NONSMOOTH DATA-BASED REINFORCEMENT LEARNING FOR ONLINE APPROXIMATE OPTIMAL CONTROL

By

MAX LEWIS GREENE

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2022

To my mother, Lorraine, and my father, Terry, for their endless love and support

ACKNOWLEDGMENTS

I thank my advisor, Dr. Warren E. Dixon, for his guidance, insight, and patience. His mentorship has allowed me to grow professionally and personally. I also extend my gratitude to my committee members, Dr. Prabir Barooah, Dr. William Hager, and Dr. Matthew Hale for their recommendations and insights. I am grateful to my colleagues at the University of Florida, Air Force Research Laboratory, and NASA Armstrong Flight Research Center, who have challenged and encouraged me throughout my career. Finally, thank you to my friends and family for your relentless support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADP	Approximate Dynamic Programming
a.e.	Almost Everywhere
BE	Bellman Error
BF	Barrier Function
CAN	Controller Area Network
CL	Concurrent Learning
DNN	Deep Neural Network
EMK	Exact Model Knowledge
FES	Functional Electrical Stimulation
HJB	Hamilton-Jacobi-Bellman (Equation)
LP	Linearly Parameterizable
NN	Neural Network
PD	Positive Definite
PE	Persistence of Excitation
PSD	Positive Semi-Definite
RL	Reinforcement Learning
R-MBRL	Regional Model-based Reinforcement Learning
RMS	Root Mean Squared
SNN	Sparse Neural Network
UUB	Uniformly Ultimately Bounded

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

NONSMOOTH DATA-BASED REINFORCEMENT LEARNING FOR ONLINE APPROXIMATE OPTIMAL CONTROL

By

Max Lewis Greene

May 2022

Chair: Warren E. Dixon
Major: Mechanical Engineering

Autonomous systems are often constrained by time-critical mission constraints and limited power. Such constraints motivate optimality in mission execution. Reinforcement learning (RL) has become a tool to facilitate learning of a desired optimal control policies online, which achieve a desired objective. Approximate dynamic programming (ADP) is a RL-based techniques that generates a forward-in-time approximation of the optimal optimal value function (and in-turn the control policy) for dynamical systems with continuous state and action spaces. Developments in regional model-based RL (R-MBRL) facilitate improved online approximation of the value function. R-MBRL approximates the value function over a compact set of the state space and facilitates learning by approximating and evaluating the optimal value function at multiple points on this compact set. This dissertation investigates numerous modifications to R-MBRL ADP to improve computational efficiency, for application to a broader class of dynamical systems, and to incorporate different function approximation techniques. These modifications introduce discontinuities into the otherwise smooth signals, which are analyzed via Lyapunov-based techniques.

Chapter 3 presents a technique to reduce the computational expense of performing R-MBRL across an arbitrarily large number of points in the state space. Without modification, existing R-MBRL algorithms evaluate the quality of the value function approximation at many user-defined points on the state space using a conventional

neural network (NN). The method presented in Chapter 3 improves on the existing techniques by segmenting the state space and using sparse neural networks (SNNs) to facilitate learning. By segmenting the state space, the cognitive agent can switch between different subsets of the state space over which to evaluate the optimal policy. Furthermore, using a SNN reduces the overall number of operations needed to evaluate the optimal policy. Combined, these two modifications improve the computational efficiency of R-MBRL.

Chapter 4 builds upon the result in Chapter 3 by modifying the state space with barrier function (BF) transformations. The BFs, which are described by invariant sets, transform the dynamics and cost of the state space. The system is penalized significantly more as the state moves closer to the boundary of the user-defined invariant set(s). This technique develops safety certificates for R-MBRL, i.e., formal guarantees that the state will not leave the user-defined invariant set.

Chapter 5 investigates the application of R-MBRL ADP to a finite family of switched systems with a countably infinite number of arbitrary switches. This chapter outlines additional constraints that must be imposed on the system to guarantee stability of the overall switching sequence. In doing so, a minimum dwell-time condition is developed. The dwell-time condition is a conservative condition that dictates the minimum time that one subsystem must be active before switching to another subsystem. The technique developed in Chapter 5 enables the application of ADP onto a larger class of systems, such as path planning in static, but unknown, environments, and functional electrical stimulation (FES) control.

Existing results show that, despite having no knowledge (or uncertainty) of a system's dynamics *a priori*, online system identifiers can be used to approximate the model of a system, which can be used in conjunction with ADP to approximate the optimal control policy. Chapter 6 leverages an online deep neural network (DNN)-based system identifier to simultaneously approximate the system dynamics and optimal

control policy. The DNN identifier updates in two timescales. The output-layer weights update in real-time, and the inner-layer features update discretely via batch updates. While existing results propose a multi-timescale DNN system identifier, convergence of the output-layer weights is not guaranteed. This chapter modifies the output-layer weight update policy with a concurrent learning (CL)-based term, which enables convergence of the overall ADP algorithm, i.e., approximately optimal trajectory tracking is achieved.

CHAPTER 1
INTRODUCTION

1.1 Background

RL is a technique that facilitates learning in many computational problems, including robotics, video game playing, supply chain management, and automatic control. Generally, RL-based techniques interact with an environment, sense the state of the system, and perform an action that seeks to minimize or maximize a cost function [1]. The cost depends on the environment, state, and previous action(s) of the system. RL, unlike other supervised learning methods, can evaluate the performance of a particular action without a teacher. This makes RL well-posed to determine policies in which examples, or models, of desired behavior do not exist. Leveraging function approximation architectures, RL-based techniques have been developed to approximately solve optimal control problems for continuous-time and discrete-time deterministic systems with finite state-spaces and stationary environments by computing the optimal control policy based on an approximation of the optimal cost-to-go function, i.e., the value function [2–13].

Optimal control problems can be solved via the Hamilton-Jacobi-Bellman (HJB) equation; the solution is the optimal value function, which is used to determine the optimal control policy [14]. However, the HJB equation is a nonlinear partial differential equation that lacks a general solution. To combat the difficulty of solving the HJB for the optimal value function, ADP is used to approximate the value function online [15] and [16]. If the value function is successfully approximated, then a stabilizing optimal control policy can be determined. ADP uses parametric methods, such as NNs, to approximate the solution of the HJB, i.e., the value function.

In RL-based online approximate optimal control, the HJB equation, along with an estimate of the state derivative [7] and [10], or an integral form of the HJB [17] and [18], is used to measure the quality of the value function approximation evaluated at each

visited state along the system trajectory. This measurement is called the Bellman error (BE).

In online RL-based techniques, approximations of the uncertain parameters in the value function are updated using the BE as a performance metric. Hence, the approximated value function parameters are updated based on the evaluation of the BE along the system trajectory. Online RL-based techniques can be implemented in either model-based or model-free form. Generally speaking, both approaches have their respective advantages and disadvantages. Model-free approaches learn optimal actions without requiring knowledge of the system [19]. Model-based RL approaches improve data efficiency by observing that if the system dynamics are known, then the state derivative, and hence the BE, can be evaluated at any desired point in the state-space [19].

Methods that seek online solutions to optimal control problems are comparable to adaptive control (see [3, 8, 10, 12, 20, 21] and the references therein), in which the approximations of the uncertain parameters in the plant model are updated using the tracking error as a performance metric. Similarly, in ADP the BE is used as a performance metric. Parameter convergence has long been a focus of research in adaptive control.

Least-squares and gradient-based update laws are used in RL-based techniques to solve optimal control problems online [19] and [22]. Such update laws generally require the persistence of excitation (PE) condition in the system state to guarantee convergence (i.e., value function approximation), which cannot be generally verified for nonlinear systems. Hence, a challenge exists that that the update law must be persistently exciting so that the system trajectory sufficiently explores the state-space to sufficiently approximate the optimal value function over the domain of operation. However, excessive exploration of the state space prohibits a cognitive agent from completing its desired task. This challenge, referred to as the exploration versus

exploitation problem, is often addressed in the related literature [5, 8, 10, 23–29] by adding an exploration signal to the control input. However, no analytical methods exist to compute the appropriate exploration signal for a nonlinear dynamical system. Existing ADP literature [19] proposes a method that simultaneously evaluates the BE at on- and off-trajectory points in the state space. This process is called BE extrapolation. R-MBRL uses BE extrapolation on user-defined regions of the state space. BE extrapolation can be used to relax the PE condition and circumvent the need for an exploration signal. However, the benefits of R-MBRL come at a significant computational expense.

Dynamic programming-based methods suffer from the curse of dimensionality (i.e., exponential growth of computational complexity with the increased number of states). This curse is necessitated by the need to explore the state space to obtain a sufficiently accurate approximation over the state space. Parametric methods, such as NNs, approximate functions over a compact set of the state space. Increasing the size of this compact set may require additional neurons in the value function approximation. Furthermore, neurons must be placed with appropriate density to sufficiently approximate the value function. Without knowledge of the system, a large number of neurons must be used to approximate the value function. In the aforementioned BE extrapolation technique, a large number of extrapolation points must be used to facilitate learning over the desired region of the state space. Hence, it is computationally demanding to use a large number of neurons for value function approximation in tandem with BE extrapolation. While R-MBRL has advantages over PE-based ADP methods, there are computational issues that limit its deployment.

SNNs are a tool to facilitate learning in uncertain systems [30–33]. SNNs have been used to reduce computational complexity in NNs by decreasing the number of active neurons. By reducing the number of active neurons, then the number of overall computations is also reduced. SNN-based adaptive controllers have been developed to activate a smaller number of neurons in certain locations of the state space [32]. Making

a NN more sparse, i.e., sparsification, encourages local learning through intelligently segmenting the state space [31]. Sparsification enables local approximation within each segment, which characterizes regions with significantly varying dynamics or unknown uncertainties. Furthermore, SNNs yield computational benefits due to activating fewer active neurons in comparison to traditional NNs. Motivated by the method in [31], the developed method simultaneously uses a segmented state space and SNNs to reduce computational loads due to intelligent switching, segmentation, and sparsification.

A longstanding problem in the development of machine learning-based algorithms is the difficulty to guarantee system performance. Notably, in uncertain environments, BFs have been used to generate safety certificates for control systems [34] and [35]. Such certificates guarantee safety in implementation by proving that they will not leave an invariant set, i.e., they will stay within a region defined by the BF(s). BFs have a natural relationship with Lyapunov-like functions, set invariance, and multi-objective control. BFs have been previously used with ADP [36], but not always in the context of safety certificates. The results in [37] and [38] provide a united framework for solving the optimal control problem online while providing formal performance guarantees. However, these existing results rely on the PE condition to guarantee convergence, which may be undesirable in practice.

At a high level, optimal control solutions guarantee system convergence and provide varying performance (e.g., rise time, overshoot, etc.). Gain scheduling is a method in which different classes of controllers are used to govern a dynamical system in different scenarios (potentially with differing dynamics) [39] and [40]. For example, an aircraft may use different controllers (or control gains) depending on its speed, angle of attack, and altitude. The controller associated with a certain state is active when the aircraft visits that state. It may be impossible to design controllers for unknown or uncertain environments. Or, in the case of gain scheduling, it may be impossible to design a family of controllers for unknown or uncertain environments.

Existing ADP results, such as [19, 22, 41], generate stabilizing control policies in unknown environments; however, these methods lack the ability to switch between different optimal control policies. The inability to switch between different optimal control policies may be due to the inability to account for switching between multiple dynamic models and cost parameters. Previous results such as [42–46] use optimal control methods to minimize cost function(s) of a switched system. These methods use a fixed mode sequence (see [42, 45–47]) or fixed switching instances [44]. In unknown environments, switching may occur arbitrarily – either by the user's design or from environmental effects. The aforementioned methods do not account for an arbitrary switching sequence.

Existing ADP methods have accounted for uncertain dynamics (e.g., [19]) or completely unknown dynamics (e.g., [48]). In [48] an additional NN is used to characterize the unknown model of the dynamic system online, which facilitates approximation of the value function. However, this online system identification method uses a single hidden-layer NN. Recent advancements in adaptive control [49–52] use DNNs instead of single or double hidden-layer NNs to approximate the dynamics online. DNN function approximation methods empirically show improved performance, but these methods often lack performance guarantees. Hence, DNN-based methods may have limited adoption for safety-critical applications. Results in [49] and [50] leverage a multi-timescale deep model reference adaptive controller. The method in [52] uses a multi-timescale DNN to estimate the unknown system dynamics, which facilitates a trajectory tracking objective. Specifically, in [52], the output-layer weights of the DNN are estimated using an unsupervised learning algorithm to provide responsiveness and guaranteed tracking performance with real-time feedback. The inner-layer features of the DNN are trained with collected data sets, which are collected in real-time, to increase performance. The inner-layer features are updated once a large amount of data is collected. The output-layer weights are updated with a gradient-based adaptation policy in real-time. Hence,

simultaneous to real-time execution, input-output data is stored and used to iteratively update the inner-layer features using traditional offline DNN function approximation methods. In ADP, the ideal NN weights of the system identifier must be exactly learned. However, the method in [52] does not guarantee convergence of the weight approximations to their ideal values. Hence, the technique in [52] cannot be trivially extended to ADP.

1.2 Outline of the Dissertation

In Chapter 2, the infinite-horizon optimal control problem and the value function approximation technique are introduced.

Chapter 3 presents a framework for improving the computational efficiency of R-MBRL methods by using SNNs to facilitate BE extrapolation. The BE is used as a performance metric in ADP. BE indirectly measures the quality of the estimation of the value function along the system trajectory. Previous works (e.g., [19] and [48]) show that if the system dynamics are successfully estimated, then the BE can be evaluated at an arbitrary number of points in a system's state space. This process is called BE extrapolation. Results such as [22] and [53] perform BE extrapolation in a neighborhood of the current state, which facilitates value function approximation in that neighborhood. Since a value function approximation is sought over a large region of the state space, BE extrapolation is sometimes performed over large regions of the state space, which is computationally expensive. Increasing the number of basis functions may improve value function approximation; however, this comes at a high computational cost. Together, BE extrapolation and a high-dimension basis functions for value function approximation place a significant computational load on the computing resource.

The contribution of Chapter 3 is to analyze the stability of using a sparse, switched BE term with a NN-based estimator within the existing R-MBRL ADP framework. This chapter extends beyond previous results by considering the optimal tracking problem with completely unknown drift dynamics and by quantifying the benefit of using SNNs in

R-MBRL via simulations. There are two simulations in Chapter 3 . The first simulation study demonstrates the ability to simultaneously estimate the system dynamics and optimal control policy. The second simulation assumes knowledge of the system dynamics to isolate the benefit of using sparse BE extrapolation; numerous simulation cases are studied to quantify the benefit of using sparse BE extrapolation compared existing results.

Chapter 4 leverages the BF transformation developed for the ADP controller in [38] in the development of an R-MBRL ADP controller. The model-free result in [38] evaluates the BE only along the system trajectory, resulting in the typical exploration versus exploitation tradeoff. A contribution of Chapter 4 is the development of a framework for sparse BE extrapolation (motivated by [19, 53, 54]) for off-trajectory learning of the value function, while also adhering to BF constraints (unlike [19, 53, 54]). Specifically, Chapter 4 provides an investigation of sparse BE extrapolation using a state-constraint BF transform. The unique combination of BE extrapolation with the use of a BF raises new questions such as – which states should be transformed? Is there a computational penalty for each state transformation? Should the BF transformation be applied before or after the BE extrapolation (i.e., in which space should the extrapolation be performed)? What are the implications of extrapolation stack updates in the transformed state-space? The subsequent design and Lyapunov-based stability analysis provides the first exploration of such questions in a manner that yields uniformly ultimately bounded (UUB) convergence of the transformed states and approximation of the optimal control policy. Simulation results are presented for a two-state dynamical system to compare the developed method to existing model-free and model-based ADP methods. Specifically, the developed R-MBRL approach with BFs, segmentation, and sparse BE extrapolation can be applied to systems to achieve online approximate optimal control with additional safety guarantees.

In Chapter 5 a switched ADP method is developed. ADP-based controllers directly tune system performance by assigning costs to the state and control variables. Altering the parameters of the cost function in an ADP-based controller affects system performance by modifying the reward gained from the system trajectory. Chapter 5 proposes a method by which the parameters of the cost function and the system dynamics can be discretely varied, and the corresponding optimal controllers can be simultaneously learned and implemented in a way that maintains closed-loop stability during the learning phase. That is, different controller properties can be achieved by varying the cost of the states.

Previous ADP results consider fixed state and control cost matrices within the reward function, such as [19, 22, 41]. Works such as [55] and [56] examine the use of ADP-based methods for switched discrete-time nonlinear systems. Previous results such as [42–46] use optimal control methods to minimize cost function(s) of a switched system. These methods use a fixed mode sequence (see [42, 45–47]) or fixed switching instances (see [44]). In comparison, the developed method considers an arbitrary switching sequence that satisfies a dwell-time condition to approximate the value functions of a finite number of continuous-time subsystems. Unlike the aforementioned methods, Chapter 5 develops a Lyapunov-based framework to prove convergence of a control policy to the neighborhood of an optimal policy while maintaining closed-loop stability. While Chapter 5 focuses on a framework for switching between multiple ADP-based controllers and modifying control system performance by using different rewards and dynamical models, it does not address the optimality of the overall switched system.

A complication in Lyapunov-based analyses for switched systems is the growth and discontinuity of Lyapunov functions at switching instances. To overcome this issue, a dwell-time analysis is performed to prove stability of the overall switching sequence. The included dwell-time analysis accounts for the worst-case growth and discontinuity between multiple Lyapunov-like functions during switching instances by explicitly

determining the minimum time required before the system can switch to a different subsystem. Overall stability of the system for an arbitrary switching sequence of UUB stable subsystems is established using the developed technique.

Chapter 6 develops a DNN-based system identifier for ADP. Results in [49] and [50] leverage a multi-timescale deep model reference adaptive controller. Similarly, the method in [52] uses a multi-timescale DNN to estimate the unknown system dynamics, which facilitates a trajectory tracking objective. In [52], a gradient-based adaptation policy is used to estimate the output layer weights of the DNN in real-time. Simultaneous to real-time execution, input-output data is stored and used to update the inner-layer weights using traditional offline DNN function approximation methods.

However, the adaptive update policy in [52] cannot be trivially extended to system identification within the ADP framework. To prove stability of the overall system with an ADP controller, the adaptive update policy of the output-layer DNN weights must include the CL modification from [57], which complicates the stability analysis (cf., model-based ADP analyses in [19] and [48]). Furthermore, iteratively updating the DNN introduces nonsmooth signals, meaning that a more rigorous Lyapunov-like analysis must be performed [58].

The primary contribution of Chapter 6 is to analyze the stability of the ADP tracking problem while using the multi-timescale DNN system identification approach. Simulation results are included to illustrate the effectiveness of the developed technique in comparison to existing optimal control and ADP-based results.

Chapter 7 concludes the dissertation by highlighting the contributions of each chapter and proposing future extensions of the work in this dissertation.

CHAPTER 2
PRELIMINARIES

2.1 Notation

Let $\mathbb{R}$, $\mathbb{Z}$, and $\mathbb{N}$ denote the set of real numbers, integers, and natural numbers, respectively. Furthermore, let $\mathbb{R}_{\geq 0} \triangleq [0, \infty)$, $\mathbb{R}_{>0} \triangleq (0, \infty)$, $\mathbb{Z}_{\geq 0} \triangleq \mathbb{R}_{\geq 0} \cap \mathbb{Z}$, $\mathbb{Z}_{>0} \triangleq \mathbb{R}_{>0} \cap \mathbb{Z}$, $\mathbb{N}_{\geq 0} \triangleq \mathbb{R}_{\geq 0} \cap \mathbb{N}$, $\mathbb{N}_{>0} \triangleq \mathbb{R}_{>0} \cap \mathbb{N}$. Generally, let a matrix $A \in \mathbb{R}^{p \times q}$ be a real-valued $p \times q$ matrix where $p, q \in \mathbb{Z}_{>0}$. If $p = q$ and A has real eigenvalues, then the maximum and minimum eigenvalues of A are denoted by $\lambda_{\max}\{A\} \in \mathbb{R}$ and $\lambda_{\min}\{A\} \in \mathbb{R}$, respectively. The positive definite (PD) and positive semi-definite (PSD) square matrices $A, B \in \mathbb{R}^{p \times p}$ are denoted by $A > 0$ and $B \geq 0$, respectively. Negative definite and negative semi-definite matrices are denoted similarly.

Unless otherwise noted, the domain of all functions is assumed to be $\mathbb{R}_{\geq 0}$. Functions with the domain $\mathbb{R}_{\geq 0}$ are defined with abuse of notation using only their image, e.g., $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is instead defined as $x \in \mathbb{R}^n$. Generally, the gradient $\left[\frac{\partial f(x,y)}{\partial x_1}^T, \ldots, \frac{\partial f(x,y)}{\partial x_n}^T\right]^T$ is denoted by $\nabla_x f(x, y)$.

The Euclidean norm of a vector $r \in \mathbb{R}^p$ is denoted by $\|r\| \triangleq \sqrt{r^T r}$. The Frobenius norm of a matrix $A \in \mathbb{R}^{n \times m}$ is denoted by $\|A\| \triangleq \sqrt{\operatorname{tr}(A^T A)}$. $\|\cdot\|$ denotes both the Euclidean norm for vectors and Frobenius norm for matrices. $\operatorname{tr}(\cdot)$ denotes the trace operator.

The diagonal matrix whose entries consist of $x \triangleq [x_1, x_2, ..., x_p] \in \mathbb{R}^p$ is denoted by $\operatorname{diag}(x_1, x_2, ..., x_p) \in \mathbb{R}^{p \times p}$.

The Moore-Penrose pseudoinverse of $A \in \mathbb{R}^{p \times q}$ is denoted by $A^+ \in \mathbb{R}^{q \times p}$.

The $p \times q$ zero matrix and the $p \times 1$ zero vector are denoted by $\mathbf{0}_{p \times q}$ and $\mathbf{0}_p$, respectively. Let $\mathbf{1}_p \in \mathbb{R}^p$ denote a column vector with all entries being 1. The $p \times p$ identity matrix is denoted by I_p.

2.2 Approximate Dynamic Programming

This dissertation focuses on obtaining online approximate solutions to infinite-horizon optimal control problems. Consider a class of nonlinear control-affine systems

$$\dot{x} = f(x) + g(x)u, \tag{2–1}$$

where $x \in \mathbb{R}^n$ denotes the system state, $u \in \mathbb{R}^m$ denotes the control input, $f : \mathbb{R}^n \to \mathbb{R}^n$ denotes the drift dynamics, and $g : \mathbb{R}^n \to \mathbb{R}^{m\times n}$ denotes the control effectiveness matrix, where $n > m$ and the pseudoinverse of $g(x)$ exists.

The assumptions in this section facilitate the formulation of the approximate optimal controller and stability analyses in Chapters 3-6.

Assumption 2.1. The function f is locally Lipschitz and $f(0) = 0$. Furthermore, $\nabla_x f : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is continuous.

Assumption 2.2. The function g is a locally Lipschitz function, has full column rank for all $x \in \mathbb{R}^n$, and is bounded such that $\|g(x)\| \leq \overline{g}$, where $\overline{g} \in \mathbb{R}_{>0}$ is the maximum singular value of $g(x)$.

The control objective is to solve the infinite-horizon optimal regulation problem for each subsystem, i.e., determine a control policy u that minimizes the infinite horizon cost functional, $J : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$, defined as

$$J(x,u) \triangleq \int_{t_0}^{\infty} r(x(\tau), u(\tau))\, d\tau, \tag{2–2}$$

subject to (2–1) while regulating the system states to the origin (i.e., $\|x\| = 0$), where $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ is the instantaneous cost defined as $r(x,u) \triangleq Q(x) + u^T R u$, where $Q : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is a user-defined PD function, and $R \in \mathbb{R}^{m\times m}$ is a constant user-defined PD symmetric matrix.

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

The infinite horizon value function (i.e., the cost to go) is denoted by $V^* : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ and given by

$$V^*(x) = \min_{u(\tau)\in U,\, \tau\in\mathbb{R}_{\geq t}} \int_t^\infty r\left(x(\tau), u(\tau)\right) d\tau, \tag{2–3}$$

where $U \subseteq \mathbb{R}^m$ denotes the action space. Provided an optimal control policy exists, the value function is characterized by the corresponding HJB equation

$$0 = \nabla_x V^*(x)\left(f(x) + g(x)\, u^*\right) + r\left(x, u^*\right), \tag{2–4}$$

with the boundary condition $V^*(0) = 0$. Generally, the HJB equation cannot be solved analytically, with the exception of a few special cases, specifically, linear and scalar systems.

Assumption 2.3. The value function V^* is continuously differentiable.

Provided the HJB in (2–4) admits a continuously differentiable PD solution, then the optimal closed-loop control policy $u^* : \mathbb{R}^n \to \mathbb{R}^m$ is

$$u^*(x) = -\frac{1}{2} R^{-1} g(x)^T \left(\nabla_x V^*(x)\right)^T. \tag{2–5}$$

Remark 2.1. Under Assumptions 2.1-2.3, 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 [59].

The HJB equation in (2–4) requires knowledge of the optimal value function, which, generally, is an unknown function for nonlinear systems. Parametric methods can be used to approximate the value function over a compact domain. To facilitate the solution of (2–4), let $\Omega \subset \mathbb{R}^n$ be a compact set containing the origin. The universal function approximation property of single-layer NNs is used to represent the value function V^* for all $x \in \Omega$ as

$$V^*(x) = W^{*T} \sigma(x) + \epsilon(x), \tag{2–6}$$

where $W^* \in \mathbb{R}^L$ is an unknown bounded vector of weights, $\sigma : \mathbb{R}^n \to \mathbb{R}^L$ is a user-defined vector of basis functions, and $\epsilon : \mathbb{R}^n \to \mathbb{R}$ is the bounded function approximation error. Substituting (2–6) into (2–5), the optimal control policy u^* can be expressed in terms of the optimal value function V^* gradient as

$$u^*(x) = -\frac{1}{2} R^{-1} g(x) \left(\nabla_x \sigma(x)^T W^* + \nabla_x \epsilon(x)^T \right). \tag{2–7}$$

Property 2. There exists a set of constants that bound the unknown ideal weight vector W^*, the user-defined activation function σ, and function reconstruction error ϵ, from above such that $\|W^*\| \leq \overline{W^*}$, $\sup_{x \in \Omega} \|\sigma(x)\| \leq \overline{\sigma}$, $\sup_{x \in \Omega} \|\nabla_x \sigma(x)\| \leq \overline{\nabla_x \sigma}$, $\sup_{x \in \Omega} \|\epsilon(x)\| \leq \overline{\epsilon}$, $\sup_{x \in \Omega} \|\nabla_x \epsilon(x)\| \leq \overline{\nabla_x \epsilon}$, where $\overline{W^*}$, $\overline{\sigma}$, $\overline{\nabla_x \sigma}$, $\overline{\epsilon}$, $\overline{\nabla_x \epsilon} \in \mathbb{R}_{\geq 0}$ [60].

2.2.1 Value Function Approximation

Since the ideal weights W^* are unknown, a parametric estimate, called the critic weight vector $\hat{W}_c \in \mathbb{R}^L$, is substituted to approximate the optimal value function in (2–6) to yield $\hat{V} : \mathbb{R}^n \times \mathbb{R}^L \to \mathbb{R}$, where

$$\hat{V}\left(x, \hat{W}_c\right) = \hat{W}_c^T \sigma(x). \tag{2–8}$$

An actor weight vector $\hat{W}_a \in \mathbb{R}^L$, is used to provide an approximate version of (2–7), the approximate optimal control policy $\hat{u} : \mathbb{R}^n \times \mathbb{R}^L \to \mathbb{R}$ is given by

$$\hat{u}\left(x, \hat{W}_a\right) = -\frac{1}{2} R^{-1} g(x)^T \left(\nabla_x \sigma(x)^T \hat{W}_a \right). \tag{2–9}$$

2.2.2 Bellman Error

The HJB equation in (2–4) is equal to zero under optimal conditions; however, substituting (2–8) and (2–9) into (2–4) results in a residual term $\delta : \mathbb{R}^n \times \mathbb{R}^L \times \mathbb{R}^L \to \mathbb{R}$, which is referred to as the BE, defined as

$$\delta\left(x, \hat{W}_c, \hat{W}_a\right) \triangleq \nabla_x \hat{V}\left(x, \hat{W}_c\right)\left(f(x) + g(x)\,\hat{u}\left(x, \hat{W}_a\right)\right) + \hat{u}\left(x, \hat{W}_a\right)^T R \hat{u}\left(x, \hat{W}_a\right) + Q(x), \tag{2–10}$$

where $\nabla_x \hat{V}\left(x, \hat{W}_c\right) = \hat{W}_c^T \nabla_x \sigma\left(x\right)$. The BE is indicative of how close the actor and critic weight estimates are to the ideal weights. By defining the mismatch between the estimates and the ideal values as $\tilde{W}_c \triangleq W^* - \hat{W}_c$ and $\tilde{W}_a \triangleq W^* - \hat{W}_a$, substituting (2–6) and (2–7) in (2–4), and subtracting from (2–10) yields

$$\delta = \frac{1}{4}\tilde{W}_a^T G_\sigma \tilde{W}_a - \omega^T \tilde{W}_c + O\left(x\right), \tag{2–11}$$

where $\omega : \mathbb{R}^n \times \mathbb{R}^L \to \mathbb{R}^n$ is defined as

$$\omega\left(x, \hat{W}_a\right) \triangleq \nabla_x \sigma\left(x\right)\left(f\left(x\right) + g\left(x\right)\hat{u}\left(x, \hat{W}_a\right)\right), \tag{2–12}$$

and $O\left(x\right) \triangleq \frac{1}{2}W^{*T}\nabla_x \sigma\left(x\right) G \nabla_x \epsilon^T + \frac{1}{4}G_\epsilon - \nabla_x \epsilon f\left(x\right)$. G, G_σ, and G_ϵ are defined as $G = G\left(x\right) \triangleq g\left(x\right) R^{-1} g\left(x\right)^T$, $G_\sigma = G_\sigma\left(x\right) \triangleq \nabla_x \sigma\left(x\right) G\left(x\right) \nabla_x \sigma\left(x\right)^T$, and $G_\epsilon = G_\epsilon\left(x\right) \triangleq \nabla_x \epsilon\left(x\right) g\left(x\right) \nabla_x \epsilon\left(x\right)^T$, respectively.

Remark 2.2. The expressions in (2–10) and (2–11) are equivalent for the BE. However, (2–10) is used in implementation, while (2–11) is used in the Lyapunov-based stability analysis.

2.3 Extension to Tracking Problems

Following the development in [61], the following section modifies the above ADP problem formulation to facilitate a trajectory tracking objective. The control objective is to track a time-varying continuously differentiable signal $x_d \in \mathbb{R}^n$ bounded such that $\sup_{t \in \mathbb{R}_{\geq 0}} \|x_d\left(t\right)\| \leq \overline{x_d}$. To quantify the tracking objective, the tracking error is defined as $e \triangleq x - x_d$.

Assumption 2.4. The desired trajectory is bounded from above by an positive constant $\overline{x_d} \in \mathbb{R}_{\geq 0}$ such that $\sup_{t \in \mathbb{R}_{\geq 0}} \|x_d\| \leq \overline{x_d}$.

Assumption 2.5. There exists a locally Lipschitz function $h_d : \mathbb{R}^n \to \mathbb{R}^n$, such that $h_d\left(x_d\right) \triangleq \dot{x}_d$ and $g\left(x_d\right) g^+\left(x_d\right)\left(h_d\left(x_d\right) - f\left(x_d\right)\right) = h_d\left(x_d\right) - f\left(x_d\right), \forall t \in \mathbb{R}_{\geq 0}$, where $g^+ : \mathbb{R}^n \to \mathbb{R}^{m \times n}$ is defined as $g^+\left(x\right) \triangleq \left(g^T\left(x\right) g\left(x\right)\right)^{-1} g^T\left(x\right)$.

Based on Assumptions 2.1-2.5, the control policy $u_d : \mathbb{R}^n \to \mathbb{R}^m$, which tracks the desired trajectory (i.e., trajectory tracking component of the controller), is $u_d(x_d) \triangleq g^+(x_d)(h_d(x_d) - f(x_d))$.

Define the transient portion of the controller as $\mu \triangleq u - u_d(x_d)$. Following the development in [61] and [48], the concatenated state dynamics are written as

$$\dot{\zeta} = F(\zeta) + G(\zeta)\mu, \tag{2–13}$$

where $\zeta \in \mathbb{R}^{2n}$ is the concatenated state vector $\zeta \triangleq \left[e^T, x_d^T\right]^T$, $F : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ is defined as

$$F(\zeta) \triangleq \left[f(e + x_d)^T - h_d(x_d)^T + u_d(x_d)^T g(e + x_d)^T, h_d(x_d)^T\right]^T, \tag{2–14}$$

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

$$G(\zeta) \triangleq \left[g(e + x_d)^T, \mathbf{0}_{m \times n}\right]^T, \tag{2–15}$$

where $\mathbf{0}_{m \times n}$ is a matrix of zeros with m rows and n columns.

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

$$J(\zeta, \mu) = \int_0^\infty r(\zeta(\tau), \mu(\tau))\, d\tau, \tag{2–16}$$

subject to (2–13) while eliminating tracking error, 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 defined as $\overline{Q}(\zeta) \triangleq Q(e)$,where $Q : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is a user-defined PD function and $R \in \mathbb{R}^{m \times m}$ is a constant user-defined PD symmetric matrix.

Property 3. The function $\overline{Q}$ satisfies $\underline{q}(\|e\|) \leq \overline{Q}(\zeta) \leq \overline{q}(\|e\|)$, where $\underline{q}, \overline{q} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$.

The scalar infinite-horizon value function for the optimal solution, i.e. the cost-to-go, denoted by $V^* : \mathbb{R}^{2n} \to \mathbb{R}_{\geq 0}$, is given by

$$V^*(\zeta) = \min_{\mu(\tau) \in U,\, \tau \in \mathbb{R}_{\geq 0}} \int_t^{\infty} r(\zeta(\tau), \mu(\tau))\, d\tau, \tag{2–17}$$

where $U \subseteq \mathbb{R}^m$ denotes the action space. If the optimal value function is continuously differentiable, then the optimal control policy $\mu^* : \mathbb{R}^{2n} \to \mathbb{R}^m$ is the unique solution to the corresponding HJB equation

$$0 = \nabla_\zeta V^*(\zeta)\left(F(\zeta) + G(\zeta)\mu^*\right) + \overline{Q}(\zeta) + \mu^*(\zeta)^T R \mu^*(\zeta), \tag{2–18}$$

which has the boundary condition $V^*(0) = 0$.

Value Function Approximation

The HJB equation in (2–18) requires knowledge of the optimal value function. To facilitate the solution of (2–18), let $\Omega \subset \mathbb{R}^{2n}$ be a compact set. The universal function approximation property of NNs is used to represent the optimal value function as

$$V^*(\zeta) = W^{*T}\sigma(\zeta) + \epsilon(\zeta), \tag{2–19}$$

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 basis functions, and $\varepsilon : \mathbb{R}^{2n} \to \mathbb{R}$ is the bounded function approximation error. Like the regulation case, the optimal control policy μ^* can be expressed in terms of the optimal value function V^* gradient as

$$\mu^*(\zeta) = -\frac{1}{2} R^{-1} G(\zeta)\left(\nabla_\zeta \sigma(\zeta)^T W^* + \nabla_\zeta \epsilon(\zeta)^T\right). \tag{2–20}$$

Remark 2.3. Like ADP for state regulation, under Assumptions 2.1-2.5, 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 [59].

The ideal weights W^* in (2–19) and (2–20) are unknown, hence, an approximation of W^* is sought. Specifically, the critic estimate, $\hat{W}_c \in \mathbb{R}^L$ is substituted to estimate the value function $\hat{V} : \mathbb{R}^{2n} \times \mathbb{R}^L \to \mathbb{R}$ denoted as

$$\hat{V}\left(\zeta, \hat{W}_c\right) = \hat{W}_c^T \sigma\left(\zeta\right). \tag{2–21}$$

Using an actor-critic approach [16, 62, 63], an actor estimate, $\hat{W}_a \in \mathbb{R}^L$ is substituted to estimate the optimal control policy $\hat{\mu} : \mathbb{R}^{2n} \times \mathbb{R}^L \to \mathbb{R}$ defined by

$$\hat{\mu}\left(\zeta, \hat{W}_a\right) \triangleq -\frac{1}{2} R^{-1} G\left(\zeta\right)^T \left(\nabla_\zeta \sigma\left(\zeta\right)^T \hat{W}_a\right). \tag{2–22}$$

The omitted development of the BE follows that of (2–10) but the notation used to facilitate the optimal tracking problem formulation. Similarly, Property 6 holds for the NN parameterization in (2–19).

CHAPTER 3
REINFORCEMENT LEARNING WITH SPARSE BELLMAN ERROR EXTRAPOLATION FOR INFINITE HORIZON APPROXIMATE OPTIMAL CONTROL

This chapter and the work in [64] and [65] provide an approximate online adaptive solution to the infinite-horizon optimal tracking problem for control-affine continuous-time nonlinear systems with uncertain drift dynamics; computational demands are reduced through the use of state space segmentation and SNNs. A model-based ADP approach, which is facilitated by the use of a CL-based system identifier, is used to relax the PE condition. To reduce computational complexity the state space is segmented into a user-defined number of segments. In addition, within each segment a SNN can be used instead of a conventional NN for BE extrapolation. Off-policy trajectories are selected over each segment to facilitate learning of the value function weight estimates. SNNs enable a framework for switching and state space segmentation as well as computational benefits due to the a smaller number of neurons being activated. Within each sparse segment, off-policy trajectories are used to extrapolate the BE across their respective segments to provide an optimal policy for each segment. Discontinuities occur in the weight update laws since different groupings of extrapolated BEs 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. Three simulation results are included to demonstrate the performance and validity of the developed method. An additional simulation result shows that using the sparse, switched BE extrapolation method developed in this chapter reduces the computation time by 85.6% when compared to traditional BE extrapolation.

3.1 Online System Identification

On any compact set $\mathcal{C} \subset \mathbb{R}^n$, the tracking drift dynamics f can be represented using a NN as $f(x) = \theta^T \sigma_\theta \left(Y^T x_\theta(x)\right) + \epsilon_\theta(x)$, where $x_\theta : \mathbb{R}^n \to \mathbb{R}^{n+1}$ is defined as $x_\theta(x) \triangleq \left[1, x^T\right]^T$, $\theta \in \mathbb{R}^{(p+1)\times n}$ is a constant, unknown output-layer weight matrix, $Y \in \mathbb{R}^{(n+1)\times p}$ denotes a hidden-layer weight matrix, $\sigma_\theta : \mathbb{R}^p \to \mathbb{R}^{p+1}$ is a NN basis

function, $\epsilon_\theta : \mathbb{R}^n \to \mathbb{R}^n$ 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 θ and known finite constants $\overline{\theta}$, $\overline{\epsilon_\theta}$, and $\overline{\nabla_x \epsilon} \in \mathbb{R}_{\geq 0}$, such that $\|\theta\| \leq \overline{\theta}$, $\sup_{x \in \mathcal{C}} \|\epsilon_\theta(x)\| \leq \overline{\epsilon_\theta}$, and $\sup_{x \in \mathcal{C}} \|\nabla_x \epsilon_\theta(x)\| \leq \overline{\nabla_x \epsilon_\theta}$ [60].

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 $\hat{f}\left(x, \hat{\theta}\right) \triangleq \hat{\theta}^T \sigma_\theta\left(Y^T x_\theta(x)\right)$. Hence, a state estimator can be developed as

$$\dot{\hat{x}} = \hat{f}\left(x, \hat{\theta}\right) + g(x)\, u + k\tilde{x}, \tag{3–1}$$

where $\tilde{x} \triangleq x - \hat{x}$ and $k \in \mathbb{R}_{>0}$ is a user-selected estimator learning gain. The update law of the system identification NN weight estimates are updated using the CL-based update law

$$\dot{\hat{\theta}} = \Gamma_\theta \sigma_\theta\left(Y^T x_\theta(x_j)\right)\tilde{x}^T + k_\theta \Gamma_\theta \sum_{j=1}^{M} \sigma_{\theta j}\left(\dot{\bar{x}}_j - g(x_j)\, u_j - \hat{\theta}^T \sigma_{\theta j}\right)^T, \tag{3–2}$$

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

Assumption 3.1. 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 $\lambda_{\min}\left\{\sum_{j=1}^M \sigma_{\theta j}\sigma_{\theta j}^T\right\} > 0$ and $\left\|\dot{\bar{x}}_j - \dot{x}_j\right\| < \overline{d}\ \forall j$ are available *a priori*, where $\overline{d} \in \mathbb{R}_{>0}$ is a known constant, $\dot{x}_j = f(x_j) + g(x_j)\, u_j$, $\sigma_{\theta j} \triangleq \sigma_\theta\left(Y^T x_\theta(x_j)\right)$ [66].

Remark 3.1. The availability of the system identification history stack *a priori* is not necessary [48]. Assumption 3.1 is used to focus the scope of this chapter and simplify the stability analysis.

3.2 Bellman Error

Recall, the HJB equation in (2–18) is equal to zero under optimal conditions; however, substituting (2–21), (2–22), and the approximated drift dynamics $\hat{F}\left(x, \hat{\theta}\right)$ as $F(x)$ in (2–18) 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

$$\hat{\delta}\left(\zeta,\hat{\theta},\hat{W}_c,\hat{W}_a\right) \triangleq \hat{\mu}\left(\zeta,\hat{W}_a\right)^T R\hat{\mu}\left(\zeta,\hat{W}_a\right) + \overline{Q}\left(\zeta\right) + \nabla_\zeta \hat{V}\left(\zeta,\hat{W}_c\right)\left(F_\theta\left(\zeta,\hat{\theta}\right) + F_1\left(\zeta\right) + G\left(\zeta\right)\hat{\mu}\left(\zeta,\hat{W}_a\right)\right), \tag{3–3}$$

where $F_\theta : \mathbb{R}^{2n} \times \mathbb{R}^{(p+1)\times n} \to \mathbb{R}^{2n}$ is defined as

$$F_\theta\left(\zeta,\hat{\theta}\right) \triangleq \left[\hat{f}\left(x,\hat{\theta}\right) - g\left(x\right)g^+\left(x_d\right)\hat{f}\left(x_d,\hat{\theta}\right),\ 0_{n\times 1}\right], \tag{3–4}$$

and $F_1 : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ is defined as

$$F_1\left(\zeta\right) \triangleq \left[\left(-h_d\left(x_d\right) + g\left(x\right)g^+\left(x_d\left(t\right)\right)h_d\left(x_d\right)\right)^T,\ h_d\left(x_d\right)^T\right]^T. \tag{3–5}$$

The BE in (3–3) 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 (2–21) and (2–22) into (3–3), and subtracting from (3–3) yields the analytical form of the BE, given by

$$\hat{\delta}\left(\zeta,\hat{\theta},\hat{W}_c,\hat{W}_a\right) = -W^{*T}\nabla_\zeta\sigma\left(F_\theta\left(\zeta,\theta\right) - F_\theta\left(\zeta,\hat{\theta}\right)\right) - \omega^T\tilde{W}_c + \frac{1}{4}\tilde{W}_a^T G_\sigma \tilde{W}_a + O\left(\zeta\right), \tag{3–6}$$

where $\omega : \mathbb{R}^{2n} \times \mathbb{R}^L \times \mathbb{R}^{(p+1)\times n} \to \mathbb{R}^L$ is defined as $\omega\left(\zeta,\hat{W}_a,\hat{\theta}\right) \triangleq \nabla_\zeta\sigma\left(\zeta\right)F_\theta\left(\zeta,\hat{\theta}\right) + \nabla_\zeta\sigma\left(\zeta\right)F_1\left(\zeta\right) + \nabla_\zeta\sigma\left(\zeta\right)G\left(\zeta\right)\hat{\mu}\left(\zeta,\hat{W}_a\right)$, and $O\left(\zeta\right) \triangleq \frac{1}{2}\nabla_\zeta\epsilon\left(\zeta\right)G_R\nabla_\zeta\sigma^T\left(\zeta\right)W^* + \frac{1}{4}G_\epsilon - W^{*T}\nabla_\zeta\sigma\left(\zeta\right)\epsilon_\theta\left(\zeta\right) - \nabla_\zeta\epsilon\left(\zeta\right)F_\theta\left(\zeta,\theta\right) - \nabla_\zeta\epsilon\left(\zeta\right)\epsilon_\theta\left(\zeta\right) - \nabla_\zeta\epsilon\left(\zeta\right)F_1\left(\zeta\right)$. The notation G_R, G_σ, and G_ϵ is defined as $G_R = G_R\left(\zeta\right) \triangleq G\left(\zeta\right)R^{-1}G\left(\zeta\right)^T$, $G_\sigma = G_\sigma\left(\zeta\right) \triangleq \nabla_\zeta\sigma\left(\zeta\right)G_R\left(\zeta\right)\nabla_\zeta\sigma\left(\zeta\right)^T$, and $G_\epsilon = G_\epsilon\left(\zeta\right) \triangleq \nabla_\zeta\epsilon\left(\zeta\right)G\left(\zeta\right)\nabla_\zeta\epsilon\left(\zeta\right)^T$, respectively.

Sparse Bellman Error Extrapolation

At each time instant $t \in \mathbb{R}_{\geq 0}$, the estimated BE in (3–3) and policy in (2–22) are evaluated using the current system state, critic estimate, and actor estimate to get the instantaneous BE and control policy, which are denoted by $\hat{\delta} \triangleq \hat{\delta}\left(\zeta,\hat{\theta},\hat{W}_c,\hat{W}_a\right)$ and

$\hat{\mu} \triangleq \hat{\mu}\left(\zeta, \hat{W}_a\right)$, respectively. However, using only the on-trajectory BE and control policy requires the traditional PE condition to be satisfied to show exponential convergence.

Motivated to increase computational efficiency, local BE extrapolation has been performed in [22] and [53] around unexplored areas of the state space by using more efficient computational capabilities compared with previous methods. Similarly, using SNNs improves computational efficiency and use segmentation for BE extrapolation. This allows the BE to be estimated 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 estimated across the entire operating region within the state space without the computational burden of nonsparse methods. The use of BE extrapolation provides simulation of experience, which provides excitation via off-policy learning.

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, 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. The segments are predetermined by the user and are state dependent (e.g. in [32] the states: altitude, angle of attack, and Mach number determine segment activation). Using the extrapolated trajectories $\zeta_i \in \Omega_j$ for $j \in S$, the BE in (3–3) is evaluated such that $\hat{\delta}_i \triangleq \hat{\delta}\left(\zeta_i, \hat{\theta}, \hat{W}_c, \hat{W}_a\right)$. For a given $j \in S$, the tuple $\left(\Sigma_c^j, \Sigma_a^j, \Sigma_\Gamma^j\right)$ 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 $\omega_i \triangleq \omega\left(\zeta_i, \hat{\theta}, \hat{W}_a\right)$, $\rho_i = \rho\left(\zeta_i, \hat{\theta}, \hat{W}_a\right) = 1 + \nu\omega_i^T\Gamma\omega_i$, $\Gamma \in \mathbb{R}^{L\times L}$ is a subsequently defined user-initialized learning gain, and Assumption 3.2 is provided to facilitate the subsequent stability analysis.

Assumption 3.2. 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 $0 < \underline{c} \triangleq \inf_{t \in \mathbb{R}_{\geq 0}, j \in S} \lambda_{\min} \left\{\Sigma_\Gamma^j\right\}$ for all $t \in \mathbb{R}_{\geq 0}$.

Remark 3.2. The constant $\underline{c}$ is the lower bound of the value of each input-output data pairs' minimum eigenvalues.

Remark 3.3. The BE extrapolations 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 time-derivative of the candidate Lyapunov function, requiring a more nuanced stability analysis with generalized solution to differential inclusions.

3.3 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

$$\dot{\hat{W}}_c = -\eta_{c1}\Gamma\frac{\omega}{\rho}\hat{\delta} - \eta_{c2}\Gamma\Sigma_c^j, \tag{3–7}$$

$$\dot{\Gamma} = \left(\lambda \Gamma - \eta_{c1} \frac{\Gamma \omega \omega^T \Gamma}{\rho^2} - \Gamma \eta_{c2} \left(\Sigma_\Gamma^j \right) \Gamma \right) \mathbf{1}_{\{\underline{\Gamma} \leq \|\Gamma\| \leq \overline{\Gamma}\}}, \tag{3–8}$$

$$\dot{\hat{W}}_a = -\eta_{a1} \left(\hat{W}_a - \hat{W}_c \right) - \eta_{a2} \hat{W}_a + \frac{\eta_{c1} G_\sigma^T \hat{W}_a \omega^T}{4\rho} \hat{W}_c + \eta_{c2} \Sigma_a^j \hat{W}_c, \tag{3–9}$$

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. $\|\Gamma(t)\|$ is upper and lower bounded by user-defined saturation constants, $\overline{\Gamma}$ and $\underline{\Gamma}$, respectively. Using (4–25) ensures that $\underline{\Gamma} \leq \|\Gamma(t)\| \leq \overline{\Gamma}$ for all $t \in \mathbb{R}_{>0}$. The indicator function in (4–25) can be removed with additional assumptions and modifications outlined in [22].

3.4 Stability Analysis

To facilitate the stability analysis, let $\tilde{\theta} \triangleq \theta - \hat{\theta}$, and $Z \in \mathbb{R}^{n(3+p)+2L}$ denote a concatenated state $Z \triangleq \left[e^T, \tilde{W}_c^T, \tilde{W}_a^T, \tilde{x}^T, \operatorname{vec}\left(\tilde{\theta}\right)^T \right]^T$. The function $\overline{Q}$ is PSD, therefore $V^*(\zeta)$ is also PSD. Hence, V^* is not a valid Lyapunov function. The result in [61] 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 $V_L : \mathbb{R}^{n(3+p)+2L} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ be a candidate Lyapunov function defined as

$$\begin{aligned} V_L(Z,t) \triangleq V^*(e,t) + \frac{1}{2} \tilde{W}_c(t)^T \Gamma(t)^{-1} \tilde{W}_c(t) + \frac{1}{2} \tilde{W}_a(t)^T \tilde{W}_a(t) \\ + \frac{1}{2} \tilde{x}(t)^T \tilde{x}(t) + \frac{1}{2} \text{tr}\left(\tilde{\theta}(t)^T \Gamma_\theta^{-1} \tilde{\theta}(t) \right). \end{aligned} \tag{3–10}$$

Using the properties of $V_{na}^*(e,t)$ and [67, Lemma 4.3], then (3–10) 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 (3–8), the normalized regressors $\frac{\omega}{\rho}$ and $\frac{\omega_i}{\rho_i}$ can be bounded as $\sup_{\zeta \in \Omega} \left\| \frac{\omega}{\rho} \right\| \leq \frac{1}{2\sqrt{\nu \underline{\Gamma}}}$ and $\sup_{\zeta_i \in \Omega_j, j \in \mathbb{S}} \left\| \frac{\omega_i}{\rho_i} \right\| \leq \frac{1}{2\sqrt{\nu \underline{\Gamma}}}$. 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 $\sup_{\zeta \in \Omega} \|G_\sigma\| \leq \left(\overline{\nabla_\zeta \sigma G} \right)^2 \lambda_{\max}\{R^{-1}\} \triangleq \overline{G_\sigma}$, respectively.

Theorem 3.1. *Provided the dynamics in (2–13), Assumption 3.2, and the sufficient gain conditions*

$$\eta_{a1} + \eta_{a2} \geq \frac{1}{\sqrt{\nu \underline{\Gamma}}} \left(\eta_{c1} + \eta_{c2}\right) \overline{W^* G_\sigma}, \tag{3–11}$$

$$\underline{c} \geq 3\frac{\eta_{a1}}{\eta_{c2}} + \frac{3\left(\left(\eta_{c1} + \eta_{c2}\right) \overline{W^* G_\sigma}\right)^2}{16\eta_{c2}\nu\underline{\Gamma}\left(\eta_{a1} + \eta_{a2}\right)} + \frac{9\left(\left(\eta_{c1} + \eta_{c2}\right) \overline{W^* \nabla_\zeta \sigma \phi}\right)^2}{8\eta_{c2} k_\theta \nu \underline{\Gamma} \lambda_{\min}\left\{\sum_{j=1}^{M} \phi_j \phi_j^T\right\}} \tag{3–12}$$

$$\nu_l^{-1}\left(l\right) < \alpha_2^{-1}\left(\alpha_1\left(r\right)\right), \tag{3–13}$$

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 UUB.

Proof. Let $r \in \mathbb{R}_{>0}$ be the radius of a compact ball $\chi \subset \mathbb{R}^{n(3+p)+2L}$ centered at the origin. Let $Z\left(t\right)$ for $t \in \mathbb{R}_{\geq 0}$ be a Filippov solution to the differential inclusion $\dot{Z} \in K\left[h\right]\left(Z\right)$, where $K\left[\cdot\right]$ is defined in [68] and $h : \mathbb{R}^{n(4+p)+2L+L^2} \to \mathbb{R}^{n(4+p)+2L+L^2}$ is defined as $h \triangleq \left[\dot{\zeta}^T, \dot{\tilde{W}}_c^T, \dot{\tilde{W}}_a^T, \text{vec}\left(\dot{\Gamma}^{-1}\right)^T, \dot{\tilde{x}}^T, \text{vec}\left(\dot{\tilde{\theta}}\right)^T\right]^T$. Due to the discontinuity in the update laws in (3–7)-(3–9), the time derivative of (3–10) exists almost everywhere (a.e., i.e., for almost all $t \in \mathbb{R}_{\geq 0}$) and $\dot{V}_L\left(Z\right) \overset{a.e.}{\in} \dot{\tilde{V}}_L\left(Z\right)$, where $\dot{\tilde{V}}_L$ is the generalized time-derivative of (3–10) along the Filippov trajectories of $\dot{Z} = h\left(Z\right)$ [69]. Using the class of dynamics in (2–1); the calculus of $K\left[\cdot\right]$ from [69]; $\dot{V}^*\left(\zeta\right) = \nabla_\zeta V^*\left(\zeta\right)\left(F\left(\zeta\right) + G\left(\zeta\right)\mu\right)$; substituting (3–1), (3–2), and (3–6)-(3–9); using Young's Inequality and nonlinear damping; Assumption 3.1 and 3.2; and substituting the sufficient conditions in (3–11) and (3–12) yields $\dot{V}_L \overset{a.e.}{\leq} -\nu_l\left(\left\|Z\right\|\right), \forall \nu_l^{-1}\left(l\right) \leq \left\|Z\right\| \leq \alpha_2^{-1}\left(\alpha_1\left(r\right)\right)$, where $\nu_l\left(\left\|Z\right\|\right) \triangleq \frac{q(\|e\|)}{2} + \frac{\eta_{c2}\underline{c}}{12}\left\|\tilde{W}_c\right\|^2 + \frac{\eta_{a1}+\eta_{a2}}{16}\left\|\tilde{W}_a\right\|^2 + \frac{k}{4}\left\|\tilde{x}\right\|^2 + \frac{k_\theta \lambda_{\min}\left\{\sum_{j=1}^{M} \phi_j \phi_j^T\right\}}{6}\left\|\text{vec}\left(\tilde{\theta}\right)\right\|^2$. Since (3–10) is a common Lyapunov-like function across each segment $j \in \mathbb{S}$, [67, Theorem 4.18] can be invoked to conclude that Z is UUB such that $\limsup_{t\to\infty}\left\|Z\right\| \leq \alpha_1^{-1}\left(\alpha_2\left(\nu_l^{-1}\left(l\right)\right)\right)$

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 $u \in \mathcal{L}_\infty$. □

Remark 3.4. The sufficient condition in (3–11) can be satisfied by increasing the gains η_{a2} and ν, and selecting a penalty weight matrix R such that $\lambda_{\max}\{R^{-1}\}$ is small. Selecting a R with a large minimum eigenvalue and a large gain ν will also help satisfy the gain condition in (3–12) by decreasing the right-hand-side. The sufficient condition in (3–12) can be satisfied by selecting off-policy trajectories for sparse BE extrapolation in each Ω_j such that the minimum eigenvalue $\underline{c} \leq \underline{c}_j \triangleq \inf_{t \in \mathbb{R}_{\geq}} \left\{\Sigma_\Gamma^j(t)\right\}$ is large enough for each $j \in \mathbb{S}$. The minimum eigenvalue of each $\Sigma_\Gamma^j(t)$ can be increased by collecting redundant data, i.e., selecting $N_j \gg L$ for each segmented neighborhood $\Omega_j \subset \Omega$ and $j \in \mathbb{S}$. Provided the basis functions used for approximation are selected such that $\overline{\nabla_s \sigma}$, $\bar{\epsilon}$, and $\overline{\nabla_s \epsilon}$ are small, and η_{a2}, $\lambda_{\max}\{R\}$, ν, and $\underline{c}$ are selected sufficiently large, then the sufficient condition in (3–13) can be satisfied [48].

3.5 Simulation Results

Simulations were performed in MATLAB® Simulink®. The simulation PC has an Intel® Core™ i7-8750H CPU @ 2.20 GHz with 16 GB DDR3 RAM.

3.5.1 2-State System with Unknown Dynamics

In the following section, the developed technique is applied and compared to a linear quadratic tracking problem. The cost function is selected as $r(\zeta, \mu) = e^T Q e + \mu^T R \mu$. The linear system

$$\dot{x} = \begin{bmatrix} -1 & 1 \\ -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{3–14}$$

is leveraged in this simulation due to the fact that an analytical solution to the HJB equation in (2–18) can be calculated. The system in (3–14) was selected because it has been used in previous ADP works [48]. The control objective is to track the desired

trajectory

$$x_d(t) = \left[4\sin(t), 4\cos(t) + 4\sin(t)\right]^T \tag{3–15}$$

and to minimize the infinite horizon cost function in (2–17), where $Q = 10 \times I_2$ and $R = 1$. For value function approximation, the basis is selected as

$$\sigma(\zeta) = \left[e_1^2,\ e_1 e_2,\ e_1 x_{d1},\ e_1 x_{d2},\ e_2^2,\ e_2 x_{d1},\ e_2 x_{d2}\right]^T. \tag{3–16}$$

The drift dynamics are unknown, but are approximated using the developed method. The unknown drift dynamics are approximated with the linear basis $\sigma_\theta(x) = [x_1, x_2]^T$. To facilitate the sparse BE extrapolation, two segments are defined: $\Omega_1 \subset \mathbb{R}^2$ and $\Omega_2 \subset \mathbb{R}^2$, where $\Omega_1 \triangleq \{e \in \mathbb{R}^2 : 6 < |e_1| \leq 15,\ 6 < |e_2| \leq 15\}$ and $\Omega_2 \triangleq \{e \in \mathbb{R}^2 : |e_1| \leq 6,\ |e_2| \leq 6\}$. The basis used over Ω_1 is

$$\sigma_i(\zeta_i) = \left[e_{1,i}^2,\ e_{1,i}e_{2,i},\ e_{1,i}x_{d1,i},\ e_{1,i}x_{d2,i},\ e_{2,i}^2,\ e_{2,i}x_{d1,i}, e_{2,i}x_{d2,i}\right]^T \ \forall \zeta_i \in \Omega_1 \tag{3–17}$$

with $N_1 = 150$ extrapolated trajectories. The basis used over Ω_2 is

$$\sigma_i(\zeta_i) = \left[e_{1,i}^2,\ e_{1,i}e_{2,i},\ 0,\ 0,\ e_{2,i}^2,\ e_{2,i}x_{d1,i},\ e_{2,i}x_{d2,i}\right]^T \ \forall \zeta_i \in \Omega_2 \tag{3–18}$$

with $N_2 = 90$ extrapolated trajectories. The initial conditions used for the simulated system are $x(0) = [-10,\ 10]^T$, $\hat{W}_c(0) = 10 \cdot \mathbf{1}_7$, $\hat{W}_a(0) = 5 \cdot \mathbf{1}_7$, $\Gamma(0) = 1000 \cdot I_7$, and $\hat{\theta}(0) = \mathbf{0}_4$,[1] where $\mathbf{1}_n$ and $\mathbf{0}_n$ are vectors of ones and zeros with n entries, respectively. The gains were selected as $\eta_{c1} = 0.5$, $\eta_{c2} = 10$, $\eta_{a1} = 10$, $\eta_{a2} = 0.001$, $\lambda = 0.1$, $\nu = 0.005$, $\overline{\Gamma} = 10^4$, $\underline{\Gamma} = 1$, $k_\theta = 500$, and $\Gamma_\theta = 0.01 \times I_7$.

3.5.2 2-State System Simulation Results

Figure 3-1 illustrates the convergence of the error trajectories to zero (Figure 3-1(a)) and the state space portrait for the two-state system in (3–14) (Figure 3-1(b)). The

[1] From (3–14), $\theta = \left[-1, 1, -\frac{1}{2}, -\frac{1}{2}\right]^T$.

value function and control policy weight estimates are shown in Figure 3-2. The critic NN weights converge within approximately 3 seconds. The actor NN weights converge within approximately 3.5 seconds. The actor is expected to converge slower than the critic because the actor tracks the critic weights. Figure 3-3 presents the trajectories of the system identification NN weights compared to their actual values. Note that the NN system ID weight converge faster than the actor-critic ADP weights converge. Figure 3-4 indicates that the ADP control policy converges to the optimal control policy in approximately 0.5 seconds. The optimal value function is analytically determined by solving the HJB equation for the linear system in (3–14). Figure 3-5 presents the optimal control policy compared the estimated optimal control policy, which takes approximately 4.5 seconds to converge. It is shown in this simulation example that the approximated optimal control policy converges close to the optimal control policy. This simulation illustrates the ability to use a sparse basis for BE extrapolation while switching extrapolation stacks to achieve trajectory tracking, convergence to the optimal control policy, and convergence to the optimal value function.

3.5.3 Two-Link Manipulator with Exact Model Knowledge Simulation

To further validate the performance of the developed technique, an additional simulation is performed on a two-link robotic manipulator, in which the dynamics are known. In this simulation case the dynamics are assumed to be known. The two-link manipulator dynamics are described using Euler-Lagrange dynamics as $u = M(q)\ddot{q} + V_m(q)\dot{q} + F_d\dot{q} + F_s(\dot{q})$,where $q \triangleq [q_1, q_2]^T$, $\dot{q} = [\dot{q}_1, \dot{q}_2]^T$, and $\ddot{q} = [\ddot{q}_1, \ddot{q}_2]^T$ are the angular positions in radians, the angular velocities in radians per second of each link, and the angular acceleration in radians per second squared of each link, respectively. The inertia matrix $M : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$ is defined as $M(q) \triangleq \begin{bmatrix} p_1 + 2p_3\cos(q_2) & p_2 + p_3\cos(q_2) \\ p_2 + p_3\cos(q_2) & p_2 \end{bmatrix}$; the centripetal-Coriolis matrix $V_m : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$ is

defined as $V_m(q) \triangleq \begin{bmatrix} -p_3 \sin(q_2)\dot{q}_2 & -p_3 \sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ p_3 \sin(q_2)\dot{q}_1 & 0 \end{bmatrix}$; the viscous friction term $F_d \in \mathbb{R}^{2\times 2}$ is defined as $F_d \triangleq \text{diag}(f_{d1}, f_{d2})$, and the static friction term $F_s : \mathbb{R}^2 \to \mathbb{R}^2$ is defined as $F_s(\dot{q}) \triangleq [f_{s1} \tanh(\dot{q}_1),\ f_{s2} \tanh(\dot{q}_2)]^T$. In this simulation $p_1 = 3.473$, $p_2 = 0.196$, $p_3 = 0.242$, $f_{d1} = 5.3$, $f_{d2} = 1.1$, $f_{s1} = 8.45$, and $f_{s2} = 2.35$.

The objective is to determine a policy μ to ensure that the state $x = [q_1,\ q_2,\ \dot{q}_1,\ \dot{q}_2]^T$ 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 for the two-link robotic system can be rewritten in the form

$$f(x) = \begin{bmatrix} \dot{q} \\ \left(M^{-1}(q)\left(\left(-V_m(q) - F_d\right)\dot{q} - F_s(\dot{q})\right)\right)^T \end{bmatrix}, \tag{3–19}$$

$$g(x) = \left[\begin{bmatrix} 0, & 0 \end{bmatrix}^T, \begin{bmatrix} 0, & 0 \end{bmatrix}^T, \left(M^{-1}(q)\right)^T\right],^T \tag{3–20}$$

$$g^+(x_d) = \left[\begin{bmatrix} 0, & 0 \end{bmatrix}^T, \begin{bmatrix} 0, & 0 \end{bmatrix}^T, M^T(x_d)\right], \tag{3–21}$$

$h_d(x_d) = [x_{d3},\ x_{d4},\ -0.25x_{d1},\ x_{d2}]^T$,which, using the development in (2–13)-(2–15), can be written in the form (2–13), where $\zeta \triangleq \left[e^T,\ x_d^T\right]^T$.

Simulations were run in Simulink using a discrete-time differential equation solver at a frequency of 100 Hz on the same machine. Each simulation case is run for 150 seconds of simulated time. For each simulation case the cost parameters are $Q = [20, 20, 2, 2] \cdot I_{4\times 4}$, $R = 10 \cdot I_2$, the gains are $\eta_{c1} = 0.6$, $\eta_{c2} = 0.075$, $\eta_{a1} = 0.5$, $\eta_{a2} = 0.005$, $\overline{\Gamma} = 10 \cdot 10^3$, $\underline{\Gamma} = 1$, $\lambda = 0.002$, $\nu = 0.005$, and the initial conditions are $\hat{W}_c(0) = 10 \cdot \mathbf{1}_{23}$, $\hat{W}_a(0) = 6 \cdot \mathbf{1}_{23}$, $\Gamma(0) = 200 \cdot I_{23}$, and $x(0) = [1.8,\ 1.6,\ 0,\ 0]^T$.

The basis selected for value function approximation is a polynomial basis with 23 elements given by [61]

$$\sigma(\zeta) = \frac{1}{2}\left[\zeta_2^2,\ \zeta_1^2,\ \zeta_1\zeta_3,\ \zeta_1\zeta_4,\ \zeta_2\zeta_3,\ \zeta_2\zeta_4,\ \zeta_1^2\zeta_2^2,\ \zeta_1^2\zeta_5^2, \zeta_1^2\zeta_6^2,\ \zeta_1^2\zeta_7^2,\ \zeta_1^2\zeta_8^2,\ \zeta_2^2\zeta_5^2,\right.$$
$$\left.\zeta_2^2\zeta_6^2,\ \zeta_2^2\zeta_7^2,\ \zeta_2^2\zeta_8^2,\ \zeta_3^2\zeta_5^2,\ \zeta_3^2\zeta_6^2,\ \zeta_3^2\zeta_7^2,\ \zeta_3^2\zeta_8^2,\ \zeta_4^2\zeta_5^2,\ \zeta_4^2\zeta_6^2,\ \zeta_4^2\zeta_7^2,\ \zeta_4^2\zeta_8^2\right]^T, \tag{3–22}$$

where, generally, ζ_i refers to the i^{th} entry of ζ. The basis in (3–22) was used in each simulation case. A total of eight different test cases were performed.

Case 1 uses the result from [61], which requires the PE condition to be satisfied. To ensure that the PE condition is satisfied, a probing signal

$$p(t) = \begin{bmatrix} 2.55\tanh\left(20\sin\sqrt{232}\pi t\right)\cos\left(\sqrt{20}\pi t\right) \\ +6\sin\left(18e^2t\right) + 20\cos\left(40t\right)\cos\left(21t\right) \\ 0.01\tanh\left(2t\right)\left(20\sin\sqrt{132}\pi t\cos\left(\sqrt{10}\pi t\right)\right) \\ +6\sin\left(8et\right) + 20\cos\left(10t\right)\cos\left(11t\right) \end{bmatrix} \tag{3–23}$$

is added to the first 37.5 seconds of the simulation (see [61]).

Cases 2-8 are model-based and do not require a probing signal. Instead, Cases 2-8 use BE extrapolation. More specifically, Case 2 (the method in [22]) uses traditional BE extrapolation, Cases 3-7 use sparse BE extrapolation, and Case 8 uses sparse BE extrapolation and switches the extrapolation stacks.

In the R-MBRL methods, a total of 576 extrapolation points were used in each case. The extrapolation points were selected from the interval $\zeta_i \in [-1.5, 1.5]\ \forall i = 1, 2, ..., 8$. Table 3-1 lists which nodes were eliminated in each case. Case 8 uses a total of 576 unique extrapolation points. However, not all 576 points are simultaneously used in the extrapolation stack. If $|\zeta_1| \leq 0.75$ or $|\zeta_2| \leq 0.75$, then the extrapolation point was assigned to Ω_1. Otherwise it was assigned to Ω_2. Furthermore, in the assignment of extrapolation points, every sixth point was also assigned to Ω_3 (i.e., $\Omega_3 \subset (\Omega_1 \cap \Omega_2)$). Furthermore, the

$\zeta_1^2\zeta_5^2$, $\zeta_1^2\zeta_6^2$, $\zeta_1^2\zeta_7^2$, $\zeta_1^2\zeta_7^2$, $\zeta_1^2\zeta_8^2$ nodes are eliminated in Ω_1 and Ω_2, but Ω_3 uses a nonsparse basis.

Table 3-1. Simulation Case Parameters

Simulation Case	Nodes Eliminated	Extrapolation Segments (Ω_j)	Total Extrapolation Points
Case 1 ([61])	N/A	0	0
Case 2 ([48])	N/A	1	576
Case 3	$\zeta_1^2\zeta_5^2 = 0$	2	576
Case 4	$\zeta_1^2\zeta_5^2, \zeta_1^2\zeta_6^2 = 0$	2	576
Case 5	$\zeta_1^2\zeta_5^2, \zeta_1^2\zeta_6^2, \zeta_1^2\zeta_8^2 = 0$	2	576
Case 6	$\zeta_1^2\zeta_5^2, \zeta_1^2\zeta_6^2, \zeta_1^2\zeta_7^2, \zeta_1^2\zeta_8^2 = 0$	2	576
Case 7	$\zeta_1^2\zeta_5^2, \zeta_1^2\zeta_6^2, \zeta_1^2\zeta_7^2, \zeta_1^2\zeta_7^2, \zeta_1^2\zeta_8^2 = 0$	2	576
Case 8	$\zeta_1^2\zeta_5^2, \zeta_1^2\zeta_6^2, \zeta_1^2\zeta_7^2, \zeta_1^2\zeta_7^2, \zeta_1^2\zeta_8^2 = 0$	3	576

3.5.4 Two-Link Manipulator Simulation Results

The median computation time, integral of error (i.e., $\int_0^{150} \|e(\tau)\| d\tau$), 5% rise time, and root mean squared (RMS) error are shown in Table 3-2. 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 affect 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. I.e., only the computation time differed.

Table 3-2. Simulation results for the two-link robotic manipulator tracking problem.

Controller	Case 1 ([61])	Case 2 ([48])	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
Median Computation Time (s)	21.70	120.40	117.41	114.81	90.51	71.12	65.4	25.90
Integral of Error (rad·s)	54.96	33.72	32.99	23.25	25.12	27.11	26.25	27.97
5% Rise Time (s)	133.06	33.33	69.61	40.65	41.34	41.57	41.44	44.29
RMS Steady-state Error (rad)	34.29e-3	6.92e-3	11.18e-3	5.69e-3	7.86e-3	6.45e-3	6.61e-3	5.57e-3

Case 8 has the shortest computation time. By combining the switched extrapolation stacks and sparse BE, the computation time is reduced by 78.4% with respect to the non-sparse BE extrapolation method (cf., [22]).The computation time is the amount of real-world time it takes to run 150 seconds of simulation time. There is a clear

trend in the computation times of each case. Case 1 has the lowest computation time since no BE extrapolation is performed. Case 2 has the highest computation time because it performs the highest amount of BE extrapolation. As the BE extrapolation becomes more sparse (from Case 3 to Case 7) the computation time significantly decreases. Case 8 uses a switched extrapolation stack with a mix of sparse and non-sparse BE extrapolation. By decreasing the number of points in each extrapolation stack, the computation time is greatly decreased. Furthermore, by performing sparse BE extrapolation on the smaller extrapolated stacks, the computation time is further reduced. As the BE extrapolation becomes more sparse and switching extrapolations tacks occurs, the computation time significantly decreases.

The integral of error for Case 1 is high because of the probing noise. Generally, as the BE extrapolation becomes more sparse, the integral of error slightly decreases.

The 5% rise time is the amount of time it takes for the error to reach 5% of its initial value. The rise time of Case 1 is the highest because the controller uses the least amount of data. The rise time of Case 2 is the smallest because it uses the most amount of data. While there is no clear correlation for the rise time between Cases 3-8. Generally, as the BE extrapolation becomes more sparse, the rise time improves.

The RMS steady-state error is highest for Case 1 due to the probing noise. Cases 2 and 4-8 have similar RMS steady-state error. From this fact, we can conclude that the amount of sparsity has little impact on the RMS steady-state error. Figure 3-6 illustrates the performance of the ADP method in Case 2 applied to the robot manipulator.

Figure 3-7 illustrates the performance of the ADO method in Case 8 applied to the robot manipulator. Noticeably, the error decreases more steadily in Case 8, however Case 2 has faster overall convergence.

3.5.5 Euler-Lagrange System with Unknown Dynamics

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

$$u = a_1 \ddot{y} + a_2 \dot{y}, \tag{3–24}$$

where $y, \dot{y}, \ddot{y} \in \mathbb{R}^2$, is leveraged in this simulation due to the fact that an analytical solution to the HJB equation in (2–18) can be calculated. The concatenated state x is defined as $x \triangleq \left[y^T, \dot{y}^T\right]^T$. The matrix $a_1 \in \mathbb{R}^{2\times 2}$ is defined as $a_1 \triangleq \begin{bmatrix} 1 & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{5} \end{bmatrix}$, and $a_2 \in \mathbb{R}^{2\times 2}$ is defined as $a_2 \triangleq \begin{bmatrix} 1 & -1 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$. 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 (3–24) can be rewritten in the form $A \triangleq \begin{bmatrix} \mathbf{0}_{2\times 2} & I_{2\times 2} \\ \mathbf{0}_{2\times 2} & a_1^{-1} a_2 \end{bmatrix}$, $f(x) = Ax$, $g(x) = \left[\mathbf{0}_{2\times 2}^T, \left(a_1^{-1}\right)^T\right]^T$, $g^+(x_d) = [\mathbf{0}_{2\times 2}, a_1]$, $h_d(x_d) = [x_{d3}, x_{d4}, -0.25x_{d1}, -x_{d2}]^T$, which can be expressed as in (2–13), where $\zeta \triangleq \left[e^T, x_d^T\right]^T$.

To achieve the desired objective, the developed value function approximation method is used. The basis selected for value function approximation is (3–22). The drift dynamics are unknown, but are approximated using the developed system identification method. The unknown drift dynamics are approximated with the linear basis function $\phi(x) = [x_1, x_2, x_3, x_4]^T$. 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 3-3. Case 1 (i.e., [48]) 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 3-3. Simulation Case Parameters

Simulation Case	Nodes Eliminated	Extrapolation Segments (Ω_j)
Case 1 ([48])	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

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: $\Omega_1 \subset \mathbb{R}^4$ and $\Omega_2 \subset \mathbb{R}^4$, 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 subsequent Table 3-3. The active BE extrapolation stack Ω_j is selected via the policy

$$j \triangleq \begin{cases} 1 & \|e\| > 2 \\ 2 & \|e\| \leq 2 \end{cases}. \tag{3–25}$$

For each simulation case the cost parameters are $Q = [1000, 1000, 0.2, 0.2]^T \cdot I_4$ and $R = 10 \cdot I_2$, the gains are $\eta_{c1} = 0.012$, $\eta_{c2} = 0.001$, $\eta_{a1} = 0.005$, $\eta_{a2} = 0.005$, $\lambda = 0.075$, $\nu = 0.005$, $k_\theta = 100$, $\Gamma_\theta = 0.02 \cdot I_4$. $\overline{\Gamma} = 10^3$, $\underline{\Gamma} = 10$, and the initial conditions are

$\hat{W}_c(0) = 10 \cdot \mathbf{1}_{23}$, $\hat{W}_a(0) = 6 \cdot \mathbf{1}_{23}$, $\Gamma(0) = 200 \cdot I_{23}$, $\hat{\theta}(0) = \mathbf{0}_{4\times4}$,[2] $\hat{x}(0) = \mathbf{0}_4$, and $x(0) = [15, -15, 0, 0]^T$.

3.5.6 Euler-Lagrange Simulation Results

The tracking errors for Case 1 and Case 5 are compared in Figure 3-8. The purpose of Figure 3-8(a) is to show the performance of an existing, non-sparse result. To contrast Figure 3-8(a), Figure 3-8(b) presents the performance of the most sparse simulation case. These figures, when paired with Table 3-4, exhibit the benefits and drawbacks to using the techniques described in Case 1 and Case 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 [48]).

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 $\hat{V}\left(\zeta, \hat{W}_c\right)$ can be compared to the optimal value function $V^*(\zeta)$. This comparison is shown in Figure 3-9.

To examine the effects of increased sparse BE extrapolation, data was collected from each simulation case to facilitate a quantitative comparison. 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 affect 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. The median computation time, integral of

[2] From (3–24), $\theta = A^T$.

error (i.e., $\int_0^{120} \|e(\tau)\| d\tau$), 5% rise time, and RMS error for each case are shown in Table 3-4.

Table 3-4. Simulation results for simulation Cases 1-5.

Controller	Case 1 ([48])	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

The computation time is the amount of real-world time it takes to run 120 seconds 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 non-sparse BE extrapolation method in [48]. As the BE extrapolation becomes more sparse (from Case 2 to Case 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., $\|e(t)\| \leq 0.05 \cdot \|e(0)\|$). 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. 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.

3.5.7 Ease of Sparsification

As the processing power of modern computing platforms increases, the need for computational efficiency gradually decreases. For the industrial application of the developed sparse BE extrapolation, one may be required to make a decision to either purchase additional man-hours for a designer to implement this technique or to purchase more sophisticated hardware. Like many product implementation decisions, there is a break-even point (or possibly many points) at which it may be more cost-effective to purchase upgraded computing hardware instead of paying a designer. While there are many parameters used to determine the most cost-effective solution, the general trend is not clear. However, once a designer is familiar with the developed technique, modifying [48] to include sparse BE extrapolation should take less than one man-hour.

In many applications it is not possible to use a faster processor due to architecture (PC104, Controller Area Network bus) restrictions, qualifications of the processor (e.g., for military or space applications, or processing power demands. In such applications, the developed sparse BE extrapolation method will be beneficial because the reduced computational load reserves computational capabilities for other processes.

3.6 Concluding Remarks

In this chapter, 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. Using SNNs in BE extrapolation yields

computational benefits due to the smaller number of active neurons. 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. One simulation example for a two-state dynamical system 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. A third simulation example is used to show that using sparse, switched BE extrapolation reduces the computation time by 85.6% when compared to the method in [48].

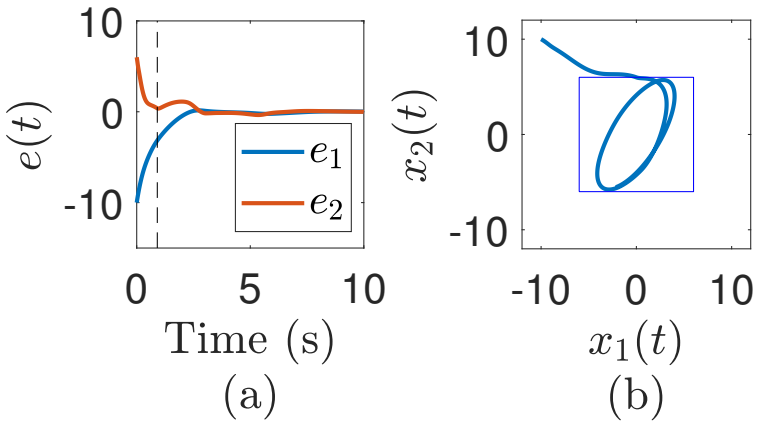


Figure 3-1. Error trajectories and state space portrait of the two-state simulation example. Figure (a) shows the error trajectories. Figure (b) shows the state space portrait for the two-state dynamical system described in (3–14). In figure (b) the blue boxed area represents where the Ω_2 segment is active, whereas the visible area outside of the box represent where the Ω_1 segment is active.

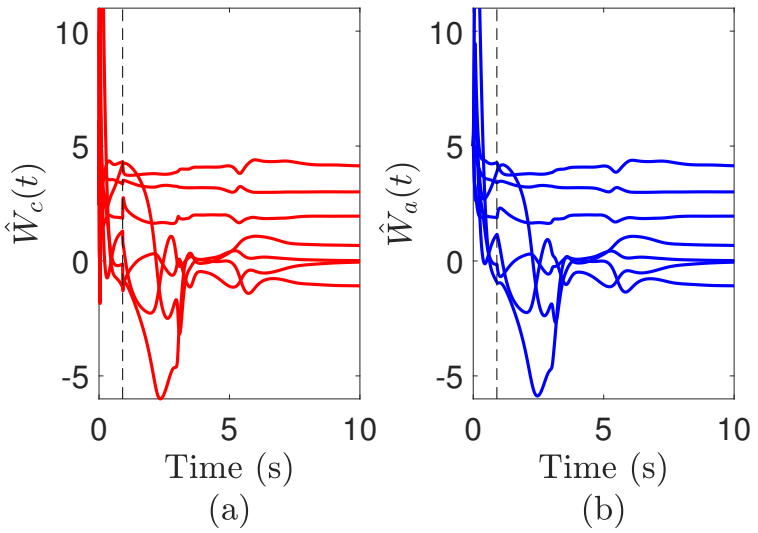


Figure 3-2. . Critic and actor weight values of the two-state simulation example. Figure (a) shows the trajectories of the critic weights $\hat{W}_c(t)$. Figure (b) shows the trajectories of actor weights $\hat{W}_a(t)$. The dashed vertical line indicates the time at which the extrapolation stack is switched from Ω_1 to Ω_2.

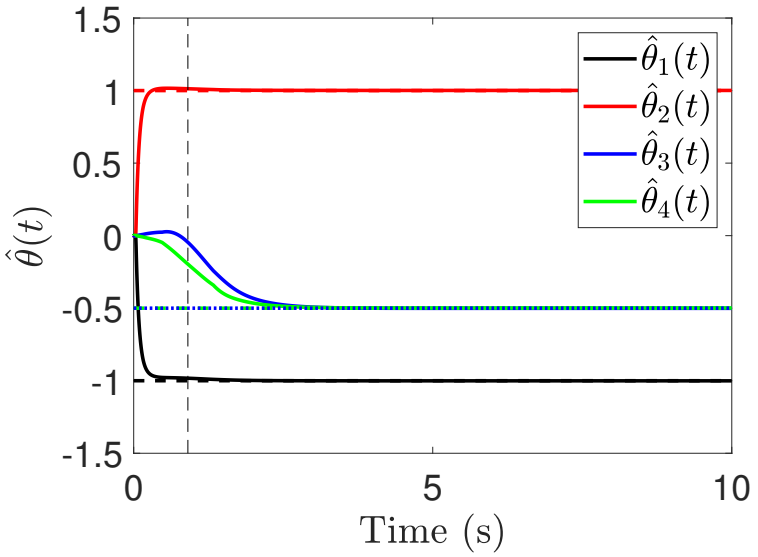


Figure 3-3. Estimate of the drift dynamics. This figure illustrates that system identification is completed around the 4 second mark.

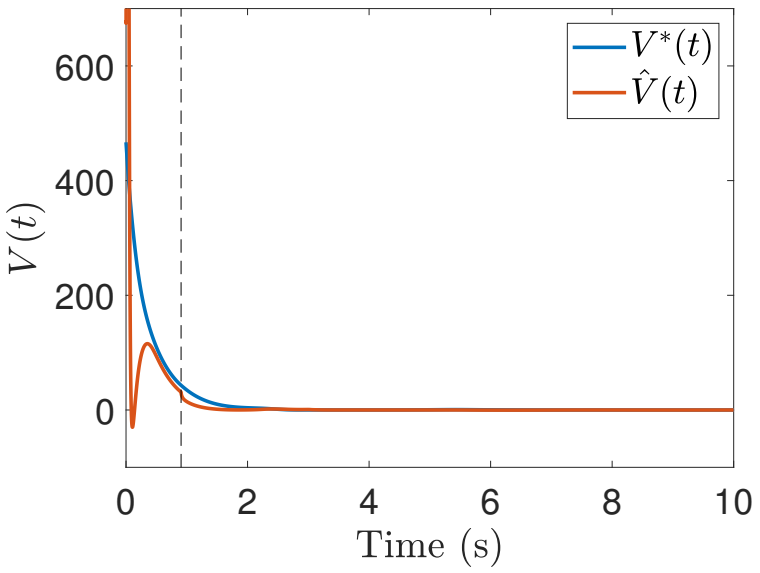


Figure 3-4. Comparison of the optimal value function $V^*(\zeta)$ to the approximated optimal value function $\hat{V}\left(\zeta, \hat{W}_c\right)$.

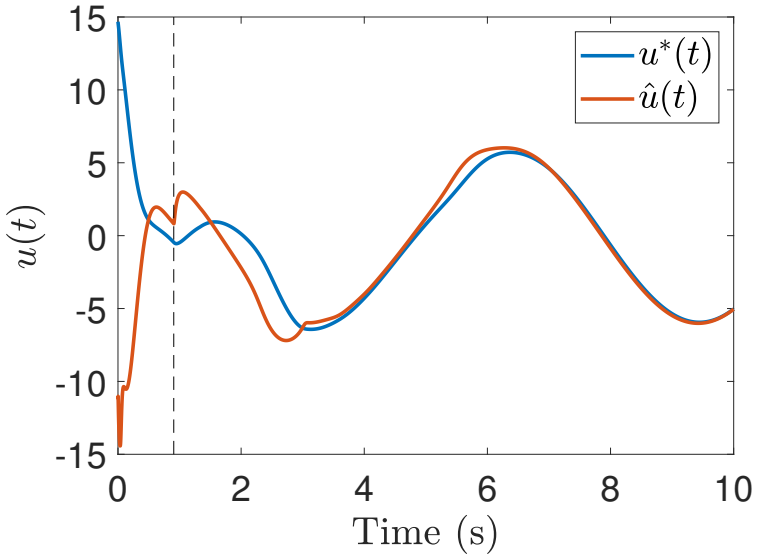


Figure 3-5. Comparison of the optimal value function $u^*(\zeta)$ to the approximated optimal value function $\hat{u}\left(\zeta, \hat{W}_a\right)$.

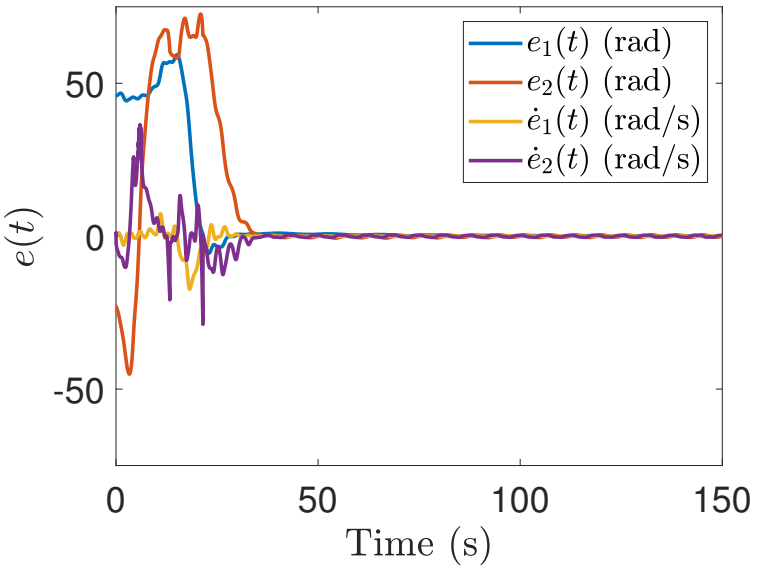


Figure 3-6. Errors $e_1,\ e_2, \dot{e}_1,\ \dot{e}_2$ for Case 2.

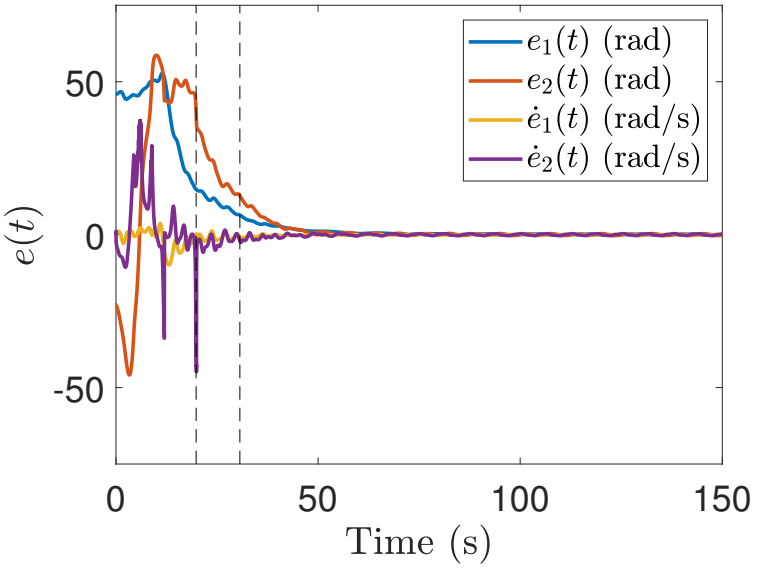


Figure 3-7. Errors $e_1,\ e_2, \dot{e}_1,\ \dot{e}_2$ for Case 8. The dashed vertical lines indicate the time at which the extrapolation stack is switched from Ω_1 to Ω_2 to Ω_3, respectively.

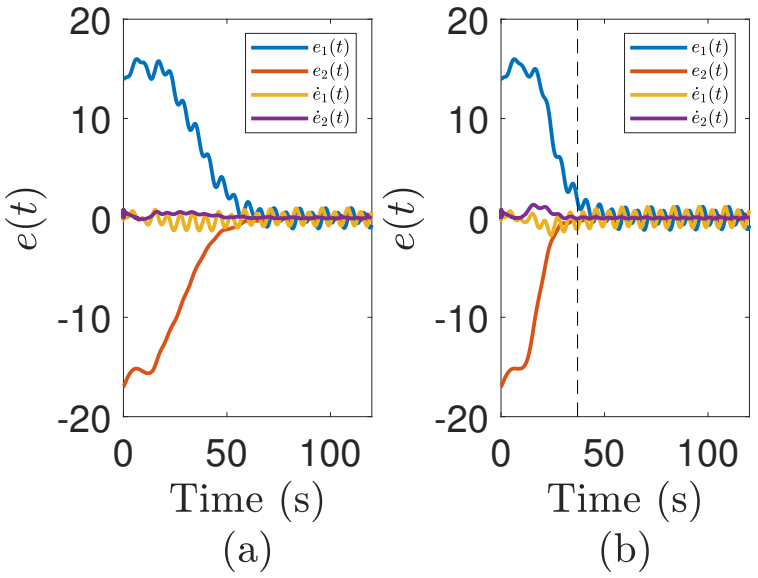


Figure 3-8. Error signals e_1, e_2, $\dot{e}_1$, $\dot{e}_2$ for Case 1 and Case 5 as outlined in Table 3-4. Figure (a) shows the errors for Case 1. Figure (b) shows the errors for Case 5. The vertical line represents the time at which the system switched BE extrapolation data stacks due to (3–25). Case 5 has faster convergence than Case 1; however, Case 1 has a smaller steady-state error.

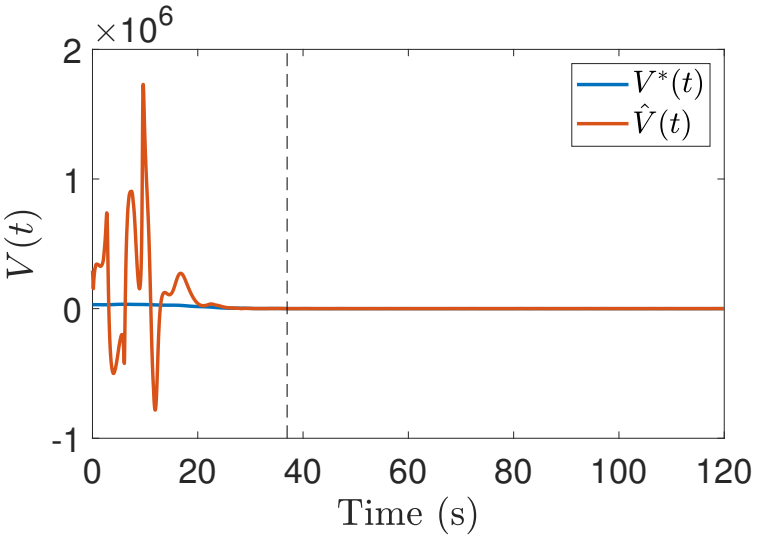


Figure 3-9. Comparison of the optimal value function $V^*(\zeta)$ to the approximated optimal value function $\hat{V}\left(\zeta, \hat{W}_c\right)$ for Case 5. The vertical line represents the time at which the system switched BE extrapolation data stacks due to (3–25).

CHAPTER 4
SPARSE LEARNING-BASED APPROXIMATE DYNAMIC PROGRAMMING WITH BARRIER CONSTRAINTS

This chapter and the result in [64] provide an approximate online adaptive solution to the infinite-horizon optimal control problem for control-affine continuous-time nonlinear systems while formalizing system safety using barrier certificates. The use of a BF transform provides safety certificates to formalize system behavior. Specifically, using a BF, the system is transformed to aid in developing a controller which maintains the system state in a user-defined constrained region. To aid in online learning of the value function, the state-space is segmented into a number of user-defined segments. Off-policy trajectories are selected in each segment, and sparse BE extrapolation is performed within each respective segment to generate an optimal policy within each segment; the process of state space segmentation for BE extrapolation is detailed in Chapter 3. A Lyapunov-like stability analysis is included which proves UUB regulation in the presence of the BF transform and discontinuities. Simulation results are provided for a two-state dynamical system to compare the performance of the developed method to existing methods.

4.1 Barrier Functions

Consider the continuous-time control-affine nonlinear dynamical system

$$\dot{x} = f(x) + g(x)\,u \tag{4–1}$$

with initial condition $x(0) = x_0 \in \mathbb{R}^n$, where $x \in \mathbb{R}^n$ denotes the system state, $u \in \mathbb{R}^m$ denotes the control input, $f : \mathbb{R}^n \to \mathbb{R}^n$ denotes the drift dynamics, and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ is the control effectiveness. The goal is to design a control policy u for the system in (4–1) while regulating the system state, $x = [x_1, \ldots, x_n]^T$, to the origin while also ensuring the states lie within distinct user-specified sets (i.e., within the user-defined barriers) such that

$$x_i(t) \in (a_i, A_i) \; \forall \, i = 1, \ldots, n \text{ and } t \in \mathbb{R}_{\geq 0}, \tag{4–2}$$

where $a \in \mathbb{R}^n$ and $A \in \mathbb{R}^n$ represent the vectors of all lower and upper bounds of the sets, respectively, with $a_i \in \mathbb{R}$ and $A_i \in \mathbb{R}$ being the i^{th} row of a and A, respectively, where $i \in \{1, \ldots, n\}$. Let a logarithmic BF, $b : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, be defined as

$$b(z_i, a_i, A_i) \triangleq \ln\left(\frac{A_i}{a_i}\frac{a_i - z_i}{A_i - z_i}\right), \forall \, z_i \in (a_i, A_i), \tag{4–3}$$

such that the constants a_i and A_i satisfy $a_i < 0 < A_i$, $z_i \in \mathbb{R}$, and the inverse of the BF in (4–3), is

$$b^{-1}(z_i, a_i, A_i) = a_i A_i \frac{e^{z_i} - 1}{e^{z_i} a_i - A_i}. \tag{4–4}$$

The logarithmic BF in (4–4) is as a tool to ensure the system remains within the user-defined barriers. To this end, the derivative of (4–4) is taken with respect to z_i to yield

$$\frac{db^{-1}(z_i, a_i, A_i)}{dz_i} = \frac{a_i^2 A_i - a_i A_i^2}{a_i^2 e^{z_i} - 2A_i a_i + A_i^2 e^{-z_i}}. \tag{4–5}$$

Let $s_i \in \mathbb{R}$ be the state-space to barrier-space coordinate transformed state such that

$$s_i = b(x_i, a_i, A_{,i}). \tag{4–6}$$

Using (4–4), the transformation from the barrier-space to the state-space is

$$x_i = b^{-1}(s_i, a_i, A_{,i}), \tag{4–7}$$

Taking the time-derivative of (4–7) and rearranging yields

$$\dot{s}_i = \frac{(a_i^2 e^{s_i} - 2A_i a_i + A_i^2 e^{-s_i})}{a_i^2 A_i - a_i A_i^2} \dot{x}_i. \tag{4–8}$$

Using (4–1) in (4–8) results in the transformed state

$$\dot{s}_i = F_i\left(s_i, a_i, A_i\right) + G_i\left(s_i, a_i, A_i\right) u\left(t\right), \tag{4–9}$$

where

$$F_i\left(s_i, a_i, A_i\right) \triangleq \left(\frac{a_i^2 e^{s_i} - 2A_i a_i + A_i^2 e^{-s_i}}{a_n^2 A_n - a_n A_n^2}\right) \cdot f_i\left(b^{-1}\left(s_i, a_i, A_{,i}\right)\right), \tag{4–10}$$

$$G_i\left(s_i, a_i, A_i\right) \triangleq \left(\frac{a_i^2 e^{s_i} - 2A_i a_i + A_i^2 e^{-s_i}}{a_n^2 A_n - a_n A_n^2}\right) \cdot g_i\left(b^{-1}\left(s_i, a_i, A_{,i}\right)\right), \tag{4–11}$$

and $b^{-1}\left(s_i, a_i, A_{,i}\right)$ with $f_i : \mathbb{R} \to \mathbb{R}$ and $g_i : \mathbb{R} \to \mathbb{R}^{1\times m}$ being the i^{th} row of the functions f and g in (4–1), respectively. The transformed states $s \triangleq [s_1, \ldots, s_n]^T \in \mathbb{R}^n$ can be written using (4–8) in a compact form as

$$\dot{s} = F\left(s\right) + G\left(s\right) u\left(t\right), \tag{4–12}$$

where $F\left(s\right) \triangleq \left[F_1\left(s_1, a_1, A_1\right), \ldots, F_n\left(s_n, a_n, A_n\right)\right]^T$ and $G\left(s\right) \triangleq \left[G_1\left(s_1, a_1, A_1\right), \ldots, G_n\left(s_n, a_n, A_n\right)\right]^T$.

The drift dynamics F is assumed to be a locally Lipschitz function with $F\left(0\right) = 0$, where $\nabla_x F : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is continuous. There exists a constant b_f, such that for $s \in \Omega$, $\left\|F\left(s\right)\right\| \leq b_f \left\|s\right\|$, where $\Omega \subset \mathbb{R}^n$ is a compact set containing the origin. The system is assumed to be controllable over the compact set Ω, and the control effectiveness, G, is assumed to be a locally Lipschitz function and bounded such that $0 < \left\|G\left(x\right)\right\| \leq \overline{G}$, where $\overline{G} \in \mathbb{R}_{\geq 0}$.

4.2 Approximate Optimal Controller Development

The development in this section follows that in Chapter 2, however, this section presents the ADP problem formulation for the barrier states in (4–6). The control objective is to solve the infinite-horizon optimal regulation problem, i.e., determine a control policy, u, that minimizes the infinite horizon cost function, $J : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$,

defined as

$$J(s,u) \triangleq \int_{t_0}^{\infty} r(s(\tau), u(\tau))\, d\tau, \tag{4–13}$$

subject to (4–12) while regulating the system states to the origin (i.e., $s = \mathbf{0}_n$), where $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ is the instantaneous cost defined as $r(s,u) \triangleq s^T Q s + u^T R u$, $Q \in \mathbb{R}^{n \times n}$ is a constant user-defined symmetric PD matrix, and $R \in \mathbb{R}^{m \times m}$ is a constant user-defined PD symmetric matrix.[1]

Remark 4.1. The state cost matrix, Q, satisfies $\underline{q} I_n \leq Q \leq \overline{q} I_n$ where $\underline{q}, \overline{q} \in \mathbb{R}_{>0}$.

The infinite horizon value function (i.e, the cost to go) for the optimal solution is denoted by $V^* : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ and given by

$$V^*(s) = \min_{u(\tau) \in U,\, \tau \in \mathbb{R}_{\geq t}} \int_t^{\infty} r(s(\tau), u(\tau))\, d\tau, \tag{4–14}$$

where $U \subseteq \mathbb{R}^m$ denotes the action space. Provided and optimal control policy exists, the value function is characterized by the corresponding HJB

$$0 = \min_{u(\tau) \in U} \left(\nabla_s V^*(s) \left(F(s) + G(s) u \right) + s^T Q s + u^T R u \right), \tag{4–15}$$

with the boundary condition $V^*(0) = 0$. Provided the HJB in (4–15) admits a continuously differentiable PD solution, then the optimal closed-loop control policy $u^* : \mathbb{R}^n \to \mathbb{R}^m$ is

$$u^*(s) = -\frac{1}{2} R^{-1} G(s)^T \left(\nabla_s V^*(s) \right)^T. \tag{4–16}$$

4.2.1 Value Function Approximation

The HJB in (4–15) requires knowledge of the optimal value function, which, generally, is an unknown function for nonlinear systems. Parametric methods can be used

[1] As in Chapter 2, the matrix PD Q can be replaced with a PD function $Q(s)$.

to approximate the value function over a compact domain. To facilitate the solution of (4–15), let $\Omega \subset \mathbb{R}^n$ be a compact set containing the origin with $s \in \Omega$. The universal function approximation property of single-layer NNs is used to parameterize the value function V^* as

$$V^*(s) = W^{*T}\sigma(s) + \epsilon(s), \tag{4–17}$$

where $W^* \in \mathbb{R}^L$ is an unknown bounded weight, $\sigma : \mathbb{R}^n \to \mathbb{R}^L$ is a user-defined vector of basis functions, and $\epsilon : \mathbb{R}^n \to \mathbb{R}$ is the bounded function approximation error. Solving (4–15) for u and using (4–17), the approximate optimal control policy u^* can be expressed in terms of the gradient of the value function V^* as

$$u^*(s) = -\frac{1}{2}R^{-1}G(s)\left(\nabla_s\sigma(s)^T W^* + \nabla_s\epsilon(s)^T\right). \tag{4–18}$$

Property 4. There exists a set of constants that upper bound the unknown weight vector, W^*, the user-defined basis vector, σ, and approximation error, ϵ, such that $\|W^*\| \leq \overline{W^*}$, $\sup_{s\in\Omega}\|\sigma(s)\| \leq \overline{\sigma}$, $\sup_{s\in\Omega}\|\nabla_s\sigma(s)\| \leq \overline{\nabla_s\sigma}$, $\sup_{s\in\Omega}\|\epsilon(s)\| \leq \overline{\epsilon}$, $\sup_{s\in\Omega}\|\nabla_s\epsilon(s)\| \leq \overline{\nabla_s\epsilon}$, where $\overline{W^*}$, $\overline{\sigma}$, $\overline{\nabla_s\sigma}$, $\overline{\epsilon}$, $\overline{\nabla_s\epsilon} \in \mathbb{R}_{>0}$ [60].

Since the ideal weights are unknown, a parametric estimate, called a critic weight, $\hat{W}_c \in \mathbb{R}^L$, is substituted to estimate the optimal value function, $\hat{V} : \mathbb{R}^n \times \mathbb{R}^L \to \mathbb{R}$ where

$$\hat{V}\left(s, \hat{W}_c\right) = \hat{W}_c^T\sigma(s). \tag{4–19}$$

An actor weight estimate, $\hat{W}_a \in \mathbb{R}^L$, is used to provide an estimated version of (4–18), $\hat{u} : \mathbb{R}^n \times \mathbb{R}^L \to \mathbb{R}^m$, given by

$$\hat{u}\left(s, \hat{W}_a\right) = -\frac{1}{2}R^{-1}G(s)^T\left(\nabla_s\sigma(s)^T\hat{W}_a\right). \tag{4–20}$$

4.2.2 Bellman Error

The HJB in (4–15) is equal to zero under optimal conditions; however, substituting (4–19) and (4–20) into (4–15) results in a residual term, $\delta : \mathbb{R}^n \times \mathbb{R}^L \times \mathbb{R}^L \to \mathbb{R}$, which is

referred to as the BE, defined as

$$\delta\left(s,\hat{W}_c,\hat{W}_a\right) \triangleq \nabla_s \hat{V}\left(s,\hat{W}_c\right)\left(F\left(s\right)+G\left(s\right)\hat{u}\left(s,\hat{W}_a\right)\right)+r\left(s,\hat{u}\left(s,\hat{W}_a\right)\right) \tag{4–21}$$

where $\nabla_s \hat{V}\left(s,\hat{W}_c\right) = \hat{W}_c^T \nabla_s \sigma\left(s\right)$ denotes the gradient of the value function estimate. The BE is indicative of how close the actor and critic weight estimates are to the ideal weights. By defining the mismatch between the estimates and the ideal values as $\tilde{W}_c \triangleq W - \hat{W}_c$ and $\tilde{W}_a \triangleq W - \hat{W}_a$, substituting (4–19) and (4–20) in (4–15), and subtracting from (4–21) yields

$$\delta = \frac{1}{4}\tilde{W}_a^T G_\sigma \tilde{W}_a - \omega^T \tilde{W}_c + O\left(s\right), \tag{4–22}$$

where $\omega : \mathbb{R}^n \times \mathbb{R}^L \to \mathbb{R}^n$ is defined as

$$\omega\left(s,\hat{W}_a\right) \triangleq \nabla_s \sigma\left(s\right)\left(F\left(s\right)+G\left(s\right)\hat{u}\left(s,\hat{W}_a\right)\right), \tag{4–23}$$

and $O\left(s\right) \triangleq \frac{1}{2}W^{*T}\nabla_s \sigma\left(s\right) G_R \nabla_s \epsilon\left(s\right)^T + \frac{1}{4}G_\varepsilon - \nabla_s \epsilon\left(s\right) F$. The notation G_R, G_σ, and G_ε is defined as $G_R = G_R\left(s\right) \triangleq G\left(s\right) R^{-1} G\left(s\right)^T$, $G_\sigma = G_\sigma\left(s\right) \triangleq \nabla_s \sigma\left(s\right) G_R\left(s\right) \nabla_s \sigma\left(s\right)^T$, and $G_\varepsilon = G_\varepsilon\left(s\right) \triangleq \nabla_s \epsilon\left(s\right) G\left(s\right) \nabla_s \epsilon\left(s\right)^T$, respectively.

4.2.3 Sparse Bellman Error Extrapolation

At each time instant, the BE in (4–21) is calculated using the control policy given by (4–18) evaluated using the current system state, critic weight estimates, and actor weight estimates to obtain the instantaneous BE denoted by $\delta\left(t\right) \triangleq \delta\left(s\left(t\right),\hat{W}_c\left(t\right),\hat{W}_a\left(t\right)\right)$ and control policy denoted by $u\left(t\right) \triangleq \hat{u}\left(s\left(t\right),\hat{W}_a\left(t\right)\right)$. Our previous work in [33] explored using the computational efficiency of SNNs and segmentation to extrapolate the BE, so that the BE can be active across the active state-space, thereby relaxing the traditional PE condition. The benefit to performing BE extrapolation across multiple segments is that the process can be performed in parallel. Since SNNs are more efficient than traditional full-weight neurons in this application, BE extrapolation within multiple segments can be computed simultaneously.

To relax the strictness of the PE condition, virtual excitation using BE extrapolation is performed. The state-space is divided into a user-specified number of segments. Let the operating domain Ω be a partition 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 $\Omega = \bigcup_{j=1}^{S} \Omega_{j.}$

Each segment is assigned a user-specified number and location of off-trajectory points, $\{s_{i,j} : s_{i,j} \in \Omega_j\}_{i=1}^{N_j}$, where $N_j \in \mathbb{N}$ denotes the user-specified number of points in the segment Ω_j, and $x_{i,j} = b^{-1}(s_{i,j}, a_i, A_i)$. Using the extrapolated barrier-space trajectories, $s_{i,j}$ for a given $j \in S$, the tuple $\left(\Sigma_c^j, \Sigma_a^j, \Sigma_\Gamma^j\right)$ is defined as the history stack corresponding to Ω_j where $\Sigma_c^j \triangleq \frac{1}{N_j}\sum_{i=1}^{N_j} \frac{\omega_{i,j}(t)}{\rho_{i,j}(t)} \delta_{i,j}(t)$, $\Sigma_a^j \triangleq \frac{1}{N_j}\sum_{i=1}^{N_j} \frac{G_{\sigma i,j}^T \hat{W}_a(t) \omega_{i,j}^T(t)}{4\rho_{i,j}(t)}$, $\Sigma_\Gamma^j \triangleq \frac{1}{N_j}\sum_{i=1}^{N_j} \frac{\omega_{i,j}(t)\omega_{i,j}^T(t)}{\rho_{i,j}(t)}$, $\omega_{i,j}(t) \triangleq \omega\left(s_{i,j}, \hat{W}_a\right) = \nabla_s \sigma(s_{i,j}) \left(F(s_{i,j}) + G(s_{i,j}) \hat{u}\left(s_{i,j}, \hat{W}_a\right)\right)$, $\rho_{i,j}(t) = 1 + \nu \omega_{i,j}^T(t) \Gamma(t) \omega_{i,j}(t)$, and $\nu \in \mathbb{R}_{>0}$ are user-defined gains. Recall, the notation $G_R = G(s) R^{-1} G(s)^T$, $G_\sigma = G_\sigma(s) \triangleq \nabla_s \sigma(s) G_R(s) \nabla_s \sigma(s)^T$, and $G_\varepsilon = G_\varepsilon(s) \triangleq \nabla_s \epsilon(s) G(s) \nabla_s \epsilon(s)^T$

Remark 4.2. BE extrapolation is performed in the barrier-space since function approximation is taken in a compact set over the barrier-space.

Assumption 4.1. Over each segment $j \in S$, there exists a finite set of trajectories $\{s_{i,j} : s_{i,j} \in \Omega_j\}_{i=1}^{N_j}$ such that $0 < \underline{c} \triangleq \inf_{t \in \mathbb{R}_{\geq 0},\, j \in S} \lambda_{\min}\left\{\Sigma_\Gamma^j\right\}$ for all $t \in \mathbb{R}_{\geq 0}$.

Remark 4.3. The constant $\underline{c}$ is the lower bound of the value of each input-output data pair's minimum eigenvalues.

4.2.4 Update Laws for Actor and Critic Weights

Using the instantaneous BE $\delta(t)$, policy $u(t)$, and extrapolated BEs $\delta_{i,j}(t)$, the critic and actor weights are updated according to

$$\dot{\hat{W}}_c(t) = -\eta_{c1} \Gamma \frac{\omega(t)}{\rho(t)} \delta(t) - \eta_{c2} \Sigma_c^j(t), \tag{4–24}$$

$$\dot{\Gamma}(t) = \left(\lambda \Gamma(t) - \eta_{c1} \frac{\Gamma(t) \omega(t) \omega(t)^T \Gamma(t)}{\rho(t)} - \Gamma(t) \eta_{c2} \Sigma_\Gamma^j \Gamma(t)\right) \mathbf{1}_{\left\{\underline{\Gamma} \leq \|\Gamma\| \leq \overline{\Gamma}\right\}}, \tag{4–25}$$

$$\dot{\hat{W}}_a(t) = -\eta_{a1}\left(\hat{W}_a(t) - \hat{W}_c(t)\right) - \eta_{a2}\hat{W}_a(t) + \frac{\eta_{c1} G_\sigma^T(t)\hat{W}_a(t)\omega^T(t)}{4\rho(t)}\hat{W}_c(t) + \eta_{c2}\Sigma_a^j \hat{W}_c(t), \tag{4–26}$$

where $\eta_{c1}, \eta_{c2}, \eta_{a1}, \eta_{a2}, \lambda$ are positive constant learning gains, $\overline{\Gamma}, \underline{\Gamma} \in \mathbb{R}_{>0}$ are upper and lower bound saturation constants, and $\mathbf{1}_{\{\cdot\}}$ denotes the indicator function. $\|\Gamma(t)\|$ is upper and lower bounded by user-defined saturation constants, $\overline{\Gamma}$ and $\underline{\Gamma}$, respectively. Using (4–25) ensures that $\underline{\Gamma} \leq \|\Gamma(t)\| \leq \overline{\Gamma}$ for all $t \in \mathbb{R}_{>0}$. The indicator function in (4–25) can be removed with additional assumptions and modifications outlined in [22].

4.3 Stability Analysis

To facilitate the analysis, the notation $\overline{(\cdot)}$ is defined as $\overline{(\cdot)} \triangleq \sup_{x\in\Omega}(\cdot)$. Let $r \triangleq \left[s^T, \tilde{W}_c^T, \tilde{W}_a^T\right]^T$ denote a concatenated state, and let $V_L : \mathbb{R}^{n+2L} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ be a candidate Lyapunov function defined as

$$V_L(r,t) = V^*(s) + \frac{1}{2}\tilde{W}_c^T\Gamma^{-1}(t)\tilde{W}_c + \frac{1}{2}\tilde{W}_a^T\tilde{W}_a, \tag{4–27}$$

which, using the positive definiteness of V^* and [67, Lemma 4.3], can be bounded as $\underline{v}_l(\|r\|) \leq V_L(r,t) \leq \overline{v}_l(\|r\|)$ for class $\mathcal{K}_\infty$ functions $\underline{v}_l, \overline{v}_l : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Using (4–25), the normalized regressors $\frac{\omega}{\rho}$ and $\frac{\omega_{i,j}}{\rho_{i,j}}$ can be bounded as $\sup_{t\in\mathbb{R}_{\geq 0}}\left\|\frac{\omega}{\rho}\right\| \leq \frac{1}{2\sqrt{\nu\underline{\Gamma}}}$ for all $s \in \Omega$ and $\sup_{t\in\mathbb{R}_{\geq 0}}\left\|\frac{\omega_{i,j}}{\rho_{i,j}}\right\| \leq \frac{1}{2\sqrt{\nu\underline{\Gamma}}}$ for all $s_i \in \Omega_j$ for all $j \in \mathbb{S}$. The matrices G_R and G_σ can be bounded as $\sup_{s\in\Omega}\|G_R\| \leq \lambda_{\max}\{R^{-1}\}\overline{G}^2$ and $\sup_{s\in\Omega}\|G_\sigma\| \leq \left(\overline{\nabla_s \sigma G}\right)^2 \lambda_{\max}\{R^{-1}\}$.

Theorem 4.1. *Provided the class of dynamics in (4–12), Assumption 4.1, and the sufficient gain conditions*

$$\eta_{a1} + \eta_{a2} > \frac{1}{\sqrt{\nu\underline{\Gamma}}}(\eta_{c1} + \eta_{c2})\overline{W^* G_\sigma}, \tag{4–28}$$

$$\underline{c} > \frac{3\eta_{a1}}{\eta_{c2}} + \frac{3(\eta_{c1} + \eta_{c2})^2 \overline{W^*}^2 \overline{G_\sigma}^2}{16\nu\underline{\Gamma}\eta_{c2}(\eta_{a1} + \eta_{a2})}, \tag{4–29}$$

$$v_l^{-1}(l) < \overline{v}_l^{-1}\left(\underline{v}_l(r)\right), \tag{4–30}$$

hold, then the system state $s(t)$ *in* (4–6)*, weight estimation errors* $\tilde{W}_c(t)$ *and* $\tilde{W}_a(t)$*, and policy* $u(t)$ *are UUB.*

Proof. 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 [68] and $h : \mathbb{R}^{n+2L+L^2} \to \mathbb{R}^{n+2L+L^2}$ is defined as $h \triangleq \left[\dot{s}^T, \dot{\tilde{W}}_c^T, \dot{\tilde{W}}_a^T, \text{vec}\left(\dot{\Gamma}^{-1}\right)^T\right]^T$. Due to the discontinuity in the update laws in (4–24)-(4–26), the time derivative of (4–27) exists almost everywhere and $\dot{V}_L(z) \overset{a.e.}{\in} \dot{\tilde{V}}_L(z)$, where $\dot{\tilde{V}}_L$ is the generalized time-derivative of (4–27) along the Filippov trajectories of $\dot{z} = h(z)$ [69]. Using the calculus of $K[\cdot]$ from [69], and $\dot{V}^*(s) = \nabla_s V^*(s)(F(s) + G(s)u(t))$, then substituting (4–24)-(4–26) yields

$$\begin{aligned}\dot{\tilde{V}}_L \subseteq{}& \nabla_s V^*(F+Gu) - \tilde{W}_c^T\Gamma^{-1}\left(-\eta_{c1}\Gamma\frac{\omega}{\rho}\hat{\delta} - \eta_{c2}K\left[\Sigma_c^j\right]\right)\\ &- \frac{1}{2}\tilde{W}_c^T\Gamma^{-1}\left(\lambda\Gamma - \eta_{c1}\frac{\Gamma\omega\omega^T\Gamma}{\rho} - \Gamma\eta_{c2}K\left[\Sigma_\Gamma^j\right]\Gamma\right)\Gamma^{-1}\tilde{W}_c\\ &- \tilde{W}_a^T\left(-\eta_{a1}\left(\hat{W}_a(t) - \hat{W}_c(t)\right) - \eta_{a2}\hat{W}_a(t)\right)\\ &- \tilde{W}_a^T\left(\frac{\eta_{c1}G_\sigma^T(t)\hat{W}_a(t)\omega^T(t)}{4\rho(t)}\hat{W}_c(t) + \eta_{c2}K\left[\Sigma_a^j\right]\hat{W}_c\right).\end{aligned} \tag{4–31}$$

Using the class of dynamics in (4–12), Assumption 4.1, and substituting the sufficient conditions in (4–28) and (4–29) yields

$$\dot{V}_L \overset{a.e.}{\leq} -v_l(\|r\|),\ \forall \|r\| \geq v_l^{-1}(l), \forall t \in \mathbb{R}_{\geq 0}, \tag{4–32}$$

where

$$v_l(\|r\|) \leq \frac{\underline{q}\|x\|^2}{2} + \frac{(\eta_{a1}+\eta_{a2})}{16}\left\|\tilde{W}_a\right\|^2 + \frac{\eta_{c2}\underline{c}}{12}\left\|\tilde{W}_c\right\|^2, \tag{4–33}$$

where l is a known positive constant. Since (4–27) is a common Lyapunov-like function across each segment $j \in \mathbb{S}$, [67, Theorem 4.18] can be invoked to conclude that r is UUB such that $\limsup_{t\to\infty}\|r\| \leq \underline{v}_l^{-1}\left(\overline{v}_l\left(v_l^{-1}(l)\right)\right)$. Since $r \in \mathcal{L}_\infty$, it follows that $s, \tilde{W}_c, \tilde{W}_a \in \mathcal{L}_\infty$,$\hat{W}_c, \hat{W}_a \in \mathcal{L}_\infty$, and $u \in \mathcal{L}_\infty$. Moreover, if $s \in \mathcal{L}_\infty$, by (4–4) $x \in \mathcal{L}_\infty$ and satisfies $x_i \in (a_i, A_i)$ for each $i \in \{1, \cdots, n\}$. □

Remark 4.4. See Remark 3.4 for details on satisfying Assumption 4.1.

4.4 Simulation Results

To demonstrate the performance of the developed method for a nonlinear system, simulation results are performed for the two-state dynamical system described in [70]. The simulation is performed with the system given in (4–1) where $x = [x_1, x_2]^T$,

$$f(x) = \begin{bmatrix} -x_1 + x_2 \\ -\frac{1}{2}x_1 - \frac{1}{2}x_2\left(1 - (\cos(2x_1) + 2)^2\right) \end{bmatrix}, \tag{4–34}$$

and

$$g(x) = \begin{bmatrix} 0 \\ \cos(2x_1) + 2 \end{bmatrix}. \tag{4–35}$$

The control objective is to minimize (4–13), where $Q = \text{diag}(0.01, 0.1)$ and $R = 0.1$. To approximate the value function, a polynomial basis is selected as $\sigma(s) = [s_1^2, s_1 s_2, s_2^2]^T$. The barrier sets as defined in (4–2) are $x_1 \in (-5.25, 0.25)$ and $x_2 \in (-0.25, 5.25)$. To facilitate the sparse BE extrapolation, two segments are selected as $\Omega_1 \subset \mathbb{R}^2$ and $\Omega_2 \subset \mathbb{R}^2$ where

$$\begin{aligned} \Omega_1 \triangleq \{ s \in \mathbb{R}^2 : b(-5.25, -5.25, 0.25) < s_1 < b(-3.5, -5.25, 0.25), \\ b(3.5, -0.25, 5.25) < s_2 < b(5.25, -0.25, 5.25) \} \end{aligned} \tag{4–36}$$

and

$$\begin{aligned} \Omega_2 \triangleq \{ s \in \mathbb{R}^2 : b(-3.5, -5.25, 0.25) < s_1 < b(0.25, -5.25, 0.25), \\ b(-0.25, -0.25, 5.25) < s_2 < b(3.5, -0.25, 5.25) \}. \end{aligned} \tag{4–37}$$

The basis used over segment 1 is $\sigma_i(s_i) = \left[s_{1,i}^2, s_{1,i}s_{2,i}, s_{2,i}^2\right]^T$ for all $s_i \in \Omega_1$ with $_1 = 16$ extrapolated trajectories, while in segment 2 the second element is turned off such that $\sigma_i(s_i) = \left[s_{1,i}^2, 0, s_{2,i}^2\right]^T$ for all $s_i \in \Omega_2$ with $N_2 = 144$ extrapolated trajectories used for

BE extrapolation. The initial conditions for the system (i.e., $t = 0$) are $x(0) = [-5, 5]^T$, $\hat{W}_a(0) = \hat{W}_c(0) = \left[\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right]^T$, $\Gamma(0) = 250 \times I_3$. The gains were selected as $\eta_{c1} = 0.001$, $\eta_{c2} = 5$, $\lambda = 0.5$, $\eta_{a1} = 25$, $\eta_{a2} = 0.1$, $\nu = 0.005$.

Figure 4-1 shows that the state converge to the origin while staying within the user-specified barriers. System parameters were selected to show the impact of the barriers on the system. As shown, as the state nears $x_1(t) = 0$, but does not cross over the boundary at $A_1 = 0.25$. By design, the state trajectory comes close to the barrier without intersecting it. As $x_2(t)$ approaches a_2, the controller forces the system trajectory away from the boundary, and toward the origin. The developed method and the method from [38] obey the barrier constraints, whereas the methods in [33] and [53] do not. Figure 4-2 illustrates the convergence of the systems' states to the origin with respect to the barriers. The colors and line styles correspond to those in Figure 4-1. All of the simulated methods converge to the origin. The developed method converges first. The noisy behavior of the simulation of [38] is partially due to added noise in the controller to satisfy the PE condition. The initial control input in [38] is high since the state is close to the barriers and due to the structure of (4–35).

Table 4-1 compares the developed method to [33, 38, 53] in terms of execution time of a 15 second simulation. Each method was simulated in MATLAB. Based on this simulation, BFs constrain an ADP algorithm that uses sparse BE extrapolation, which enables the system to converge to the origin while switching history stacks between active segments, thus formalizing system safety constraints. The developed method converges faster than [33, 38, 53] but is more computationally expensive.

Table 4-1. Simulation Execution Time

Method	Computation Time (s)
Sparse BE Extrapolation with BF	3.39
Sparse BE Extrapolation [33]	2.58
StaF ADP [53]	2.71
Standard ADP with BF [38]	1.97

4.5 Concluding Remarks

This chapter presents a framework that combines the use of sparse BE extrapolation with BFs. Using this framework, a dynamic model can be used to evaluate the BE over unexplored areas of the state-space when the states have been transformed using BFs. An online approximate optimal controller is developed using sparse, segmented BE extrapolation and BFs to optimally regulate a dynamical system while providing formal safety guarantees. A BF transform is applied to a fully-constrained dynamical system to generate an unconstrained optimization problem. RL is used to solve the optimization problem online, leading to the development of an approximate optimal controller. The value function is approximated via sparse BE extrapolation over segments of the state-space. A Lyapunov-like stability analysis in the presence of discontinuities shows UUB regulation of the system states to the neighborhood of the origin and convergence of the control policy to the neighborhood of the optimal policy. A simulation of a two-state dynamical system compares the developed method to related existing methods.

Future research thrusts in the combination of optimal control and BFs can examine the use of more complicated (e.g., coupled) state constraints. While multiple variations of ADP have been used to impose constraints on individual states, the problem of implementing more complex combined state constraints exists. While it is possible to place BF-based constraints on $\hat{\mu}$ in (4–20), such a constraint would invalidate the optimality objective. A more complicated (and currently open problem) is how to perform a BF-based transformation of the state (e.g., the one developed in (4–2)-(4–12)) to include more complex state constraints.

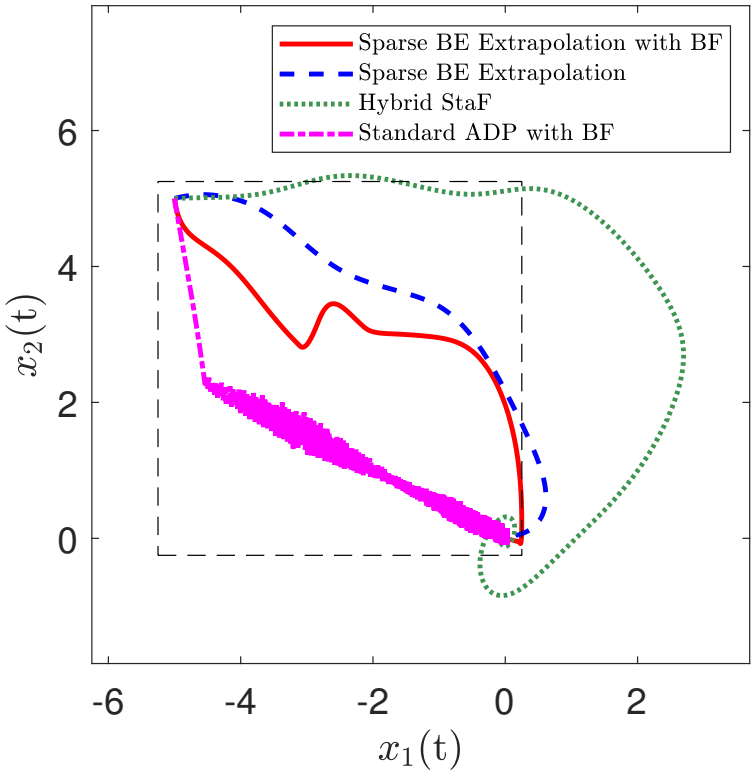


Figure 4-1. State-space portrait for the system in (4–35). The black dashed lines represent the barriers. The solid red line denotes the trajectory of the developed method, the dashed blue line denotes the trajectory using the method in [33], the dotted green line denotes the trajectory using the method in [53], the dash-dotted magenta line denotes the trajectory using the method in [38] without input saturation. Each method is simulated with the same Q and R values.

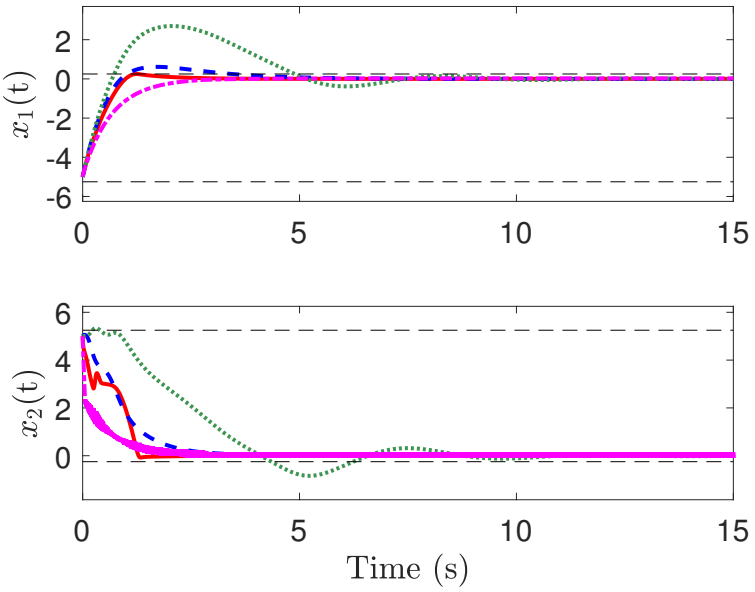


Figure 4-2. State trajectory for the system in (4–35). The black horizontal dashed lines represent the barriers of each state. The developed method is shown to converge to the origin first. The legend for this figure is omitted since it is the same as Figure 4-1.

CHAPTER 5
MODEL-BASED REINFORCEMENT LEARNING FOR OPTIMAL FEEDBACK CONTROL OF SWITCHED SYSTEMS

This chapter and the results in [71–73] examine the use of RL-based controllers to approximate multiple value functions of specific classes of subsystems while following a switching sequence. Each subsystem may have varying characteristics, such as different cost or different system dynamics. Stability of the overall switching sequence is proven using Lyapunov-based analysis techniques. Specifically, Lyapunov-based methods are developed to prove boundedness of individual subsystems and to determine a minimum dwell-time condition to ensure stability of the overall switching sequence. UUB regulation of the states, approximation of the value function, and approximation of the optimal control policy is achieved for arbitrary switching sequences provided the minimum dwell-time condition is satisfied. Simulation results for a three-state dynamical system are presented to demonstrate the performance of the developed technique.

5.1 Switched ADP Development

In the subsequent chapter, the ADP problem is outlined again for a family of dynamical systems. The development of an ADP controller for each subsystem is identical to that in Chapter 2, except a separate ADP policy governs each subsystem. Generally, the subscript p denotes the p^{th} subsystem (e.g., f_p represents the drift dynamics for the p^{th} subsystem).

Let $p \in \mathbb{P}$, where $\mathbb{P} \subset \mathbb{N}$ and $|\mathbb{P}| < \infty$, represent a family of switched subsystems. Consider the p^{th} continuous-time control-affine nonlinear dynamical system $\dot{x} = f_p(x) + g_p(x)\,u$ where $x \in \mathbb{R}^n$ denotes the system state, $u \in \mathbb{R}^m$ denotes the control input, $f_p : \mathbb{R}^n \to \mathbb{R}^n$ denotes the drift dynamics, and $g_p : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ is the control effectiveness, where $n \geq m$ and the pseudoinverse of $g(x)$ exists. The control objective

is to track a time-varying continuously differentiable signal $x_d \in \mathbb{R}^n$.[1] To quantify the tracking objective, the tracking error is defined as $e \triangleq x - x_d$. Using the technique in [48] to transform the time-varying tracking problem into an infinite horizon regulation problem, the control-affine dynamics can be rewritten as

$$\dot{\zeta} = F_p(\zeta) + G_p(\zeta)\,\mu_p, \tag{5–1}$$

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

$$F_p(\zeta) \triangleq \begin{bmatrix} f_p(e + x_d) - h_{d,p}(x_d) + g_p(e + x_d)\,u_{d,p}(x_d) \\ h_{d,p}(x_d) \end{bmatrix}, \tag{5–2}$$

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

$$G_p(\zeta) \triangleq \left[g_p(x)^T, \mathbf{0}_{m\times n}\right]^T. \tag{5–3}$$

The action space for μ_p is $U \subseteq \mathbb{R}^m$. The following assumptions facilitate the development of the approximate optimal tracking controller [48].

Assumption 5.1. The function f_p is continuously differentiable and $f_p(0) = 0$ for all $p \in \mathbb{P}$.

Assumption 5.2. Each function g_p is locally Lipschitz and bounded such that $0 < \|g_p(x)\| \leq \overline{g_p}$, where $\overline{g_p} \in \mathbb{R}_{>0}$ is the known constant maximum singular value of $g_p(x)$. It follows that $0 < \|G_p(\zeta)\| \leq \overline{G_p}$, where $\overline{G_p} \in \mathbb{R}_{>0}$ is a known constant.

Assumption 5.3. There exist locally Lipschitz functions $h_{d,p} : \mathbb{R}^n \to \mathbb{R}^n$ that define the desired trajectories such that $h_{d,p}(x_d) \triangleq \dot{x}_d$ and $g_p(x_d)\,g_p^+(x_d)\,(h_{d,p}(x_d) - f_p(x_d)) =$

[1] x_d need not be continuously differentiable at the switching instances.

$h_{d,p}(x_d) - f_p(x_d)\ \forall t \in \mathbb{R}_{\geq 0}$ and $p \in \mathbb{P}$, where $g_p^+ : \mathbb{R}^n \to \mathbb{R}^{m\times n}$ is defined as $g_p^+(x) \triangleq \left(g_p^T(x)\, g_p(x)\right)^{-1} g_p^T(x)$.

The control objective is to solve the infinite-horizon optimal tracking problem for each subsystem, i.e., determine a transient control policy μ_p that minimizes the infinite horizon cost functional, $J_p : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$, defined as

$$J_p(\zeta, \mu_p) \triangleq \int_{t_0}^{\infty} \overline{Q}_p(\zeta) + \mu_p^T R_p \mu_p \, d\tau, \tag{5–4}$$

subject to (5.1) while tracking the desired trajectory of the p^{th} mode, where $\overline{Q}_p \in \mathbb{R}^{2n} \to \mathbb{R}_{\geq 0}$ is a PSD user-defined state cost function p^{th} subsystem, and $R_p \in \mathbb{R}^{m\times m}$ is user-defined PD symmetric input cost matrix for the p^{th} subsystem. Let $\overline{Q}_p(\zeta) \triangleq Q_p(e)$, where $Q_p : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is a PD user-defined cost function that penalizes the error e and not the desired trajectory x_d.

Property 5. Each state cost function $\overline{Q}_p$ is PSD and satisfies $\underline{q}_p(\|e\|) \leq \overline{Q}_p(\zeta) \leq \overline{q}_p(\|e\|)\ \forall p \in \mathbb{P}$, where $\underline{q}_p, \overline{q}_p : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$.

The infinite horizon value function (i.e., the cost-to-go) for the p^{th} mode $V_p^* : \mathbb{R}^{2n} \to \mathbb{R}_{\geq 0}$ is defined as

$$V_p^*(\zeta) \triangleq \min_{\mu_p \in U} \int_t^{\infty} \overline{Q}_p(\zeta) + \mu_p^T R_p \mu_p \, d\tau, \tag{5–5}$$

which has the boundary condition $V_p^*(0) = 0$. If the optimal value function is continuously differentiable, then the optimal control policy $\mu_p^* : \mathbb{R}^{2n} \to \mathbb{R}^m$ can be obtained from the corresponding HJB equation

$$0 = \nabla_\zeta V_p^*(\zeta) \left(F_p(\zeta) + G_p(\zeta)\, \mu_p^*\right) + \overline{Q}_p(\zeta) + \mu_p^{*T} R_p \mu_p^*, \tag{5–6}$$

with the boundary condition $V_p^*(0) = 0$. Provided the HJB in (5–6) admits a continuously differentiable PD solution, then the optimal closed-loop transient control policy μ_p^* :

$\mathbb{R}^{2n} \to \mathbb{R}^m$ is

$$\mu_p^*(\zeta) = -\frac{1}{2} R_p^{-1} G_p(\zeta)^T \left(\nabla_\zeta V_p^*(\zeta)\right)^T. \tag{5–7}$$

Value Function Approximation

The HJB in (5–6) requires knowledge of the optimal value function, which is unknown for general nonlinear systems. Parametric NN-based methods can be used to approximate the optimal value function over a compact domain.[2] To facilitate the solution of (5–6), let $\Omega_p \subset \mathbb{R}^{2n}$ be a compact set $\Omega_p \subset \mathbb{R}^{2n}$ be a compact set such that $\zeta \in \Omega_p$. The universal function approximation property of single-layer NNs is used to represent the value function of the p^{th} mode V_p^* as

$$V_p^*(x) = W_p^{*T} \phi_p(\zeta) + \epsilon_p(\zeta), \tag{5–8}$$

where $W_p^* \in \mathbb{R}^L$ is an unknown bounded vector of weights, $\phi_p : \mathbb{R}^{2n} \to \mathbb{R}^L$ is a user-defined vector of basis functions, and $\epsilon_p : \mathbb{R}^n \to \mathbb{R}$ is the function approximation error. Each subsystem p can have finitely many neurons L. The number of neurons in each subsystem's NN can be different (e.g., L for subsystem 2 need not be the same as L in subsystem 1). However, to focus the subject of this manuscript and to minimize the amount of notation, generally, L represents the number of neurons in $\phi_p \ \forall p \in \mathbb{P}$. Substituting (5–8) into (5–7), the transient optimal control policy of the p^{th} mode μ_p^* is expressed as

$$\mu_p^*(\zeta_p) = -\frac{1}{2} R_p^{-1} G_p(\zeta) \left(W_p^* \nabla_\zeta \phi_p(\zeta) + \nabla_\zeta \epsilon_p(\zeta)\right)^T. \tag{5–9}$$

Property 6. There exists a set of constants that bound each unknown ideal weight vector W_p^*, the user-defined activation functions ϕ, and function reconstruction error ϵ_p,

[2] The subsequent stability analyses proves that if ζ is initialized within a compact set, then it will remain in that compact set.

from above such that $\left\|W_p^*\right\| \leq \overline{W_p^*}$, $\sup_{\zeta\in\Omega_p} \|\phi_p(\zeta)\| \leq \overline{\phi_p}$, $\sup_{\zeta\in\Omega_p} \|\nabla_\zeta \phi_p(\zeta)\| \leq \overline{\nabla_\zeta \phi_p}$, $\sup_{\zeta\in\Omega_p} \|\epsilon_p(\zeta)\| \leq \overline{\epsilon_p}$, $\sup_{\zeta\in\Omega_p} \|\nabla_\zeta \epsilon_p(\zeta)\| \leq \overline{\nabla_x \epsilon_p}$, where $\overline{W_p^*}$, $\overline{\phi_p}$, $\overline{\nabla_\zeta \phi_p}$, $\overline{\epsilon_p}$, $\overline{\nabla_\zeta \epsilon_p} \in \mathbb{R}_{\geq 0}$ [60].

The ideal weights W_p^* in (5–8) and (5–9) are unknown; hence, an approximation of W_p^* is sought. Specifically, the critic estimate, $\hat{W}_{c,p} \in \mathbb{R}^L$ is substituted to estimate the value function $\hat{V}_p : \mathbb{R}^{2n} \times \mathbb{R}^L \to \mathbb{R}$ denoted as

$$\hat{V}_p\left(\zeta, \hat{W}_{c,p}\right) \triangleq \hat{W}_{c,p}^T \phi_p(\zeta). \tag{5–10}$$

Similarly, another estimate for W_p, called the actor weight vector $\hat{W}_{a,p} \in \mathbb{R}^L$, is used to provide an approximate version of (5–9), the approximate optimal control policy $\hat{\mu}_p : \mathbb{R}^n \times \mathbb{R}^L \to \mathbb{R}$ is given by

$$\hat{\mu}_p\left(\zeta, \hat{W}_{a,p}\right) = -\frac{1}{2} R_p^{-1} G_p(\zeta)^T \left(\nabla_\zeta \phi_p(\zeta)^T \hat{W}_{a,p}\right). \tag{5–11}$$

5.2 System Identification

One contribution of this chapter is the extension of ADP to certain classes of switched systems. In many cases, a model of the drift dynamics f_p, and therefore F_p, may be known *a priori*. However, if the drift dynamics contain uncertainty (i.e., contains parametric uncertainties), then online system identification must be performed to estimate the system model in real-time.

To facilitate the system identification objective, let $\hat{f}_p : \mathbb{R}^n \times \mathbb{R}^s \to \mathbb{R}^n$ be a parametric estimate of the drift dynamics f_p. The number of parametric uncertainties $s \in \mathbb{N}$ represents the number of uncertain parameters for each subsystem p. Each subsystem p may have a different number of uncertainties and, therefore, different value of s. For notational brevity, let $s = s_p \, \forall p \in \mathbb{P}$ represent the number of parametric uncertainties for each subsystem. Let f_p be linearly parameterizable, i.e., $f_p(x) \triangleq Y_p(x)\theta_p$, where $Y_p : \mathbb{R}^n \to \mathbb{R}^{n\times s}$ is a constant regression matrix and $\theta_p \in \mathbb{R}^s$ denotes the unknown

constant parameters of f_p. The developed technique can be extended to a linear-in-the-parameters NNs [48] or DNNs [52] to capture a larger class of nonlinear systems. To contain the scope of this manuscript, the dynamics are assumed to be linear-in-the-parameters. Using an approximation of the uncertain parameter vector $\hat{\theta}_p \in \mathbb{R}^s$, an approximation of the p^{th} mode's uncertain drift dynamics $\hat{f}_p : \mathbb{R}^n \times \mathbb{R}^s \to \mathbb{R}^n$ is defined as $\hat{f}_p\left(x, \hat{\theta}_p\right) \triangleq Y_p(x)\,\hat{\theta}_p$. The parameter estimates are updated using the ICL-based update policy

$$\dot{\hat{\theta}}_p \triangleq -\Gamma_{p,\theta} \sum_{j=1}^{M} \mathcal{Y}_{p,j}^T \left(x(t_j) - x(t_j - \Delta t) - \mathcal{U}_{p,j} - \mathcal{Y}_{p,j}\hat{\theta}_p\right) \tag{5–12}$$

from [74], where $\Gamma_{p,\theta} \in \mathbb{R}^{s\times s}$ is a PD learning gain, $\mathcal{Y}_{p,j} \triangleq \mathcal{Y}_p(t_j)$, $\mathcal{U}_{p,j} \triangleq \mathcal{U}_p(t_j)$, $\mathcal{Y}_p(t) \triangleq \int_{\max[t-\Delta t,0]}^{t} Y_p(x(\tau))\,\mathrm{d}\tau$, and $\mathcal{U}_p(t) \triangleq \int_{\max[t-\Delta t,0]}^{t} g_p(x(\tau))\,u(\tau)\,\mathrm{d}\tau$.

Assumption 5.4. An history stack of recorded state and control inputs $\{x(t_j), u(t_j)\}_{j=1}^{M_p}$ that satisfy $\underline{\mathcal{Y}}_p \triangleq \lambda_{\min}\left\{\sum_{j=1}^{M} \mathcal{Y}_{p,j}^T \mathcal{Y}_{p,j}\right\} > 0$ are available *a priori* for all subsystems $p \in \mathbb{P}$.

Remark 5.1. To relax the common PE condition, the update law in (5–12) uses a history stack of recorded state and input data. The finite excitation condition (Assumption 5.4) from [74] is necessary to facilitate parameter convergence in the subsequent stability analysis. With a minor modification to the parameter update law in (5–12), the availability of the history stack *a priori* is not necessary [48]. Assumption 5.4 is used to focus the scope of this manuscript and simplify the subsequent stability analysis.

To facilitate the subsequent stability analysis, define $\tilde{\theta}_p \triangleq \theta_p - \hat{\theta}_p$. The update law in 5–12 can be rewritten in an analytical form as

$$\dot{\tilde{\theta}}_p = -k_{ICL,p}\Gamma_{p,\theta} \sum_{j=1}^{M} \mathcal{Y}_{p,j}^T \mathcal{Y}_{p,j} \tilde{\theta}_p \tag{5–13}$$

If f is unknown, then the trajectory tracking component of the controller $u_d(x_d)$ contains uncertainty. An approximation of the trajectory tracking component $\hat{u}_d : \mathbb{R}^n \times \mathbb{R}^s \to \mathbb{R}^m$ is defined as $\hat{u}_{d,p}\left(x_d, \hat{\theta}\right) \triangleq g_p^+(x_d)\left(h_{d,p}(x_d) - \hat{f}_p\left(x, \hat{\theta}\right)\right)$. Hence, the

applied control policy is

$$u \triangleq \hat{\mu}_p\left(\zeta, \hat{W}_{a,p}\right) + \hat{u}_{d,p}\left(x_d, \hat{\theta}_p\right). \tag{5–14}$$

5.3 Bellman Error

The HJB equation in (5–6) is equal to zero under optimal conditions; however, substituting (5–10) and (5–11) into (5–6) results in a residual term $\hat{\delta}_p : \mathbb{R}^n \times \mathbb{R}^L \times \mathbb{R}^L \to \mathbb{R}$, which is referred to as the BE, defined as

$$\begin{aligned}\hat{\delta}_p\left(\zeta, \hat{\theta}_p, \hat{W}_{c,p}, \hat{W}_{a,p}\right) \triangleq \nabla_\zeta \hat{V}_p\left(\zeta, \hat{W}_{c,p}\right)\left(F_{\theta,p}\left(\zeta, \hat{\theta}_p\right) + F_{1,p}(\zeta) + G_p(\zeta)\hat{\mu}_p\left(\zeta, \hat{W}_{a,p}\right)\right) \\ + \overline{Q}_p(\zeta) + \hat{\mu}_p\left(\zeta, \hat{W}_{a,p}\right)^T R_p \hat{\mu}_p\left(\zeta, \hat{W}_{a,p}\right),\end{aligned} \tag{5–15}$$

where $F_{\theta,p}\left(\zeta, \hat{\theta}_p\right) \triangleq \left[\left(\hat{f}_p\left(x, \hat{\theta}_p\right) - g_p(x) g_p^+(x_d) \hat{f}_p\left(x_d, \hat{\theta}_p\right)\right)^T, \mathbf{0}_{1\times n}\right]^T$ and $F_{1,p}(\zeta) \triangleq \left[\left(-h_{d,p}(x_d) + g_p(x) g_p^+(x_d) h_{d,p}(x_d)\right)^T, h_{d,p}(x_d)^T\right]^T$. The BE is in indirect measure of the proximity of the actor and critic weight estimates to the ideal weights. By defining the mismatch between the estimates and the ideal values as $\tilde{W}_{c,p} \triangleq W_p^* - \hat{W}_{c,p}$ and $\tilde{W}_{a,p} \triangleq W_p^* - \hat{W}_{a,p}$, substituting (5–8) and (5–9) in (5–6), and subtracting from (5–15) yields the analytical form of the BE, which is used in the subsequent stability analysis, as

$$\begin{aligned}\hat{\delta}_p\left(\zeta, \hat{\theta}_p, \hat{W}_{c,p}, \hat{W}_{a,p}\right) = -W_p^T \nabla_\zeta \phi_p(\zeta)\left(F_{\theta,p}(\zeta, \theta_p) - F_{\theta,p}\left(\zeta, \hat{\theta}_p\right)\right) - \omega_p^T \tilde{W}_{c,p} \\ + \frac{1}{4}\tilde{W}_{a,p}^T G_{\phi,p} \tilde{W}_{a,p} + O_p(\zeta),\end{aligned} \tag{5–16}$$

where $\omega_p : \mathbb{R}^{2n} \times \mathbb{R}^L \times \mathbb{R}^s \to \mathbb{R}^{2n}$ is defined as

$$\omega_p\left(\zeta, \hat{W}_{a,p}, \hat{\theta}_p\right) \triangleq \nabla_\zeta \phi_p(\zeta)\left(F_{p,\theta}\left(\zeta, \hat{\theta}_p\right) + F_{1,p}(\zeta) + G_p(\zeta)\hat{\mu}\left(\zeta, \hat{W}_{a,p}\right)\right), \tag{5–17}$$

$O(\zeta) \triangleq \frac{1}{2}\nabla_\zeta \epsilon_p(\zeta) G_{R,p} \nabla_\zeta \phi_p(\zeta)^T W_p^* + \frac{1}{4} G_{\epsilon,p} - \nabla_\zeta \epsilon_p(\zeta) F_{p,\theta}(\zeta, \theta_p) - \nabla_\zeta \epsilon_p(\zeta) F_{1,p}(\zeta)$, $G_{R,p} = G_{R,p}(\zeta) \triangleq G_p(\zeta) R_p^{-1} G_p(\zeta)^T$, $G_{\phi,p} = G_{\phi,p}(\zeta) \triangleq \nabla_\zeta \phi_p(\zeta) G_{R,p}(\zeta) \nabla_\zeta \phi_p(\zeta)^T$, and $G_{\epsilon,p} = G_{\epsilon,p}(\zeta) \triangleq \nabla_\zeta \epsilon_p(\zeta) G_p(\zeta) \nabla_\zeta \epsilon_p(\zeta)^T$.

Remark 5.2. The expressions in (5–15) and (5–16) are equivalent for the BE. However, (5–15) is used in implementation, while (5–16) is used in the stability analysis.

Bellman Error Extrapolation

At each time instant, the BE in (5–15) is calculated using the control policy given by (5–11) evaluated using the current system state, critic weight estimates, and actor weight estimates to obtain the instantaneous BE denoted by $\hat{\delta}_p \triangleq \hat{\delta}_p\left(\zeta, \hat{\theta}_p, \hat{W}_{c,p}, \hat{W}_{a,p}\right)$.

A classical problem in learning-based control is exploration versus exploitation. Results such as [26] add an exploration signal to sufficiently explore the operating domain. However, no analytical methods exist to compute the appropriate exploration signal. Alternatively, results such as [19] evaluate the BE along the system trajectory and at any desired point in the state space (i.e., so-called BE extrapolation). The BE extrapolation technique provides simulation of experience to avoid using an exploration signal.

Specifically, BE is extrapolated from a user-specified number and location of off-trajectory points $\{\zeta_{i,p} : \zeta_{i,p} \in \Omega_p\}_{i=1}^{N_p}$, where $N_p \in \mathbb{N}$ denotes a user-specified number of points in the compact set Ω_p. The data is represented by the tuple $(\Sigma_{c,p}, \Sigma_{a,p}, \Sigma_{\Gamma,p})$, defined as $\Sigma_{c,p} \triangleq \frac{1}{N_p}\sum_{i=1}^{N_p} \frac{\omega_{i,p}}{\rho_{i,p}}\hat{\delta}_{i,p}$, $\Sigma_{a,p} \triangleq \frac{1}{N_p}\sum_{i=1}^{N_p} \frac{G_{\sigma i,p}^T \hat{W}_{a,p}\omega_{i,p}^T}{4\rho_{i,p}}$, $\Sigma_{\Gamma,p} \triangleq \frac{1}{N_p}\sum_{i=1}^{N_p} \frac{\omega_{i,p}\omega_{i,p}^T}{\rho_{i,p}}$, where $\hat{\delta}_{i,p} \triangleq \hat{\delta}_p\left(\zeta_{i,p}, \hat{\theta}_p, \hat{W}_{c,p}, \hat{W}_{a,p}\right)$, $\omega_{i,p} \triangleq \omega_p\left(\zeta_{i,p}, \hat{W}_{a,p}, \hat{\theta}_p\right)$, and $\rho_{i,p} = 1 + \nu_p \omega_{i,p}^T \Gamma_p \omega_{i,p}$, $\nu_p \in \mathbb{R}_{>0}$ is a user-defined gain, and $\Gamma : \mathbb{R} \to \mathbb{R}^{L\times L}$ is a time-varying least-squares gain matrix. Each subsystem, p, must have distinct sets of data, gain values, and update laws.

Assumption 5.5. Over the compact set, Ω_p, a finite set of off-trajectory points $\{\zeta_{i,p} : \zeta_{i,p} \in \Omega_p\}_{i=1}^{N_p}$ exists such that $0 < \underline{c}_p \triangleq \inf_{t\in\mathbb{R}_{\geq 0}} \lambda_{\min}\{\Sigma_{\Gamma,p}(t)\}$ for all $t \in \mathbb{R}_{\geq 0}$ and $p \in \mathbb{P}$, where $\underline{c}_p$ is a constant scalar lower bound of the value of each input-output data pair's minimum eigenvalues for the p^{th} subsystem.

5.4 Update Laws for Actor and Critic Weights

Using the extrapolated BEs $\hat{\delta}_{i,p}$, the critic and actor weights are updated according to the following policies. In the following definitions, $\eta_{c,p}$, $\eta_{a1,p}$, $\eta_{a2,p}$, $\lambda_p \in \mathbb{R}$ are positive constant learning gains, and $\underline{\Gamma}_p$, $\overline{\Gamma}_p \in \mathbb{R}_{>0}$ are upper and lower bounds of the least-squares learning gains of subsystem p. The critic update law of the p^{th} mode is

$$\dot{\hat{W}}_{c,p} \triangleq -\eta_{c,p}\Gamma_p\Sigma_{c,p}. \tag{5–18}$$

The actor update law of the p^{th} mode is

$$\dot{\hat{W}}_{a,p} \triangleq -\eta_{a1,p}\left(\hat{W}_{a,p} - \hat{W}_{c,p}\right) - \eta_{a2,p}\hat{W}_{a,p} + \eta_{c2,p}\Sigma_{a,p}\hat{W}_{c,p}. \tag{5–19}$$

The least-squares gain matrix update law of the p^{th} mode is

$$\dot{\Gamma}_p \triangleq \left(\lambda_p\Gamma_p - \eta_{c,p}\Gamma_p\Sigma_{\Gamma,p}\Gamma_p\right)\cdot \mathbf{1}_{\left\{\underline{\Gamma}_p \le \|\Gamma_p\| \le \overline{\Gamma}_p\right\}}, \tag{5–20}$$

where $\mathbf{1}_{\{\cdot\}}$ denotes the indicator function, and $\overline{\Gamma}_p, \underline{\Gamma}_p \in \mathbb{R}_{>0}$ are user-defined saturation gains that bound $\|\Gamma_p\|$ such that $\underline{\Gamma}_p \le \|\Gamma_p(t)\| \le \overline{\Gamma}_p$ for all $t \in \mathbb{R}_{>0}$ and $p \in \mathbb{P}$.

The update policies in (5–12) and (5–18)-(5–20) are distinct for each subsystem. However, to prove stability of the overall switched system, each update policy must be active simultaneously. Note that the aforementioned update policies do not rely on the current state of the system to update the learning parameters. The update policy in (5–12) relies on the history stack of state-action data, and the update policies in (5–18)-(5–20) depend on the values of other parameters and data collected from BE extrapolation.

5.5 Stability Analysis

Generally, the trajectory of a switched system can diverge even when all the subsystems that compose the switched system are stable. Hence, the switching signal must be properly designed to keep the overall system stable. Before the switching signal is designed, the stability of each subsystem must be analyzed. In the following

development, p subsystems, each with a class of dynamics in (5–1), will be analyzed with the control policy and update laws outlined in (5–1) and (5–18)-(5–20).

5.5.1 Subsystem Stability Analysis

Recall from Property 5 that the function $\overline{Q}$ and, therefore, the optimal value function V^* in (5–8) is PSD. Hence, V^* is not a valid Lyapunov function. The result in [61] 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 $V_{na,p}^*(e,t) \triangleq V^*(\zeta)$, is PD and decrescent. Hence, $V_{na}^*(0,t) = 0$ and there exist class $\mathcal{K}_\infty$ functions $\underline{v}, \overline{v} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that bound $\underline{v}_p(\|e\|) \leq V_{na,p}^*(e,t) \leq \overline{v}_p(\|e\|)\ \forall e \in \mathbb{R}^n, t \in \mathbb{R} \geq 0$. Hence, $V_{na}^*(e,t)$ is a valid Lyapunov function. Let $Z \in \mathbb{R}^{n+|\mathbb{P}|(2L+s)}$ denote a concatenated state defined as $Z \triangleq \left[e^T, \tilde{W}_{c,1}^T, \ldots, \tilde{W}_{c,}^T, \tilde{W}_{a,1}^T, \ldots, \tilde{W}_{a,}^T, \tilde{\theta}_1^T, \ldots, \tilde{\theta}^T\right]^T$. Let $V_{L,p} : Z \in \mathbb{R}^{n+|\mathbb{P}|(2L+s)} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ be a candidate Lyapunov function defined as

$$V_{L,p}(Z,t) \triangleq V_{na,p}^*(e,t) + \frac{1}{2}\sum_{p\in\mathbb{P}} \tilde{W}_{c,p}^T \Gamma_p(t)^{-1} \tilde{W}_{c,p} + \frac{1}{2}\sum_{p\in\mathbb{P}} \tilde{W}_{a,p}^T \tilde{W}_{a,p} + \frac{1}{2}\sum_{p\in\mathbb{P}} \tilde{\theta}_p^T \Gamma_{\theta,p}^{-1} \tilde{\theta}_p. \quad (5\text{–}21)$$

Using the properties of $V_{na}^*(e,t)$ and [67, Lemma 4.3], then (5–21) be bounded as $\alpha_{1,p}(\|Z\|) \leq V_{L,p}(Z,t) \leq \alpha_{2,p}(\|Z\|)$ using class $\mathcal{K}$ functions $\alpha_{1,p}, \alpha_{2,p} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Using (5–20), the normalized regressors $\frac{\omega}{\rho}$ and $\frac{\omega_e}{\rho_e}$ can be bounded as $\sup_{t\in\mathbb{R}_{\geq 0}} \left\|\frac{\omega_p}{\rho_p}\right\| \leq \frac{1}{2\sqrt{\nu_p \underline{\Gamma}_p}}$ for all $\zeta_p \in \Omega_p$ and $\sup_{t\in\mathbb{R}_{\geq 0}} \left\|\frac{\omega_{e,p}}{\rho_{e,p}}\right\| \leq \frac{1}{2\sqrt{\nu_p \underline{\Gamma}_p}}$ for all $\zeta_{e,p} \in \Omega_p$. The matrices $G_{R,p}$ and $G_{\sigma,p}$ can be bounded as $\sup_{\zeta_p\in\Omega_p} \|G_R\| \leq \lambda_{\max}\left(R_p^{-1}\right)\overline{G_p}^2 \triangleq \overline{G_{R,p}}$ and $\sup_{\zeta_p\in\Omega_p} \|G_{\sigma,p}\| \leq \left(\overline{\nabla_\zeta \phi_p G_p}\right)^2 \lambda_{\max}\left(R_p^{-1}\right) \triangleq \overline{G_{\sigma,p}}$, respectively. Furthermore, define $\mathcal{R}_p \in \mathbb{R}_{>0}$ as the radius of a compact ball centered at the origin $\mathcal{B}_{\mathcal{R}_p} \subset \mathbb{R}^{n+|\mathbb{P}|(2L+s)}$.

Theorem 5.1. *Provided Assumptions 5.4 and 5.5 hold, the weight update laws in (5–18)-(5–20) are used, and the gain conditions*

$$\eta_{a1,p} + \eta_{a2,p} > \frac{\eta_{c1,p} + \eta_{c2,p}}{\sqrt{\nu_p \underline{\Gamma}_p}} \overline{W_p^* G_{\phi,p}}, \quad (5\text{–}22)$$

$$\underline{c}_p > \frac{3\eta_{a1,p}}{\eta_{c,p}} + \frac{3\eta_{c,p}^2 \overline{W_p^*}^2 \overline{G_{\phi,p}}^2}{16\nu_p \underline{\Gamma}_p (\eta_{a1,p} + \eta_{a2,p})\eta_{c2,p}}, \quad (5\text{–}23)$$

$$L_p < \alpha_{2,p}^{-1}\left(\alpha_{1,p}\left(\mathcal{R}_p\right)\right), \tag{5–24}$$

are satisfied for all $p \in \mathbb{P}$*, where* L_p *and* $\mathcal{R}_p$ *are positive constants, then the tracking error* e*, weight estimation errors* $\tilde{W}_{c,p}$ *and* $\tilde{W}_{a,p}$*, and parameter estimation error* $\tilde{\theta}_p$ *are UUB. Hence, the error between the transient control policy for each mode* $\hat{\mu}_p$ *in* (5–11) *and its respective optimal control policy* μ_p^* *in* (5–7) *is UUB.*

Proof. Taking the time derivative of (5–21), the fact $\frac{d}{dt}\Gamma^{-1} = \Gamma^{-1}\dot{\Gamma}\Gamma^{-1}$, along with Assumptions 5.1-5.5 and the sufficient conditions in (5–22)-(5–24) yields $\dot{V}_{L,p} \leq -v_{L,p}\left(\|Z\|\right), \ \forall v_{L,p}^{-1}\left(L_p\right) \leq \|Z\| \leq \alpha_{2,p}^{-1}\left(\alpha_{1,p}\left(\mathcal{R}\right)\right)$, where $v_{L,p}\left(\|Z\|\right) \triangleq \frac{1}{2}\underline{q}_p\left(\|e\|\right) + \sum_{p=1}^{|\mathbb{P}|}\left[\frac{\eta_{c,p}\underline{c}_p}{12}\left\|\tilde{W}_{c,p}\right\|^2 + \frac{\eta_{a1,p}+\eta_{a2,p}}{20}\left\|\tilde{W}_{a,p}\right\|^2 + \frac{k_{ICL,p}\underline{\mathcal{Y}}_p}{6}\left\|\tilde{\theta}_p\right\|^2\right]$ and L_p is a positive constant. While each individual subsystem is active, [67, Theorem 4.18] can be invoked to conclude that Z is UUB such that $\limsup_{t\to\infty}\|Z\left(t\right)\| \leq \alpha_{1,p}^{-1}\left(\alpha_{2,p}\left(v_{L,p}^{-1}\left(L_p\right)\right)\right)$. Hence, it can also be shown that $\hat{\mu}_p$ converges to a neighborhood of the optimal policy μ_p^*. Furthermore, since $Z \in \mathcal{L}_\infty$, it follows that $e, \tilde{W}_{c,1}, \ldots, \tilde{W}_{c,|\mathbb{P}|}, \tilde{W}_{a,1}, \ldots, \tilde{W}_{a,|\mathbb{P}|}, \tilde{\theta}_1, \ldots, \tilde{\theta}_{|\mathbb{P}|} \in \mathcal{L}_\infty$, hence $x, \hat{W}_{c,1}, \ldots, \hat{W}_{c,|\mathbb{P}|}, \hat{W}_{a,1}, \ldots, \hat{W}_{a,|\mathbb{P}|}, \hat{\theta}_1, \ldots, \hat{\theta}_{|\mathbb{P}|} \in \mathcal{L}_\infty$ and $u \in \mathcal{L}_\infty$. □

5.5.2 Dwell-Time Analysis

Theorem 5.1 indicates that each subsystem is UUB. However, this does not account for switching between subsystems. Switching between control policies may result in instantaneous growth when switching between multiple Lyapunov-like functions. To ensure that the switched system is stable, a dwell-time must be designed to switch between subsystems. Hence, continuity is not guaranteed between Lyapunov-like functions $V_{L,p}$ across all subsystems.

Remark 5.3. The notation in this subsection is meant to be self-contained. To generalize the following development, the conventional notation used in Lyapunov-based analyses is introduced.

Suppose we have a family f_p, $p \in \mathbb{P}$ of functions from $f_p : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ where $\mathbb{P}$ is a switching index set. This yields a family of systems

$$\dot{x} = f_p(x,t),\ p \in \mathbb{P} \tag{5–25}$$

that evolve on $\mathbb{R}^n$. $\sigma : \mathbb{R}_{\geq 0} \to \mathbb{P}$ is a piecewise constant switching signal that is continuous from the right everywhere ($\sigma(t) = \lim_{\tau \to t^+} \sigma(\tau)$ for each $\tau \geq 0$). To simplify notation and denote switching, let

$$\dot{x} = f_{\sigma(t)}(x,t),\ p \in \mathbb{P}. \tag{5–26}$$

This stability analysis approach relies on multiple Lyapunov functions. While each candidate Lyapunov function V_p for $p \in \mathbb{P}$ is continuous, the general candidate Lyapunov function $V_{\sigma(t)}$ is discontinuous (i.e., the value of the Lyapunov-like function may instantaneously change value at the switching instances). While each $V_p(x,t)$ decreases or is bounded within a defined UUB region while active, $V_p(x,t)$ may increase when the p^{th} system is inactive.

Definition 5.1. Given an infinite sequence of switching times $t_\sigma \triangleq \{t_0, t_1, \ldots, t_i, t_j, \ldots\}$, the dwell-time $\tau \in \mathbb{R}_{>0}$ is defined as the time between switching instances. Specifically, $\tau(t_i, t_j) \triangleq t_j - t_i$ such that $\sigma(t_i) \neq \sigma(t_j)$.

Theorem 5.2. *Let $\dot{x} = f_p(x,t)$ be a finite family of UUB stable subsystems and $V_p : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ be a family of corresponding Lyapunov-like functions that satisfy*

$$\alpha_{1,p}(\|x\|) \leq V_p(x,t) \leq \alpha_{2,p}(\|x\|), \tag{5–27}$$

$$\frac{\partial V_p}{\partial t} + \frac{\partial V_p}{\partial x} f_p(x,t) \leq -W_p(x), \tag{5–28}$$

and

$$\max_{p \in \mathbb{P}} \alpha_{2,p}(\mu_p) < \min_{p \in \mathbb{P}} \alpha_{1,p}(r) \tag{5–29}$$

for all $x \in \Lambda_p$, $p \in \mathbb{P}$, *and* $t \geq 0$, *where* $\Lambda_p \triangleq \{x \,|\, 0 \leq \mu_p \leq \|x\| \leq r\}$, $\alpha_{1,p}$, $\alpha_{2,p} : [0, r] \to \mathbb{R}_{\geq 0}$ *are class* $\mathcal{K}$ *functions, and* $W_p : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ *is a continuous PD function. Furthermore, let* $t_\sigma = \{t_0, t_1, t_2, \ldots\}$ *represent a sequence of switching times. If the conditions in* (5–27)-(5–29) *and the minimum dwell-time condition*

$$\tau(t_i) \geq \begin{cases} \frac{\alpha_{2,\sigma(t_i)}(\|x(t_i)\|) - \alpha_{1,\sigma\left(t_i^-\right)}(\|x(t_i)\|)}{\kappa} & \forall V_{\sigma(t_i)}(x(t_i), t_i) > \overline{\alpha} \\ > 0 & \forall V_{\sigma(t_i)}(x(t_i), t_i) \leq \overline{\alpha} \end{cases} \tag{5–30}$$

are satisfied for all $p \in \mathbb{P}$ *and for every switching instant* $t_i \in t_\sigma$, *where* V_j *represents the Lyapunov-like function of the* j^{th} *subsystem,* κ *is a subsequently defined positive constant, and* $t_i \in t_\sigma$ *represents a general switching instance, then the trajectories of the switched system* $\dot{x} = f_p(x, t)$ *initialized in the set* $\left\{x \mid \|x\| \leq \min_{p,q \in \mathbb{P}} \alpha_{2,p}^{-1}(\alpha_{1,q}(r))\right\}$ *converge to a bounded region given by* $\lim_{t \to \infty} \|x(t)\| \leq \max_{p,q \in \mathbb{P}} \alpha_{1,p}^{-1}(\alpha_{2,q}(\mu_q))$.

Proof. Consider a pair of switching times $\{t_0, t_1\} \subseteq t_\sigma$ such that $t_0 < t_1$, $\sigma(t) = p \in \mathbb{P}\ \forall t \in [t_0, t_1)$. Let $q \in \mathbb{P}$ represent the subsystem active before p, i.e., $\sigma\left(t_0^-\right) = q$.

If (5–29) holds and if $\alpha_{2,p}(\mu_p) < V_p(x, t) \leq \alpha_{1,p}(r)$, then $x \in \Lambda_p$ and $\dot{V}_p(x, t) < 0$ from (5–28). As a result, if (5–29) holds and if $V_p(x(t_0), t_0) \leq \alpha_{2,p}(\mu_p)$, then $V_p(x(t), t) \leq \alpha_{2,p}(\mu_p)$ for all $t \in [t_0, t_1]$. Similarly, if (5–29) holds and if $V_p(x(t_0), t_0) \leq \overline{\alpha} \triangleq \max_{p \in \mathbb{P}}\{\alpha_{2,p}(\mu_p)\}$, then either $V_p(x(t_0), t_0) \leq \alpha_{2,p}(\mu_p)$ or $x(t_0) \in \Lambda_p$. In either case, (5–28) implies that $V_p(x(t), t) \leq \max\{V_p(x(t_0), t_0), \alpha_{2,p}(\mu_p)\}$ for all $t \in [t_0, t_1]$. Thus, if (5–29) holds, then the $\overline{\alpha}$-sublevel set and the $\alpha_{1,p}(r)$-sublevel set of V_p are forward invariant over $[t_0, t_1]$. From (5–28), whenever $x \in \Lambda_p$, $\dot{V}_p(x, t) \leq -W_p(x) \leq -\min_{p \in \mathbb{P}} \min_{x \in \Lambda_p} W_p(x) \triangleq \kappa$. Using forward invariance of $\alpha_{1,p}(r)$ and $\alpha_{2,p}(\mu_p)$, and sublevel sets of V_p, it can be concluded that $V_p(x(t), t) \leq \max\{V_p(x(t_0), t_0) - \kappa(t - t_0), \alpha_{2,p}(\mu_p)\}\ \forall t \in [t_0, t_1]$ whenever $V_p(x(t_0), t_0) \leq \alpha_{1,p}(r)$.

The difference between the Lyapunov-like functions of the q^{th} and p^{th} subsystems, at the time of exit, is bounded from above as $V_p(x(t_1), t_1) - \lambda V_q(x(t_0), t_0) \leq$

$\max\{V_p(x(t_0),t_0) - \lambda V_q(x(t_0),t_0) - \kappa(t_1-t_0), \overline{\alpha} - \lambda V_q(x(t_0),t_0)\}$ for some $\lambda \in (0,1]$. If the inequality

$$V_p(x(t_0),t_0) - \lambda V_q(x(t_0),t_0) - \kappa(t_1-t_0) \leq 0 \tag{5–31}$$

is satisfied, then

$$V_p(x(t_1),t_1) - \lambda V_q(x(t_0),t_0) \leq \max\{0, \overline{\alpha} - \lambda V_q(x(t_0),t_0)\}. \tag{5–32}$$

If $\overline{\alpha} - V_q(x(t_0),t_0) \geq 0$, then $\overline{\alpha} - \lambda V_q(x(t_0),t_0) \geq 0$. Thus, (5–32) implies that $V_p(x(t_1),t_1) \leq \overline{\alpha}$. If $\overline{\alpha} - \lambda V_q(x(t_0),t_0) \leq 0$ (i.e., the value of the Lyapunov-like function is bounded within $\overline{\alpha}$), then (5–32) implies that $V_p(x(t_1),t_1) \leq \lambda V_q(x(t_0),t_0)$ (i.e. that the value of the Lyapunov-like function has not increased at the time of exit).

In conclusion, $V_p(x(t_0),t_0) \leq \alpha_{1,p}(r)$ and (5–31) imply that $V_p(x(t_1),t_1) \leq \max\{\overline{\alpha}, \lambda V_q(x(t_0),t_0)\}$. If it can be guaranteed (by (5–29)) that

$$\overline{\alpha} \leq \alpha_{1,p}(r) \text{ and } \lambda V_q(x(t_0),t_0) \leq \alpha_{1,p}(r) \tag{5–33}$$

then, recursively, $V_{\sigma(t_{i+1})}(x(t_{i+1}),t_{i+1}) \leq \max\left\{\overline{\alpha}, \lambda V_{\sigma(t_i)}(x(t_i),t_i)\right\}$ for all $t_i \in t_\sigma$. That is, the sequence of Lyapunov-like functions decays to $\overline{\alpha}$, and since $\overline{\alpha}$-sublevel sets are forward invariant over $[t_i, t_{i+1}]$ under (5–29), the trajectories decay to $\bigcup_{p\in\mathbb{P}}\{x \mid V_p(x(t),t) \leq \overline{\alpha}\} \subseteq \left\{x \mid \|x\| \leq \max_{p\in\mathbb{P}} \alpha_{1,p}^{-1}(\overline{\alpha})\right\}$.

Sufficient conditions for (5–33) can be derived as $x(t_0) \in \{x \mid \alpha_{2,p}(\|x(t)\|) \leq \alpha_{1,p}(r) \ \forall p \in \mathbb{P}\}$ and $\overline{\alpha} \leq \min_{p\in\mathbb{P}} \alpha_{1,p}(r)$, and a sufficient dwell-time condition to satisfy the inequality in (5–31) can be derived as

$$\tau(t_0) \geq \tau^*(t_0) = \frac{\alpha_{2,p}(\|x(t_0)\|) - \alpha_{1,q}(\|x(t_0)\|)}{\kappa}, \tag{5–34}$$

The dwell-time restriction in (5–34) can be relaxed if the state at the switching time is in the set $\{x \,|\, V_p(x,t) \leq \overline{\alpha}\ \forall p \in \mathbb{P}\} \subseteq \{x \,|\, \alpha_{2,p}(\|x\|) \leq \overline{\alpha}\ \forall p \in \mathbb{P}\}$, which yields the final dwell-time condition in (5–30). □

5.5.3 Application to Switched ADP

From Theorem 5.1, every individual subsystem is UUB; i.e., each subsystem satisfies (5–27)-(5–29). Hence, given that the dwell-time condition in (5–34) is satisfied, then Theorem 5.2 can be trivially applied to show that $\lim_{t\to\infty} \|Z(t)\| \leq \max_{p,q\in\mathbb{P}} \alpha_{1,p}^{-1}(\alpha_{2,q}(\mu_q))$. Hence, it can also be shown that each $\hat{\mu}_p$ converges to a neighborhood of the respective optimal policy μ_p^* for all $p \in \mathbb{P}$. Furthermore, since $Z \in \mathcal{L}_\infty$, it follows that $e, \tilde{W}_{c,1}, \ldots, \tilde{W}_{c,|\mathbb{P}|}, \tilde{W}_{a,1}, \ldots, \tilde{W}_{a,|\mathbb{P}|}, \tilde{\theta}_1, \ldots, \tilde{\theta}_{|\mathbb{P}|} \in \mathcal{L}_\infty$, hence $e, \hat{W}_{c,1}, \ldots, \hat{W}_{c,|\mathbb{P}|}, \hat{W}_{a,1}, \ldots, \hat{W}_{a,|\mathbb{P}|}, \hat{\theta}_1, \ldots, \hat{\theta}_{|\mathbb{P}|} \in \mathcal{L}_\infty$ and $u \in \mathcal{L}_\infty$.

5.6 Simulation Results

To demonstrate the performance of the developed method, the ADP controller is applied to a family of dynamical systems. The simulation is performed on the control-affine systems in (5–35)-(5–37). The dynamic models are based on the continuous-time F-16 longitudinal dynamics from [18]. The dynamics of the first mode are

$$\dot{x} = \begin{bmatrix} -1 & 0.9 & -0.002 \\ 0.8 & -1.1 & -0.2 \\ 0 & 0 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u, \tag{5–35}$$

the dynamics of the second mode are

$$\dot{x} = \begin{bmatrix} -0.8 & 0.2 & -0.01 \\ 0.6 & -1.3 & -0.1 \\ 0 & 0 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix} u, \tag{5–36}$$

and the dynamics of the third mode are

$$\dot{x} = \begin{bmatrix} -1 & 0.5 & -0.02 \\ 0.9 & -0.8 & -0.4 \\ 0 & 0 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u, \tag{5–37}$$

where $x = \begin{bmatrix} x_1, & x_2, & x_3 \end{bmatrix}^T$, $x \in \mathbb{R}$ is measured in radians, and $u \in \mathbb{R}$. The initial condition is $x(0) = \begin{bmatrix} 0.35 & 0.26 & -0.35 \end{bmatrix}^T$. The mode described by (5–35) is the closest to the dynamic model given in [18]. (5–36) and (5–37) vary from (5–35). A different mode was arbitrarily selected every 5 second as the active subsystem to highlight this method's ability to switch between different dynamical systems. The simulated switching sequence is $\{1, 2, 3, 1, 3, 2\}$. The basis function is $\phi_p(x) = \begin{bmatrix} x_1^2, & x_1x_2, & x_1x_3, & x_2^2, & x_2x_3, & x_3^2 \end{bmatrix}^T$ for all $p \in \mathbb{P}$. Modes 1-3 have different cost matrices and gains, which alters V_p^*,and hence, the performance. The simulation parameters for each mode are listed in Table 5-1.

Table 5-1. Switched Subsystem Simulation Parameters

Parameter	Mode 1	Mode 2	Mode 3
Q_p	diag$(1,\ 1,\ 1)$	diag$(5,\ 5,\ 5)$	diag$(3,\ 3,\ 3)$
R_p	0.5	2	1
$\overline{\Gamma}_p$	10^3	10^3	10^3
$\underline{\Gamma}_p$	500	500	50
λ_p	0.4	0.5	0.5
ν_p	0.005	0.005	0.005
$\eta_{c1,p}$	3	1	1
$\eta_{c2,p}$	5	2.5	5
$\eta_{a1,p}$	20	10	5
$\eta_{a2,p}$	1	0.75	1
N_p	10	10	10

Figure 5-1 illustrates that the system states are driven to the origin with an arbitrary switching sequence and sufficiently long dwell-time. Since the dynamical systems are linear, the analytical value function can be determined by solving the Algebraic Riccati Equation. Solving the Algebraic Riccati Equation provides a matrix which corresponds

to the value function weights W_p^*, and, hence, the value functions $V_p^*(x)$. Figure 5-2 compares the value of the approximate value function to analytical value function while switching between modes. Figure 5-3 presents the evolution of the critic weights $\hat{W}_{c,p}$, while switching. A mode's weights update regardless of mode (in)activity. Note that the weights of mode 1 and 2 converge before switching to another mode for the first time, while mode 3 is switched before it finishes learning. This illustrates that the weights do not need to be learned before the switching occurs.

5.7 Concluding Remarks

A set of online approximate optimal controllers are developed for an arbitrary sequence of subsystems. Each controller is proven to regulate the state to within a neighborhood of the origin. Furthermore, the control policies are shown to converge to the neighborhood of the optimal policy using a Lyapunov-based analysis, while switching between different dynamic models and cost matrices. Simulation results show that switching according to an arbitrary sequence yields different tracking performance while maintaining overall system stability.

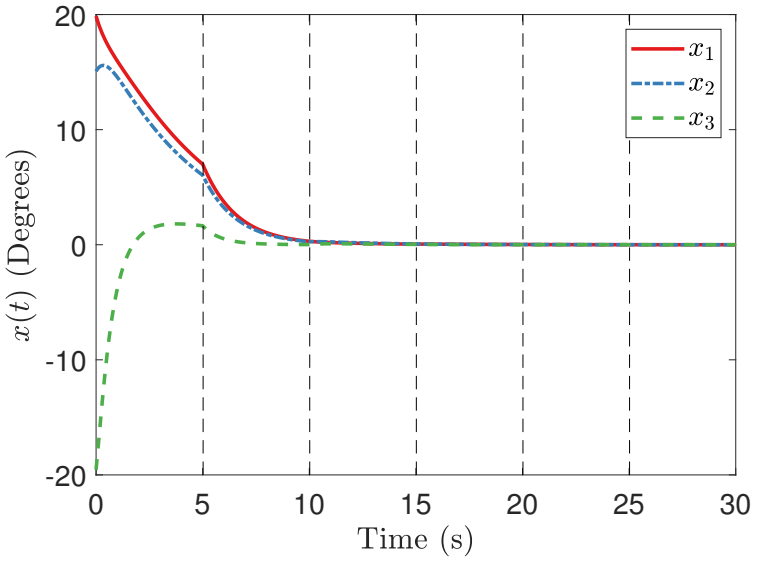


Figure 5-1. System states. The vertical dashed lines represent the time instances at which the mode was switched.

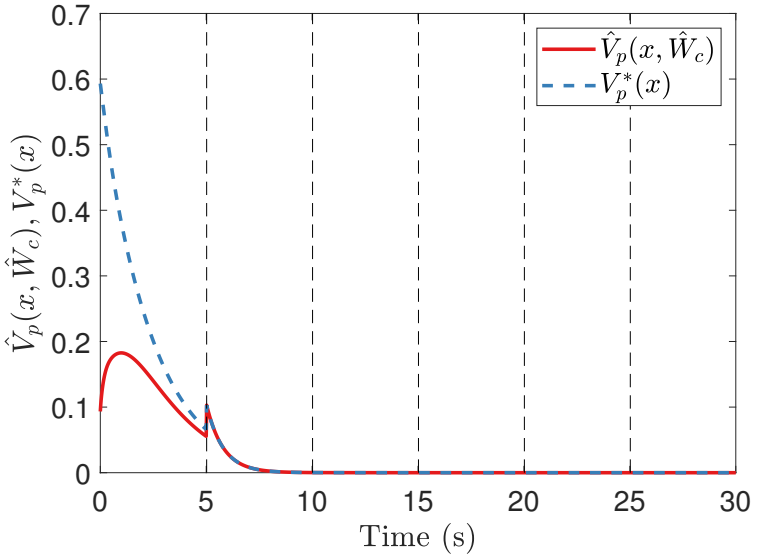


Figure 5-2. Comparison of the analytical value functions, $V_p^*(x)$, and the approximate value functions, $\hat{V}_p\left(x, \hat{W}_{c,p}\right)$. The vertical dashed lines represent the time instances at which the mode was switched.

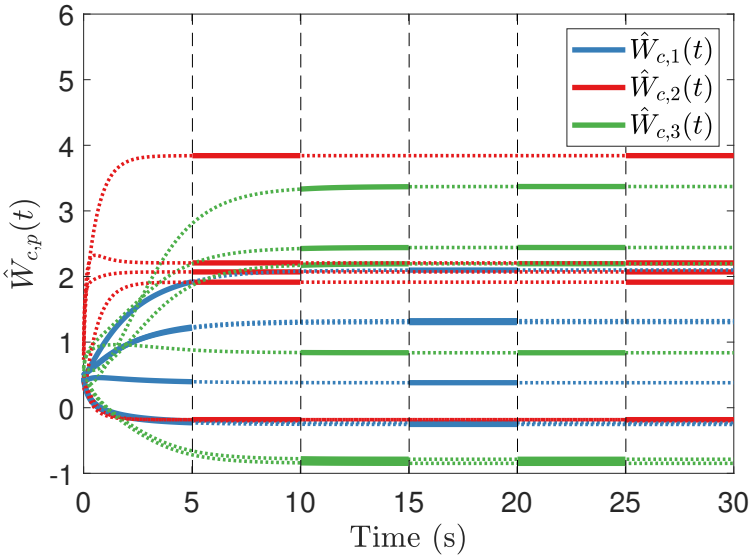


Figure 5-3. Critic weight estimates of each mode, $\hat{W}_{c,p}$. The vertical dashed lines represent the time instances at which the mode was switched. The active vector of critic weights is denoted with a solid instead of a dashed line.

CHAPTER 6
DEEP NEURAL NETWORK-BASED APPROXIMATE OPTIMAL TRACKING FOR UNKNOWN NONLINEAR SYSTEMS

Results in [49] and [50] leverage a multi-timescale deep model reference adaptive controller. Similarly, the method in [52] uses a multi-timescale DNN to estimate the unknown system dynamics, which facilitates a trajectory tracking objective. In [52], a gradient-based adaptation policy is used to estimate the output layer weights of the DNN in real-time. Simultaneous to real-time execution, input-output data is stored and used to update the inner-layer weights using traditional offline DNN function approximation methods.

However, the adaptive update policy in [52] cannot be easily extended to system identification within the ADP framework. To prove stability of the overall system with an ADP controller, the adaptive update policy of the output-layer DNN weights must include the CL modification from [57], which complicates the stability analysis (cf., model-based ADP analyses in [19] and [48]).

The primary contribution of this chapter is to analyze the stability of tracking problem while using the multi-timescale DNN system identification approach. Simulation results are presented to illustrate the effectiveness of the developed technique in comparison to existing ADP-based results.

6.1 DNN-based System Identification

There exist numerous DNN architectures can approximate continuous functions on a compact set; the ability to do so is based on universal approximation theorems that can be invoked case-by-case for specific DNN architectures [75]. The drift dynamics f

can be approximated on a compact set $\mathcal{C} \subset \mathbb{R}^n$ as[1]

$$f(x) = \theta^T \phi(\Phi^*(x)) + \epsilon_\theta^*(x) \ \forall x \in \mathcal{C}, \tag{6–1}$$

where $\theta \in \mathbb{R}^{h \times n}$ is an unknown bounded ideal output-layer weight matrix, $\phi : \mathbb{R}^p \to \mathbb{R}^h$ is an vector of activation functions, $\Phi^* : \mathbb{R}^n \to \mathbb{R}^p$ is the ideal unknown inner-layer features of the DNN, and $\epsilon_\theta^* : \mathbb{R}^n \to \mathbb{R}^n$ is a bounded function approximation error. For example, the unknown ideal DNN Φ^* can be expressed as $\Phi^*(x) = V_k \varrho_k(V_{k-1}, \varrho_{k-1}(V_{k-2}, \varrho_{k-2}(...x)))$, where $k \in \mathbb{N}$ denotes the number of inner-layers of the DNN, V_k and $\varrho_k(\cdot)$ denote the corresponding inner-layer weights and activation functions of the DNN, respectively.

Based on the DNN representation in (6–1), the i^{th} DNN-based estimate of the drift dynamics $\hat{f}_i : \mathbb{R}^n \times \mathbb{R}^{h \times n}, \to \mathbb{R}^n$ is defined as

$$\hat{f}_i\left(x, \hat{\theta}\right) = \hat{\theta}^T \phi\left(\hat{\Phi}_i(x)\right), \tag{6–2}$$

where $\hat{\theta} \in \mathbb{R}^{h \times n}$ is the estimate of the ideal output-layer weight matrix θ, and $\hat{\Phi}_i : \mathbb{R}^p \to \mathbb{R}^n$ is the i^{th} iteration selection of the inner features with user-selected activation functions and estimated internal-layer weights. To facilitate the convergence of the DNN-based online system identifier, (6–2) can be used to develop an estimator

$$\dot{\hat{x}} = \hat{f}_i\left(x, \hat{\theta}\right) + g(x)u + k_o \tilde{x}, \tag{6–3}$$

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

Using the universal function approximation property of NNs [60], there exists constant weights θ^* and known finite constants $\overline{\theta^*}$, $\overline{\epsilon_\theta^*}$, and $\overline{\nabla_x \epsilon_\theta^*} \in \mathbb{R}_{\geq 0}$, such that $\|\theta^*\| \leq \overline{\theta^*}$, $\sup_{x \in \mathcal{C}} \|\epsilon_\theta^*(x)\| \leq \overline{\epsilon_\theta^*}$, and $\sup_{x \in \mathcal{C}} \|\nabla_x \epsilon_\theta^*(x)\| \leq \overline{\nabla_x \epsilon_\theta^*}$.

[1] The subsequent stability analysis in Theorem 4.1 proves that if x is initialized within an appropriately-sized subset of $\mathcal{C}$, then it will remain in $\mathcal{C}$.

Assumption 6.1. The i^{th} user-selected inner-layer features of the DNN Φ^* and $\hat{\Phi}_i$ are selected such that $\Phi^*(x) - \hat{\Phi}_i(x) \leq \tilde{\Phi}_i(x)$, where $\tilde{\Phi}_i : \mathbb{R}^n \to \mathbb{R}^p$ is the inner-layer DNN function reconstruction error of the i^{th} iteration, and $\sup_{x \in \mathcal{C}, i \in \mathbb{N}} \left\| \tilde{\Phi}_i(x) \right\| \leq \overline{\tilde{\Phi}}$, where $\overline{\tilde{\Phi}} \in \mathbb{R}_{\geq 0}$ is a bounded constant for all i. Using the Mean Value Theorem, $\left\| \phi(\Phi^*(x)) - \phi\left(\hat{\Phi}_i(x)\right) \right\| \leq \overline{\nabla_x \phi}\, \overline{\tilde{\Phi}}$.

Assumption 6.2. A history stack of input-output data pairs $\{x_j, u_j\}_{j=1}^M$ and history stack of numerically-computed state derivatives $\left\{\dot{\bar{x}}_j\right\}_{j=1}^M$, which satisfies $\lambda_{\min}\left\{\sum_{j=1}^M \hat{\phi}_i\left(\hat{\Phi}_i(x_j)\right) \hat{\phi}_i\left(\hat{\Phi}_i(x_j)\right)^T\right\} > 0$ and $\left\|\dot{\bar{x}}_j - \dot{x}_j\right\| < \overline{d} \ \ \forall j$, are available *a priori* for each index j of x_j, where $\overline{d} \in \mathbb{R}_{>0}$ is a known constant and $\dot{x}_j \triangleq f(x_j) + g(x_j) u_j$ [57].

Remark 6.1. Availability of the system identification history stack (i.e., the tuple $\{x_j, u_j, \dot{\bar{x}}_j\}_{j=1}^M$) *a priori* is not necessary [48]. Assumption 6.2 is used to focus the scope of this chapter and simplify the subsequent stability analysis. A traditional, PE-based ADP controller (e.g., [10]) can be used during the initial stage in which Assumption 6.2 cannot be verified. Provided that the system is sufficiently exited and the history stack is recorded within a finite time, then the developed controller can be used. Switching between a PE-based controller and the developed controller results in a switched subsystem with one switching event. In this case, stability of the overall system is determined from the stability of the individual subsystems.

In the developed method, a DNN with uncertain output-layer parameters $\hat{\theta}$ is used to facilitate system identification in the sense that $\hat{F}$ approximates F. To enable convergence of $\hat{F}$ to F, CL-based parameter update laws are developed that use recorded data for learning. This CL strategy is leveraged to modify the output-layer weight update law in [52]. As shown in the subsequent stability analysis, this modification enables $\hat{\theta}$ to converge to a region containing θ. The output-layer DNN weight estimates are updated using a CL-based update law

$$\dot{\hat{\theta}} = \Gamma_\theta \phi\left(\widehat{\Phi}_i(x)\right) \tilde{x}^T + k_\theta \Gamma_\theta \sum_{j=1}^M \phi\left(\widehat{\Phi}_i(x_j)\right) \left(\dot{\bar{x}}_j - g_j(x_j) u_j - \hat{\theta}^T \phi\left(\widehat{\Phi}_i(x_j)\right)\right)^T, \tag{6–4}$$

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

Remark 6.2. While the contribution of this section focuses on updating the output layer weights in real-time, (re)training of the DNN system identifier is key to the DNN system identifier framework outlined in [52]. The history stack can be collected *a priori* and/or online. The history stack, which is also used to update the output layer weights in (6–4), can simultaneously be used to update the inner-layer features and weights of the DNN (i.e., update $\phi\left(\hat{\Phi}_i(x)\right)$ from i to $i+1$) iteratively. To improve the DNN estimate of the dynamics, additional data should be collected online and used to update the inner-layer features and weights.

6.2 Bellman Error

The HJB equation in (2–18) is equal to zero under optimal conditions; however, substituting (2–21), (2–22), and the approximated drift dynamics $\hat{f}_i\left(x,\hat{\theta}\right)$ into (2–18) results in a residual term $\hat{\delta}:\mathbb{R}^{2n}\times\mathbb{R}^{h\times n}\times\mathbb{R}^{L}\times\mathbb{R}^{L}\to\mathbb{R}$, which is referred to as the BE, defined as

$$\hat{\delta}\left(\zeta,\hat{\theta},\hat{W}_c,\hat{W}_a\right) \triangleq \hat{\mu}\left(\zeta,\hat{W}_a\right)^T R\hat{\mu}\left(\zeta,\hat{W}_a\right) + \overline{Q}(\zeta) + \nabla_\zeta \hat{V}\left(\zeta,\hat{W}_c\right)\left(\hat{F}_i\left(\zeta,\hat{\theta}\right) + G(\zeta)\,\hat{\mu}\left(\zeta,\hat{W}_a\right)\right), \tag{6–5}$$

where $\hat{F}_i:\mathbb{R}^{2n}\times\mathbb{R}^{h\times n}\to\mathbb{R}^{2n}$ is defined as

$$\hat{F}_i\left(\zeta,\hat{\theta}\right) \triangleq \left[\hat{f}_i\left(e+x_d,\hat{\theta}\right)^T - h_d(x_d)^T + u_d(x_d)^T g(e+x_d)^T, h_d(x_d)^T\right]^T. \tag{6–6}$$

The BE in (6–5) 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 (2–21) and (2–22) into (2–18), and subtracting from (6–5) yields the analytical form of the BE, given by

$$\hat{\delta}\left(\zeta,\hat{\theta},\hat{W}_c,\hat{W}_a\right) = -\omega^T\tilde{W}_c - W^{*T}\nabla_\zeta\sigma\left(F(\zeta) - \hat{F}_i\left(\zeta,\hat{\theta}\right)\right) + \frac{1}{4}\tilde{W}_a^T G_\sigma \tilde{W}_a + O\left(\epsilon,\nabla_\zeta\epsilon,\epsilon_\theta^*\right), \tag{6–7}$$

where $\omega : \mathbb{R}^{2n} \times \mathbb{R}^{L} \times \mathbb{R}^{h \times n} \to \mathbb{R}^{h}$ is defined as $\omega\left(\zeta, \hat{W}_a, \hat{\theta}\right) \triangleq \nabla_{\zeta}\sigma\left(\zeta\right)\left(\hat{F}_i\left(\zeta, \hat{\theta}\right) + G\left(\zeta\right)\hat{\mu}\left(\zeta, \hat{W}_a\right)\right)$, $O\left(\zeta\right) \triangleq \frac{1}{2}\nabla_{\zeta}\epsilon\left(\zeta\right)G_R\nabla_{\zeta}\sigma\left(\zeta\right)^T W^* + \frac{1}{4}G_{\varepsilon} - W^{*T}\nabla_{\zeta}\sigma\left(\zeta\right)\epsilon_{\theta}^*\left(e + x_d\right) - \nabla_{\zeta}\varepsilon\left(\zeta\right)F\left(\zeta\right)$, $G_R = G_R\left(\zeta\right) \triangleq G\left(\zeta\right)R^{-1}G\left(\zeta\right)^T$, $G_{\sigma} = G_{\sigma}\left(\zeta\right) \triangleq \nabla_{\zeta}\sigma\left(\zeta\right)G_R\left(\zeta\right)\nabla_{\zeta}\sigma\left(\zeta\right)^T$, and $G_{\varepsilon} = G_{\varepsilon}\left(\zeta\right) \triangleq \nabla_{\zeta}\epsilon\left(\zeta\right)G\left(\zeta\right)\nabla_{\zeta}\epsilon\left(\zeta\right)^T$.

Bellman Error Extrapolation

At each time instant $t \in \mathbb{R}_{\geq 0}$, the estimated BE in (6–5) and policy in (2–22) are evaluated using the current system state, critic estimate, and actor estimate to get the instantaneous BE and control policy, which are denoted by $\hat{\delta} \triangleq \hat{\delta}\left(\zeta, \hat{\theta}, \hat{W}_c, \hat{W}_a\right)$ and $\hat{\mu} \triangleq \hat{\mu}\left(\zeta, \hat{W}_a\right)$, respectively. However, using only the on-trajectory BE and control policy requires the traditional PE condition to be satisfied to show convergence.

To simulate PE and extrapolate BE over off-policy trajectories, the off-policy trajectories $\{\zeta_e : \zeta_e \in \Omega\}_{e=1}^{N}$ are selected, where $N \in \mathbb{N}$ denotes the number of extrapolated trajectories in Ω. The extrapolation points are selected *a priori* by the user and are state dependent. Using the extrapolated trajectories $\zeta_e \in \Omega$, the BE in (6–5) is evaluated such that $\hat{\delta}_e \triangleq \hat{\delta}\left(\zeta_e, \hat{\theta}, \hat{W}_c, \hat{W}_a\right)$. Let the tuple $(\Sigma_c, \Sigma_a, \Sigma_{\Gamma})$ define the extrapolation stacks corresponding to Ω such that $\Sigma_c \triangleq \frac{1}{N}\sum_{e=1}^{N}\frac{\omega}{\rho}\hat{\delta}$, $\Sigma_a \triangleq \frac{1}{N}\sum_{e=1}^{N}\frac{G_{\sigma e}^T\hat{W}_a\omega_e^T}{4\rho_e}$, and $\Sigma_{\Gamma} \triangleq \frac{1}{N}\sum_{e=1}^{N}\frac{\omega_e\omega_e^T}{\rho_e}$, where $\omega_e \triangleq \omega\left(\zeta_e, \hat{\theta}, \hat{W}_a\right)$, $\rho_e \triangleq \rho\left(\zeta_e, \hat{\theta}, \hat{W}_a\right) = 1 + \nu\omega_e^T\Gamma\omega_e$, $\Gamma \in \mathbb{R}^{L \times L}$ is a subsequently defined user-initialized learning gain, and Assumption 6.3 is provided to facilitate the subsequent stability analysis.

Assumption 6.3. There exist a finite set of trajectories $\{\zeta_e : \zeta_e \in \Omega\}_{e=1}^{N}$ such that $0 < \underline{c} \triangleq \inf_{t \in \mathbb{R}_{\geq 0}} \lambda_{\min}\{\Sigma_{\Gamma}\}$ for all $t \in \mathbb{R}_{\geq 0}$.

Remark 6.3. The constant $\underline{c}$ is the lower bound of the value of each input-output data pairs' minimum eigenvalues.

Remark 6.4. The computational expense of BE extrapolation can be reduced by using the sparse BE extrapolation method in [64].

6.3 Actor and Critic Weight Update Laws

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

$$\dot{\hat{W}}_c = -\eta_{c1}\Gamma\frac{\omega}{\rho}\hat{\delta} - \eta_{c2}\Gamma\Sigma_c, \tag{6–8}$$

$$\dot{\Gamma} = \left(\lambda\Gamma - \eta_{c1}\frac{\Gamma\omega\omega^T\Gamma}{\rho^2} - \Gamma\eta_{c2}\Sigma_\Gamma\Gamma\right)\mathbf{1}_{\{\underline{\Gamma}\leq\|\Gamma\|\leq\overline{\Gamma}\}}, \tag{6–9}$$

$$\dot{\hat{W}}_a = -\eta_{a1}\left(\hat{W}_a - \hat{W}_c\right) - \eta_{a2}\hat{W}_a + \eta_{c1}\frac{G_\sigma^T\hat{W}_a\omega^T}{4\rho}\hat{W}_c + \eta_{c2}\Sigma_a\hat{W}_c, \tag{6–10}$$

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. $\|\Gamma(t)\|$ is upper and lower bounded by two user-defined saturation constants, $\overline{\Gamma}$ and $\underline{\Gamma}$, respectively. Using the indicator function in (6–9) ensures that $\underline{\Gamma} \leq \|\Gamma(t)\| \leq \overline{\Gamma}$ for all $t \in \mathbb{R}_{>0}$, where $\underline{\Gamma} \in \mathbb{R}_{>0}$. The indicator function in (6–9) can be removed provided additional conditions are met [53].

6.4 Stability Analysis

Note that the function $\overline{Q}$ and, therefore, V^* are positive semidefinite. Hence, V^* is not a valid Lyapunov function. However, the result in [61] 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. Hence, $V^*(0,t) = 0$ 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. Let $Z \in \mathbb{R}^{2n+2L+hn}$ denote a concatenated state defined as $Z \triangleq \left[e^T, \tilde{W}_c^T, \tilde{W}_a^T, \tilde{x}^T, \text{vec}\left(\tilde{\theta}\right)^T\right]^T$. Let $V_L : \mathbb{R}^{2n+2L+hn} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ be a candidate Lyapunov function defined as

$$V_L(Z,t) \triangleq V_{na}^*(e,t) + \frac{1}{2}\tilde{W}_c^T\Gamma(t)^{-1}\tilde{W}_c + \frac{1}{2}\tilde{W}_a^T\tilde{W}_a + \frac{1}{2}\tilde{x}^T\tilde{x} + \frac{1}{2}\text{tr}\left(\tilde{\theta}^T\Gamma_\theta^{-1}\tilde{\theta}\right). \tag{6–11}$$

Using the properties of $V_{na}^{*}(e,t)$ and [67, Lemma 4.3], then (6–11) be bounded as $\underline{v}_l(\|Z\|) \leq V_L(Z,t) \leq \overline{v}_l(\|Z\|)$ for class $\mathcal{K}_\infty$ functions $\underline{v}_l, \overline{v}_l : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Using (6–9), the normalized regressors $\frac{\omega}{\rho}$ and $\frac{\omega_i}{\rho_i}$ can be bounded as $\sup_{t\in\mathbb{R}_{\geq 0}} \left\|\frac{\omega}{\rho}\right\| \leq \frac{1}{2\sqrt{\nu\underline{\Gamma}}}$ for all $\zeta \in \Omega$ and $\sup_{t\in\mathbb{R}_{\geq 0}} \left\|\frac{\omega_i}{\rho_i}\right\| \leq \frac{1}{2\sqrt{\nu\underline{\Gamma}}}$ for all $\zeta_i \in \Omega$. 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 $\sup_{\zeta\in\Omega}\|G_\sigma\| \leq \left(\overline{\nabla_\zeta \sigma G}\right)^2 \lambda_{\max}\{R^{-1}\} \triangleq \overline{G_\sigma}$, respectively. Furthermore, g_d^+ can be bounded as $\sup_{t\in\mathbb{R}_{\geq 0}}\left\|g_d^+\right\| \leq \overline{g_d^+}$.

Theorem 6.1. *Given the dynamics in (2–1), given that Assumptions 6.2 and 6.3 are satisfied, and given that the sufficient gains conditions*

$$\eta_{a1} + \eta_{a2} \geq (\eta_{c1} + \eta_{c2}) \frac{\overline{W}\,\overline{G_\sigma}}{\sqrt{\nu\underline{\Gamma}}}, \tag{6–12}$$

$$\underline{c} \geq 4\frac{\eta_{a1}}{\eta_{c2}} + \frac{(\eta_{c1}+\eta_{c2})^2 \overline{W^*}^2 \overline{G_\sigma}^2}{4\nu\underline{\Gamma}(\eta_{a1}+\eta_{a2})} + \frac{3(\eta_{c1}+\eta_{c2})^2 \overline{W}^2 \overline{\nabla_\zeta \sigma}^2 \left(\overline{\phi} + \overline{g_d^+}\,\overline{\phi}\overline{g}\right)^2}{2\eta_{c2} k_\theta \nu\underline{\Gamma} \lambda_{\min}\left\{\sum_{j=1}^{M} \phi\left(\widehat{\Phi}_i(x_j)\right)\phi\left(\widehat{\Phi}_i(x_j)\right)^T\right\}}, \tag{6–13}$$

$$\nu_l^{-1}(l) < \overline{v}_l^{-1}\left(\underline{v}_l(\|Z\|)\right), \tag{6–14}$$

are satisifed, where l is a known positive constant, then the system state ζ, weight estimation errors $\tilde{W}_c$ and $\tilde{W}_a$, state estimation error $\tilde{x}$,and output-layer weight matrix error $\tilde{\theta}$ are UUB. Hence, the applied control policy $\hat{u}$ converges to a neighborhood of the optimal control policy u^*.

Proof. Let $r \in \mathbb{R}_{>0}$ be the radius of a compact ball $\chi \subset \mathbb{R}^{2n+2L+hn}$ centered at the origin. Using (2–13), $\dot{V}^*(\zeta) = \nabla_\zeta V^*(\zeta)(F(\zeta) + G(\zeta)\mu)$, (6–3), (6–4), (6–8)-(6–10), Young's Inequality, nonlinear damping, Assumption 6.2, Assumption 6.3, the sufficient conditions in (6–12) and (6–13) yields $\dot{V}_L \leq -\nu_l(\|Z\|), \forall \nu_l^{-1}(l) \leq \|Z\| \leq \overline{v}_l^{-1}\left(\underline{v}_l(r)\right)\ \forall t \in \mathbb{R}_{\geq 0}$,where $\nu_l(\|Z\|) \triangleq \frac{1}{2}\underline{q}(\|e\|) + \frac{1}{16}\eta_{c2}\underline{c}\left\|\tilde{W}_c\right\|^2 + \frac{1}{16}(\eta_{a1}+\eta_{a2})\left\|\tilde{W}_a\right\|^2 + \frac{k_o}{4}\|\tilde{x}\|^2 + \frac{k_\theta}{6}\lambda_{\min}\left\{\sum_{j=1}^{M}\phi\left(\widehat{\Phi}_i(x_j)\right)\phi\left(\widehat{\Phi}_i(x_j)\right)^T\right\}\left\|\text{vec}\left(\tilde{\theta}\right)\right\|^2$. Since the discontinuities in the update laws in (6–3), (6–4), and (6–8)-(6–10) are piecewise continuous in time and (6–11)

is a common Lyapunov-like function across each DNN iteration i, [67, Theorem 4.18] can be invoked to conclude that Z is UUB such that $\limsup_{t\to\infty} \|Z\| \leq \underline{v}_l^{-1}\left(\overline{v}_l\left(\nu_l^{-1}(l)\right)\right)$ and $\hat{\mu}$ converges to a neighborhood around the optimal policy μ^*. 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 $\mu \in \mathcal{L}_\infty$.

The result in [67, Theorem 4.18] can be invoked to show that every trajectory $Z(t)$ that satisfies the initial condition $\|Z(0)\| \leq \overline{v}_l^{-1}(\underline{v}_l(r))$ 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.[2] 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 DNN-based system identification. □

Remark 6.5. For insight into satisfying the conditions in (6–12)-(6–14), see [48].

6.5 Simulation Results

The following section will apply the developed technique to a linear quadratic tracking (LQT) problem, which has a cost function $r(\zeta, \mu) = e^T Q e + \mu^T R \mu$. The linear system

$$\dot{x} = \begin{bmatrix} -1 & 1 \\ -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{6–15}$$

is studied in the following simulation. Since linear system dynamics are used, an analytical solution to the HJB equation in (2–18) can be calculated for comparison purposes. The linear system in (6–15) was selected because it has been used in previous ADP works (e.g., [48]) and an analytical solution to the HJB equation in (2–18) can be calculated for comparison purposes. The control objective is to track the time-varying desired trajectory $x_d(t) = [4\sin(t), 4\cos(t) + 4\sin(t)]^T$ and to minimize the infinite horizon cost function in (2–16).

[2] See [16, Algorithm A.2] for discussion on establishing the compact set χ.

The drift dynamics are unknown and approximated using the developed DNN-based system identification method. The DNN used in this simulation was composed of 4 layers, each with 10, 6, 7, and 2 neurons, respectively. The DNN architecture is illustrated in Figure 6-1. Note that the DNN architecture utilizes jump connections that are often seen in residual NNs. Since the developed framework is constructed to permit DNN structures with a linear output layer, this parameterization matches that in (6–1). The first, second, and third layers use Elliot symmetric sigmoid, logarithmic sigmoid, and tangent sigmoid activation functions, respectively. Additionally, biases are included in the first, second, and third layers. The learning rate (i.e., the learning gain parameter used to determine the step size in retraining the DNN weights at each iteration) was fixed as $\eta = 0.001$. The mean squared error was used as the loss function for training. The Levenberg-Marquardt algorithm was used to train the weights of the DNN. For each DNN training iteration, 70% of the data was used for training, 15% was used for validation, and 15% was used for testing.

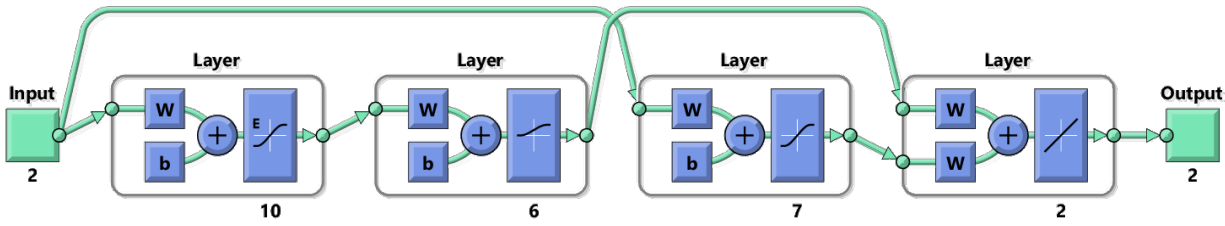


Figure 6-1. The DNN is composed of 4 layers, each with 10, 6, 7, and 2 neurons, respectively.

The cost function in (2–17) is selected as $r(\zeta, \mu) = e^T Q e + u^T R u$, where $Q = 5 \cdot I_2$ and $R = 1$. The basis selected for value function approximation is $\sigma(\zeta) = [e_1^2,\ e_1 e_2,\ e_1 x_{d1},\ e_1 x_{d2},\ e_2^2,\ e_2 x_{d1},\ e_2 x_{d2}]^T$. $N = 50$ BE extrapolation points were uniformly selected across $\Omega \triangleq \{\zeta \in \mathbb{R}^4 : -15 < e_{1,2} \leq 15,\ -4 < x_{d1,2} \leq 4,\}$. The initial conditions used for the simulated system are $x(0) = [-1,\ 1]^T$, $\hat{x}(0) = \mathbf{1}_7$, $\hat{W}_c(0) = \mathbf{1}_7$, $\hat{W}_a(0) = \mathbf{1}_7$, $\Gamma(0) = 2500 \cdot I_7$, and $\hat{\theta}(0) = \mathbf{1}_{13\times 2}$. The gains were selected as $\eta_{c1} = 0.005$, $\eta_{c2} = 0.1$, $\eta_{a1} = 10$, $\eta_{a2} = 0.1$, $\lambda = 0.4$, $\nu = 0.005$, $\overline{\Gamma} = 10^4$, $\underline{\Gamma} = 0.1$, $k_\theta = 30$, $k_o = 10$ and $\Gamma_\theta = 20 \cdot I_7$.

The following simulation results are divided into two sections. Section 6.5.1 provides qualitative and quantitative comparisons between the different ADP methods and the analytical optimal control policy; to focus the presented figures, none of the DNN methods update their inner features online. Section 6.5.2 provides a more direct comparison between the randomly initialized DNN ADP methods; one updates the inner-layer weights online and the other does not.

6.5.1 ADP Simulation Comparisons

This section presents simulation results for an analytical (Analyt.) optimal control policy, exact model knowledge (EMK) ADP, linearly parameterizable (LP) ADP, randomly initialized DNN ADP, transfer learning DNN ADP, and pretrained DNN ADP. All of the ADP methods in this simulation comparison are model-based (i.e., use BE extrapolation). The analytical optimal control policy calculates feedback gains *a priori* and is a baseline measurement for the best possible optimal control performance. EMK ADP uses exact model knowledge of $f(x)$, so the results present the best possible performance for an ADP-based controller. LP ADP assumes that $f(x)$ is linearly parameterizable (i.e., $f(x) = Y(x)\theta$, where $Y(x)$ exactly parameterizes the dynamics), as typically seen in adaptive control literature [76]. LP requires some, but not exact model knowledge. Furthermore, LP dynamics are a special class of dynamics. Since the dynamics in this simulation happen to be LP, and LP is common in adaptive control literature, it is included in this comparison study. Pretrained DNN ADP pretrains the DNN on the actual dynamics $f(x)$ (e.g., (6–15)).This method requires some data from the system *a priori*, which may not always be possible. Transfer learning DNN ADP pretrains the DNN on a dynamical system that is similar, but not exactly the same, as the one it will be implemented on (e.g., (6–15)). Specifically, for this training case $A = \begin{bmatrix} -2 & -2 \\ -1 & -2 \end{bmatrix}$ and B is identical. The pretraining can be done on a similar system or from prior experiments *a priori*. However, transfer learning DNN ADP does not require

exact model knowledge. Randomly initialized DNN ADP uses the developed DNN ADP controller, but there is no pretraining. The initial values of the inner and output-layer weights are arbitrarily selected from the interval $[-1, 1]$. This simulation case shows that without any model knowledge, the desired trajectory can still be tracked.

Figure 6-2 compares the above methods qualitatively. The analytical and EMK ADP methods were expected to perform best, and are used for comparison purposes. Both of these methods require exact model knowledge, whereas the other methods do not. The LP ADP performs very well, but also requires some knowledge about the model. The pretrained DNN ADP method performs well, but does not perform as well as EMK ADP. The transfer learning ADP performs slightly better than the randomly initialized ADP, but both are noticeably worse than the other methods, specifically around the transient changes in error caused by a rapid change in the desired trajectory. The random and transfer learning DNN cases did not successfully identify the dynamics in regions with a rapidly changing desired trajectory.

Table 6-1 quantitatively compares the performance of each method. Column one compares the total error of each simulation (i.e., $\int_0^{15+1} e(\tau)\ \mathrm{d}\tau$). Recall that the analytical and EMK ADP were expected to have the best performance. Pretrained DNN ADP performs the best out of the methods without model exact model knowledge, followed by LP ADP, transfer learning DNN ADP, and random DNN ADP, in that order. The second column of Table 6-1 compares the ADP methods with the integral of the difference between their state trajectory and the EMK ADP state trajectory. Similarly pretrained DNN ADP performs the best, followed by LP DNN ADP, transfer learning DNN ADP, and random DNN ADP, in that order. A noticeable trend in Table 6-1 is that if a system has more model knowledge *a priori*, then performance improves.

Table 6-1. Simulation Results and ADP Comparison

Control Type	Total Integral Error	Integral Error from EMK
Analytical Solution	3.43	–
EMK ADP	5.04	–
LP ADP	5.19	0.27
Pretrained DNN ADP	4.93	0.22
Transfer Learning DNN ADP	7.00	2.91
Random DNN ADP	10.57	6.33

6.5.2 Multi-Timescale Simulation Results

This simulation section presents a simulation case in which the inner-layer weights are updated. This section will present the performance improvements that occur after online retraining. For comparison purposes, all of the gains, costs, etc. are identical to those in Section 6.5.1. Both cases are initialized as the random DNN ADP. The internal weights of one controller are updated in one case, which is henceforth referred to as "retrained DNN." The internal weights of the other controller are not updated in the other case, which is henceforth referred to as "random DNN." A history stack of DNN training data is collected for 5 seconds. The time of 5 seconds was arbitrarily selected. Collecting more data, generally, should result in improved training of the inner-layer weights at the expense of additional computation time. After 5 seconds, the internal DNN weights begin retraining. Once retraining is complete, the new internal weights are implemented, at which point the difference between the retrained DNN and random DNN controllers is notable. The retrained DNN ADP controller has significantly better performance compared to the random case. Figure 6-3 illustrates the magnitude of the error between the retrained DNN and random DNN case. The cases are identical, as intended, until the updated internal DNN weights are implemented. The retrained DNN has improved performance after retraining completes.

Figure 6-4 depicts a phase plot that compares the performance of the retrained DNN and random DNN ADP controllers. The red, random DNN ADP controller tracks the general shape of the desired trajectory; however, its poor approximation of $f(x)$ led to increased tracking error. In contrast, the blue, retrained DNN ADP controller progressively converges toward the desired trajectory, suggesting that the DNN retraining facilitated learning of $f(x)$.

Figures 6-5a and 6-5b show the actor and critic weights $\hat{W}_c$ and $\hat{W}_a$, respectively, for the retrained DNN ADP simulation case. While a majority of the learning occurs during the first 1-2 seconds, additional learning takes place after DNN retraining.

Figure 6-6 shows the output-layer weight estimates of the DNN $\hat{\theta}$ for the retrained DNN simulation case. Note that the Lyapunov-based analysis in Theorem 1 does not guarantee convergence of the DNN weights to their actual values, which are unknown. Learning initially occurs in the first 5 seconds. Once the internal weight are trained (in the retrained DNN case), then the output-layer weights adjust to compensate for changes in the inner-layer DNN.

Figure 6-7 compares the analytically-obtained optimal value function $V^*(\zeta)$ and the value function estimate for the retrained case $\hat{V}\left(\zeta, \hat{W}_c\right)$. Note that $V^*(\zeta)$ is not the same as the analytical simulation in Section 6.5.1. The analytical simulation in Section 6.5.1 applies the optimal value function and optimal control policy, which have been determined *a priori*, directly. In contrast, the $V^*(\zeta)$ presented in Figure 6-7 uses the same analytically-determined value function V^* that is instead analyzed along the trajectories generated by the retrained DNN ADP controller. Initially, the value function is poorly approximated, which is likely due to the poor approximation of the dynamics $f(x)$. The poor approximation is clear around the 6 second mark. After the retraining, the value function approximation improves.Figure 6-8 compares the analytically-determined optimal control policy and approximate optimal control policy for the retrained case $u^*(\zeta)$ and $\hat{u}\left(\zeta, \hat{W}_a\right)$, respectively. Like the comparison in Figure 6-7,

the values of $u^*(\zeta)$ are not from the analytical simulation in Section 6.5.1. u^* is the same function as the control policy in the analytical simulation case, but it is evaluated along the trajectory generated by the retrained DNN ADP controller. $\hat{u}\left(\zeta, \hat{W}_a\right)$ tracks $u^*(\zeta)$ better after the retraining; however, there are isolated increases in the control effort, which are likely due to the poor approximation of $f(x)$ by the DNN.

Overall, these simulation studies confirm the effectiveness of a DNN-based ADP controller with iterative inner-layer weight updates. Furthermore, the simulation results suggest that the DNN-based methods are not necessarily superior to having exact model knowledge or a known LP case, which is expected. The benefit to the developed technique is that the drift dynamics $f(x)$ can be learned without any model knowledge *a priori*. When some model knowledge of the dynamics are known, then the developed method may not always be the best solution, but it is a is a promising technique for instances when the system model is completely unknown.

6.6 Concluding Remarks

This chapter develops a framework for using the DNN-based system identifier in [52] within the model-based ADP framework, which was initially developed in [48], to solve the infinite horizon optimal tracking control problem. A CL-based continuous-time update law is used to update the weights of the DNN. A Lyapunov-based analysis was performed to prove UUB convergence of the trajectory tracking error and DNN weights estimation error to a neighborhood of zero. Hence, approximation of the applied control policy to the optimal control policy is proven. Simulations results are presented to illustrate the performance of the developed method compared to existing methods. Future works will investigate using a DNN to estimate the value function in conjunction with a DNN-based system identifier.

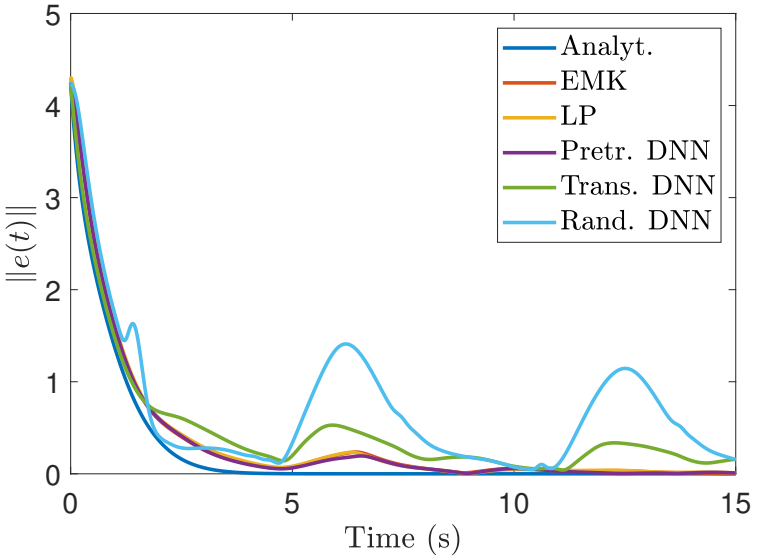


Figure 6-2. Error comparisons between different ADP methods and the analytical solution.

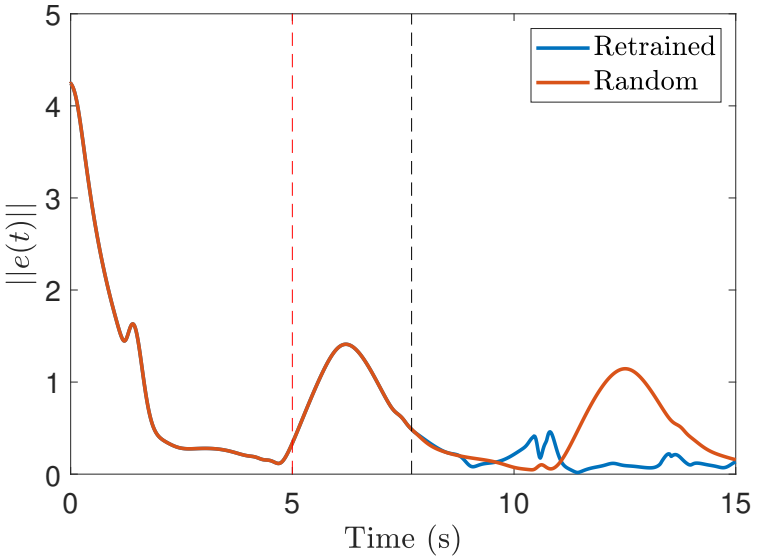


Figure 6-3. Error comparison of DNN ADP with and without an online internal weight update. The red dashed line at $t = 5$ seconds represents the beginning of the retraining, and the black dashed line at approximately $t = 7.5$ seconds represents the end the retraining and when the new internal DNN weights are implemented.

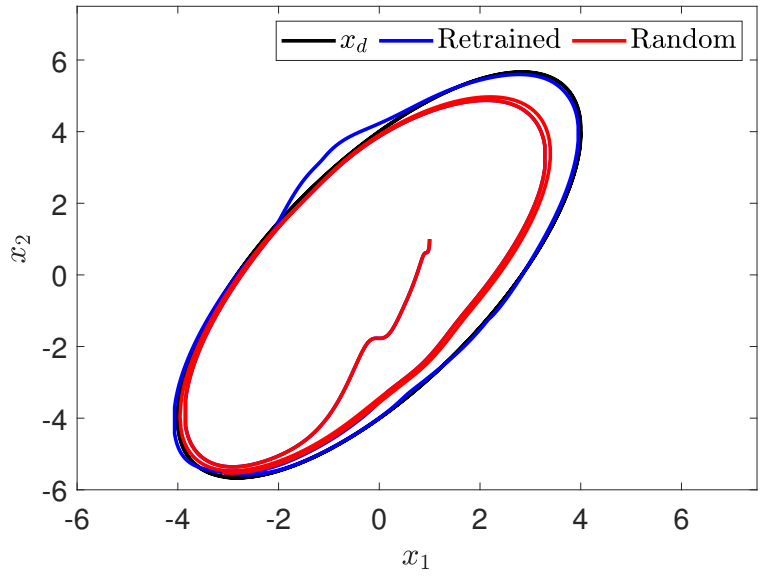
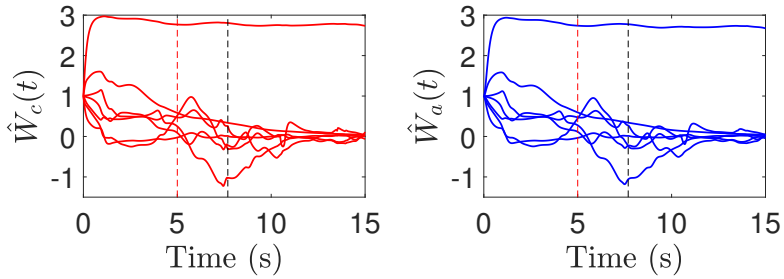


Figure 6-4. Phase plot comparison of retrained DNN and random DNN ADP trajectories.



(a) Approximated critic weights $\hat{W}_c$.

(b) Approximated actor weights $\hat{W}_a$.

Figure 6-5. The actor and critic weights $\hat{W}_c$ and $\hat{W}_a$, respectively, of the retrained case. As detailed in 6-3, the red dashed line represents the beginning of the retraining and the black dashed line represents when the new internal DNN weights are implemented.

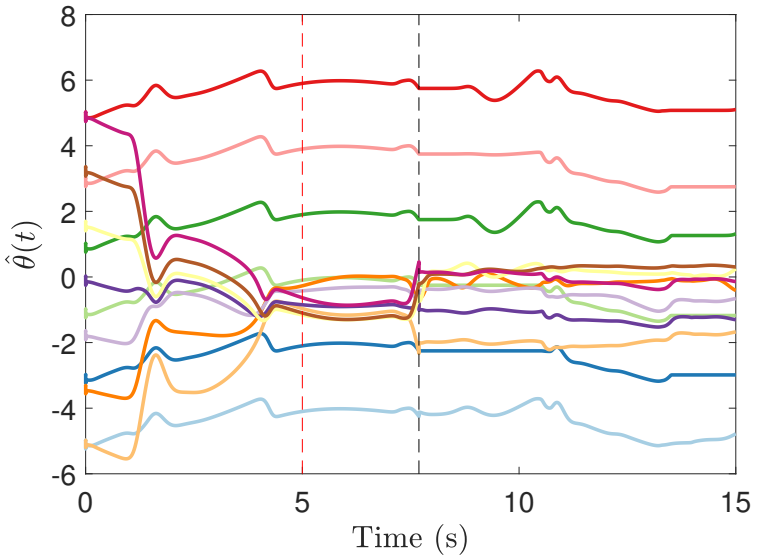


Figure 6-6. Output layer DNN weights $\hat{\theta}$ for the retrained DNN case. As stated in Figure 6-3, the red and black dashed lines represents the beginning of the retraining and implementation of the new internal DNN weights, respectively.

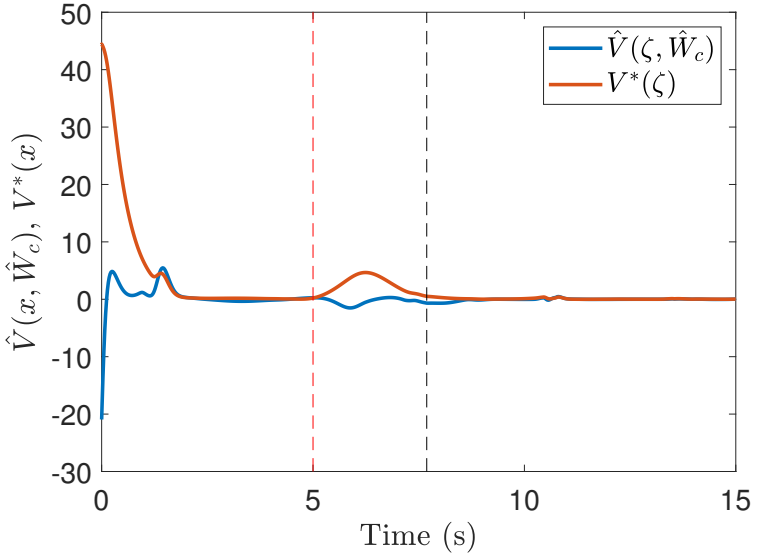


Figure 6-7. Comparison of analytical and approximated value function $V^*(\zeta)$ and $\hat{V}\left(\zeta, \hat{W}_c\right)$, respectively. As stated in Figure 6-3, the red and black dashed lines represents the beginning of the retraining and implementation of the new internal DNN weights, respectively.

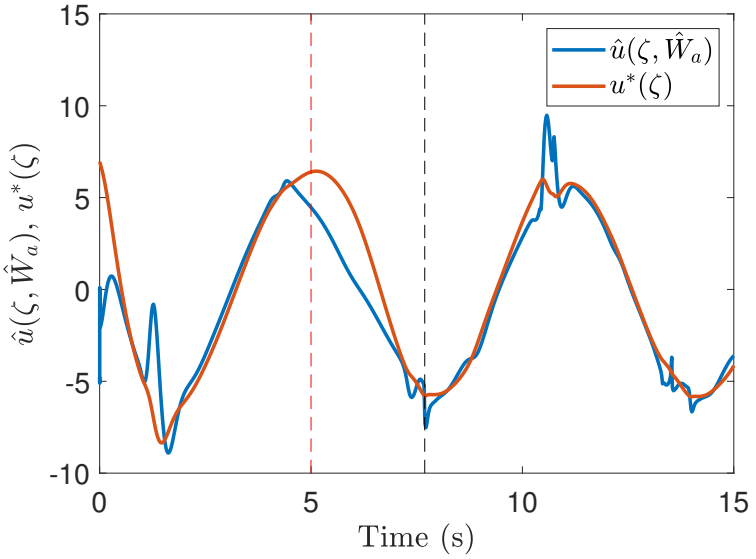


Figure 6-8. Comparison of analytical and approximated optimal control policy $u^*(\zeta)$ and $\hat{u}\left(\zeta, \hat{W}_a\right)$. As stated in Figure 6-3, the red and black dashed lines represents the beginning of the retraining and implementation of the new internal DNN weights, respectively.

CHAPTER 7
CONCLUSIONS

RL is a valuable tool for learning parametric uncertainties and approximating optimal control policies in dynamical systems. The application of RL-based techniques to nonlinear dynamical systems is complicated due to practical system constraints. This dissertation develops methods that advance the state-of-the-art of ADP-based controllers. Specifically, sparse BE extrapolation is leveraged to reduce the computational load associated with BE extrapolation while retaining its improved regional approximation properties, barrier function-based system constraints are leveraged to improve system safety during the online learning phase, the ADP framework has been extended to nonlinear switched systems, and a DNN-based system identifier is used to improve function approximation. The barrier function, switched systems, and DNN system identifier add significant computational expenses to their respective variation of ADP; hence, the sparse BE extrapolation makes these variations feasible in real-time applications.

In Chapter 3, a method is developed that significantly decreases the computational expense of R-MBRL techniques. This chapter has significant practical application, as it may be infeasible for systems with limited computational resources, such as mobile robots or multirotor aircraft, to implement traditional R-MBRL algorithms. While the concept of BE extrapolation is retained from R-MBRL, the developed method introduces the idea of sparse BE extrapolation within subsets of the operating domain. That is, sparse BE extrapolation is used within user-defined regions of the operating domain; within each region, a subset of BE extrapolation points are used to provide sufficient excitation to prove convergence. A limitation of Chapter 3 is a lack of guidance on how to segment the operating domain and which nodes of the NN to modify to promote sparsity while maintaining sufficient data richness. Currently, these tasks are done iteratively with a tuning-like method. Future research will investigate the idea of having an algorithm determine how to segment the state space, how many BE extrapolation

points to select, the locations of the BE extrapolation points, and what basis function to use for sparse BE extrapolation.

Chapter 4 pairs the computational benefits of sparse BE extrapolation with BFs to provide safety certificates. The BFs are enforced by a state space transformation that artificially increases the cost of the state around user-defined boundaries. Previous work in BF-augmented ADP has required the restrictive PE condition to be satisfied to prove system convergence. The result in Chapter 4 extends existing work to relax the PE condition via BE extrapolation. However, the state space transformation used to generate the BFs require additional computations; hence, the sparse BE extrapolation method in 3 is added to offset the computational expense of the state space transformation. Future work will investigate the case in which the initial state of the system lies outside of the user-defined barriers, which may require a switched systems-based Lyapunov-like analysis.

In Chapter 5 a framework is developed that investigates ADP applied to families of switched systems. Current ADP techniques are designed for a single continuous systems; however, numerous practical systems have multiple sets of continuous dynamics that are discretely activated (i.e., switched to) by a time-based signal. Hence, the stabilizing ADP-based controller on the first set of dynamics (e.g., subsystem) may not stabilize the second active subsystem. Hence, multiple ADP learning systems must be implemented in real-time to separately approximate each subsystem's value function to show overall system convergence.

Unlike previous ADP results, it is necessary to evaluate the stability of each subsystem to determine convergence of the overall switched system. Typically, in switched systems, stability is proved with the existence of a common Lyapunov function. However, for ADP, a common Lyapunov function has not yet been determined. Hence, a switched systems analysis with multiple Lyapunov functions must be performed. In the process of developing this analysis, it became clear that the broader problem, which

is the analysis of switched individually UUB stable subsystems with multiple Lyapunov functions has not yet been addressed. Hence, we have developed a general theorem to show stability of nonautonomous switched UUB stable subsystems via multiple Lyapunov functions. This theorem, and its application to ADP, has thus far been useful for the development of an ADP-based functional electrical stimulation cycling controller and ADP-based hierarchical RL supervisory controller. While the aforementioned result is significant, a more complicated problem remains for state-based switching (cf. time-based switching), which will be the focus of future research efforts.

Chapter 6 incorporates multi-timescale DNN-based system identifier within the R-MBRL framework. While the general DNN used in this framework has been used in existing robust control results, its use in ADP requires parameter (i.e., output-layer weight error) convergence, which the robust control results do not require. Hence, the output-layer weights are modified with a CL-based adaptation policy to faciliatate UUB convergence of the overall system. Furthermore, the work in this chapter presents comparisons between the DNN identifier and alternative methods, including linear-in-the-parameters system identification and exact model knowledge. Generally, single-layer NN approximations are less accurate than DNN approximations are. Hence, to further capture nonlinearities in the value function approximation, future efforts will use two separate multi-timescale DNNs for value function approximation and online system identification to better approximate optimal control policies via R-MBRL.

All of these chapters consider nonsmooth modifications to the traditional R-MBRL method. However, the methods presented in this dissertation do not consider state-based switched, hybrid dynamics, and stochastic systems. These omissions limit the application of ADP to a broader class of practical systems. This motivates investigation into the application of ADP-based controller to state-based switching and hybrid systems. Patchy Lyapunov functions, which have been used in the analysis of hybrid systems, may be a useful tool in the analysis of general state-based switching for UUB

stable subsystems [77]. However, significant efforts will be required to mature such a result for general classes of UUB stable subsystems to subsystems with ADP-based controllers. With the rise of RL within the machine learning community over the past decade, it is clear that the methods in this dissertation will play a role in the development of future intelligent data-based autonomous systems.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[2] K. Doya, "Reinforcement learning in continuous time and space," *Neural Comput.*, vol. 12, no. 1, pp. 219–245, 2000.

[3] R. Padhi, N. Unnikrishnan, X. Wang, and S. Balakrishnan, "A single network adaptive critic (SNAC) architecture for optimal control synthesis for a class of nonlinear systems," *Neural Netw.*, vol. 19, no. 10, pp. 1648–1660, 2006.

[4] 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. Part B Cybern.*, vol. 38, pp. 943–949, 2008.

[5] 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, 2009.

[6] T. Dierks, B. Thumati, and S. Jagannathan, "Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence," *Neural Netw.*, vol. 22, no. 5-6, pp. 851–860, 2009.

[7] P. Mehta and S. Meyn, "Q-learning and pontryagin's minimum principle," in *Proc. IEEE Conf. Decis. Control*, pp. 3598–3605, Dec. 2009.

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

[9] H. Zhang, L. Cui, X. Zhang, and Y. Luo, "Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method," *IEEE Trans. Neural Netw.*, vol. 22, pp. 2226–2236, Dec. 2011.

[10] S. Bhasin, R. Kamalapurkar, M. Johnson, K. G. Vamvoudakis, F. L. Lewis, and W. E. Dixon, "A novel actor-critic-identifier architecture for approximate optimal control of uncertain nonlinear systems," *Automatica*, vol. 49, pp. 89–92, Jan. 2013.

[11] H. Zhang, L. Cui, and Y. Luo, "Near-optimal control for nonzero-sum differential games of continuous-time nonlinear systems using single-network adp," *IEEE Trans. Cybern.*, vol. 43, no. 1, pp. 206–216, 2013.

[12] H. Zhang, D. Liu, Y. Luo, and D. Wang, *Adaptive Dynamic Programming for Control Algorithms and Stability*. Communications and Control Engineering, London: Springer-Verlag, 2013.

[13] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[14] D. Liberzon, *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2012.

[15] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis, *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*. The Institution of Engineering and Technology, 2013.

[16] R. Kamalapurkar, P. S. Walters, J. A. Rosenfeld, and W. E. Dixon, *Reinforcement learning for optimal feedback control: A Lyapunov-based approach*. Springer, 2018.

[17] D. Vrabie, *Online Adaptive Optimal Control For Continuous-time Systems*. PhD thesis, University of Texas at Arlington, 2010.

[18] K. G. Vamvoudakis, D. Vrabie, and F. L. Lewis, "Online adaptive algorithm for optimal control with integral reinforcement learning," *Int. J. of Robust and Nonlinear Control*, vol. 24, no. 17, pp. 2686–2710, 2014.

[19] R. Kamalapurkar, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for approximate optimal regulation," *Automatica*, vol. 64, pp. 94–104, 2016.

[20] P. He and S. Jagannathan, "Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 37, no. 2, pp. 425–436, 2007.

[21] H. Zhang, Q. Wei, and Y. Luo, "A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy hdp iteration algorithm," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 38, no. 4, pp. 937–942, 2008.

[22] 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.

[23] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Model-free q-learning designs for linear discrete-time zero-sum games with application to H_∞ control," *Automatica*, vol. 43, pp. 473–481, 2007.

[24] K. G. Vamvoudakis and F. L. Lewis, "Multi-player non-zero-sum games: Online adaptive learning solution of coupled hamilton-jacobi equations," *Automatica*, vol. 47, pp. 1556–1569, 2011.

[25] K. G. Vamvoudakis, F. L. Lewis, and G. R. Hudas, "Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality," *Automatica*, vol. 48, no. 8, pp. 1598–1611, 2012.

[26] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 10, pp. 1513–1525, 2013.

[27] B. Kiumarsi, F. L. Lewis, H. Modares, A. Karimpour, and M.-B. Naghibi-Sistani, "Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics," *Automatica*, vol. 50, pp. 1167–1175, Apr. 2014.

[28] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems," *Automatica*, vol. 50, no. 1, pp. 193–202, 2014.

[29] H. Modares and F. L. Lewis, "Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning," *Automatica*, vol. 50, no. 7, pp. 1780–1792, 2014.

[30] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Int. Conf. Artif. Intell. Stat.*, pp. 315–323, 2011.

[31] 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. Navig. Control Conf.*, Jan. 2017.

[32] S. A. Nivison and P. Khargonekar, "A sparse neural network approach to model reference adaptive control with hypersonic flight applications," in *Proc. AIAA Guid. Navig. Control Conf.*, p. 0842, 2018.

[33] 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 Conf. Decis. Control*, (Nice, Fr), pp. 1959–1964, Dec. 2019.

[34] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proc. Vol.*, vol. 40, no. 12, pp. 462–467, 2007.

[35] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *International Workshop on Hybrid Systems: Computation and Control*, pp. 477–492, Springer, 2004.

[36] A. Keshavarz, Y. Wang, and S. Boyd, "Imputing a convex objective function," in *Proc. IEEE Int. Symp. on Intell. Control*, pp. 613–619, IEEE, 2011.

[37] Y. Yang, D.-W. Ding, H. Xiong, Y. Yin, and D. C. Wunsch, "Online barrier-actor-critic learning for H_∞ control with full-state constraints and input saturation," *J. Franklin Inst.*, 2019.

[38] Y. Yang, Y. Yin, W. He, K. G. Vamvoudakis, H. Modares, and D. C. Wunsch, "Safety-aware reinforcement learning framework with an actor-critic-barrier structure," in *Proc. Am. Control Conf.*, pp. 2352–2358, IEEE, 2019.

[39] J. Shamma and M. Athans, "Gain scheduling: Potential hazards and possible remedies," *IEEE Control System Magazine*, vol. vol. 12, no. no. 3, pp. 101–107, 1992.

[40] L. Eugene, W. Kevin, and D. Howe, "Robust and adaptive control with aerospace applications," 2013.

[41] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, 2017.

[42] 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, 2004.

[43] W. Zhang, J. Hu, and A. Abate, "On the value functions of the discrete-time switched lqr problem," *IEEE Trans. Autom. Control*, vol. 54, no. 11, pp. 2669–2674, 2009.

[44] A. Heydari and S. N. Balakrishnan, "Optimal switching and control of nonlinear switching systems using approximate dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 6, pp. 1106–1117, 2013.

[45] Y. Wardi, M. Egerstedt, and M. Hale, "Switched-mode systems: gradient-descent algorithms with Armijo step sizes," *Discrete Event Dyn. Syst.*, vol. 25, no. 4, pp. 571–599, 2015.

[46] A. Heydari, "Optimal switching with minimum dwell time constraint," *Journal of the Franklin Institute*, vol. 354, no. 11, pp. 4498–4518, 2017.

[47] 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.

[48] 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, 2017.

[49] G. Joshi and G. Chowdhary, "Deep model reference adaptive control," in *IEEE Conf. Decis. Control*, pp. 4601–4608, IEEE, 2019.

[50] G. Joshi, J. Virdi, and G. Chowdhary, "Design and flight evaluation of deep model reference adaptive controller," in *AIAA Scitech 2020 Forum*, p. 1336, 2020.

[51] G. Joshi, J. Virdi, and G. Chowdhary, "Asynchronous deep model reference adaptive control," in *Conf. Robot Learn.*, 2020.

[52] R. Sun, M. Greene, D. Le, Z. Bell, G. Chowdhary, and W. E. Dixon, "Lyapunov-based real-time and iterative adjustment of deep neural networks," *IEEE Control Syst. Lett.*, vol. 6, pp. 193–198, 2021.

[53] P. Deptula, J. 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, pp. 2154–2166, June 2018.

[54] P. Walters, R. Kamalapurkar, F. Voight, E. Schwartz, and W. E. Dixon, "Online approximate optimal station keeping of a marine craft in the presence of an irrotational current," *IEEE Trans. Robot.*, vol. 34, pp. 486–496, April 2018.

[55] 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, 2014.

[56] C. Qin, H. Zhang, and Y. Luo, "Online optimal tracking control of continuous-time linear systems with unknown dynamics by using adaptive dynamic programming," *Int. J. Control*, vol. 87, no. 5, pp. 1000–1009, 2014.

[57] G. Chowdhary, T. Yucelen, M. Mühlegg, and E. N. Johnson, "Concurrent learning adaptive control of linear systems with exponentially convergent bounds," *Int. J. Adapt. Control Signal Process.*, vol. 27, no. 4, pp. 280–301, 2013.

[58] R. Kamalapurkar, J. A. Rosenfeld, A. Parikh, A. R. Teel, and W. E. Dixon, "Invariance-like results for nonautonomous switched systems," *IEEE Trans. Autom. Control*, vol. 64, pp. 614–627, Feb. 2019.

[59] 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, June 2020.

[60] F. L. Lewis, S. Jagannathan, and A. Yesildirak, *Neural network control of robot manipulators and nonlinear systems*. Philadelphia, PA: CRC Press, 1998.

[61] 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.

[62] F. L. Lewis and D. Liu, *Reinforcement learning and approximate dynamic programming for feedback control*, vol. 17. John Wiley & Sons, 2013.

[63] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," in *Handbook of intelligent control: Neural, fuzzy, and adaptive approaches* (D. A. White and D. A. Sorge, eds.), vol. 15, pp. 493–525, Nostrand, New York, 1992.

[64] 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, pp. 743–748, July 2020.

[65] M. L. Greene, P. Deptula, R. Kamalapurkar, and W. E. Dixon, *Handbook of Reinforcement Learning and Control*, ch. Mixed Density Methods for Approximate Dynamic Programming, pp. 139–172. Cham: Springer International Publishing, 2021.

[66] G. V. Chowdhary and E. N. Johnson, "Theory and flight-test validation of a concurrent-learning adaptive controller," *J. Guid. Control Dynam.*, vol. 34, pp. 592–607, Mar. 2011.

[67] H. K. Khalil, *Nonlinear Systems*. Upper Saddle River, NJ: Prentice Hall, 3 ed., 2002.

[68] A. F. Filippov, "Differential equations with discontinuous right-hand side," in *Fifteen papers on differential equations*, vol. 42 of *American Mathematical Society Translations - Series 2*, pp. 199–231, American Mathematical Society, 1964.

[69] 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. 34, pp. 73–82, Jan. 1987.

[70] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis, "Adaptive optimal controllers based on generalized policy iteration in a continuous-time framework," in *Proc. Mediterr Conf. Control Autom.*, pp. 1402–1409, IEEE, 2009.

[71] M. Greene, M. Abudia, R. Kamalapurkar, and W. E. Dixon, "Model-based reinforcement learning for optimal feedback control of switched systems," in *Proc. IEEE Conf. Decis. Control*, pp. 162–167, 2020.

[72] M. L. Greene, P. Deptula, B. Bialy, and W. E. Dixon, "Model-based approximate optimal feedback control of a hypersonic vehicle," in *AIAA SCITECH*, Jan. 2022. AIAA 2022-0613.

[73] W. A. Makumi, M. L. Greene, K. J. Stubbs, and W. E. Dixon, "Model-based switched approximate dynamic programming for functional electrical stimulation cycling," in *Proc. Am. Control Conf.*, June 2022.

[74] A. Parikh, R. Kamalapurkar, and W. E. Dixon, "Integral concurrent learning: Adaptive control with parameter convergence using finite excitation," *Int J Adapt Control Signal Process*, vol. 33, pp. 1775–1787, Dec. 2019.

[75] P. Kidger and T. Lyons, "Universal approximation with deep narrow networks," in *Conf. Learn. Theory*, pp. 2306–2327, PMLR, 2020.

[76] W. E. Dixon, A. Behal, D. M. Dawson, and S. Nagarkatti, *Nonlinear Control of Engineering Systems: A Lyapunov-Based Approach*. Birkhauser: Boston, 2003.

[77] R. Goebel, C. Prieur, and A. R. Teel, “Smooth patchy control lyapunov functions,” *Automatica*, vol. 45, no. 3, pp. 675–683, 2009.

BIOGRAPHICAL SKETCH

Max Lewis Greene was born in Middletown, Connecticut in 1996. He received his B.S. and M.S. in mechanical engineering from the University of Florida in May 2018 and May 2020, respectively. In August 2018, Max joined the Nonlinear Controls and Robotics Laboratory, under the supervision of Dr. Warren E. Dixon, at the University of Florida to pursue a Ph.D. Subsequently, Max received his Ph.D. in mechanical engineering from the University of Florida in May 2022. Max's research interests include Lyapunov-based, adaptive, and reinforcement learning-based control of dynamical systems.