon, u)\}_{i=0}^{\infty}$ and $\{\underline{Q}^{(i)}(x, \upsilon, u)\}_{i=0}^{\infty}$ be sequences generated by (19) and (20) with $\underline{Q}^{(0)}(x, \upsilon, u) = \hat{Q}^{(0)}(x, \upsilon, u) = \overline{Q}^{(0)}(x, \upsilon, u) = 0$, respectively. Then, the following inequality holds for $\forall i$:

$$\underline{Q}^{(i)}(x, \upsilon, u) \le \hat{Q}^{(i)}(x, \upsilon, u) \le \overline{Q}^{(i)}(x, \upsilon, u) \quad \forall i. \quad (18)$$

Proof: Let the sequence $\{\overline{Q}^{(i)}(x, \upsilon, u)\}_{i=0}^{\infty}$ be generated by

$$\overline{Q}^{(i+1)}(x_k, \upsilon_k, u_k) = U(x_k, \upsilon_k, u_k) + \beta U(x_k, \upsilon_k, u_k) + \min_{\upsilon \in \mathcal{P}} \inf_{u \in \mathbb{R}^m} \overline{Q}^{(i)}(x_{k+1}, \upsilon, u) \quad (19)$$

while let the sequence $\{\underline{Q}^{(i)}(x, \upsilon, u)\}_{i=0}^{\infty}$ be generated by

$$\underline{Q}^{(i+1)}(x_k, \upsilon_k, u_k) = U(x_k, \upsilon_k, u_k) - \beta U(x_k, \upsilon_k, u_k) + \min_{\upsilon \in \mathcal{P}} \inf_{u \in \mathbb{R}^m} \underline{Q}^{(i)}(x_{k+1}, \upsilon, u) \quad (20)$$

with $\underline{Q}^{(0)}(x, \upsilon, u) = \hat{Q}^{(0)}(x, \upsilon, u) = \overline{Q}^{(0)}(x, \upsilon, u) = 0$.

The boundedness of $\hat{Q}^{(i)}(x, \upsilon, u)$ can be derived by using mathematical induction method. First, it is obvious that (18) holds when $i = 0$. Assume that (18) holds for the ith iteration. Then, according to (17) and (19), one has

$$\overline{Q}^{(i+1)}(x, \upsilon, u) - \hat{Q}^{(i+1)}(x, \upsilon, u) = \beta U(x, \upsilon, u) - \varepsilon^{(i)}(x, \upsilon, u). \quad (21)$$

Since $|\varepsilon^{(i)}(x, \upsilon, u)| \le \beta U(x, \upsilon, u)$, it is obvious that $\overline{Q}^{(i+1)}(x, \upsilon, u) \ge \hat{Q}^{(i+1)}(x, \upsilon, u)$. On the other hand, according to (17) and (20), one has

$$\underline{Q}^{(i+1)}(x, \upsilon, u) - \hat{Q}^{(i+1)}(x, \upsilon, u) = -\beta U(x, \upsilon, u) - \varepsilon^{(i)}(x, \upsilon, u). \quad (22)$$

Similarly, the inequality $\underline{Q}^{(i+1)}(x, \upsilon, u) \le \hat{Q}^{(i+1)}(x, \upsilon, u)$ can be derived if $|\varepsilon^{(i)}(x, \upsilon, u)| \le \beta U(x, \upsilon, u)$ holds. Therefore, we know that (18) holds for $\forall i$. ■

Note that $\{\overline{Q}^{(i)}(x, \upsilon, u)\}_{i=0}^{\infty}$ can be regarded as the sequence of exact HNAF with the performance function $\overline{J} = \sum_{k=0}^{\infty} U(x_k, \upsilon_k, u_k) + \beta U(x_k, \upsilon_k, u_k)$. On the other hand, $\{\underline{Q}^{(i)}(x, \upsilon, u)\}$ can be regarded as the sequence of exact HNAF with the performance function $\underline{J} = \sum_{k=0}^{\infty} U(x_k, \upsilon_k, u_k) - \beta U(x_k, \upsilon_k, u_k)$. Although the approximation error cannot be completely eliminated, it can be significantly reduced by selecting appropriate network structure. Then, according to Theorems 1 and 2, if β is a small value, the sequence of $\hat{Q}^{(i)}(x, \upsilon, u)$ will converge to a near-optimal solution.

TABLE I
PARAMETER SETTINGS FOR SIMULATION CASES

Parameter	Value		
	Case1	Case 2	Case 3
Learning rate (α)	0.0001	0.0001	0.0001
Batch size (B)	128	128	64
Replay buffer size (D)	20000	20000	20000
Iteration number (I)	5000	2000	2000
Max Steps per iteration (M)	50	30	100
Hidden layer units (h)	20	20	20

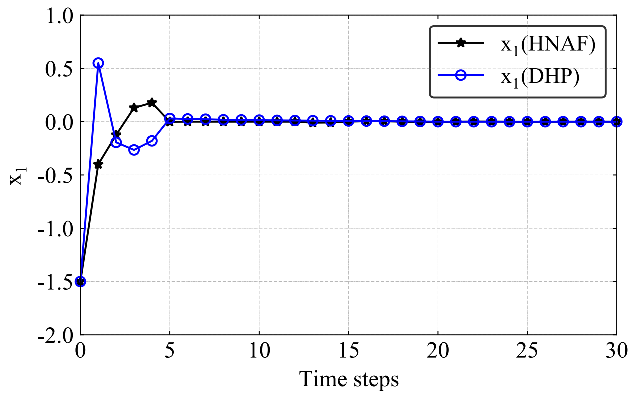

Fig. 2. Trajectory of x_1 with $x_0 = [-1.5, 0.5]^T$.

V. SIMULATION STUDIES

In this section, three simulation examples are presented to verify the effectiveness of the proposed algorithm. In order to demonstrate the effectiveness of the HNAF algorithm, the results are compared with the DHP method [35], the GDHP algorithm [36], and the two-stage VI method (which is presented in Section III-A), respectively. The simulations are run at a desktop with AMD Core 3600 CPU @ 3.60 GHz and Nvidia GTX 1660 super GPU under the Ubuntu 18.04 Operation System. In addition, the Pytorch toolbox is utilized to train NNs.

Case 1: Consider the switching system with the following dynamics [35]:

$$x_{k+1} = \begin{bmatrix} -0.8x_{2_k} \\ \sin(0.8x_{1_k} - x_{2_k}) + 1.8x_{2_k} \end{bmatrix} + \begin{bmatrix} 0 \\ -x_{2_k} \end{bmatrix} u_k, \quad v_k = 1$$
$$x_{k+1} = \begin{bmatrix} 0.5x_{1_k}^2 x_{2_k} \\ x_{1_k} + 0.8x_{2_k} \end{bmatrix} + \begin{bmatrix} 0 \\ x_{1_k} \end{bmatrix} u_k, \quad v_k = 2 \tag{23}$$

where $x_k = [x_{1_k}, x_{2_k}]^T \in \mathbb{R}^2$ denotes the system, $u_k \in \mathbb{R}$ denotes the continuous parameter, and $v_k \in \{1, 2\}$ denotes the discrete action. Define the performance function as (3) with $Q_1 = Q_2 = I_Q$ and $R_1 = R_2 = I_R$, where I_Q and I_R are both identity matrices with appropriate dimensions. The NAF network with the one hidden layer is employed to implement the algorithm. The activation function is RELU [49]. The hyperparameters are listed in column 2 of Table I.

Let the initial state be $x_0 = [-1.5, 0.5]^T$, and apply the trained policy for 30 time steps. The trajectories of system states are shown in Figs. 2 and 3, while the trajectories of continuous parameter and discrete action are given in Figs. 4 and 5, respectively.

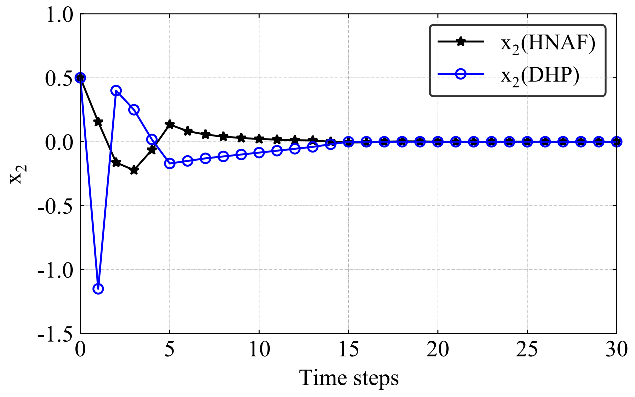

Fig. 3. Trajectory of x_2 with $x_0 = [-1.5, 0.5]^T$.

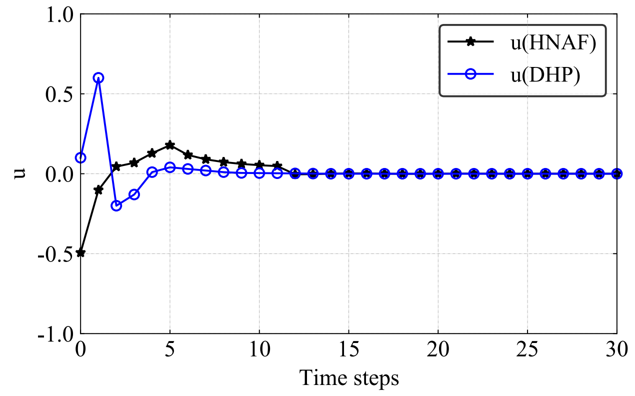

Fig. 4. Trajectory of the continuous parameter with $x_0 = [-1.5, 0.5]^T$.

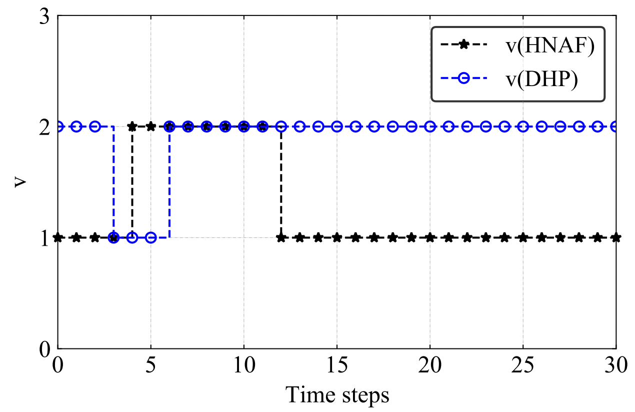

Fig. 5. Trajectory of the discrete action with $x_0 = [-1.5, 0.5]^T$.

Furthermore, the simulation results are compared with those of the model-based DHP algorithm without consideration of control constraints [35]. Starting from the same initial state, the corresponding trajectories are shown in Figs. 2–5 using the dotted blue lines with circle, respectively. By substituting the generated sequence of states and parameters into (3), we can obtain the performance function value of HNAF as 3.2046, while the cost function value of DHP is 5.0302, which demonstrates the effectiveness of our proposed method for non-linear switching systems.

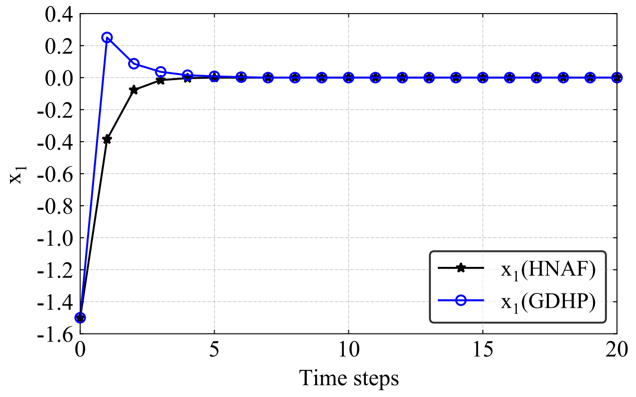

Fig. 6. Trajectory of x_1 with $x_0 = [-1.5, 0.5]^T$.

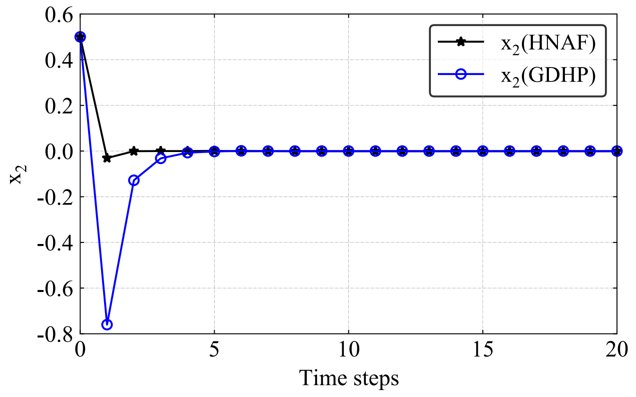

Fig. 7. Trajectory of x_2 with $x_0 = [-1.5, 0.5]^T$.

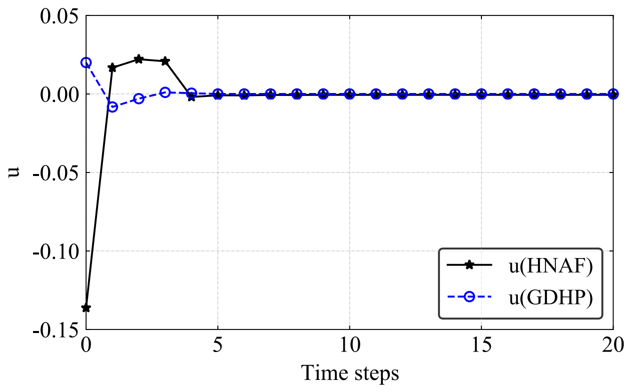

Fig. 8. Trajectory of the continuous parameter with $x_0 = [-1.5, 0.5]^T$.

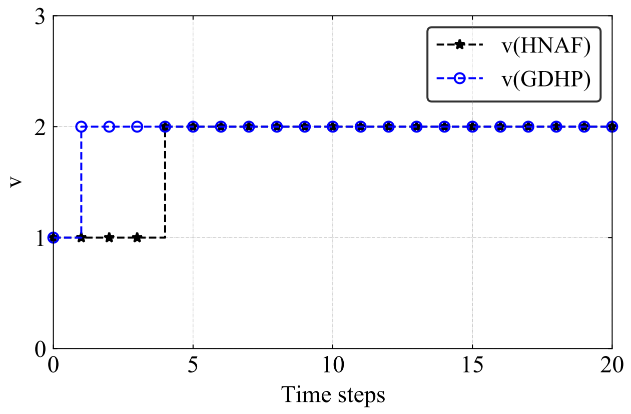

Fig. 9. Trajectory of the discrete action with $x_0 = [-1.5, 0.5]^T$.

Case 2: Consider a non-linear switching system with two modes with the following dynamics [36]:

$$x_{k+1} = \begin{bmatrix} 0.2x_{1_k}e^{x_{2_k}^2} \\ 0.3x_{2_k}^3 \end{bmatrix} + \begin{bmatrix} 0 \\ x_{2_k} \end{bmatrix} u_k, \quad v_k = 1$$
$$x_{k+1} = \begin{bmatrix} -\sin(0.5x_{2_k}) \\ -\cos(1.4x_{2_k})\sin(0.9x_{1_k}) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k, \quad v_k = 2 \qquad (24)$$

where $x_k = [x_{1_k}, x_{2_k}]^T \in \mathbb{R}^2$ denotes the system, $u_k \in \mathbb{R}$ denotes the continuous parameter, and $v_k \in \{1, 2\}$ denotes the discrete action. The performance function is defined as (3) with $Q_1 = Q_2 = I_Q$ and $R_1 = R_2 = I_R$, where I_Q and I_R are both identity matrices with appropriate dimensions. The NAF network is utilized to implement the HNAF algorithm, and the hyperparameters are listed in Table I.

Let the initial state be $x_0 = [-1.5, 0.5]^T$, and apply the trained policy for 20 time steps. The trajectories of system states are shown in Figs. 6 and 7, while the trajectories of continuous parameter and discrete action are given in Figs. 8 and 9, respectively.

Moreover, the simulation results are compared with those of the GDHP method [36]. Figs. 6–9 show the corresponding trajectories using the dotted blue line with circles starting from the same initial states. By substituting the generated sequence of states and parameters into (3), we can obtain the cost function value of the HNAF algorithm as 2.6753, while the cost function value of the GDHP algorithm is 3.1677, which demonstrates the proposed method achieves a better performance without training an extra model network.

Case 3: Finally, the optimal control problem of a hybrid power system with a gasoline-driven mode and an electric-driven mode is considered [50]. The system dynamics is given as follows:

$$x(k+1) = A_1x(k) + B_1u(k), \quad v(k) = 1$$
$$x(k+1) = A_2x(k) + B_2u(k), \quad v(k) = 2 \qquad (25)$$

where $x = [x_1, x_2]^T$ is the system state, and $u(k)$ is the continuous control signal. $v(k) = 1$ denotes the gasoline-driven mode, and $v(k) = 2$ denotes the electric-driven mode. When $v(k) = 1$, the matrices are

$$A_1 = \begin{bmatrix} 1 & 0.05 \\ -0.05 & 0.95 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0 \\ 0.05 \end{bmatrix}. \qquad (26)$$

When $v(k) = 2$, the matrices are

$$A_2 = \begin{bmatrix} 1 & 0.05 \\ -0.05 & 0.975 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0 \\ 0.04 \end{bmatrix}. \qquad (27)$$

The cost function of the gasoline-driven mode is

$$U(x, 1, u) = 100x_1^Tx_1 + 200x_2^Tx_2 + 400u^Tu \qquad (28)$$

while the cost function of the electric-driven mode is

$$U(x, 2, u) = 250x_1^Tx_1 + 200x_2^Tx_2 + 50u^Tu. \qquad (29)$$

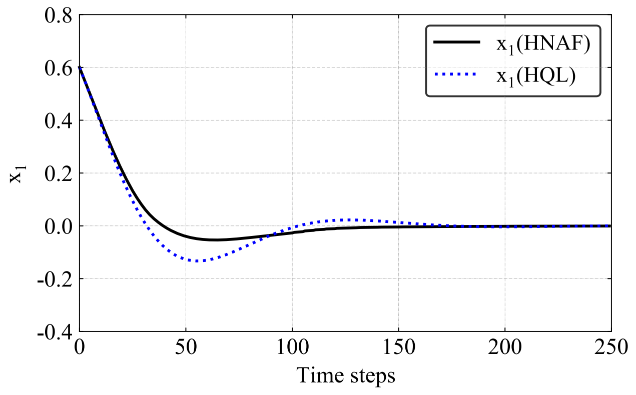

Fig. 10. Trajectory of x_1 with $x_0 = [0.6, -0.4]^T$.

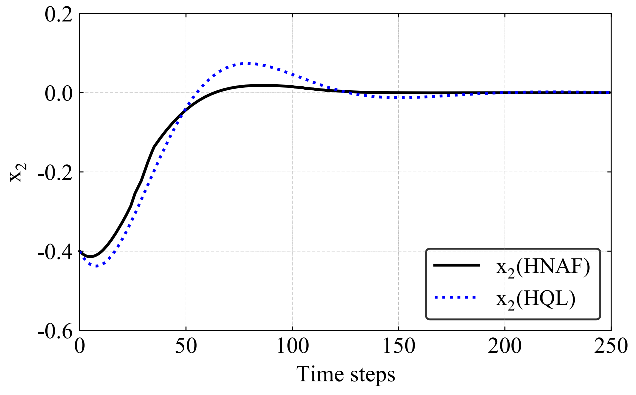

Fig. 11. Trajectory of x_2 with $x_0 = [0.6, -0.4]^T$.

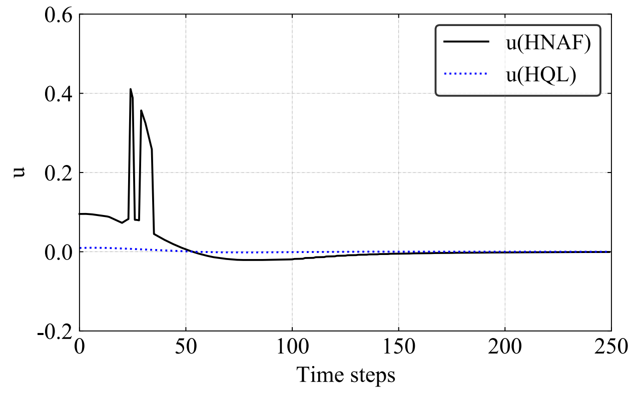

Fig. 12. Trajectory of the continuous control input.

The NAF network is utilized to implement the proposed algorithm, and the hyperparameters are listed in column 4 of Table I. By contrast, we also apply the aforementioned two-stage VI method to this hybrid power system. Note that this algorithm can also learn policies completely from data and requires no information on the system dynamics.

Let the initial state be $x_0 = [0.6, -0.4]^T$. The trajectories of system states are shown in Figs. 10 and 11. In addition, the trajectories of the continuous control input and active mode are given in Figs. 12 and 13, respectively. According to the results, it is obvious that the state of the HNAF

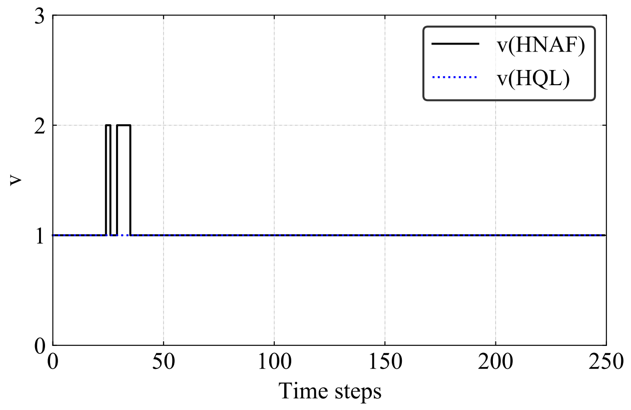

Fig. 13. Trajectory of the active mode.

method converges to the origin much faster. By substituting the generated trajectories into (3), we can obtain the performance function value of the HNAF algorithm is 1379.94, while the value of the VI algorithm is 1473.88, which also verifies the conclusion. Comparing the trajectories, the switching signal of the HNAF method will switch to the electric-driven mode during 25–35 time steps. In addition, when the electric-driven mode is active, the continuous control input is much larger than that in the gasoline-driven mode, which leads to fast convergence of state. According to the cost function [see (28) and (29)], we know that the gasoline-driven mode generates a larger penalty on x_1, while the electric-driven mode generates a larger penalty on u. From this point of view, the policy of the HNAF method makes "smart" decisions and achieves better performance.

VI. Discussion

We propose a data-driven hybrid RL algorithm with a special NN structure to codesign the switching policy and the continuous policy of the controlled switching system. The simulation results demonstrate that the HNAF can always achieve better performance in comparison to DHP, GDHP, and VI methods. Note that no information on system dynamics is required for the HNAF algorithm. We postulate that the online learning manner of the HNAF algorithm makes it able to explore the state and action spaces sufficiently and find better policies. In contrast, the comparison algorithms are all trained on rather small datasets, and there is no guarantee that every mode is sufficiently explored equally. In particular, as for the GDHP algorithm, the identification error of the model network may also degrade the performance. Finally, it is worthwhile to mention that, in general, the advantage function does not need to be quadratic, and exploring other parametric forms may provide substantial improvement.

VII. Conclusion

In this article, a novel data-based scheme is proposed to deal with the hybrid action space of the non-linear switching system. In the first place, a transformed HJB equation is derived, and a hybrid RL algorithm is designed with convergence analysis. By utilizing the HNAF structure, the policies of continuous parameters and discrete action (i.e., the Q-function) can be updated simultaneously. The simulation results demonstrate that the HNAF algorithm can achieve better performance by online exploring the state and action spaces. However, the quadratic formulation of the advantage function may degrade its performance. Therefore, other parametric formulations will be investigated in the future. The optimal control of the switching system with control constraint is also an interesting topic to study.

REFERENCES

[1] D. Liberzon, *Switching in Systems and Control*. Boston, MA, USA: Birkhäuser, 2003.

[2] M. Soler, A. Olivares, E. Staffetti, and D. Zapata, "Framework for aircraft trajectory planning toward an efficient air traffic management," *J. Aircraft*, vol. 49, no. 1, pp. 341–348, Jan./Feb. 2012.

[3] M. Rinehart, M. Dahleh, D. Reed, and I. Kolmanovsky, "Suboptimal control of switched systems with an application to the DISC engine," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 2, pp. 189–201, Mar. 2008.

[4] A. Kouvelas, K. Aboudolas, M. Papageorgiou, and E. B. Kosmatopoulos, "A hybrid strategy for real-time traffic signal control of urban road networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 3, pp. 884–894, Sep. 2011.

[5] A. Tanwani, H. Shim, and D. Liberzon, "Observability for switched linear systems: Characterization and observer design," *IEEE Trans. Autom. Control*, vol. 58, no. 4, pp. 891–904, Apr. 2013.

[6] Y.-E. Wang, J. Zhao, and B. Jiang, "Stabilization of a class of switched linear neutral systems under asynchronous switching," *IEEE Trans. Autom. Control*, vol. 58, no. 8, pp. 2114–2119, Aug. 2013.

[7] F. Zhu and P. J. Antsaklis, "Optimal control of hybrid switched systems: A brief survey," *Discrete Event Dyn. Syst.*, vol. 25, no. 3, pp. 345–364, Sep. 2014.

[8] X. Xu and P. J. Antsaklis, "Optimal control of switched autonomous systems," in *Proc. 41st IEEE Conf. Decis. Control*, vol. 4, Dec. 2002, pp. 4401–4406.

[9] H. Axelsson, M. Boccadoro, M. Egerstedt, P. Valigi, and Y. Wardi, "Optimal mode-switching for hybrid systems with varying initial states," *Nonlinear Anal., Hybrid Syst.*, vol. 2, no. 3, pp. 765–772, 2008.

[10] X. Xu and P. J. Antsaklis, "Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions," *Int. J. Control*, vol. 75, nos. 16–17, pp. 1406–1426, 2002.

[11] H. Axelsson, M. Egerstedt, Y. Wardi, and G. Vachtsevanos, "Algorithm for switching-time optimization in hybrid dynamical systems," in *Proc. IEEE Int. Symp. Mediterrean Conf. Control Autom. Intell. Control*, Jun. 2005, pp. 256–261.

[12] A. Heydari and S. N. Balakrishnan, "Optimal orbit transfer with ON-OFF actuators using a closed form optimal switching scheme," in *Proc. AIAA Guid., Navigat., Control (GNC) Conf.*, Aug. 2013, p. 4635.

[13] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control: Model and optimal control theory," *IEEE Trans. Autom. Control*, vol. 43, no. 1, pp. 31–45, Jan. 1998.

[14] S. C. Bengea and R. A. DeCarlo, "Optimal control of switching systems," *Automatica*, vol. 41, no. 1, pp. 11–27, 2005.

[15] M. Kamgarpour and C. Tomlin, "On optimal control of non-autonomous switched systems with a fixed mode sequence," *Automatica*, vol. 48, no. 6, pp. 1177–1181, 2012.

[16] X. Xu and P. J. Antsaklis, "Optimal control of switched systems based on parameterization of the switching instants," *IEEE Trans. Autom. Control*, vol. 49, no. 1, pp. 2–16, Jan. 2004.

[17] W. Zhang and J. Hu, "On optimal quadratic regulation for discrete-time switched linear systems," in *Hybrid Systems: Comput. Control*, M. Egerstedt and B. Mishra, Eds. Berlin, Germany: Springer, 2008, pp. 584–597.

[18] W. Zhang, J. Hu, and A. Abate, "Infinite-horizon switched LQR problems in discrete time: A suboptimal algorithm with performance analysis," *IEEE Trans. Autom. Control*, vol. 57, no. 7, pp. 1815–1821, Jul. 2012.

[19] D. Gorges, M. Izak, and S. Liu, "Optimal control and scheduling of switched systems," *IEEE Trans. Autom. Control*, vol. 56, no. 1, pp. 135–140, Jan. 2011.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[21] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[22] D. Silver *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018.

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

[24] 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.

[25] X. Li, L. Dong, and C. Sun, "Data-based optimal tracking of autonomous nonlinear switching systems," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 1, pp. 227–238, Jan. 2021.

[26] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 943–949, Aug. 2008.

[27] C. Mu, D. Wang, and H. He, "Novel iterative neural dynamic programming for data-based approximate optimal control design," *Automatica*, vol. 81, pp. 240–252, Jul. 2017.

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

[29] H. Zhang, R. Song, Q. Wei, and T. Zhang, "Optimal tracking control for a class of nonlinear discrete-time systems with time delays based on heuristic dynamic programming," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 62–1851, 2011.

[30] H. Zhang, Y. Luo, and D. Liu, "Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1490–1503, Sep. 2009.

[31] L. Dong, X. Zhong, C. Sun, and H. He, "Adaptive event-triggered control based on heuristic dynamic programming for nonlinear discrete-time systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 7, pp. 1594–1605, Jul. 2016.

[32] A. Heydari, "Optimal switching of DC–DC power converters using approximate dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 3, pp. 586–596, Mar. 2018.

[33] A. Heydari, "Optimal switching with minimum dwell time constraint," *J. Franklin Inst.*, vol. 354, no. 11, pp. 4498–4518, Jul. 2017.

[34] 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.

[35] H. Zhang, C. Qin, and Y. Luo, "Neural-network-based constrained optimal control scheme for discrete-time switched nonlinear system using dual heuristic programming," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 3, pp. 839–849, Jul. 2014.

[36] C. Mu, K. Liao, L. Ren, and Z. Gao, "Approximately optimal control of discrete-time nonlinear switched systems using globalized dual heuristic programming," *Neural Process. Lett.*, vol. 52, no. 2, pp. 1089–1108, Oct. 2020.

[37] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 2829–2838.

[38] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*. New York, NY, USA: Wiley, 2012.

[39] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*. Philadelphia, PA, USA: SIAM, 2000.

[40] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[41] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Conf. Operating Syst. Design Implement.*, vol. 16, Nov. 2016, pp. 265–283.

[42] T. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–14.

[43] W. Masson, P. Ranchod, and G. Konidaris, "Reinforcement learning with parameterized actions," in *Proc. 30th AAAI Conf. Artif. Intell. (AAAI)*, 2016, pp. 1934–1940.

[44] M. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," 2015, *arXiv:1511.04143*.

[45] J. Xiong *et al.*, "Parametrized deep Q-networks learning: Reinforcement learning with discrete-continuous hybrid action space," 2018, *arXiv:1810.06394*.

[46] C. J. Bester, S. D. James, and G. D. Konidaris, "Multi-pass Q-networks for deep reinforcement learning with parameterised action spaces," 2019, *arXiv:1905.04388*.

[47] A. Heydari, "Optimal codesign of control input and triggering instants for networked control systems using adaptive dynamic programming," *IEEE Trans. Ind. Electron.*, vol. 66, no. 1, pp. 482–490, Jan. 2019.

[48] A. Heydari, "Theoretical analysis of stabilizing value iteration with approximation errors," in *Proc. Amer. Control Conf. (ACC)*, May 2017, pp. 5312–5317.

[49] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[50] W. Lu, P. Zhu, and S. Ferrari, "A hybrid-adaptive dynamic programming approach for the model-free control of nonlinear switched systems," *IEEE Trans. Autom. Control*, vol. 61, no. 10, pp. 3203–3208, Oct. 2016.

Xiaofeng Li received the B.S. and M.S. degrees in engineering from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2012 and 2016, respectively, and the Ph.D. degree in control science and engineering from Southeast University, Nanjing, in 2021.

He was a Joint Ph.D. Student with the Department of Electrical, Computer, and Biomedical Engineering, The University of Rhode Island, Kingston, RI, USA, from 2018 to 2019. He is currently a Post-Doctoral Researcher with the School of Artificial Intelligence, Anhui University, Heifei, China. His current research interests include reinforcement learning, adaptive dynamic programming, robot systems, and optimal control.

Lu Dong (Member, IEEE) received the B.S. degree in physics and the Ph.D. degree in electrical engineering from Southeast University, Nanjing, China, in 2012 and 2017, respectively.

She is currently an Associate Professor with the School of Cyber Science and Engineering, Southeast University. Her current research interests include adaptive dynamic programming, event-triggered control, nonlinear system control, and optimization.

Lei Xue (Member, IEEE) received the Ph.D. degree in control science and engineering from Southeast University, Nanjing, China, in 2017.

From September 2013 to September 2014, he was a Visiting Ph.D. Student with the Applied Computational Intelligence Laboratory, Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO, USA. He is currently an Associate Professor with the School of Automation, Southeast University. His research interests include game theory, multiagent systems, and optimization control.

Changyin Sun (Senior Member, IEEE) received the B.S. degree in applied mathematics from the College of Mathematics, Sichuan University, Chengdu, China, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from Southeast University, Nanjing, China, in 2001 and 2004, respectively.

He is currently a Professor with the School of Automation, Southeast University. His current research interests include intelligent control, flight control, and optimal theory.

Dr. Sun is also an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *Neural Processing Letters*, and the IEEE/CAA JOURNAL OF AUTOMATICA SINICA.