Approximate Optimal Trajectory Tracking With Sparse Bellman Error Extrapolation

Max L. Greene, *Member, IEEE*, Patryk Deptula, Scott Nivison, *Member, IEEE*, and Warren E. Dixon, *Fellow, IEEE*

***Abstract*—This article provides an approximate online adaptive solution to the infinite-horizon optimal tracking problem for control-affine continuous-time nonlinear systems with uncertain drift dynamics. A model-based approximate dynamic programming (ADP) approach, which is facilitated using a concurrent learning-based system identifier, approximates the optimal value function. To reduce the computational complexity of model-based ADP, the state space is segmented into user-defined segments (i.e., regions). Off-policy trajectories are selected within each segment to facilitate learning of the value function weight estimates; this process is called Bellman error (BE) extrapolation. Within certain segments of the state space, sparse neural networks are used to reduce the computational expense of BE extrapolation. Discontinuities occur in the weight update laws since different groupings of extrapolated BE trajectories are active in certain regions of the state space. A Lyapunov-like stability analysis is presented to prove boundedness of the overall system in the presence of discontinuities. Simulation results are included to demonstrate the performance and validity of the developed method. The simulation results demonstrate that using the sparse, switched BE extrapolation method developed in this article reduces the computation time by 85.6% when compared to the traditional BE extrapolation method.**

***Index Terms*—Adaptive control, nonlinear control, optimal control, reinforcement learning.**

I. Introduction

Reinforcement learning (RL)-based methods, such as [1]–[4], have been used to obtain online approximate solutions to optimal control problems for systems with finite state spaces and stationary environments. Generally, when formulating optimal control problems, the Hamilton–Jacobi–Bellman (HJB) equation provides an optimality condition. The designed optimal control policy depends on the value function [5]. It is generally difficult to solve the HJB equation due to uncertainties and nonlinearities in the system. To combat this deficiency, approximate dynamic programming (ADP) has become a popular method to approximate the value function online (e.g., [6]–[10]). By approximating the optimal value function, a stabilizing and approximately optimal control policy can be determined.

In trajectory tracking optimal control problems, the value function explicitly depends on time. Since universal function approximators can approximate functions with arbitrary accuracy only on compact domains, the infinite-horizon optimal tracking value function cannot be directly approximated. Developments, such as [11], provide frameworks to reformulate the system into a stationary environment by separating the controller into steady-state and transient components. In such approaches, the optimal control problems can be reformulated to penalize the transient portion of the controller and eliminate the time-dependency of the value function. The caveat is that this method requires the exact model knowledge to identify the steady-state component of the controller. To alleviate the need for exact model knowledge, a concurrent learning (CL)-based adaptation law can be used to identify the drift dynamics online.

Traditional adaptive control-based ADP methods require a persistence of excitation (PE) condition to be satisfied to successfully approximate the value function and drift dynamics [2]. Satisfying the PE conditions often motivates *ad hoc* methods, which may affect performance or destabilize the system. To relax the PE condition, the method in this article uses a CL-based system identifier (see [12] and [13]) and a model-based RL (MBRL)-based technique to approximate the value function online (see [14] and [15]).

MBRL methods are useful for applications in which a model of the system is available (e.g., robotic manipulators) [16] and [17]. Utilizing a model to complete an RL-based objective leads to faster convergence to an approximate optimal solution [16]. The Bellman error (BE) is used as a performance metric in ADP. The BE indirectly measures the quality of the estimation of the value function along the system trajectory. Previous works (e.g., [14] and [15]) show that if the system dynamics are successfully approximated, then the BE can be evaluated at an arbitrary number of points in a system's state space. This process is called BE extrapolation. To facilitate improved data richness and value function approximation, BE extrapolation is performed at a sufficiently large number of user-selected off-trajectory points. These off-trajectory extrapolation points are sometimes placed over large regions of the state space in high density. Each BE extrapolation point requires additional computations. Furthermore, since value function approximation is sought over a large compact domain of the state space, neurons are similarly distributed across a large region of the state space. Like BE extrapolation points, every additional neuron in the basis function for value function approximation increases the total number of computations. Together, additional BE extrapolation points and neurons compound the computational complexity of performing regional model-based ADP. Results in [18] and [19] address the computational drawbacks of BE extrapolation by leveraging local and computationally efficient state-following (StaF) kernels. However, StaF kernels do not facilitate function approximation

Manuscript received 21 February 2022; accepted 9 July 2022. Date of publication 27 July 2022; date of current version 29 May 2023. This work was supported in part by the Office of Naval Research Grant N00014-21-1-2481, in part by AFRL Award FA8651-21-F-1027, and in part by the AFOSR Grant FA9550-19-1-0169. Recommended by Associate Editor M. Guay. *(Corresponding author: Max L. Greene.)*

Max L. Greene is with the Aurora Flight Sciences, a Boeing Company, Cambridge, MA 02142 USA (e-mail: greene.max@aurora.aero).

Patryk Deptula is with the Charles Stark Draper Laboratory, Inc., Cambridge, MA 02139 USA (e-mail: pdeptula@draper.com).

Scott Nivison is with the Johns Hopkins University Applied Physics Laboratory, Fort Walton Beach, FL 32548 USA (e-mail: scott.nivison@jhuapl.edu).

Warren E. Dixon is with the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: wdixon@ufl.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TAC.2022.3194040.

Digital Object Identifier 10.1109/TAC.2022.3194040

across the entire operating domain. Computational efficiency is necessary, and it is desirable to perform traditional BE extrapolation over the entire operating domain to improve value function approximation. Hence, this article develops two modifications to existing BE extrapolation to increase computational efficiency for the model-based ADP tracking problem. The first modification is to use sparse neural networks (SNNs) to alleviate the computational expense of BE extrapolation, and the second is to dynamically select BE extrapolation points and basis functions based on the system state.

SNNs are a tool to facilitate learning in uncertain systems [20]–[23] by decreasing the number of active neurons in a neural network (NN). By reducing the number of active neurons, then the number of overall computations is also reduced. SNN-based adaptive controllers have been developed to update a smaller number of neurons within certain regions in the state space [22]. Making an NN more sparse (i.e., sparsification) encourages local learning through intelligently segmenting the state space [21]. Sparsification enables local approximation within each segment, which characterizes regions with significantly varying dynamics or unknown uncertainties. Motivated by the idea of segmenting the state space and to facilitate improved learning, the developed method divides the state space into smaller regions. Each region contains a unique distribution of extrapolation points and NNs, some of which may be SNNs, that facilitate BE extrapolation. In doing so, a subset of the total number of BE extrapolation points are used and the number of computations associated with the NN basis functions is decreased while retaining sufficient data richness. However, as the system state moves between the divided regions of the state space, discontinuities are introduced into the weight update laws. Such discontinuities require specific attention within the Lyapunov-like analysis to prove stability.

The contribution of this article and its preliminary works in [23] and [24] is to analyze the stability of using a sparse switched BE term with an NN-based estimator. The primary contribution extends beyond our preliminary works in [23] and [24] by extending them to the optimal tracking problem with completely unknown drift dynamics and by quantifying the benefit of using SNNs in BE extrapolation via simulations. Unlike the previous ADP-based approximately optimal trajectory tracking result in [11], this article examines the impact of SNN-based BE extrapolation and simultaneous system identification on the stability of the dynamical system. A simulation study is performed to demonstrate the ability to simultaneously approximate the system dynamics and optimal control policy; furthermore, this study quantifies the benefit of using sparse BE extrapolation by examining five simulation cases with varying degrees of sparsity. The simulation result shows that the developed sparse BE extrapolation method reduces simulation computation time by 85.6% when compared to the traditional BE extrapolation method.

The remainder of this article is structured as follows. Section II formulates the tracking ADP problem and objectives. Section III outlines the NN-based system identifier, which is used to approximate the system dynamics online. Section IV introduces sparse BE extrapolation, which requires the NN-based model from Section III. Section V defines the update laws that facilitate the stability analysis. Section VI presents a nonsmooth Lyapunov-like stability analysis. Section VII presents multiple simulation results to illustrate the effectiveness of the developed technique in comparison to an existing ADP-based result in [15]. Finally, Section VIII concludes this article.

A. Notation

In the following manuscript, for notational brevity, time-dependence is omitted while denoting trajectories of the dynamical systems. For example, the trajectory $x(t)$, where $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, is denoted as $x \in \mathbb{R}^n$ and referred to as x instead of $x(t)$. For example, an equation of the form $f + h(y,t) = g(x)$ should be interpreted as $f(t) + h(y(t),t) = g(x(t)) \, \forall t \in \mathbb{R}_{\geq 0}$. The gradient $[\frac{\partial f(x,y)}{\partial x_1}^T, \ldots, \frac{\partial f(x,y)}{\partial x_n}^T]^T$ is denoted by $\nabla_x f(x,y)$. $\|\cdot\|$ denotes both the Euclidean norm for vectors and Frobenius norm for matrices. $\mathbf{1}_{n\times m}$ and $\mathbf{0}_{n\times m}$ denote matrices of ones and zeros with n rows and m columns, respectively. $I_{n\times n}$ denotes an $n \times n$ identity matrix.

II. Background Information

A. Problem Formulation

Consider a class of nonlinear control-affine systems $\dot{x} = f(x) + g(x)u$, where $x \in \mathbb{R}^n$ denotes the system state, $u \in \mathbb{R}^m$ denotes the control input, $f : \mathbb{R}^n \to \mathbb{R}^n$ is the drift dynamics, and $g : \mathbb{R}^n \to \mathbb{R}^{m\times n}$ is the control effectiveness matrix, where $n > m$ and the pseudoinverse of $g(x)$ exists. The control objective is to track a time-varying continuously differentiable signal $x_d \in \mathbb{R}^n$. To quantify the tracking objective, the tracking error is defined as $e \triangleq x - x_d$. Using the technique in [15] to transform the time-varying tracking problem into an infinite horizon regulation problem, the control-affine dynamics can be rewritten as

$$\dot{\zeta} = F(\zeta) + G(\zeta)\mu \tag{1}$$

where $\zeta \in \mathbb{R}^{2n}$ is the concatenated state vector $\zeta \triangleq [e^T, x_d^T]^T$, $\mu \triangleq u - u_d(x_d)$ is the transient portion of the controller, $h_d : \mathbb{R}^n \to \mathbb{R}^n$ is subsequently-defined, $u_d : \mathbb{R}^n \to \mathbb{R}^m$ is the subsequently-defined trajectory tracking component of the controller, $F : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ is defined as

$$F(\zeta) \triangleq \begin{bmatrix} f(e + x_d) - h_d(x_d) + g(e + x_d)u_d(x_d) \\ h_d(x_d) \end{bmatrix} \tag{2}$$

and $G : \mathbb{R}^{2n} \to \mathbb{R}^{2n\times m}$ is defined as

$$G(\zeta) \triangleq \begin{bmatrix} g(e + x_d)^T, \mathbf{0}_{m\times n} \end{bmatrix}^T. \tag{3}$$

The action space for μ is $U \subset \mathbb{R}^m$. The following assumptions facilitate the formulation of the approximate optimal tracking controller [15].

Assumption 1: The function f is continuously differentiable and $f(0) = 0$.

Assumption 2: The function g is a known locally Lipschitz and bounded such that $0 < \|g(x)\| \leq \overline{g} \, \forall x \in \mathbb{R}^n$, where $\overline{g} \in \mathbb{R}_{>0}$ is the supremum over all x of the maximum singular values of $g(x)$. It follows that $0 < \|G(\zeta)\| \leq \overline{G} \, \forall \zeta \in \mathbb{R}^{2n}$, where $\overline{G} \in \mathbb{R}_{>0}$ is a known constant.

Assumption 3: The desired trajectory is bounded from above by a known positive constant $\overline{x_d} \in \mathbb{R}$ such that $\sup_{t\in\mathbb{R}_{\geq 0}} \|x_d(t)\| \leq \overline{x_d}$.

Assumption 4: There exists a locally Lipschitz function h_d such that $h_d(x_d) \triangleq \dot{x}_d$ and $g(x_d)g^+(x_d)(h_d(x_d) - f(x_d)) = h_d(x_d) - f(x_d)$, $\forall t \in \mathbb{R}_{\geq 0}$, where $g^+ : \mathbb{R}^n \to \mathbb{R}^{m\times n}$ is defined as $g^+(x) \triangleq (g^T(x)g(x))^{-1}g^T(x)$.

Based on Assumptions 2–4, the trajectory tracking component of the controller $u_d(x_d)$ is defined as $u_d(x_d) \triangleq g^+(x_d)(h_d(x_d) - f(x_d))$. The trajectory tracking component $u_d(x_d)$ requires knowledge of the drift dynamics f. Since f is unknown, an approximation of the trajectory tracking component $\hat{u}_d$ is developed in Section III.

B. Control Objective

The control objective is to solve the infinite-horizon optimal tracking problem, i.e., to find a control policy μ that minimizes the cost functional

$$J(\zeta, \mu) = \int_0^\infty r(\zeta(\tau), \mu(\tau)) \, d\tau \tag{4}$$

subject to (1) while eliminating tracking error (i.e., $e = 0$), where $r : \mathbb{R}^{2n} \times \mathbb{R}^m \to \mathbb{R}$ is the instantaneous cost, which is defined

as $r(\zeta,\mu) \triangleq \overline{Q}(\zeta) + \mu^T R\mu$, where $\overline{Q} \in \mathbb{R}^{2n} \to \mathbb{R}_{\geq 0}$ is a positive semidefinite (PSD) user-defined state cost function and $R \in \mathbb{R}^{m\times m}$ is user-defined positive definite (PD) symmetric input cost matrix. Let $\overline{Q}(\zeta) \triangleq Q(e)$, where $Q : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is a PD user-defined cost function that penalizes the error e and not the desired trajectory x_d.

Property 1: The function $\overline{Q}$ is PSD and satisfies $\underline{q}(\|e\|) \leq \overline{Q}(\zeta) \leq \overline{q}(\|e\|)$ for $\underline{q}, \overline{q} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$.

The scalar infinite-horizon optimal value function, i.e., the cost-to-go, denoted by $V^* : \mathbb{R}^{2n} \to \mathbb{R}_{\geq 0}$, is given by $V^*(\zeta) = \min_{\mu\in U} \int_t^\infty r(\zeta(\tau),\mu(\tau))d\tau$, which has the boundary condition $V^*(0) = 0$. If the optimal value function is continuously differentiable, then the optimal control policy $\mu^* : \mathbb{R}^{2n} \to \mathbb{R}^m$ can be obtained from the corresponding HJB equation

$$0 = \min_{\mu\in U} \nabla_\zeta V^*(\zeta)\left(F(\zeta) + G(\zeta)\mu\right) + \overline{Q}(\zeta) + \mu(\zeta)^T R\mu(\zeta) \tag{5}$$

which yields $\mu^*(\zeta) = -\frac{1}{2}R^{-1}G(\zeta)^T(\nabla_\zeta V^*(\zeta))^T$. Generally, the HJB equation cannot be solved analytically with the exception of a few cases, such as linear or scalar systems.

Remark 1: Under Assumptions 1–4, the optimal value function can be shown to be the unique PD solution of the HJB equations. Approximation of the PD solution to the HJB equation is guaranteed by appropriately selecting initial weight estimates and Lyapunov-based update laws [25].

C. Value Function Approximation

The solution to the HJB equation in (5) is the optimal value function. Parametric methods can be used to approximate the value function over a compact domain $\zeta \in \Omega \subset \mathbb{R}^{2n}$.[1] The universal function approximation property of NNs is used to represent the value function as

$$V^*(\zeta) = W^T\sigma(\zeta) + \varepsilon(\zeta) \tag{6}$$

where $W \in \mathbb{R}^L$ is an unknown bounded weight vector, $\sigma : \mathbb{R}^{2n} \to \mathbb{R}^L$ is a user-defined vector of continuous basis functions, and $\varepsilon : \mathbb{R}^{2n} \to \mathbb{R}$ is the bounded function approximation error [2, Eq. 7.3].

Assumption 5: There exist constants $\overline{W}, \overline{\sigma}, \overline{\nabla_\zeta\sigma}, \overline{\varepsilon}, \overline{\nabla_\zeta\varepsilon} \in \mathbb{R}_{>0}$ such that the unknown weights W, user-defined vector of activation functions σ, and function approximation error ε, can be bounded such that $\|W\| \leq \overline{W}$, $\sup_{\zeta\in\Omega}\|\sigma(\zeta)\| \leq \overline{\sigma}$, $\sup_{\zeta\in\Omega}\|\nabla_\zeta\sigma(\zeta)\| \leq \overline{\nabla_\zeta\sigma}$, $\sup_{\zeta\in\Omega}\|\varepsilon(\zeta)\| \leq \overline{\varepsilon}$, and $\sup_{\zeta\in\Omega}\|\nabla_\zeta\varepsilon(\zeta)\| \leq \overline{\nabla_\zeta\varepsilon}$ [2, Assumptions 9.1.c-e].

Remark 2: The bound $\sup_{\zeta\in\Omega}\|\sigma(\zeta)\| \leq \overline{\sigma}$ can be easily satisfied using activation functions, such as radial basis functions and hyperbolic tangent functions. Other activation functions, such as linear rectifier units or polynomials, which are not bounded functions on $\mathbb{R}$, can be bounded on a compact subset of $\mathbb{R}$. Theorem 1 proves that if ζ is initialized in an appropriately sized compact set, then it will remain in that set. Since ζ lies on a compact set and if the activation function is continuous, then $\sup_{\zeta\in\Omega}\|\sigma(\zeta)\| \leq \overline{\sigma}$ [26, 4.13–4.16].

Solving (5) for μ^* and using (6), the optimal control policy $\mu^* : \mathbb{R}^{2n} \to \mathbb{R}^m$ is

$$\mu^*(\zeta) = -\frac{1}{2}R^{-1}G(\zeta)^T\left(\nabla_\zeta\sigma(\zeta)^T W + \nabla_\zeta\varepsilon(\zeta)^T\right). \tag{7}$$

The ideal weights W in (6) and (7) are unknown; hence, an approximation of W is sought. Specifically, the critic estimate, $\hat{W}_c \in \mathbb{R}^L$ is substituted to approximate the value function $\hat{V} : \mathbb{R}^{2n} \times \mathbb{R}^L \to \mathbb{R}$ denoted as

$$\hat{V}\left(\zeta,\hat{W}_c\right) = \hat{W}_c^T\sigma(\zeta). \tag{8}$$

Using an actor–critic approach (see [4] and [27]), the actor estimate $\hat{W}_a \in \mathbb{R}^L$ is substituted to approximate the optimal control policy $\hat{\mu} : \mathbb{R}^{2n} \times \mathbb{R}^L \to \mathbb{R}$ defined as

$$\hat{\mu}\left(\zeta,\hat{W}_a\right) \triangleq -\frac{1}{2}R^{-1}G(\zeta)^T\left(\nabla_\zeta\sigma(\zeta)^T\hat{W}_a\right). \tag{9}$$

III. System Identification

Another NN can be used to approximate the drift dynamics f over a compact domain[1] $x \in \mathcal{C} \subset \mathbb{R}^n$ as $f(x) = \theta^T\phi(Y^T x_\theta(x)) + \varepsilon_\theta(x)$, where $x_\theta : \mathbb{R}^n \to \mathbb{R}^{n+1}$ is defined as $x_\theta(x) \triangleq [1, x^T]^T$, $\theta \in \mathbb{R}^{(p+1)\times n}$ is a constant, unknown output-layer weight matrix, $Y \in \mathbb{R}^{(n+1)\times p}$ denotes the constant input-layer weight matrix, $\phi : \mathbb{R}^p \to \mathbb{R}^{p+1}$ is an NN basis function that contains an optional bias element, $\varepsilon_\theta : \mathbb{R}^n \to \mathbb{R}^n$ is the NN reconstruction error, and $p \in \mathbb{N}$ is the user-defined number of neurons in the NN. Using the universal function approximation property of single layer NNs there exists constant weights $\overline{\theta}$, $\overline{\phi}$, $\overline{\nabla_x\phi}$, $\overline{\varepsilon_\theta}$, and $\overline{\nabla_x\varepsilon_\theta} \in \mathbb{R}_{\geq 0}$, such that $\|\theta\| \leq \overline{\theta}$, $\sup_{x\in\mathcal{C}}\|\phi(x)\| \leq \overline{\phi}$, $\sup_{x\in\mathcal{C}}\|\nabla_x\phi(x)\| \leq \overline{\nabla_x\phi}$, $\sup_{x\in\mathcal{C}}\|\varepsilon_\theta(x)\| \leq \overline{\varepsilon_\theta}$, and $\sup_{x\in\mathcal{C}}\|\nabla_x\varepsilon_\theta(x)\| \leq \overline{\nabla_x\varepsilon_\theta}$ [2, Assumptions 9.1.c-e].

Let $\hat{\theta} \in \mathbb{R}^{(p+1)\times n}$ be an estimate of the ideal weight matrix θ. The drift dynamics f are approximated by the function $\hat{f} : \mathbb{R}^n \times \mathbb{R}^{(p+1)\times n} \to \mathbb{R}^n$ defined as $\hat{f}(x,\hat{\theta}) \triangleq \hat{\theta}^T\phi(Y^T x_\theta(x))$. Hence, a state estimator can be developed as

$$\dot{\hat{x}} = \hat{f}\left(x,\hat{\theta}\right) + g(x)u + k\tilde{x} \tag{10}$$

where $\tilde{x} \triangleq x - \hat{x}$ and $k \in \mathbb{R}_{>0}$ is a user-selected estimator learning gain.

Assumption 6: A history stack of input–output data pairs $\{x_j, u_j\}_{j=1}^M$ and history stack of numerically computed state derivatives $\{\dot{\bar{x}}_j\}_{j=1}^M$, which satisfies $\lambda_{\min}(\sum_{j=1}^M \phi_j\phi_j^T) > 0$ and $\|\dot{\bar{x}}_j - \dot{x}_j\| < \overline{d}$ $\forall j$ are available *a priori*, where $\overline{d} \in \mathbb{R}_{>0}$ is a positive constant, $\dot{x}_j = f(x_j) + g(x_j)u_j$, $\phi_j \triangleq \phi(Y^T x_\theta(x_j))$, and the operator $\lambda_{\min}(\cdot)$ represents the minimum eigenvalue of the argument [12].[2]

The update law of the system identification NN weight estimates are updated using the CL-based update law

$$\dot{\hat{\theta}} = \Gamma_\theta\phi\left(Y^T x_\theta(x)\right)\tilde{x}^T + k_\theta\Gamma_\theta\sum_{j=1}^M \phi_j\left(\dot{\bar{x}}_j - g(x_j)u_j - \hat{\theta}^T\phi_j\right)^T \tag{11}$$

where $\Gamma_\theta \in \mathbb{R}^{(p+1)\times(p+1)}$ and $k_\theta \in \mathbb{R}_{>0}$ are constant user-selected adaptation gains.

Since f is unknown, then the trajectory tracking component of the controller $u_d(x_d)$ is not known. An approximation of the trajectory tracking component $\hat{u}_d : \mathbb{R}^n \times \mathbb{R}^{(p+1)\times n} \to \mathbb{R}^m$ is defined as $\hat{u}_d(x_d,\hat{\theta}) \triangleq g^+(x_d)(h_d(x_d) - \hat{f}(x,\hat{\theta}))$. Hence, the applied control policy is

$$u \triangleq \hat{\mu}\left(\zeta,\hat{W}_a\right) + \hat{u}_d\left(x_d,\hat{\theta}\right). \tag{12}$$

[1] Theorem 1 proves that if ζ and x are initialized within appropriately sized compact sets, then they will remain in their respective compact sets.

[2] The availability of the system identification history stack *a priori* is not necessary [15]. Assumption 6 is used to focus the scope of this manuscript and simplify the subsequent stability analysis.

IV. Bellman Error

The right-hand side of the HJB equation in (5) is equal to zero under optimal conditions; however, substituting (8), (9), and the approximated drift dynamics $\hat{f}(x,\hat{\theta})$ into (5) results in a residual term, $\hat{\delta}:\mathbb{R}^{2n}\times\mathbb{R}^{(p+1)\times n}\times\mathbb{R}^{L}\times\mathbb{R}^{L}\to\mathbb{R}$, which is referred to as the BE, defined as

$$\hat{\delta}\left(\zeta,\hat{\theta},\hat{W}_c,\hat{W}_a\right)\triangleq\hat{\mu}\left(\zeta,\hat{W}_a\right)^T R\hat{\mu}\left(\zeta,\hat{W}_a\right)+\overline{Q}\left(\zeta\right)+\nabla_\zeta\hat{V}\left(\zeta,\hat{W}_c\right)\left(F_\theta\left(\zeta,\hat{\theta}\right)+F_1\left(\zeta\right)+G\left(\zeta\right)\hat{\mu}\left(\zeta,\hat{W}_a\right)\right)\tag{13}$$

where $F_\theta:\mathbb{R}^{2n}\times\mathbb{R}^{(p+1)\times n}\to\mathbb{R}^{2n}$ is defined as $F_\theta(\zeta,\hat{\theta})\triangleq[(\hat{f}(x,\hat{\theta})-g(x)g^+(x_d)\hat{f}(x_d,\hat{\theta}))^T,\ \mathbf{0}_{1\times n}]^T$, and $F_1:\mathbb{R}^{2n}\to\mathbb{R}^{2n}$ is defined as $F_1(\zeta)\triangleq[(-h_d(x_d)+g(x)g^+(x_d)h_d(x_d))^T,\ h_d(x_d)^T]^T$. The BE in (13) indicates how close the actor and critic weight estimates are to their respective ideal weights. The mismatch between the estimates and their ideal values are defined as $\tilde{W}_c\triangleq W-\hat{W}_c$ and $\tilde{W}_a\triangleq W-\hat{W}_a$. Substituting (6) and (7) into (5), and subtracting from (13) yields the analytical form of the BE, given by

$$\hat{\delta}\left(\zeta,\hat{\theta},\hat{W}_c,\hat{W}_a\right)=-W^T\nabla_\zeta\sigma\left(F_\theta\left(\zeta,\theta\right)-F_\theta\left(\zeta,\hat{\theta}\right)\right)-\omega^T\tilde{W}_c+\frac{1}{4}\tilde{W}_a^T G_\sigma\tilde{W}_a+O\left(\zeta\right)\tag{14}$$

where $\omega:\mathbb{R}^{2n}\times\mathbb{R}^{L}\times\mathbb{R}^{(p+1)\times n}\to\mathbb{R}^{L}$ is defined as $\omega=\omega(\zeta,\hat{W}_a,\hat{\theta})\triangleq\nabla_\zeta\sigma(\zeta)F_\theta(\zeta,\hat{\theta})+\nabla_\zeta\sigma(\zeta)F_1(\zeta)+\nabla_\zeta\sigma(\zeta)G(\zeta)\hat{\mu}(\zeta,\hat{W}_a)$, $G_R=G_R(\zeta)\triangleq G(\zeta)R^{-1}G(\zeta)^T$, $G_\sigma=G_\sigma(\zeta)\triangleq\nabla_\zeta\sigma(\zeta)G_R(\zeta)\nabla_\zeta\sigma(\zeta)^T$, $G_\varepsilon=G_\varepsilon(\zeta)\triangleq\nabla_\zeta\varepsilon(\zeta)G(\zeta)\nabla_\zeta\varepsilon(\zeta)^T$, and $O(\zeta)\triangleq\frac{1}{2}\nabla_\zeta\varepsilon(\zeta)G_R\nabla_\zeta\sigma^T(\zeta)W+\frac{1}{4}G_\varepsilon-W^T\nabla_\zeta\sigma(\zeta)\varepsilon_\theta(\zeta)-\nabla_\zeta\varepsilon(\zeta)F_\theta(\zeta,\theta)-\nabla_\zeta\varepsilon(\zeta)\varepsilon_\theta(\zeta)-\nabla_\zeta\varepsilon(\zeta)F_1(\zeta)$. Furthermore, $\rho:\mathbb{R}^{2n}\times\mathbb{R}^{L}\times\mathbb{R}^{(p+1)\times n}\times\mathbb{R}^{L\times L}\to\mathbb{R}$ is defined as $\rho\triangleq\rho(\zeta_i,\hat{\theta},\hat{W}_a,\Gamma)=1+\nu\omega(\zeta,\hat{W}_a,\hat{\theta})^T\Gamma\omega(\zeta,\hat{W}_a,\hat{\theta})$, where $\Gamma\in\mathbb{R}^{L\times L}$ is a subsequently defined user-initialized learning gain and $\nu\in\mathbb{R}_{>0}$ is a positive normalization constant.

A. Sparse BE Extrapolation

At each time instant $t\in\mathbb{R}_{\geq 0}$, the approximated BE in (13) and policy in (9) are evaluated using the current system state, critic weight estimate, and actor weight estimate to get the instantaneous BE and control policy, which are denoted by $\hat{\delta}\triangleq\hat{\delta}(\zeta,\hat{\theta},\hat{W}_c,\hat{W}_a)$ and $\hat{\mu}\triangleq\hat{\mu}(\zeta,\hat{W}_a)$, respectively. However, using only the on-trajectory BE and control policy requires the traditional PE condition to be satisfied to prove exponential convergence.

Motivated to increase computational efficiency and to provide simulation of experience, local BE extrapolation has been performed in [18] and [19] around unexplored areas of the state space by utilizing more efficient computational capabilities compared with previous methods. Similarly, SNNs improve computational efficiency and use segmentation to extrapolate the BE. This allows the BE to be approximated across a larger, combined region of the state space. Therefore, leveraging the increased computational efficiency of SNNs and segmentation to extrapolate the BE, the BE can be approximated across the entire operating region of the state space without the computational burden of nonsparse methods.

To facilitate the sparse BE extrapolation, let the operating domain Ω be partitioned into $S\in\mathbb{N}$ segments such that $\mathbb{S}\triangleq\{j\in\mathbb{N}|j\leq S\}$ defines the set of segments in the operating domain as $\Omega=\bigcup_{j=1}^{S}\Omega_j$. To simulate PE and extrapolate BE over off-policy trajectories, the segments $\{\zeta_i:\zeta_i\in\Omega_j\}_{i=1}^{N_j}$ are selected,[3] where $N_j\in\mathbb{N}$ denotes the number of extrapolated states in each segment Ω_j. Each segment is assigned a certain number of off-policy trajectories.[4] Using the extrapolated trajectories $\zeta_i\in\Omega_j$ for $j\in S$, the BE in (13) is evaluated such that $\hat{\delta}_i\triangleq\hat{\delta}(\zeta_i,\hat{\theta},\hat{W}_c,\hat{W}_a)$. For a given $j\in S$, the tuple $(\Sigma_c^j,\Sigma_a^j,\Sigma_\Gamma^j)$ is defined as the extrapolation stacks corresponding to Ω_j such that $\Sigma_c^j\triangleq\frac{1}{N_j}\sum_{i=1}^{N_j}\frac{\omega_i}{\rho_i}\hat{\delta}_i$, $\Sigma_a^j\triangleq\frac{1}{N_j}\sum_{i=1}^{N_j}\frac{G_{\sigma i}^T\hat{W}_a\omega_i^T}{4\rho_i}$, and $\Sigma_\Gamma^j\triangleq\frac{1}{N_j}\sum_{i=1}^{N_j}\frac{\omega_i\omega_i^T}{\rho_i}$, where $\omega_i\triangleq\omega(\zeta_i,\hat{\theta},\hat{W}_a)$, $\rho_i\triangleq\rho(\zeta_i,\hat{\theta},\hat{W}_a,\Gamma)=1+\nu\omega_i^T\Gamma\omega_i$, and Assumption 7 is provided to facilitate the subsequent stability analysis.

Assumption 7: Over each segment $j\in S$, there exist a finite set of trajectories $\{\zeta_i:\zeta_i\in\Omega_j\}_{i=1}^{N_j}$ such that $0<\underline{c}\triangleq\inf_{t\in\mathbb{R}_{\geq 0},\,j\in S}\lambda_{\min}\{\Sigma_\Gamma^j\}$, where $\lambda_{\min}\{\cdot\}$ is the minimum eigenvalue, and the constant $\underline{c}$ is the lower bound of the value of each input–output data pairs' minimum eigenvalues.

Remark 3: BE extrapolation can be performed in parallel if needed (i.e., BE extrapolation across multiple segments can be performed simultaneously). Since SNNs are used to improve computational efficiency, the extrapolation within multiple segments can be performed at once. For certain systems, parallel computing may be more computationally efficient in time and power when compared to methods that use traditional NNs for BE extrapolation across the entire state space. One difference in the developed technique compared to previous results is that the actor and critic update laws take a new form in which switching extrapolation stacks are introduced. The extrapolation stacks, Σ_c^j, Σ_Γ^j, and Σ_a^j correspond to user-defined segments of the state space. Upon entering a new segment of the state space, the extrapolation stacks will recall data previously recorded from when the system was last operating in that segment. This allows the user to use separate analysis tools (e.g., machine learning tools) to select segment properties (e.g., size, spacing, quantity, etc.). Switching extrapolation stacks introduces discontinuities in the Lyapunov function time-derivative, requiring a more nuanced stability analysis with generalized solutions.

V. Actor and Critic Weight Update Laws

Using the instantaneous BE $\hat{\delta}$ and extrapolated BEs $\hat{\delta}_i$, the critic and actor weights are updated according to

$$\dot{\hat{W}}_c=-\eta_{c1}\Gamma\frac{\omega}{\rho}\hat{\delta}-\eta_{c2}\Gamma\Sigma_c^j\tag{15}$$

$$\dot{\Gamma}=\left(\lambda\Gamma-\eta_{c1}\frac{\Gamma\omega\omega^T\Gamma}{\rho^2}-\eta_{c2}\Gamma\Sigma_\Gamma^j\Gamma\right)\mathbf{1}_{\{\underline{\Gamma}\leq\|\Gamma\|\leq\overline{\Gamma}\}}\tag{16}$$

$$\dot{\hat{W}}_a=-\eta_{a1}\left(\hat{W}_a-\hat{W}_c\right)-\eta_{a2}\hat{W}_a+\frac{\eta_{c1}G_\sigma^T\hat{W}_a\omega^T}{4\rho}\hat{W}_c+\eta_{c2}\Sigma_a^j\hat{W}_c\tag{17}$$

where $\eta_{c1},\eta_{c2},\eta_{a1},\eta_{a2},\lambda\in\mathbb{R}_{>0}$ are constant learning gains, $\overline{\Gamma}$ and $\underline{\Gamma}\in\mathbb{R}_{>0}$ are upper and lower bound saturation constants, and $\mathbf{1}_{\{\cdot\}}$ denotes the indicator function.[5]

[3] See [14] for guidance on selecting off-policy trajectories for BE extrapolation.

[4] The segments are predetermined by the user and are state dependent (e.g., in [22], the states: altitude, angle of attack, and Mach number determine segment activation).

[5] Using the indicator function in (16) ensures that $\underline{\Gamma}\leq\|\Gamma(t)\|\leq\overline{\Gamma}$ for all $t\in\mathbb{R}_{>0}$. The indicator function in (16) can be removed with minor changes and additional assumptions [19].

VI. Stability Analysis

To facilitate the stability analysis, let $\tilde{\theta} \triangleq \theta - \hat{\theta}$, and $Z \in \mathbb{R}^{n(3+p)+2\,L}$ denote a concatenated state $Z \triangleq [e^T, \tilde{W}_c^T, \tilde{W}_a^T, \tilde{x}^T, \text{vec}(\tilde{\theta})^T]^T$. From Property 1, $\overline{Q}$ is PSD, therefore, $V^*(\zeta)$ is also PSD. Hence, V^* is not a valid Lyapunov function. The result in [11] can be used to show that a nonautonomous form of V^*, denoted as $V_{na}^* : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ and defined as $V_{na}^*(e,t) \triangleq V^*(\zeta)$, is PD and decrescent. Furthermore, $V_{na}^*(0,t) = 0\,\forall t \in \mathbb{R}_{\geq 0}$ and there exist class $\mathcal{K}_\infty$ functions $\underline{v}, \overline{v} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that bound $\underline{v}(\|e\|) \leq V^*(e,t) \leq \overline{v}(\|e\|)\,\forall e \in \mathbb{R}^n, t \in \mathbb{R}_{\geq 0}$. Hence, $V_{na}^*(e,t)$ is a valid Lyapunov function candidate. Let $V_L : \mathbb{R}^{n(3+p)+2\,L} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ be a candidate Lyapunov function defined as

$$V_L(Z,t) \triangleq V_{na}^*(e,t) + \frac{1}{2}\tilde{W}_c^T \Gamma(t)^{-1} \tilde{W}_c + \frac{1}{2}\tilde{W}_a^T \tilde{W}_a + \frac{1}{2}\tilde{x}^T \tilde{x} + \frac{1}{2}\text{tr}\left(\tilde{\theta}^T \Gamma_\theta^{-1} \tilde{\theta}\right). \tag{18}$$

Using the properties of $V_{na}^*(e,t)$ and [28, Lemma 4.3], then (18) be bounded as $\alpha_1(\|Z\|) \leq V_L(Z,t) \leq \alpha_2(\|Z\|)$ for class $\mathcal{K}$ functions $\alpha_1, \alpha_2 : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Using (16), the normalized regressors $\frac{\omega}{\rho}$ and $\frac{\omega_i}{\rho_i}$ can be bounded as $\sup_{\zeta \in \Omega} \|\frac{\omega}{\rho}\| \leq \frac{1}{2\sqrt{\nu \underline{\Gamma}}}$ and $\sup_{\zeta_i \in \Omega_j, j \in \mathbb{S}} \|\frac{\omega_i}{\rho_i}\| \leq \frac{1}{2\sqrt{\nu \underline{\Gamma}}}$. The matrices G_R and G_σ can be bounded as $\sup_{\zeta \in \Omega} \|G_R\| \leq \lambda_{\max}\{R^{-1}\}\overline{G}^2 \triangleq \overline{G_R}$ and $\sup_{\zeta \in \Omega} \|G_\sigma\| \leq (\overline{\nabla_\zeta \sigma G})^2 \lambda_{\max}\{R^{-1}\} \triangleq \overline{G_\sigma}$, respectively, where $\lambda_{\max}\{\cdot\}$ denotes the maximum eigenvalue.

Theorem 1: Given the dynamics in (1), Assumptions 1–7, and the sufficient conditions

$$\eta_{a1} + \eta_{a2} \geq \frac{1}{\sqrt{\nu \underline{\Gamma}}}\left(\eta_{c1} + \eta_{c2}\right)\overline{WG_\sigma} \tag{19}$$

$$\underline{c} \geq 3\frac{\eta_{a1}}{\eta_{c2}} + \frac{3\left(\left(\eta_{c1} + \eta_{c2}\right)\overline{WG_\sigma}\right)^2}{16\eta_{c2}\nu\underline{\Gamma}\left(\eta_{a1} + \eta_{a2}\right)} + \frac{9\left(\left(\eta_{c1} + \eta_{c2}\right)\overline{W\nabla_\zeta \sigma \phi}\right)^2}{8\eta_{c2}k_\theta \nu \underline{\Gamma} \lambda_{\min}\left\{\sum_{j=1}^M \phi_j \phi_j^T\right\}} \tag{20}$$

$$\nu_l^{-1}(l) < \alpha_2^{-1}\left(\alpha_1(r)\right) \tag{21}$$

where l and r are positive constants, then the system state ζ, weight estimation errors $\tilde{W}_c$ and $\tilde{W}_a$, state estimation error $\tilde{x}$, output-layer weight matrix error $\tilde{\theta}$, and control policy $\hat{\mu}$ are uniformly ultimately bounded (UUB).

Proof: Let $r \in \mathbb{R}_{>0}$ be the radius of a compact ball $\chi \subset \mathbb{R}^{n(3+p)+2\,L}$ centered at the origin. Let $Z(t)$ for $t \in \mathbb{R}_{\geq 0}$ be a Filippov solution to the differential inclusion $\dot{Z} \in K[h](Z)$, where $K[\cdot]$ is defined in [29] and $h : \mathbb{R}^{n(4+p)+2L+L^2} \to \mathbb{R}^{n(4+p)+2L+L^2}$ is defined as $h \triangleq [\dot{\zeta}^T, \dot{\tilde{W}}_c^T, \dot{\tilde{W}}_a^T, \text{vec}(\dot{\Gamma}^{-1})^T, \dot{\tilde{x}}^T, \text{vec}(\dot{\tilde{\theta}})^T]^T$. Due to the discontinuity in the update laws in (15)–(17), the time derivative of (18) exists almost everywhere (a.e., i.e., for almost all $t \in \mathbb{R}_{\geq 0}$) and $\dot{V}_L(Z,t) \overset{a.e.}{\in} \dot{\tilde{V}}_L(Z,t)$, where $\dot{\tilde{V}}_L$ is the generalized time-derivative of (18) along the Filippov trajectories of $\dot{Z} = h(Z)$ [30]. Using the class of dynamics in (1); the calculus of $K[\cdot]$ from [30]; $\dot{V}^*(\zeta) = \nabla_\zeta V^*(\zeta)(F(\zeta) + G(\zeta)\mu)$; substituting (10), (11), and (14)–(17); using Young's Inequality and nonlinear damping; Assumption 6 and 7; and substituting the sufficient conditions in (19) and (20) yields $\dot{V}_L \overset{a.e.}{\leq} -\nu_l(\|Z\|)$, $\forall \nu_l^{-1}(l) \leq \|Z\| \leq \alpha_2^{-1}(\alpha_1(r))$, where $\nu_l(\|Z\|) \triangleq \frac{q(\|e\|)}{2} + \frac{\eta_{c2}\underline{c}}{12}\|\tilde{W}_c\|^2 + \frac{\eta_{a1}+\eta_{a2}}{16}\|\tilde{W}_a\|^2 + \frac{k}{4}\|\tilde{x}\|^2 + \frac{k_\theta \lambda_{\min}\left\{\sum_{j=1}^M \phi_j \phi_j^T\right\}}{6}\|\text{vec}(\tilde{\theta})\|^2$. Since (18) is a common Lyapunov function across each segment $j \in \mathbb{S}$, [28, Th. 4.18] can be invoked to conclude that Z is UUB such that $\limsup_{t \to \infty} \|Z\| \leq \alpha_1^{-1}(\alpha_2(\nu_l^{-1}(l)))$ and $\hat{\mu}$ converges to a neighborhood around the optimal policy μ^*. Furthermore, since $Z \in \mathcal{L}_\infty$, it follows that $e, \tilde{W}_c, \tilde{W}_a, \tilde{x}, \tilde{\theta} \in \mathcal{L}_\infty$, hence, $x, \hat{W}_c, \hat{W}_a, \hat{\theta} \in \mathcal{L}_\infty$ and $u \in \mathcal{L}_\infty$.

The result in [28, Th. 4.18] can be invoked to show that every trajectory $Z(t)$ that satisfies the initial condition $\|Z(0)\| \leq \alpha_2^{-1}(\alpha_1(r))$ is bounded for all $t \in \mathbb{R}_{\geq 0}$. That is, $Z \in \chi\,\forall t \in \mathbb{R}_{\geq 0}$. Since $Z \in \chi$ it follows that the individual states of Z lie on compact sets. Furthermore, since $x_d \leq \overline{x_d}$, then $\zeta \in \Omega$ and $x \in \mathcal{C}$, where Ω is the compact set that facilitates value function approximation, and $\mathcal{C}$ is the compact set that facilitates NN-based system identification.

Remark 4: Theorem 1 expands on previous results by examining a different candidate Lyapunov function (18) that contains system identification terms $\tilde{x}$ and $\tilde{\theta}$, considers a nonautonomous form of the value function, considers different weight update laws (15)–(17), results in different sufficient conditions due to the system uncertainty (19)–(21), and provides convergence guarantees on the system identification terms, whereas [23, Th. 1] and [24, Th. 1] do not.

Remark 5: For insight into satisfying the conditions in (19)–(21), see [15].

VII. Simulation

In this section, the developed technique is applied to a linear quadratic tracking problem, which has a cost function $r(\zeta, \mu) = e^T Q e + \mu^T R \mu$. The Euler Lagrange system

$$u = a_1 \ddot{y} + a_2 \dot{y}, \tag{22}$$

where $y, \dot{y}, \ddot{y} \in \mathbb{R}^2$, is leveraged in this simulation since the analytical solution to the HJB equation in (5) can be calculated. The concatenated state x is defined as $x \triangleq [y^T, \dot{y}^T]^T$. The matrix $a_1 \in \mathbb{R}^{2\times 2}$ is defined as $a_1 \triangleq \left[\begin{smallmatrix} 1 & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{5} \end{smallmatrix}\right]$, and $a_2 \in \mathbb{R}^{2\times 2}$ is defined as $a_2 \triangleq \left[\begin{smallmatrix} 1 & -1 \\ \frac{1}{2} & \frac{1}{2} \end{smallmatrix}\right]$. The objective is to determine a policy μ online to ensure that the concatenated state x tracks the desired trajectory $x_d = [\cos(0.5t),\, 2\cos(t),\, -0.5\sin(0.5t),\, -2\sin(t)]^T$ while minimizing the cost function, which is selected as $r(\zeta, \mu) = e^T Q e + \mu^T R \mu$.

The dynamics in (22) can be rewritten in the form $A \triangleq \left[\begin{smallmatrix} \mathbf{0}_{2\times 2} & I_{2\times 2} \\ \mathbf{0}_{2\times 2} & a_1^{-1} a_2 \end{smallmatrix}\right]$, $f(x) = Ax$, $g(x) = [\mathbf{0}_{2\times 2}^T, (a_1^{-1})^T]^T$, $g^+(x_d) = [\mathbf{0}_{2\times 2}, a_1]$, $h_d(x_d) = [x_{d3}, x_{d4}, -0.25x_{d1}, -x_{d2}]^T$, which can be expressed as in (1), where $\zeta \triangleq [e^T, x_d^T]^T$.

To achieve the desired objective, the developed value function approximation method is used. The basis selected for value function approximation is a polynomial basis function with 23 elements given by $\sigma(\zeta) = \frac{1}{2}[\zeta_2^2, \zeta_1^2, \zeta_1\zeta_3, \zeta_1\zeta_4, \zeta_2\zeta_3, \zeta_2\zeta_4, \zeta_1^2\zeta_2^2, \zeta_1^2\zeta_5^2, \zeta_1^2\zeta_6^2, \zeta_1^2\zeta_7^2, \zeta_1^2\zeta_8^2, \zeta_2^2\zeta_5^2, \zeta_2^2\zeta_6^2, \zeta_2^2\zeta_7^2, \zeta_2^2\zeta_8^2, \zeta_3^2\zeta_5^2, \zeta_3^2\zeta_6^2, \zeta_3^2\zeta_7^2, \zeta_3^2\zeta_8^2, \zeta_4^2\zeta_5^2, \zeta_4^2\zeta_6^2, \zeta_4^2\zeta_7^2, \zeta_4^2\zeta_8^2]^T$, where, generally, ζ_i refers to the ith entry of ζ. The drift dynamics are unknown, but are approximated using the developed system identification method. The unknown drift dynamics are approximated with the linear basis $\phi(x) = [x_1, x_2, x_3, x_4]^T$. Five separate simulation cases were performed that use identical gains, initial conditions, and basis function for system identification and on-trajectory BE. The differences between the simulations is that BE extrapolation is performed with different NNs, which have varying sparsity, which are specified in Table I. Case 1 (i.e., [15]) uses traditional BE extrapolation, Cases 2–4 use sparse BE extrapolation, and Case 5 uses sparse BE extrapolation and switches the extrapolation stacks depending on the system state. Each simulation case was executed in Simulink using a discrete-time differential equation solver at a frequency of 100 Hz on the same machine. Each simulation case is executed for 120 seconds of simulated time.

TABLE I
SIMULATION CASE PARAMETERS

Simulation case	Nodes eliminated	Extrapolation segments (Ω_j)
Case 1 (see [15])	N/A	1
Case 2	$\zeta_2^2\zeta_8^2 = 0$	1
Case 3	$\zeta_2^2\zeta_6^2, \zeta_2^2\zeta_7^2, = 0$	1
Case 4	$\zeta_2^2\zeta_6^2, \zeta_2^2\zeta_7^2,$ $\zeta_2^2\zeta_8^2 = 0$	1
Case 5	$\zeta_2^2\zeta_6^2, \zeta_2^2\zeta_7^2,$ $\zeta_2^2\zeta_8^2 = 0$	2

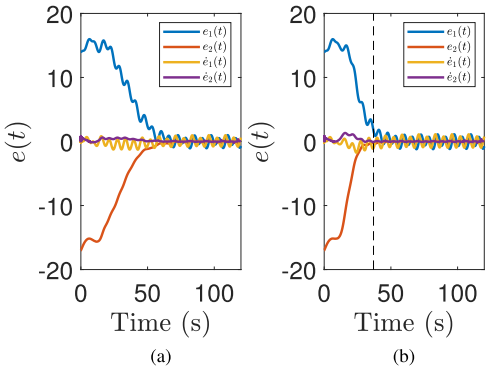

Fig. 1. (a) Errors e_1, e_2, $\dot{e}_1$, $\dot{e}_2$ in Case 1. (b) Errors e_1, e_2, $\dot{e}_1$, $\dot{e}_2$ in Case 5. The vertical line represents the time at which the system switched BE extrapolation data stacks due to (23). Case 5 has faster convergence than Case 1; however, Case 1 has a smaller steady-state error.

For Cases 1–4, BE extrapolation is performed over the domain $\Omega_1 \triangleq \{\zeta \in \mathbb{R}^8 : -5 \leq \zeta_i \leq 5\, \forall i \in [1,8]\}$ with $N_1 = 64$ extrapolated trajectories. However, for Case 5, two segments are defined: $\Omega_1 \subset \mathbb{R}^8$ and $\Omega_2 \subset \mathbb{R}^8$, where $\Omega_1 \triangleq \{\zeta \in \mathbb{R}^8 : -5 \leq \zeta_i \leq 5\, \forall i \in [1,8]\}$ with $N_1 = 64$ extrapolated trajectories. The second segment is defined such that $\Omega_2 \triangleq \Omega_1$ with $N_2 = 32$ extrapolated trajectories. In Case 5, both Ω_1 and Ω_2 use the same basis, which is defined in the Table I. The active BE extrapolation stack Ω_j is selected via the policy

$$j \triangleq \begin{cases} 1 & \|e\| > 2 \\ 2 & \|e\| \leq 2 \end{cases}. \tag{23}$$

For each simulation case the cost parameters are $Q = [1000, 1000, 0.2, 0.2]^T \cdot I_{4\times 4}$ and $R = 10 \cdot I_{2\times 2}$, the gains are $\eta_{c1} = 0.012$, $\eta_{c2} = 0.001$, $\eta_{a1} = 0.005$, $\eta_{a2} = 0.005$, $\lambda = 0.075$, $\nu = 0.005$, $k_\theta = 100$, $\Gamma_\theta = 0.02 \times I_{4\times 4}$. $\overline{\Gamma} = 10^3$, $\underline{\Gamma} = 10$, and the initial conditions are $\hat{W}_c(0) = 10 \cdot \mathbf{1}_{23\times 1}$, $\hat{W}_a(0) = 6 \cdot \mathbf{1}_{23\times 1}$, $\Gamma(0) = 200 \cdot I_{23\times 23}$, $\hat{\theta}(0) = \mathbf{0}_{4\times 4}$,[6] $\hat{x}(0) = \mathbf{0}_{4\times 1}$, and $x(0) = [15, -15, 0, 0]^T$.

The tracking errors for Cases 1 and 5 are compared in Fig. 1. The purpose of Fig. 1(a) is to show the performance of an existing, nonsparse result. To contrast Fig. 1(a), (b) presents the performance of the most sparse simulation case. These figures, when paired with Table II, exhibit the benefits and drawbacks to using the techniques described in Cases 1 and 5. Case 1 may take slightly longer to converge, but it has a lower steady-state error, which indicates that the use of additional BE extrapolation data results in improved value function approximation. The convergence rate of Case 5 (SNN with switched BE extrapolation stack) is better than that of Case 1 (standard BE extrapolation from [15]).

For this class of dynamics and cost function, the solution to the HJB equation can be determined analytically by solving the Algebraic Riccati Equation offline. Hence, the approximate value function $\hat{V}(\zeta, \hat{W}_c)$ can be compared to the optimal value function $V^*(\zeta)$. This comparison is shown in Fig. 2.

[6]From (22), $\theta = A^T$.

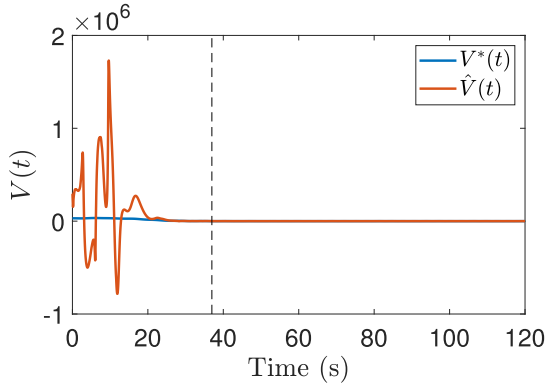

Fig. 2. Comparison of the optimal value function $V^*(\zeta)$ to the approximated optimal value function $\hat{V}(\zeta, \hat{W}_c)$ for Case 5. The vertical line represents the time at which the system switched BE extrapolation data stacks due to (23).

To examine the effects of increased sparse BE extrapolation, data were collected from each simulation case to facilitate a quantitative comparison. The median computation time,[7] integral of error (i.e., $\int_0^{120} \|e(\tau)\| d\tau$), 5% rise time, and root mean squared (RMS) error for each case are shown in Table II.

The computation time is the amount of real-world time it takes to run 120 s of simulation time. The computation times were measured by a built-in function in MATLAB. There is a clear trend in the computation times of each case. Case 1 has the highest computation time because it performs the highest amount of BE extrapolation. Case 5 has the shortest computation time. By combining the switched extrapolation stacks and sparse BE, the computation time is reduced by 85.6% compared to the nonsparse BE extrapolation method in [15]. As the BE extrapolation becomes more sparse (from Cases 2 to 4) the computation time significantly decreases. Case 5 uses a switched extrapolation stack with sparse BE extrapolation (the same SNN as Case 4). By decreasing the number of points in each extrapolation stack, the computation time is decreased. Additionally, by performing sparse BE extrapolation on the smaller extrapolated stacks, the computation time is further reduced. Hence, as the BE extrapolation becomes more sparse and more switching extrapolations stacks are used, the computation time significantly decreases.

The 5% rise time is the amount of time it takes for the error to reach 5% of its initial value (i.e., $\|e(t)\| \leq 0.05 \cdot \|e(0)\|$).[8] While the rise time is the worst for Case 1, there is no clear explanation. Increasing sparsity seems to have a minor effect on rise time. The RMS steady-state error is lowest for Case 1. This is likely due to the fact that Case 1 uses the most data and computations; however, this has a negative effect on computation time. Cases 2–5 have similar RMS steady-state error; we can conclude that the increasing amount of sparsity has little impact on the RMS steady-state error.

[7]The computation time varied between multiple instances of the same simulation case. To better measure the computation time of each case, the median computation time was determined by running each case 10 times. The median was selected to eliminate the effect of outliers because the computation times are skewed toward higher computation times. For each case, the integral of error, 5% rise time, and RMS steady-state error of each case were identical between multiple simulation trials.

[8]The 5% rise time was used as a performance metric to better compare the convergence of the test cases. If a 10% rise time were used, the performance difference between Case 1 and Cases 2–5 would not be as pronounced.

TABLE II
SIMULATION RESULTS FOR SIMULATION CASES 1–5.

Controller	Case 1 (see [15])	Case 2	Case 3	Case 4	Case 5
Median computation time (s)	66.09	13.66	13.56	13.49	9.55
Integral of error	802.88	593.97	580.36	595.13	594.91
5% Rise time	55.96	37.88	37.57	37.90	37.90
RMS steady-state error	0.80	0.92	0.96	0.93	0.93

VIII. CONCLUSION

An online approximate optimal tracking controller is developed for an initially unknown dynamical system. The value function is approximated by performing sparse BE extrapolation over segments of the state space. Motivated by reducing the computational complexity of BE extrapolation, sparse BE extrapolation is performed over user-defined subsets of the state space. UUB tracking of each agent's state to the neighborhood of the desired state and convergence of the control policy to the neighborhood of the optimal policy are proven using a Lyapunov-like stability analysis in the presence of discontinuities. A simulation study shows that this method enables the system to track a desired trajectory while approximating the value function and optimal control policy to their optimal values. Furthermore, the simulation shows that sparse, switched BE extrapolation reduces computation time by 85.6% when compared to the method in [15]. Future efforts will investigate if BE extrapolation data and system identification input–output data can be exploited together to reduce the overall amount of data used by this control technique.

ACKNOWLEDGMENT

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Aurora Flight Sciences, Draper, the Johns Hopkins Applied Physics Laboratory, or the sponsoring agencies.

REFERENCES

[1] J. Si, A. Barto, W. Powell, and D. Wunsch, Eds., *Handbook of Learning and Approximate Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2004.
[2] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis, *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*. London, U.K.: The Institution of Engineering and Technology, 2013.
[3] R. Kamalapurkar, P. S. Walters, J. A. Rosenfeld, and W. E. Dixon, *Reinforcement Learning for Optimal Feedback Control: A Lyapunov-Based Approach*. Berlin, Germany: Springer, 2018.
[4] F. L. Lewis and D. Liu, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, vol. 17. Hoboken, NJ, USA: Wiley, 2013.
[5] D. Liberzon, *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton, NJ, USA: Princeton Univ. Press, 2012.
[6] A. Kanellopoulos and K. G. Vamvoudakis, "A moving target defense control framework for cyber-physical systems," *IEEE Trans. Autom. Control*, vol. 65, no. 3, pp. 1029–1043, Mar. 2020.
[7] A. Chakrabarty, D. K. Jha, G. T. Buzzard, Y. Wang, and K. G. Vamvoudakis, "Safe approximate dynamic programming via kernelized lipschitz estimation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 405–419, Jan. 2021.
[8] B. Pang and Z.-P. Jiang, "Adaptive optimal control of linear periodic systems: An off-policy value iteration approach," *IEEE Trans. Autom. Control*, vol. 66, no. 2, pp. 888–894, Feb. 2021.
[9] W. Gao, M. Mynuddin, D. C. Wunsch, and Z.-P. Jiang, "Reinforcement learning-based cooperative optimal output regulation via distributed adaptive internal model," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–12, 2021.
[10] B. Pang, T. Bian, and Z.-P. Jiang, "Robust policy iteration for continuous-time linear quadratic regulation," *IEEE Trans. Autom. Control*, vol. 67, no. 1, pp. 504–511, Jan. 2022.
[11] R. Kamalapurkar, H. Dinh, S. Bhasin, and W. E. Dixon, "Approximate optimal trajectory tracking for continuous-time nonlinear systems," *Automatica*, vol. 51, pp. 40–48, Jan. 2015.
[12] G. Chowdhary, T. Yucelen, M. Mühlegg, and E. N. Johnson, "Concurrent learning adaptive control of linear systems with exponentially convergent bounds," *Int. J. Adaptive Control Signal Process.*, vol. 27, no. 4, pp. 280–301, 2013.
[13] G. Chowdhary, M. Mühlegg, and E. Johnson, "Exponential parameter and tracking error convergence guarantees for adaptive controllers without persistency of excitation," *Int. J. Control*, vol. 87, no. 8, pp. 1583–1603, 2014.
[14] R. Kamalapurkar, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for approximate optimal regulation," *Automatica*, vol. 64, pp. 94–104, 2016.
[15] R. Kamalapurkar, L. Andrews, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for infinite-horizon approximate optimal tracking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 753–758, Mar. 2017.
[16] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *J. Int. Robot. Syst.*, vol. 86, no. 2, pp. 153–173, 2017.
[17] M. Janner et al., "When to trust your model: Model-based policy optimization," in *Proc. Adv. Neural Inf. Process.*, vol. 32, 2019.
[18] R. Kamalapurkar, J. Rosenfeld, and W. E. Dixon, "Efficient model-based reinforcement learning for approximate online optimal control," *Automatica*, vol. 74, pp. 247–258, Dec. 2016.
[19] P. Deptula, J. A. Rosenfeld, R. Kamalapurkar, and W. E. Dixon, "Approximate dynamic programming: Combining regional and local state following approximations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2154–2166, Jun. 2018.
[20] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.
[21] S. A. Nivison and P. Khargonekar, "Improving long-term learning of model reference adaptive controllers for flight applications: A sparse neural network approach," in *Proc. AIAA Guid. Navigat. Control Conf.*, 2017, Art. no. 1249.
[22] S. A. Nivison and P. Khargonekar, "A sparse neural network approach to model reference adaptive control with hypersonic flight applications," in *Proc. AIAA Guid. Navigat. Control Conf.*, 2018, Art. no. 0842.
[23] M. L. Greene, P. Deptula, S. Nivison, and W. E. Dixon, "Reinforcement learning with sparse Bellman error extrapolation for infinite-horizon approximate optimal regulation," in *Proc. IEEE 58th Conf. Decis. Control*, 2019, pp. 1959–1964.
[24] M. L. Greene, P. Deptula, S. Nivison, and W. E. Dixon, "Sparse learning-based approximate dynamic programming with barrier constraints," *IEEE Control Syst. Lett.*, vol. 4, no. 3, pp. 743–748, Jul. 2020.
[25] P. Deptula, Z. Bell, E. Doucette, W. J. Curtis, and W. E. Dixon, "Data-based reinforcement learning approximate optimal control for an uncertain nonlinear system with control effectiveness faults," *Automatica*, vol. 116, pp. 1–10, Jun. 2020.
[26] W. Rudin, *Principles of Mathematical Analysis*. New York, NY, USA: McGraw-Hill, 1976.
[27] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, vol. 15, D. A. White and D. A. Sorge, Eds., New York, NY, USA: Nostrand, 1992, pp. 493–525.
[28] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.
[29] A. F. Filippov, "Differential equations with discontinuous right-hand side," in *American Mathematical Society Translations - Series 2 (Fifteen Papers on Differential Equations 42)*. Providence, RI, USA: Amer. Math. Soc., 1964, pp. 199–231.
[30] B. E. Paden and S. S. Sastry, "A calculus for computing Filippov's differential inclusion with application to the variable structure control of robot manipulators," *IEEE Trans. Circuits Syst.*, vol. CS-34, no. 1, pp. 73–82, Jan. 1987.