REAL-TIME LEARNING FOR SUBOPTIMAL CONTROL OF UNKNOWN SYSTEMS

By

WANJIKU APRILE MAKUMI

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

2024

To my mother, Clare, and my father, James, and my brother, Munga, for their endless love and support

I thank my advisor, Dr. Warren Dixon, for his mentorship and inspiration during my PhD career. Dr. Dixon has continuously showed unwavered support and encouragement for everything I've done and has taught me many lessons on perseverance and determination as well as challenged me to not only meet but exceed my goals. I extend my gratitude to my committee members Dr. Jane Shin, Dr. Yu Wang, and Dr. Chris Hass for their recommendations and insights. I thank Dr. Zachary Bell for advising me as an intern for two years in the Air Force Research Lab Scholars program. I am grateful to my colleagues in the Nonlinear Control and Robotics Laboratory, at the University of Florida, and at the Air Force Research Laboratory, who have challenged and encouraged me throughout my career. I appreciate my friends and family for their relentless support. Finally, I thank Aidan Leaird for lifting me up each and every day.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADP	Approximate Dynamic Programming
AQL	Approximate Q-Learning
BE	Bellman Error
CL	Concurrent Learning
DNN	Deep Neural Network
HDP	Heuristic Dynamic Programming
HJB	Hamilton-Jacobi-Bellman
HRL	Hierarchical Reinforcement Learning
ICL	Integral Concurrent Learning
Lb-DNN	Lyapunov-based Deep Neural Network
LQR	Linear Quadratic Regulator
LQT	Linear Quadratic Tracking
NN	Neural Network
PD	Positive Definite
PE	Persistence of Excitation
PSD	Positive Semi-Definite
RISE	Robust Integral of the Sign of the Error
RL	Reinforcement Learning
RMS	Root Mean Square
SNN	Shallow Neural Network
STaF	State Following
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

REAL-TIME LEARNING FOR SUBOPTIMAL CONTROL OF UNKNOWN SYSTEMS

By

Wanjiku Aprile Makumi

May 2024

Chair: Warren E. Dixon
Major: Aerospace Engineering

Approximate dynamic programming (ADP) has emerged as a leading method for solving optimal control problems using reinforcement learning (RL) with many benefits and also many open research problems. Model-based methods allow for off-trajectory learning, but they require exact model knowledge. When exact model knowledge is not readily available *a priori*, approximate models can be used to obtain approximations of the optimal value function and the optimal control policy. This dissertation focuses on the intersection of optimality and uncertainty by filling the gaps in the literature and advancing real-time learning in ADP. Specifically the methods developed in this dissertation highlight the advancements of approximate optimal control in the presence of unknown model dynamics with stability guarantees.

Chapter 1 provides an literature overview containing background on RL and RL in control, actor-critic methods, and ADP. An outline of the dissertation is also provided in this chapter. The subsequent chapters of this dissertation elaborate on the evolution of system identification techniques for ADP in the presence of unknown systems for different problems.

Chapter 2 introduces a hierarchical agent to facilitate switched ADP. The standard switched ADP result lacks guidance on how/when to switch. This chapter introduces a framework that uses hierarchical reinforcement learning (HRL) to create a switching pattern. Previous results contained unsupervised switching, and this chapter provides

a method for supervised switching to be used to achieve optimality by using a hierarchy to optimize a selected performance method. The hierarchical agent selects which subsystem to switch to based on which subsystem yields the lowest value function approximation at that time. The control objectives are to minimize the infinite-horizon cost function of each subsystem and to design a switching rule that yields a lower cost for switching between subsystems. Uniformly ultimately bounded (UUB) regulation of the system states to a neighborhood of the origin, and convergence of the approximate control policy to a neighborhood of the optimal control policy, are proven using a Lyapunov-based stability and dwell-time analysis. Simulation results are presented to show that implementing the developed HRL controller yields a total cost of 37% less than the total cost of implementing one ADP sub-controller and an improved rise time.

Chapter 3 introduces a deep system identification approach to the classical pursuit-evasion herding game. This chapter develops an approximate optimal policy for a pursuing agent to indirectly regulate an evading agent coupled by an unknown interaction dynamic. ADP is used to design a controller for the pursuing agent to optimally influence the evading agent to a goal location. Since the interaction dynamic between the agents is unknown, integral concurrent learning (ICL) is used to update a Lyapunov-based deep neural network (Lb-DNN) to facilitate sustained learning and system identification. The integral data collected online is used to update the output-layer weights and optimize the inner-layer features of the Lb-DNN in a new ICL-based deep learning technique. The contribution of this chapter lies in the fact that this is the first time ICL has been used in a DNN system identifier framework. The deep learning and approximate optimal architecture is informed by a Lyapunov-based analysis that ensures UUB convergence of the states as well as estimation of the control policy to within a neighborhood of the optimal control policy. The simulation shows that the pursuer is able to intercept and regulate the evader towards the desired goal location and that the Lb-DNN system identifier outperforms the shallow NN system identifier.

Chapter 4 uses DNNs to facilitate function approximation through multiple parts of the ADP process. Within the context of ADP, a DNN has been used for online system identification; however, a DNN has not been used for online value function approximation. The standard actor-critic framework in ADP uses a single neural network (NN) to approximate the optimal value function. This chapter presents the first method of using a DNN to improve the value function approximation, and thus improving the optimal control policy approximation, with real-time adaptation. Two separate DNNs are used for simultaneous online value function approximation and system identification. A Lyapunov-based stability analysis that accounts for piecewise-in-time continuous dynamics is used to show UUB states and derive the analytical adaptation law for the outer layer weights of the Lb-DNN. The presented simulation results show that the deep value function approximation method results in 95.04% improvement in BE minimization and 5.06% faster convergence.

In Chapter 5, an adaptive DNN-based system identifier is developed to update the dynamics fully online. This dissertation presents the first ADP result with Lyapunov-based real-time weight adaptation laws for each layer of the DNN, with stability guarantees. A robust integral of the sign of the error (RISE)-based dynamics observer is developed to provide a secondary estimate of the dynamics. The difference between the two estimates is calculated as an identification error which is used to develop a least squares adaptive update law. The least squares adaptation law is used with a bounded gain forgetting factor to update the weights of all layers of the DNN. UUB convergence of the DNN weight estimation error, provided the Jacobian of the DNN satisfies the persistence of excitation (PE) condition, is shown via a Lyapunov-based stability analysis. Simulation results show that the Lb-DNN yields 65.73% improved root mean square RMS regulation error, 31.82% improved RMS controller error, and 78.97% improved RMS function approximation error in comparison to the previously developed multi-timescale DNN.

Chapter 6 concludes the dissertation by highlighting the contributions of each chapter. Additionally, future extensions of the work in this dissertation are proposed in this chapter.

CHAPTER 1
INTRODUCTION

1.1 Literature Review

1.1.1 Reinforcement Learning

RL is a framework in which an agent (or controller) learns behavior via feedback from interacting with its environment. An agent learns behaviors through trial and error interactions with its environment, based on “reinforcement” signals from the environment. Over the past few decades, the potential of RL has motivated researchers in machine learning, intelligent systems, and artificial intelligence communities [1].

In a general RL model, an agent interacts with its environment by taking actions. Each interaction typically proceeds as follows: the agent receives inputs that indicate the state of the environment, and then the agent selects and takes an action. After taking an action in some state, the agent receives a scalar reward from the environment, which gives the agent an indication of the quality of that action. This action changes the state of the environment, transitioning it to a “better” or a “worse” state which is interpreted by the agent by either a **reward** or a **penalty** from the environment. The amount of the reward/penalty has the effect of a reinforcement signal to the agent. The function that indicates the action to take in a certain state is called the **policy**. The agent aims to find a policy that maximizes the total accumulated reward, also called the **return**, through a trial and error learning process. The function representing this estimated return is known as the **value function**. The value function allows the agent to make indirect use of past experiences to decide on future actions to take.

RL can be formulated as the process of bringing the current policy estimate and the corresponding value function estimate closer to the optimal policy and the optimal value function. The two components that ususally make up RL consist of estimating the value function based on the current policy and improving the policy by making it greedy with respect to the current estimate of the value function. Improving the policy greedily

means selecting the actions where the value function shows the highest expected return. This general process facilitates learning via reinforcement; see [2–15].

RL is thought of as a third machine learning paradigm, alongside supervised learning and unsupervised learning [16]. RL differs from supervised and unsupervised learning mainly in the kind of feedback received from the environment. In supervised learning, labeled input and output datasets are used to train/supervise learning where the model can measure its accuracy and learn overtime. In unsupervised learning, underlying hidden structures and patterns are sought out in collections of unlabeled data without any assistance or supervision. In RL, on the other hand, the agent only receives a more general, composite reward/punishment signal, and learns from this using an operating principle of increasing the amount of reward it receives over time [1].

1.1.2 Reinforcement Learning for Control

RL enables a cognitive agent to learn desirable behavior from interactions with its environment. In control theory, the desirable behavior is typically quantified using a cost function, and the control problem is formulated as the desire to find the optimal policy that minimizes a cumulative cost. RL techniques were first used for control systems in [17–19].

In [20], Richard Bellman first introduced dynamic programming to study multistage decision processess. He defined the concept of a policy as a rule that determines the optimal decision/control. Dynamic programming-based methods use a state value function or an action value function. The state value function is the expected reward from starting from a certain state under a certain policy. The action value function, or Q-function, is the expected reward from starting from a state and performing certain actions. The Q-function assigns each state-action pair a value, which is the total optimal cost when the action is performed in the state and the optimal policy is followed.

Dynamic programming is made up of two methods, policy evaluation and policy improvement, and was first introduced in the following probabalistic formulation. Policy

evaluation, which can also be thought of as policy prediction, finds the value function for a given arbitrary policy. An initial condition is arbitrarily selected, and then the Bellman equation is used as an update rule for the following approximations. Policy improvement constructs a new policy that greedily improves the policy. The combination of policy evaluation and policy improvement is called generalized policy iteration. In generalized policy iteration, starting with a random policy, the value function is evaluated, and then the policy is greedily improved, and then a new value function is evaluated, and then the policy is greedily improved again alternating until the unique optimal policy is found. In generalized policy iteration the cost of the control is not completely evaluated at each step, instead only the current cost estimate towards that value is updated. The control policy is not fully updated to the greedy policy for the new cost estimate, but the policy is only updated towards the greedy policy. Most dynamic programming-based approximate optimal control methods consist of generalized policy iteration [16, 21–24]. The policy iteration algorithm was first developed by Bellman in [20], and then a policy improvement theorem was provided in [25] for Markov processes.

In continuous time, the policy evaluation step aims to find a solution to the generalized Hamilton-Jacobi-Bellman (HJB) equation. The HJB equation is the continous-time counterpart of the Bellman equation for discrete-time systems. The HJB equation is first introduced in [26] for a fixed policy. The policy improvement step aims to improve the policy by solving the minimization problem for a fixed value function.

Some disadvantages of policy iteration are that each of its iterations involves policy evaluation which requires multiple sweeps through the state set while only reaching convergence at the limit and that an initial admissable policy is required. Through a method called value iteration, policy evaluation is shortened as it is stopped after just one sweep (one update of each state) [16]. In other words, there is no need to wait for the policy to converge. Value iteration is a generalized policy iteration algorithm for discrete-time systems. Value iteration uses Bellman's recurrence relation, which is

the discrete time counterpart of the HJB equation, as an update rule, similar to policy iteration. Also, the initialization is not required to be a Lyapunov function or a value function corresponding to an admissible policy. Therefore, value iteration does not require an initial admissible policy like policy iteration.

The applicability of classical dynamic programming techniques like policy iteration and value iteration is limited by the curse of dimensionality and the need for model knowledge. In results like [27–31], the need for model knowledge is circumvented by developing a policy iteration algorithm that converges to the optimal solution of the discrete-time linear quadratic regulator (LQR) problem using Q functions, which is known as Q-learning. Q-learning is a technique to compute the optimal policy and the associated value function based on state and input observations without model knowledge. Watkins introduced this technique in his thesis [32] and provided a complete convergence proof in [33] which was the first convergence result for dynamic programming-based RL for a continuous system.

Q-learning was first only applied to the LQR problem because of additional complexities associated with computing the feedforward term in the linear quadratic tracking (LQT) problem. An online model-free solution is developed in [28] using a reinforcement Q-learning algorithm to the infinite-horizon LQT for discrete-time systems. Results such as [29] and [30] use Q-learning for continuous linear systems. One of the disadvantages of Q-learning is that the learning process is expensive for the agent since every state-action pair needs to be visited frequently to get convergence. Q-learning is classified as an online value iteration method. Advantage updating was proposed as an extension of the Q-learning algorithm [31]. Advantage updating does not require a model to be given or learned. It can be implemented in continuous-time and provides faster convergence, but there are no stability results.

1.1.3 Approximate Optimal Control

In classic optimal control, if full state knowledge is assumed and if the system dynamics are modeled by linear dynamics and the cost functional is quadratic in the state and control, then the optimal control problem can be solved through Riccati equations. If the system is modeled with nonlinear dynamics or the cost functional is not quadratic, then the optimal control problem depends on solving the HJB equation. Since it is difficult to find an analytical solution to the HJB equation, classical optimal control techniques are of limited use for nonlinear systems, and there have been many research aims dedicated to approximating the HJB equation.

Open-loop solutions for approximating the HJB equation are presented in [34–38]. In [34], the problem is reduced to a two-point boundary-value problem which can be solved by various methods. In [35], the authors reduce the optimal bilinear control problem to successive iterations of a sequence of Riccati equations, and in [36] the same problem is further reduced to successive approximations of a sequence of Lyapunov equations. In [37], the authors reduce the bilinear control problem to a sequence of linear control problems that converge uniformly to the optimal bilinear control. In [38], the nonlinear optimal control problem is cast in the form of a nonlinear programming problem. In this method, the point that solves the nonlinear programming problem is the optimal path of the system. Unfortunately, open-loop control is undesirable for practical systems because it lacks feedback of the system. Closed-loop approximations of the HJB equation have resulted in several promising approaches.

The authors in [39–42] present a perturbation method. Perturbation methods assume that the nonlinear system is a perturbation of a linear system. The optimal cost and control are assumed to be analytic and are expanded in a Taylor series. These papers employ various techniques to find the first few terms in the series. The first term is the solution of the matrix Riccati equation resulting from linearizing the system about the origin. The next term is a third-order approximation to the control and can also be

expressed as matrix equations. The process usually terminates after the first two terms since the higher-order terms need the solution of a linear partial differential equation.

The theory of viscous solutions is another method for approximate optimal control. Normally, the solution to the HJB needs to be differentiable over the domain of interest for it to be defined in a classical sense. For the theory of viscosity solutions, a continuous function can be defined as the unique solution to the HJB equations that do not admit continuous solutions. The viscosity solution to the HJB equation equals the value function of the associated optimal control problem. Therefore, the viscosity solution and the classical solution are identical if the solution of the HJB equation is differentiable.

Other numerically approximated viscous solutions including finite difference and finite element approaches have since been investigated. In [43], a discrete approximation to the continuous-time, deterministic optimal control problem is analyzed. The approximate solutions are interpreted as value functions of discrete time control so that a minimizing sequence of constant contols can be constructed. The basic idea is to approximate the system equations by an Euler formula. Then the discrete dynamic programming problem is solved. The result in [44] shows improved convergence using a Runge-Kutta formula instead of Euler's method to approximate the system dynamic.

Using a finite-element method, [45] looks at stationary systems and uses an algorithm for finite-time problems with continuous and impulse control. Following the previous work in [46], the non-stationary case is considered. A similar approach is considered for infinite-time horizon problems in [47] where the HJB is solved via the viscosity solution related to the infinite horizon deterministic control problem. The method converges to the viscosity solution and presents the error estimates for the convergence of the algorithm.

The main disadvantage of both finite-element and finite-difference methods is that they require a discretization of the state space. Discretization is problematic because

of the computational burden and a large amount of data is required to be stored and recalled in real-time for feedback control, i.e. Bellman's curse of dimensionality.

To avoid discretizing the state space, Galerkin's spectral method (projection) was developed. Previously, successful application of the policy iteration method was limited for nonlinear systems. Galerkin's spectral method uses Galerkin's spectral approximation methods to solve the nonlinear Lyapunov equations for the policy evaluation step in the policy iteration algorithm. The curse of dimensionality does not disappear completely, as it shows up as weighted averages of the dynamics over a compact set, but this creates more possible solutions for dealing with the dimensionality issues. Galerkin's spectral method is used to approximate the generalized HJB equation, and the generalized HJB equation is used to approximate the HJB equation. The resulting control is in feedback form and stabilizes the closed-loop system. The authors in [48] build on the previous result by combining successive approximation and Galerkin approximation methods to develop closed-loop control laws with well-defined stability regions. In [49] the authors use successive approximation to reduce the HJB equation to a sequence of linear partial differential equations; then Galerkin's spectral method is used to approximate them.

Approximate Q-learning (AQL) for nonlinear continuous systems is presented in results such as [50–52]. In [50] the steepest descent Q-learning algorithm to obtain the optimal approximation of the Hamiltonian is implemented. A convex optimization problem that characterizes the best approximation of the Q-function within a given class is formulated; however, closed-loop stability of the developed controller is not analyzed. The experimental results in [51] show that the AQL method performs better than the conventional Q-learning method. AQL is used because of its ability to solve the nonlinear optimal control problem when model knowledge of the plant is unavailable.

As a result of function approximation errors, the AQL algorithms can just give a near-optimal solution, and [52] includes a quantitative analysis result of the the error bound between the optimal cost and the actual cost.

Another method, which will be the focus of the rest of this document, is approximate dynamic programming (ADP) which is essentially a juxtaposition of RL and dynamic programming ideas [1]. ADP is an approach to balance computational demands with optimal decision-making. An optimal policy is obtained by solving the HJB equation, but since the solution to the HJB is intractable in general, actor-critic-based RL methods can be used to approximate the value function and the optimal controller.

1.1.4 Actor-Critic Methodology

Before discussing ADP, the necessary actor-critic-based RL methods must be defined. Several classes of RL algorithms include actor-only, critic-only, and actor-critic methods. The methods that learn approximations of both the policy and the value function are referred to as actor-critic methods. In these methods, the actor references the learned policy and the critic references the learned value function.

The actor-only methods use optimization procedures on a parameterized family of policies. The methods are advantageous because a spectrum of the continous actions can be generated. However, usually policy gradient optimization methods are used and these suffer from high variance in the estimates of the gradient which results in slow learning [53]. In policy gradient (actor-only) methods, instead of approximating the value function, the policy is directly approximated by computing the gradient of the cost functional with respect to the unknown parameters in the approximation of the policy [54–56]. Both policy iteration and value iteration are typically critic-only methods and can be considered as special cases of generalized policy iteration [16]. The critic-only methods use temporal difference learning and have a lower variance in the estimates of expected returns [57]. Critic methods learn the policy by selecting greedy actions. Therefore, actions are selected where the value function shows the highest

expected return. The actor-critic methods, which are modern policy gradient methods, use an approximation of the value function to estimate the gradients. These methods are helpful for implementing generalized policy iteration algorithms because they combine the benefits of the actor-only and critic-only. The actor supplies the advantage of continuous actions without needing to implement optimization procedures on a value function, and the critic allows for the actor to update with gradients that have lower variance via its estimate of the expected return leading to faster learning. Additionally, the combined actor-critic methods usually have better convergence properties than critic-only methods [53].

The idea of learning with a critic first appeared in [58, 59] where the state-space was partitioned to make the computations tractable. Further development of critic based methods to learn optimal actions in sequential decision problems appears in [60–62]. Actor-critic methods initiated in [63] for systems with finite state and action-spaces, and in [64] for systems with continuous state and action-spaces using neural networks (NNs) to implement the actor and the critic. For deterministic systems, the analysis of the actor-critic convergence properties can be found in [65] and [66]. For stochastic systems, they can be found in [53].

The iterative nature of actor-critic methods makes them particularly suitable for offline computation and for discrete-time systems, which is why RL has been mostly constrained to problems where discrete actions are taken in discrete time steps based on the observation of the discrete state of the system. The drawbacks of discretization are that the control output is not smooth which can result in poor performance, the number of states and the number of iteration steps can become very large which requires a large memory storage and many learning trials, and in order to keep the number of states manageable, an elaborate partitioning of the variables has to be found using prior knowledge. Research motivated by these shortcomings eventually extended the concept of ADP to apply actor-critic methods online to continuous-time systems.

Some benefits of a continuous framework are that a smooth control performance can be achieved, an efficient control policy can be derived using the gradient of the value function, and the function approximation and numerical integration algorithms determine how to partition the state, action and time. For nonlinear continuous-time systems, the HJB equation is used since it is the continous-time counterpart of the Bellman equation.

A continuous-time formulation of actor-critic methods was first developed by Doya in [67] which is one of the first papers to use RL for continuous-time dynamical systems without a priori discretization of time, state, and action. A set of RL algorithms is proposed for nonlinear dynamical systems based on the HJB equation for infinite-horizon discounted reward problems. Algorithms are derived for estimating value functions and for improving polices with the use of function approximators.

First, the authors determine methods for learning the value function by minimizing a continuous-time form of the temporal difference error or the Bellman error (BE). Then the continuous actor-critic method and a value gradient based policy are formulated to improve the policy using the value function. In [67], the value function and the optimal policy do not depend explicitly on time, which is convenient for using function approximators to estimate them. The actor and the critic weights are tuned continuously using an adaptive update law. The developed algorithms can be used to simultaneously learn and utilize an approximate optimal feedback controller in real-time for nonlinear systems; however there are no stability or convergence results stated.

Convergence properties of actor-critic methods for continuous-time systems where both the networks are tuned simultaneously are examined in [68], and a Lyapunov-based analysis that examines both convergence and stability properties of an online implementation of the actor-critic method is developed in [69]. The standard actor-critic fomulation discussed next is based on the algorithms in [67] and the analysis techniques in [69].

ADP uses NNs to approximately solve dynamic programming forward-in-time, therefore avoiding the curse of dimensionality. The value function can be approximated using the Stone-Weierstrass Theorem to approximate a function with a single layer NN. The value function can be represented as an NN with a user-defined vector of basis functions, an ideal weight vector, and a function approximation error. The function approximation error can be made arbitrarily small by increasing the number of basis functions. Approximate policy iteration is model-free and estimates the weights with the critic weight. Approximate generalized policy iteration is model-based and the minimization occurs over a specific trajectory as opposed to the whole state space in model-free [70, 71]. Unfortunately, these algorithms cannot generally be shown to be stabilizing, and therefore are not well-suited for online optimal feedback control and learning. The two-network actor-critic framework is utilized to overcome this by ensuring stability of the system during learning.

The optimal value function is replaced with a parametric estimate resulting in the NN representation, and the optimal control policy is replaced with a parametric estimate resulting in the NN representation. Both sets of weights are used to estimate the same set of ideal weights. The accuracy of the actor and critic estimates to their ideal weight values is measured by the BE. The BE is zero without approximation; hence, it is a measure of the suboptimality. Therefore, the goal is to minimize the BE.

1.1.5 Approximate Dynamic Programming

Werbos proposed ADP as an effective adaptive learning control approach to solve optimal control problems forward-in-time [19]. Werbos defined actor-critic online learning algorithms to solve the optimal control problem based value iteration, which does not require an initial stabilizing control policy [17, 72]. He defined a family of value iteration algorithms, which he termed ADP algorithms, where he used a critic NN for value function approximation and an actor NN for approximation of the control policy. ADP

combines the advantages of the learning algorithms previously discussed in this review and can be implemented in discrete-time or continuous-time.

Solving the discrete-time HJB equation is more difficult than solving the discrete-time Riccati equation that is used for linear systems because it involves solving nonlinear partial differential equations. Several results have used dynamic programming-based techniques and NNs to investigate solving the discrete-time nonlinear optimal control problem such as [73, 74]. Heuristic dynamic programming (HDP) techniques were utilized by [73] to develop an iteration-based solution to the HJB, and it was shown that the algorithm converges to the optimal control policy and the optimal value function that solves the HJB equation. Two NNs are utilized in the implementation of this algorithm to learn the optimal control policy and to learn the optimal cost function. The reconstruction errors of the NNs are ignored, and the internal dynamics are considered unknown whereas the control coefficient matrix has to be known. An iterative solution to the generalized HJB equation is proposed in [74] and a nearly optimal state feedback control law for affine nonlinear discrete-time systems is derived. The complete dynamics of the affine nonlinear system are assumed to be known. A single NN is utilized to learn the cost function while, again, the NN reconstruction errors are considered negligible. Due to the assumption that there are no NN reconstruction errors, it is unlikely to continue to grow as a research thread where other threads are more applicable.

To implement ADP on discrete-time systems, partial knowledge of the system dynamics must be known, in addition to the value of the controlled plant one step ahead. Since it can be challenging to obtain partial knowledge of the system dynamics, [75] developed an approach where the need for partial knowledge of the system is relaxed. The authors develop a NN system identification scheme to learn the system dynamics online. The NN system identification is achieved by asserting that the reconstruction errors lie within a small-gain type norm bounded conic sector. The identification errors are proven to converge to zero asymptotically even with the NN reconstruction error.

Furthermore, ADP training takes place offline using the learned NN model to yield the optimal control law. The iterative NN system identification scheme is shown to converge to the optimal solution. Therefore, there is no requirement of explicit knowledge of the system dynamics because an online learned NN model is utilized for the offline ADP training. While intriguing, these results lack a stability analysis for guaranteed performance and cannot be applied to a patient population.

More discrete-time results include [76–79]. In [76], multilayer NNs are used to design an optimal tracking neuro-controller for discrete-time nonlinear dynamic systems, and generalized back-propogation through time is used to solve a finite horizon tracking problem with offline training of NNs. Multiple results [77–79] investigate greedy HDP based algorithms to transform nonautonomous systems into autonomous systems to achieve approximate convergence of the value function to the optimal value function. In [77], the optimal tracking controller is composed of two sub-controllers: the feedforward controller and the feedback controller. The feedforward controller is designed by an implicit function theorem, while the feedback controller uses the greedy globalized HDP iteration algorithm. Three NNs are used to facilitate the implementation; the model network, the critic network, and the actor network. In [78], an iterative ADP algorithm to design a finite-horizon near-optimal tracking controller is developed for a class of discrete-time nonlinear systems. The ADP via HDP technique is introduced to solve for the optimal tracking controller that gets the cost function within an ε error bound of the optimal value. The result in [79] is one of the first results to solve the optimal tracking control problem with ADP, specifically by the HDP iteration algorithm. The optimal tracking problem is transformed into an optimal regulation problem which then uses the greedy HDP iteration algorithm to assess the convergence of the regulation problem. The common problem with these discrete-time system approaches is the lack of an accompanying stability analysis.

In contrast to the previous results, [80] develops a time-based optimal controller design for affine nonlinear discrete-time systems without using value and policy iteration. Therefore, a closed-loop stability analysis is introduced which is not possible with value and policy iteration-based ADP methods. The framework learns the HJB equation and the optimal control policy online and forward-in-time using NNs. Real-time control is introduced where the cost function and the control policy are updated with respect to time at each sampling instance.

ADP approximation techniques are used on continuous-time systems in [81] and [82] to address the tracking problem. In these results, the value function and controller are time-varying functions of the tracking error. The authors in [82] develop a single online approximator (SOLA)-based framework to learn the HJB equation and optimal control input online and forward-in-time in comparison to [81] where two online approximators are used. The boundedness of the system states and the success of the optimal cost\HJB function and control input are ensured by an online parameter tuning law thoughout the learning process. An initial stabilizing control is not required for stability such as in the previous result in [81]. In [82], the optimal regulation and tracking control of affine nonlinear continuous-time systems with known dynamics is achieved using a SOLA-based scheme. The SOLA-based adaptive approach is designed to learn the infinite horizon continuous time HJB equation and its corresponding optimal control input.

The authors in [83] proposed a new formulation of the proportional algorithm which converges to the optimal control solution without using internal dynamics of the system. Furthermore, [70] extends the result in [83] to nonlinear continuous-time systems where the knowledge of the input-to-state dynamics is still required. The algorithms used in these results were based on sequential updates of the critic and actor NNs. For the implementation of these NNs, while one is tuned, the other remains constant. These ADP methods still cannot be applied directly to unknown general nonlinear systems.

Motivated by the previous results, [84] is the one of the first papers to apply the ADP method to the continuous-time optimal control tracking problem with unknown general nonlinear dynamics. In [84], a data-driven model based on a recurrent NN is proposed to reconstruct the unknown system dynamics. Input-output data is used so that knowledge of known nonlinear dynamics is not required. The authors add an adjustable term related to the modeling error to the data-driven model. The adjustable term guarantees that the modeling error will converge to zero. The input-output data-driven model is used to design the controller. The controller in [84] consists of the steady-state controller, the optimal feedback controller, and a robustifying term. The goal of the steady-state controller is to achieve the desired tracking performance at the steady-state stage. The goal of the optimal feedback controller is to stabilize the state tracking error dynamics at the transient stage in an optimal way. The critic and actor NNs are updated simultaneously as opposed to sequentially in most existing literature at the time. While both of these results have contributions, time does not live on a compact set for the infinite horizon optimal control problem. Therefore, it is not explained how the NNs can approximate the time-varying value function and controller since the tracking error is an input.

Also, inspired by the previous results in [83] and [70] synchronous update laws for actor and critic were first introduced in [69] as synchronous policy iteration. Synchronous policy iteration is an online adaptive algorithm which involves simultaneous tuning of both actor and critic NNs and an extremal version of the generalized policy iteration introduced in [16] where the algorithm learns the optimal solution to the HJB equation online in real-time. The method in [69] requires complete system model knowledge. Also, [69] introduces a nonstandard 'normalized' critic NN tuning algorithm, along with guarantees for its convergence based on a persistence of excitation (PE) condition, and adds nonstandard extra terms to the actor NN tuning algorithm to guarantee closed-loop stability.

Motivated by the work in [69], [85] proposes an actor–critic–identifier to approximately solve the continuous-time infinite horizon optimal control problem for uncertain nonlinear systems. A continuous-time control-affine nonlinear dynamical system is considered in the following. The difference from [69] is that knowledge of the system drift dynamics is not required. The actor-critic-identifier framework, seen in Figure 1-1, is composed of the actor and critic NNs to approximate the optimal controller and optimal value function respectively and an identifier dynamic NN to estimate the system dynamics online. Learning of the actor, critic, and identifier is simultaneous and continuous. The result in [85] is the first indirect adaptive control approach to RL where an identification-based online learning scheme is applied. The actor NN uses a gradient-based update law, the critic NN uses a least-squares-based update law, and the identifier dynamic NN uses a combination of a Hopfield-type component [86] with a robust integral of sign of the error (RISE) component [87]. Uniformly Ultimately Bounded (UUB) stability of the closed-loop system is guaranteed with the use of a PE condition as well as exponential convergence to the neighborhood of the optimal control. The diagram in Figure 1-1 contains the framework used in [85].

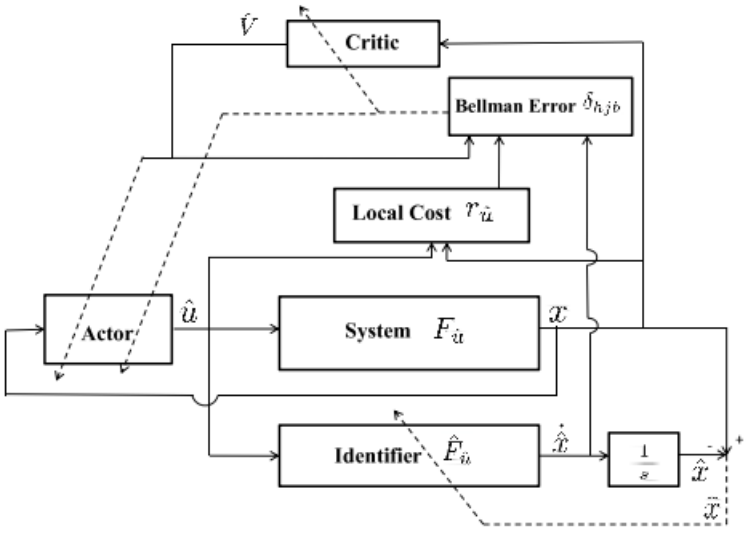

Figure 1-1. Actor-critic-identifier framework

In [88], a new controller is developed for ADP tracking problems for continuous-time nonlinear systems. In trajectory tracking problems, the value function explicitly depends on time. The tracking error and the desired trajectory both serve as inputs to the NN. A different HJB equation is required to be solved where its solution is a time-varying function of the tracking error. The partial derivative of the value function with respect to the desired trajectory is included in the HJB equation. The technical challenges that come with a time-varying optimal control problem are highlighted in [88]. The problem must be converted so that value function is a time-invariant function of the transformed states allowing the value function to be approximated using a NN. The problem must be reformulated as a stationary optimal control problem so that the time invariant value function can be approximated using the NN function approximation techniques. The time-invariant optimal value function can be interpreted as a time-varying map [88, Lemma 1.] is used to prove that it is positive definite and decrescent and therefore can be used as a candidate Lyapunov function. To transform the time-varying optimal control problem into a time-invariant optimal control problem, a new concatenated state is defined consisting of the tracking error and the desired trajectory. The value function (critic) weights are updated to minimize the BE using a normalized least squared update law, and the policy (actor) weights are updated to track the critic weights. The drawbacks of this result, are that PE is required and the dynamics must be known.

Convergence of the parameter estimates to a neighborhood of their ideal values is required to achieve convergence of the RL-based controller to a neighborhood of the optimal controller. Gradient descent-based and least squares update laws generally require PE in the system to attain parameter convergence, yet it is impossible to guarantee PE a priori. The unknown value function parameter estimates are updated based on evaluation of the BE along the system trajectory. Subsequent challenges are that the system states need to be persistently excited, and the BE needs to be

evaluated over enough points in the state space to result in a good approximation. Sufficient exploration in a neighborhood around the desired trajectory is necessary so that information around the desired trajectory can be used to learn the value function. For the stochastic systems in [16, 50, 53], a randomized stationary policy is used. For the deterministic systems in [69, 75, 85, 89], a probing noise is added to the derived control law. Other results in [28, 90, 91] add an exploration signal to the control input to ensure sufficient exploration in the desired region of the state space. However, for nonlinear systems there are no analytical methods to compute the appropriate exploration signal, and the exploration signal is not included in the stability analysis.

For the BE to measure the quality of the value function estimate, it needs to be evaluated along the system trajectories. Similarly, state derivative estimators can only estimate the state derivative along the system trajectories. In [92], the aforementioned challenges are addressed for an infinite horizon optimal regulation problem on a nonlinear, control affine plant with linear in the parameters uncertainties in the drift dynamics by observing that if the system dynamics are known, the BE can be computed at any desired point in the state space. Therefore, unknown parameters in the value function can be adjusted using least squares minimization of the BE evaluated at arbitrary points in the state space. The results of [92] indicate that convergence of the unknown parameters in the value function is guaranteed provided the selected points satisfy a rank condition that is weaker than the PE condition. Sufficient exploration can be achieved based on appropriate selection of the points in the state space. If the system dynamics are partially unknown, an estimate of the system dynamics can be used for the BE to be evaluated at any desired point in the state space.

The aforementioned method, called BE extrapolation, is used to facilitate simulation of experience, since a model is used to evaluate the BE at unexplored points in the state space and is gaining experience from exploration. Simulation of experience has been investigated in results such as [4, 15] for stochastic model-based RL, but these problems

are solved offline and system stability during the learning phase is not analyzed. In [92], a concurrent learning (CL)-based parameter estimator is developed to exponentially identify the unknown parameters in the system model, and the parameter estimates are used to compute an approximation to the BE. The unknown parameters in the value function are updated based on the approximate BE. UUB regulation of the system states to a neighborhood of the origin, UUB convergence of the parameter estimates, and UUB convergence of the developed policy to the optimal policy, is established using a Lyapunov-based analysis.

The authors combine the results in [88] and [92] to provide an approximate online adaptive solution to the infinite-horizon optimal tracking problem for control-affine continuous-time nonlinear systems with unknown drift dynamics using model-based RL with a CL-based system identifier to simulate experience in [93]. As mentioned previously, the trajectory tracking result in [88] requires exact model knowledge. Since the optimal tracking problem requires knowledge of the steady-state controller, the extension to uncertain systems presents technical challenges because obtaining a good estimate of the desired steady-state controller and using the estimation error in the stability analysis are quite complex. In [93], the authors use the technique developed in [92] to obtain an estimate of the steady-state controller using CL-based system identifiers. Since the use of an estimate in place of the true steady-state controller results in additional approximation errors that could threaten the stability during the learning phase, [93] analyzes the stability of the closed-loop system. The error between the actual steady-state controller and the estimate is included in the stability analysis by examining the trajectories of the concatenated system under the implemented control signal.

Previous work in [91] shows the PE condition is relaxed to a finite excitation condition using integral RL along with experience replay, where each evaluation of the BE is interpreted as gained experience, and these experiences are stored in a history

stack and are repeatedly used in the learning algorithm to improve data efficiency. In, [93] a different approach is used where a dynamic system identifier is developed to generate a parametric estimate of the drift dynamics. Therefore, experience can be simulated by extrapolating the BE over unexplored off-trajectory points of the state space. Optimal policy learning laws can utilize simulated experience along with experience gained and stored along the state trajectory. The CL-based identifier system from [92] is used to relax the PE condition and simulate experience by evaluating the BE over unexplored areas of the state space [94].

1.2 Outline of Dissertation

Chapter 2 leverages supervisory control methods within the ADP framework. Supervisory control methods provide alternatives to traditional continuously-tuned adaptive control laws and are useful when traditional control methodologies based on a single continuous controller do not provide satisfactory performance [95]. Switching between multiple controllers is orchestrated by a supervisory agent that uses data obtained to dictate the active control policy at each instance of time [95].

Supervised switching can be used in the context of optimality by using a hierarchy to optimize a certain performance index. The infinite-horizon value function is a valuable metric to observe because it provides the cost-to-go of implementing its respective optimal controller [96]. Supervisory control approaches have been used to obtain optimality in [97–100]. Many results in this field do not consider nonlinear systems because it is challenging to solve optimal control problems for nonlinear systems. However, recent advancements in [101–103] have created a framework for approximating optimal control policies online, and these methods can be integrated into a supervisory control problem.

For unknown systems, i.e., systems that contain unknown parametric uncertainties, the optimal value function cannot be determined offline; hence, there is a need to approximate it online. Due to the parametric uncertainties, it is difficult to know which controller yields the lowest cost for the system. The HRL framework exploits the

generalized policy iteration technique introduced in Section 1.1.2 to decide which controller to select. For multiple different cost functions, policy improvement is used to calculate an approximate optimal controller, and policy evaluation is used to calculate a value function approximation. A hierarchical agent is developed that compares the value function approximation of the several approximately optimal lower-level controllers as a metric to select which controller should be active in the feedback loop; i.e, the hierarchical agent selects the controller associated with the least approximated cost-to-go at each instance in time. Since the system model contains parametric uncertainties, then an estimate of the system model must be used for BE extrapolation. An integral concurrent learning (ICL)-based parameter identifier, as in [104], is used in the feedback loop of the HRL structure with the supervisory agent to identify the unknown drift dynamics online.

At each instance in time, the HRL closed-loop system continuously switches between control policies, resulting in a switched system. In general, switched systems are challenging to analyze due to discontinuities and instantaneous growth of the Lyapunov function(s) [105]. Switching between multiple stable subsystems can result in an unstable switched system; hence, a switched systems stability analysis is motivated [106]. Since optimal value functions are generally distinct between subsystems, and are a part of the candidate Lyapunov function for each subsystem, a common Lyapunov function cannot be constructed. Hence, a multiple Lyapunov function-based approach is motivated. One way to ensure stability using multiple Lyapunov functions is to ensure that each subsystem remains active a minimum amount of time (i.e., a minimum dwell-time analysis [105, Ch. 3]) or to establish an upper bound on the number of switches in any given time interval (i.e., an average dwell-time analysis [105, Ch. 3]). While a switched ADP technique that uses a minimum dwell-time analysis is available in [107], the analysis therein assumes that the optimal value function is upper and lower bounded by quadratic functions. In [107, Assumption 6], a very restrictive bound on

the optimal value function in the Lyapunov function is used to facilitate an exponential result; however, for general nonlinear systems, the assumption cannot be verified. The Lyapunov-based switched system stability analysis in Chapter 2 relaxes that previous assumption.

The contribution of Chapter 2 is the development of an HRL-based framework that uses a hierarchical supervisory control strategy to determine the policy corresponding to the lowest cost-to-go between lower-level ADP controllers and optimize the corresponding subsystem. The hierarchical framework identifies which controller should be active at a given time and generates a switching signal indicating the most desirable switching pattern based on comparing multiple value function estimates. A Lyapunov-based dwell-time analysis is used to establish stability while relaxing the constraints and assumptions in [107]. The dwell-time analysis uses a Lyapunov-based stability theorem that is generally applicable to switched systems where all subsystems can be shown to be UUB using multiple Lyapunov-like functions. Simulation results show that implementing the developed HRL controller yields a total cost that is 37% less than the total cost of implementing one ADP sub-controller and an improved rise time.

Chapter 3 and the result in [108] consider an indirect herding problem where a pursuing agent is tasked with intercepting and regulating an evading agent to a desired goal location through an interaction dynamic. Optimal solutions for indirect herding problems are sought in [109–111] using tools such as dynamic programming and calculus of variations. Drawbacks of such methods include computational inefficiency, due to the curse of dimensionality, and the need for known dynamics. ADP can be used as an alternative approach. The previous ADP indirect herding result in [112] used a shallow NN (SNN) to approximate the unknown drift dynamics; however, recent evidence shows that using a deep neural network (DNN) for system identification results in improved tracking performance [113].

Previously, [113] used an Lb-DNN for a control affine system with CL. CL is an adaptive update scheme that uses input/output data to guarantee parameter convergence without requiring persistent excitation. CL requires estimates of the state derivatives if the true values are not known or measurable. In Chapter 3, we are generalizing to a larger class of systems that are not control affine, and not even directly controlled, by using ICL to remove the need to measure the state derivatives. In existing literature, DNNs have never been used within an ICL-based system identification technique. The result in [112] used ICL solely for the output weights in an SNN, but now we develop a framework that uses the integral data to additionally train the Lb-DNN inner features by optimizing an integral form of the loss.

In Chapter 3, a multi-timescale Lb-DNN, similar to the one introduced in [113] and [114], is used for system identification. A multi-timescale framework is used to merge typically offline deep learning techniques with online adaptation to result in real-time deep learning (lifelong learning). In contrast to those previous works, this multi-timescale framework consists of output-layer weights being updated in real-time via an ICL-based adaptive update law and inner-layer features being updated concurrent to real-time via iterative batch updates training on integrated data sets. The challenges associated with applying this framework to the ADP-based indirect herding problem include piecewise-in-time discontinuities in the dynamics estimate, adaptation laws, and closed-loop error system from the iterative updates of the inner-layer features. These challenges restrict the adaptive update law used in [114] from being used in this problem, resulting in a new analysis used in this chapter that considers piecewise-in-time discontinuities. Simulation results demonstrate the performance of the developed method and the improved function approximation compared to an SNN.

Building on the DNN-based system identifier introduced in Chapter 3, Chapter 4 introduces a joint DNN-based system identification and DNN-based value function

approximation technique for ADP. Recent results in ADP have leveraged the multi-timescale Lb-DNN introduced in [114]. Within the context of ADP, the multi-timescale Lb-DNN in [114] has been used for online system identification in [108, 113, 115, 116] and Chapter 3; however, deep learning has not yet been used for simultaneous online value function approximation, which could be especially beneficial given the complexity of approximating the solution to the HJB equation for nonlinear systems. Some RL-based results, such as the deep Q network (DQN) approach in [117], have investigated using DNNs to learn the action-value function. The result in [117] uses the AQL method introduced in Section 1.1.3, and was one of the first to address instability concerns when nonlinear function approximators are used to approximate the action-value function (or Q-function). The popular deep deterministic policy gradient (DDPG) method in [118] combines insights from the DQN in [117] with the actor-critic approach. Specifically, in [118], DNN function approximators are used to approximate the action-value function within an actor-critic, model-free algorithm. While the methods in [117] and [118], AQL and DDPG, that train DNNs offline are useful tools, they lack the ability to adapt in real-time (i.e., lack lifelong learning). In adversarial environments, if an adversary modifies the environment or the features of the offline-trained DNN, this could result in instability of the dynamical system (e.g., a vehicular crash). Hence, to build robust tools for adversarial, changing, or unknown environments, there is motivation to implement deep RL methods that update online to facilitate lifelong learning.

In Chapter 4, leveraging the results in [108, 113, 114] and Chapter 3, two separate Lb-DNNs are used for simultaneous online system identification and value function approximation. The multi-timescale Lb-DNNs used for simultaneous system identification and value function approximation use adaptation policies to update the output-layers of the Lb-DNN weights in real-time. Concurrently to real-time execution, data is collected and DNN training algorithms are performed iteratively for the inner-layer weights using

Adam [119]. Similar to batch updates, the slower timescale update of the inner-layer features results in a time-varying switch, which introduces piecewise-in-time discontinuities into the dynamics estimate. Hence, a Lyapunov-based stability analysis that accounts for piecewise-in-time continuous dynamics is used to show uniform ultimate boundedness of the states and derive the analytical adaptation law for the outer layer weights of the Lb-DNN. The contribution of this chapter lies in the multi-timescale Lb-DNN approach to value function approximation. Value function approximation corresponds to the policy evaluation step in the generalized policy iteration method introduced in Section 1.1.2. In policy evaluation, the BE is used to update the value function approximation. This chapter provides a framework to use deep policy evalutation where the off-policy data points collected through BE extrapolation are used to update the output-layer weights of the Lb-DNN in real-time and are also used in the loss function to update the inner-layer features concurrent to real-time. The comparative simulation results show that the deep policy evaluation method results in 95.04% improvement in BE minimization and 5.06% faster convergence.

Chapter 5 advances the deep learning techniques in Chapters 3 and 4. Previous works have used linear parameterization (cf., [120–122] and Chapter 2), NNs (cf., [93, 104, 112, 123, 124]), and DNNs (cf., [108, 113, 115, 116] and Chapters 3 and 4) to obtain an approximation of the dynamics. Chapter 3 shows that DNN-based approximators yield better approximation of the dynamics when compared to single-layer NNs and [113] shows that DNN-based approximators improve system performance. However, the aforementioned results, known as multi-timescale DNNs, update only the output-layer weights in real-time, whereas the inner-layer weights are updated iteratively by minimizing a loss function based on datasets obtained over discrete training intervals. As a result, the inner-layer weights are not updated via adaptive update laws. Moreover, there are no guarantees provided on the identification of inner-layer weights under any sufficient excitation condition.

Recent advancements in adaptive control provide Lb-DNN controllers with real-time updates for all weights for several architectures in [125–131]. Although these recent online DNN results eliminate the restriction of offline training and allow for sustained learning, they are designed to address the trajectory tracking problem based on tracking error feedback, and are not applicable to perform system identification for ADP due to the lack of parameter convergence guarantees. Thus, it is desirable to construct adaptation laws to identify the system for incorporation in ADP.

A common challenge in system identification is the lack of availability of state-derivative information. Previous results such as [108, 115, 116] and Chapter 3 use integrators to eliminate the requirement of state derivative information. However, integrators do not assist in identifying the inner-layer weights due to the nonlinear parameterization of the DNN. Additionally, the nonlinear parameterization also makes it difficult to yield performance guarantees on the system identification.

This chapter introduces the first ADP method involving an Lb-DNN as an adaptive system identifier. The developed identifier uses a least squares adaptation law with a bounded gain forgetting factor to update the weights of all layers of the DNN. To overcome the challenges posed by the lack of state-derivative information, we construct a RISE-based observer that provides a secondary estimate of the dynamics. While the RISE-based observer can provide an estimate of the dynamics, it would only be an instantaneous estimate and could not be used in BE extrapolation. The difference between the two estimates is calculated as an identification error which is used to develop a least squares adaptive update law [132]. Through a combined Lyapunov-based stability analysis, the system identifier composed of the DNN and RISE-based dynamics observer is shown to exponentially converge to a neighborhood of the DNN weight estimation error, provided the Jacobian of the DNN satisfies the persistence of excitation (PE) condition. Simultaneously, the approximate DNN-based model is used to achieve approximate BE extrapolation. The resulting ADP formulation is shown to

achieve convergence of the developed control policy to a neighborhood of the optimal control policy. Comparative simulation results show that the adaptive DNN yields 65.73% improved RMS regulation error, 31.82% improved controller error, and 78.97% improved function approximation error in comparison to the previously developed multi-timescale DNN.

1.3 Notation

For notational brevity, time-dependence is omitted while denoting trajectories of the dynamic systems. For example, given the trajectories $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ and $y : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, the equation $f + h(y, t) = g(x)$ should be interpreted as $f(t) + h((y(t)), t) = g(x(t))$. The Jacobian $\left[\frac{\partial f(x,y)}{\partial x_1}^T, \ldots, \frac{\partial f(x,y)}{\partial x_n}^T\right]$ is denoted by $\nabla_x f(x, y)$. Unless otherwise specified, let $\nabla \triangleq \nabla_x$. A square diagonal matrix with elements of vector y on the main diagonal is denoted by diag(y). Matrices of ones and zeros with n rows and m columns are denoted by $\mathbf{1}_{n\times m}$ and $\mathbf{0}_{n\times m}$, respectively. An $n \times n$ identity matrix is denoted by $I_{n\times n}$. Both the Euclidean norm for vectors and the Frobenius norm for matrices are denoted by $\|\cdot\|$. The vectorization operator is denoted by $\mathrm{vec}(\cdot)$. The operator $\lambda_{\min}(\cdot)$ represents the minimum eigenvalue of the argument.The cardinality of a set A is denoted by $|A|$. Let the subscript p define the quantity or function belonging to the p^{th} subsystem of the overall system. Let $p \in \mathcal{P}$, where $\mathcal{P} \subset \mathbb{N}$ and $|\mathcal{P}| < \infty$ represent a family of switched subsystems.

The space of essentially bounded Lebesgue measurable functions is denoted by $\mathcal{L}_\infty$. The pseudo-inverse of full row rank matrix $A \in \mathbb{R}^{n\times m}$ is denoted by A^+, where $A^+ \triangleq A^\top \left(AA^\top\right)^{-1}$. The right-to-left matrix product operator is represented by $\overset{\curvearrowleft}{\prod}$, i.e., $\overset{\curvearrowleft}{\prod_{p=1}^{m}} A_p = A_m \ldots A_2 A_1$ and $\overset{\curvearrowleft}{\prod_{p=a}^{m}} A_p = I$ if $a > m$. The Kronecker product is denoted by $\otimes$. Function compositions are denoted using the symbol $\circ$, e.g., $(g \circ h)(x) = g(h(x))$, given suitable functions g and h. Given $w \in \mathbb{R}$ and some functions f and g, the notation $f(w) = \mathcal{O}^m(g(w))$ means that there exists some constants $M \in \mathbb{R}_{>0}$ and $w_0 \in \mathbb{R}$ such that $\|f(w)\| \leq M \|g(w)\|^m$ for all $w \geq w_0$. Given some matrix $A \triangleq [a_{i,j}] \in \mathbb{R}^{n\times m}$, where

$a_{i,j}$ denotes the element in the i^{th} row and j^{th} column of A, the vectorization operator is defined as $\mathrm{vec}(A) \triangleq [a_{1,1}, \ldots, a_{n,1}, \ldots, a_{1,m}, \ldots, a_{n,m}]^\top \in \mathbb{R}^{nm}$. The space of continuous functions with continuous first n derivatives is denoted by $\mathcal{C}^n$. The notation $\overset{a.a.t.}{(\cdot)}$ denotes that the relation $(\cdot)$ holds for almost all time (a.a.t.). Given any $A \in \mathbb{R}^{p\times a}$, $B \in \mathbb{R}^{a\times r}$, and $C \in \mathbb{R}^{r\times s}$, the vectorization operator satisfies the property [133, Proposition 7.1.9]

$$\mathrm{vec}(ABC) \;=\; (C^\top \otimes A)\mathrm{vec}\,(B)\,. \tag{1–1}$$

Differentiating (1–1) on both sides with respect to $\mathrm{vec}\,(B)$ yields the property

$$\frac{\partial}{\partial \mathrm{vec}\,(B)}\mathrm{vec}(ABC) \;=\; C^\top \otimes A. \tag{1–2}$$

A function $y : \mathcal{I}_y \to \mathbb{R}^n$ is called a *Filippov solution* of $\dot{y} = h(y,t)$ on the interval $\mathcal{I}_y \subseteq \mathbb{R}_{\geq 0}$, given some Lebesgue measurable and locally essentially bounded function $h : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, if y is absolutely continuous on $\mathcal{I}_y$, and $\dot{y} \in K\,[h]\,(y,t)$ for almost all $t \in \mathcal{I}_y$, where $K\,[\cdot]$ denotes the Filippov set-valued map defined in [134, Equation 2b].

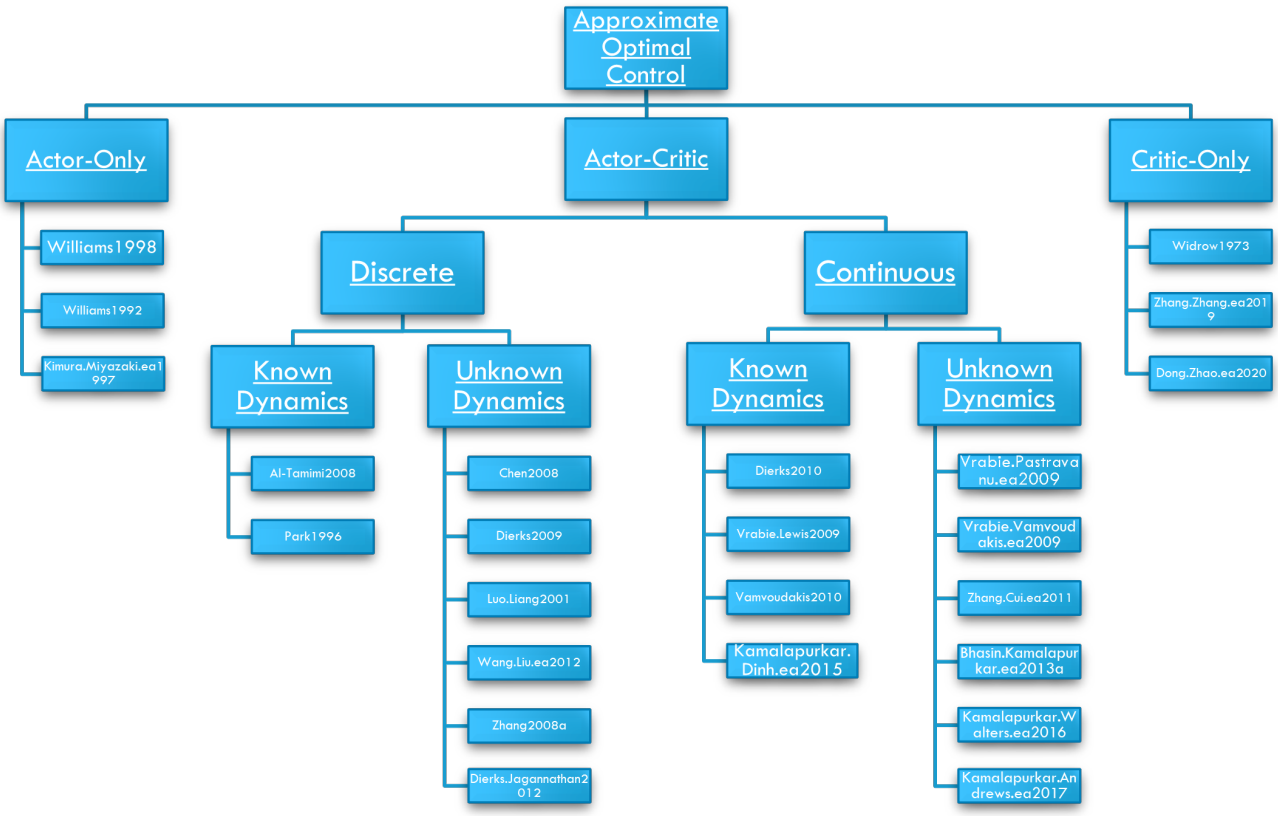

Figure 1-2. Diagram of ADP-based literature review results

CHAPTER 2
HIERARCHICAL REINFORCEMENT LEARNING-BASED SUPERVISORY CONTROL OF UNKNOWN NONLINEAR SYSTEMS

In this chapter and the work in [120], a supervisory control approach using HRL is developed to approximate the solution to optimal regulation problems for a control-affine, continuous-time nonlinear system with unknown drift dynamics. This result contains two objectives. The first objective is to approximate the optimal control policy that minimizes the infinite-horizon cost function of each ADP sub-controller. The second objective is to design a switching rule, by comparing the approximated optimal value functions of the ADP sub-controllers, to ensure that switching yields a lower cost than not switching. An ICL-based parameter identifier approximates the unknown drift dynamics. UUB regulation of the system states to a neighborhood of the origin, and convergence of the approximate control policy to a neighborhood of the optimal control policy, are proven using a Lyapunov-based stability and dwell-time analysis. Simulation results are presented which demonstrate the effectiveness of the developed method. Implementing the developed HRL controller yields a total cost that is 37% less than the total cost of implementing one ADP sub-controller and an improved rise time.

2.1 Problem Formulation

Consider a continuous-time, control-affine nonlinear dynamical system

$$\dot{x} = f(x) + g(x)\, u \tag{2–1}$$

where $x \in \mathbb{R}^n$ denotes the system state trajectory, $u \in \mathbb{R}^m$ denotes the control input, $f : \mathbb{R}^n \to \mathbb{R}^n$ denotes the drift dynamics, and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ denotes the control effectiveness.

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

Assumption 2.2. The function g is a known locally Lipschitz function, bounded such that $0 < \|g(x)\| \leq \overline{g}\ \forall x \in \mathbb{R}^n$, where $\overline{g} \in \mathbb{R}_{>0}$ is the supremum over all x of the maximum singular values of $g(x)$.

2.1.1 Control Objective

Let $\mathcal{P} \subset \mathbb{N}$ with $\mathcal{P} < \infty$ represent a family of subsystems, and let the subscript p define the quantity or function belonging to the p^{th} subsystem of the overall system. Let $p \in \mathcal{P}$, where $\mathcal{P} \subset \mathbb{N}$ and $|\mathcal{P}| < \infty$ represent a family of switched subsystems. The cost function

$$J_p(x, u_p) = \int_{t_0}^{\infty} Q_p(x) + u_p^{\top} R_p u_p \, d\tau, \tag{2–2}$$

denotes the cost of running subsystem p the entire time. The cost function

$$J_{p(t)}(x, u) = \int_{t_0}^{\infty} Q_{p(t)}(x) + u_{p(t)}^{\top} R_{p(t)} u_{p(t)} \, d\tau, \tag{2–3}$$

denotes the cost of switching between subsystems. The control objective is to solve the infinite horizon optimal regulation problem online i.e. find an optimal control policy u that minimizes the cost functional for the p^{th} subsystem and to design the switching rule so that the cost in (2–3) is smaller than the cost in (2–2).

In (2–2), $Q_p : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is a positive definite (PD) cost function where Q_p satisfies $\underline{q_p}(\|x\|) \leq Q_p(x) \leq \overline{q_p}(\|x\|)$ for $\underline{q_p}, \overline{q_p} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, and $R_p \in \mathbb{R}^{m \times m}$ is a user-defined constant PD symmetric cost matrix.

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

$$V_p^*(x) \triangleq \min_{u \in U} \int_{t}^{\infty} Q_p(x) + u_p^{\top} R_p u_p \, d\tau, \tag{2–4}$$

where $U \subseteq \mathbb{R}$ is the action space for u_p.

Remark 2.1. While each subsystem has the same set of dynamics, each has a different state penalty function Q_p, a different cost penalty matrix R_p and, thereby, a different respective controller. There are a user-defined number of cost functions that yield different desirable behavior, but since (2–1) is unknown a priori, supervised switching between the cost functions with different parameters will result in different expressions for (2–4), which motivates selecting the V_p^* with the lowest value for the specific unknown system.

Assumption 2.3. The optimal value function V_p^* is continuously differentiable for all $p \in \mathcal{P}$ [92].

The optimal value function is the solution to the corresponding HJB equation

$$0 = \nabla V_p^*(x)\left(f(x) + g(x)\,u_p^*\right) + Q_p(x) + u_p^{*\top} R_p u_p^*, \tag{2–5}$$

where $u_p^* : \mathbb{R}^n \to \mathbb{R}^m$ is the optimal control policy for the p^{th} mode. The HJB equation in (2–5) has the boundary condition $V_p^*(0) = 0$. The optimal control policy u_p^* is defined as

$$u_p^*(x) = -\frac{1}{2} R_p^{-1} g(x)^\top \left(\nabla V_p^*(x)\right)^\top. \tag{2–6}$$

Remark 2.2. Under Assumptions 2.1-2.3, the optimal value function is the unique PD solution of the HJB equation for each system. The approximation of the PD solution to the HJB is guaranteed by the appropriate selection of Lyapunov-based update laws and initial weight estimates [104].

2.1.2 Value Function Approximation

The optimal control policy in (2–6) requires knowledge of the optimal value function, which is generally unknown for nonlinear systems. Let $\Omega \subset \mathbb{R}^n$ be a compact set. The subsequent stability analysis guarantees that if x is initialized in an appropriately-sized subset of Ω, then it will stay in Ω. Using the Universal Function Approximation Theorem, the optimal value function can be approximated with an NN in Ω as

$$V_p^*(x) = W_p^\top \phi_p(x) + \epsilon_p(x) \;\forall x \in \Omega, \tag{2–7}$$

where $W_p \in \mathbb{R}^L$ is a vector of unknown weights, $\phi_p : \mathbb{R}^n \to \mathbb{R}^L$ is a user-defined vector of basis functions, and $\epsilon_p : \mathbb{R}^n \to \mathbb{R}$ is the bounded function reconstruction error. For brevity, each subsystem uses the same number of elements in the basis function vector L. Substituting (2–7) into (2–6), the NN representation of the p^{th} mode optimal control policy in (2–6) is

$$u_p^*(x) = -\frac{1}{2} R_p^{-1} g(x) \left(\nabla \phi_p(x) W_p + \nabla \epsilon_p(x)\right)^\top. \tag{2–8}$$

Assumption 2.4. There exists a set of known positive constants $\overline{W}, \overline{\phi}, \overline{\nabla \phi}, \overline{\epsilon}, \overline{\nabla \epsilon} \in \mathbb{R}_{>0}$ such that $\sup_{p \in \mathcal{P}} \|W_p\| \leq \overline{W}$, $\sup_{x \in \Omega,\, p \in \mathcal{P}} \|\phi_p(x)\| \leq \overline{\phi}$, $\sup_{x \in \Omega,\, p \in \mathcal{P}} \|\nabla \phi_p(x)\| \leq \overline{\nabla \phi}$, $\sup_{x \in \Omega,\, p \in \mathcal{P}} \|\epsilon_p(x)\| \leq \overline{\epsilon}$, and $\sup_{x \in \Omega,\, p \in \mathcal{P}} \|\nabla \epsilon_p(x)\| \leq \overline{\nabla \epsilon}$ for all p [135, Assumptions 9.1.c-e].

The critic weight estimate vector $\widehat{W}_{c,p} \in \mathbb{R}^L$ is used to approximate (2–7), resulting in the optimal value function estimate $\widehat{V}_p : \mathbb{R}^n \times \mathbb{R}^L \to \mathbb{R}$, defined as

$$\widehat{V}_p\left(x, \widehat{W}_{c,p}\right) \triangleq \widehat{W}_{c,p}^\top \phi_p(x). \tag{2–9}$$

The actor weight estimate vector $\widehat{W}_{a,p} \in \mathbb{R}^L$ is used to approximate (2–8), resulting in the optimal control policy estimate $\hat{u}_p : \mathbb{R}^n \times \mathbb{R}^L \to \mathbb{R}^m$, defined as

$$\hat{u}_p\left(x, \widehat{W}_{a,p}\right) \triangleq -\frac{1}{2} R_p^{-1} g(x)^\top \left(\nabla \phi_p(x)^\top \widehat{W}_{a,p}\right). \tag{2–10}$$

2.2 Hierarchical Agent

2.2.1 Switching Rule

The hierarchical agent is tasked with identifying which policy minimizes the infinite horizon cost functional based on a switching policy. The supervisory algorithm

$$\sigma \triangleq \operatorname{argmin}_{p \in \mathcal{P}} \left\{\widehat{V}_p\left(x, \widehat{W}_{c,p}\right)\right\} \tag{2–11}$$

returns the number of the subsystem associated with the smallest approximated cost-to-go, computed using estimates of the optimal value function corresponding to each subsystem. The switched signal in (2–11) will switch in real-time; therefore, to guarantee

closed-loop stability of the overall system, a subsequently defined dwell-time condition must be satisfied. The optimal value function approximations are used to quantitatively compare all individual ADP controllers $p \in \mathcal{P}$ in real-time. The switching rule in (2–11) evaluates all of the approximated costs-to-go and selects the applied control input u in (2–1) as

$$u = \hat{u}_\sigma \left(x, \widehat{W}_{a,p} \right), \tag{2–12}$$

that corresponds to the smallest optimal value function approximation at a given time as seen in Figure 2-1. The goal is to determine which control policy provides the least approximate cost-to-go for the system.

2.2.2 System Identification

In addition to approximating the optimal value function for each subsystem, there is also uncertainty in the drift dynamics, and those uncertain parameters are approximated using system identification. All subsystem controllers have the same dynamical system. The system parameters are being identified strictly in one drift dynamics model. To facilitate the online system identification, assume the drift dynamics f are linearly parameterizable such that $f(x) = Y(x)\theta$, where $Y : \mathbb{R}^n \to \mathbb{R}^{n \times s}$ is the known regression matrix and $\theta \in \mathbb{R}^s$ is a vector of constant unknown parameters. Let $\hat{\theta} \in \mathbb{R}^s$ be an approximation of the unknown parameter vector θ, which is updated according to the subsequently defined parameter update policy. The uncertain drift dynamics f are approximated by $\hat{f} : \mathbb{R}^n \times \mathbb{R}^s \to \mathbb{R}^n$ which is defined as $\hat{f}\left(x, \hat{\theta}\right) \triangleq Y(x)\hat{\theta}$. The parameter estimate $\hat{\theta}$ is updated with the ICL-based update policy [136]

$$\dot{\hat{\theta}}(t) \triangleq k_{ICL}\Gamma_\theta \sum_{j=1}^{M} \mathcal{Y}_j^\top \left(x(t_j) - x(t_j - \Delta t) - \mathcal{U}_j - \mathcal{Y}_j \hat{\theta} \right), \tag{2–13}$$

where $k_{ICL} \in \mathbb{R}_{>0}$ and $\Gamma_\theta \in \mathbb{R}^{s \times s}$ are user-selected PD constants, $\mathcal{Y}_j \triangleq \mathcal{Y}(t_j)$, $\mathcal{U}_j \triangleq \mathcal{U}(t_j)$, $\mathcal{Y}(t) \triangleq \int_{\max[t-\Delta t,0]}^{t} Y(x(\tau))\, d\tau$, and $\mathcal{U}(t) \triangleq \int_{\max[t-\Delta t,0]}^{t} g(x(\tau))\, u(\tau)\, d\tau$. The

parameter update law in (2–13) can be rewritten in an analytical form as

$$\dot{\hat{\theta}} = k_{ICL}\Gamma_\theta \sum_{j=1}^{M} \mathcal{Y}_j^\top \mathcal{Y}_j \tilde{\theta}, \tag{2–14}$$

where $\tilde{\theta} \triangleq \theta - \hat{\theta}$ is the parametric error.

Assumption 2.5. A history stack of recorded state and control inputs $\{x(t_j), u(t_j)\}_{j=1}^{M}$ is available that satisfies $\underline{\mathcal{Y}} \triangleq \lambda_{\min}\left\{\sum_{j=1}^{M} \mathcal{Y}^\top{}_j \mathcal{Y}_j\right\} > 0$ and ensures the finite excitation condition in [136] is satisfied *a priori*. The *a priori* availability of the history stack is used for ease of exposition but is not necessary [92].

2.3 Bellman Error

The BE indicates how close the actor and critic weight estimates are to their ideal weight values. By substituting the approximate optimal value function $\widehat{V}_p\left(x, \widehat{W}_{c,p}\right)$ and approximate optimal control policy $\hat{u}_p\left(x, \widehat{W}_{a,p}\right)$ into (2–5), the BE $\hat{\delta}_p : \mathbb{R}^n \times \mathbb{R}^L \times \mathbb{R}^L \times \mathbb{R}^s \to \mathbb{R}$ is defined as

$$\hat{\delta}_p\left(x, \widehat{W}_{c,p}, \widehat{W}_{a,p}, \hat{\theta}\right) \triangleq Q_p(x) + \hat{u}_p\left(x, \widehat{W}_{a,p}\right)^\top R_p \hat{u}_p\left(x, \widehat{W}_{a,p}\right) + \nabla \widehat{V}_p\left(x, \widehat{W}_{c,p}\right)\left(Y(x)\hat{\theta} + g(x)\hat{u}_p\left(x, \widehat{W}_{a,p}\right)\right). \tag{2–15}$$

While (2–15) is used for implementation, to facilitate the subsequent stability analysis, the BE can be expressed in terms of the weight approximation errors $\widetilde{W}_{c,p} \triangleq W_p - \widehat{W}_{c,p}$ and $\widetilde{W}_{a,p} \triangleq W_p - \widehat{W}_{a,p}$. Subtracting (2–5) from (2–15) and substituting (2–7)-(2–10), the analytical form of the BE in (2–15) can be expressed as

$$\hat{\delta}_p\left(x, \widehat{W}_{c,p}, \widehat{W}_{a,p}, \hat{\theta}\right) = -\omega_p^\top \widetilde{W}_{c,p} - W_p^\top \nabla\phi_p Y(x)\tilde{\theta} + \frac{1}{4}\widetilde{W}_{a,p}^\top G_{\phi,p}(x)\widetilde{W}_{a,p} + O_p(x), \tag{2–16}$$

where $\omega_p : \mathbb{R}^n \times \mathbb{R}^L \times \mathbb{R}^s \to \mathbb{R}^n$ is $\omega_p\left(x, \widehat{W}_{a,p}, \hat{\theta}\right) \triangleq \nabla\phi_p(x)\left(\hat{f}\left(x, \hat{\theta}\right) + g(x)\hat{u}_p\left(x, \widehat{W}_{a,p}\right)\right)$ and $O_p(x) \triangleq \frac{1}{2}\nabla\epsilon_p(x) G_{R,p} \nabla\phi_p(x)^\top W_p + \frac{1}{4} G_{\epsilon,p} - \nabla\epsilon_p(x) f(x)$. The functions $G_{R,p}$, $G_{\phi,p}$, and $G_{\epsilon,p}$ are defined as $G_{R,p}(x) \triangleq g_p(x) R_p^{-1} g_p(x)^\top$, $G_{\phi,p}(x) \triangleq \nabla\phi_p(x) G_{R,p}(x) \nabla\phi_p(x)^\top$, and $G_{\epsilon,p}(x) \triangleq \nabla\epsilon_p(x) G_{R,p}(x) \nabla\epsilon_p(x)^\top$ respectively.

Bellman Error Extrapolation

As described in [92], the BE in (2–15) can be calculated at any user-defined point in the state space using a user-selected state x_i, the critic weight estimate $\widehat{W}_{c,p}$, and the actor weight estimate $\widehat{W}_{a,p}$. To estimate the value function over the compact set, the estimate of the system model from the aforementioned online system identifier is used to evaluate the BE along a set of off-trajectory points via BE extrapolation. BE extrapolation yields simultaneous exploration and exploitation, and can provide simulation of experience, enabling faster policy learning.

To facilitate sufficient exploration, the BE is extrapolated from the user-defined off-trajectory points $\{x_i : x_i \in \Omega\}_{i=1}^{N_p}$, where $N_p \in \mathbb{N}$ denotes a user-specified number of total extrapolation trajectories in the compact set Ω. Each subsystem p has its own distinct set of gain values, data, and update laws.

Assumption 2.6. On the compact set, Ω, a finite set of off-trajectory points $\{x_i : x_i \in \Omega\}_{i=1}^{N_p}$ are user-selected such that $0 < \underline{c}_p \triangleq \inf_{t\in\mathbb{R}_{\geq 0}} \lambda_{\min}\left\{\frac{1}{N_p}\sum_{i=1}^{N_p}\frac{\omega_{i,p}\omega_{i,p}^\top}{\rho_{i,p}^2}\right\}$ for all $p \in \mathcal{P}$, where $\rho_{i,p} = 1 + \nu_p\omega_{i,p}^\top\Gamma_p\omega_{i,p}$, $\nu_p \in \mathbb{R}_{>0}$ is a user-defined gain, $\Gamma_p : \mathbb{R}^{L\times L}$ is a time-varying least-squares gain matrix, and $\underline{c}_p$ is a constant scalar lower bound of the value of each input-output data pair's minimum eigenvalues for the p^{th} subsystem [92].

2.4 Update Laws for Actor and Critic Weights

The actor and critic weights for each subsystem are updated simultaneously via BE error extrapolation. In the subsequent weight update laws, $\eta_{c,p}$, $\eta_{a1,p}$, $\eta_{a2,p}$, $\lambda_p \in \mathbb{R}_{>0}$ are positive constant adaptation gains, and $\underline{\Gamma}_p$, $\overline{\Gamma}_p \in \mathbb{R}_{>0}$ denote lower and upper bounds for Γ_p. The critic update law for the p^{th} mode $\dot{\widehat{W}}_{c,p} \in \mathbb{R}^L$ is defined as

$$\dot{\widehat{W}}_{c,p} \triangleq -\eta_{c,p}\Gamma\frac{1}{N_p}\sum_{i=1}^{N_p}\frac{\omega_{i,p}}{\rho_{i,p}}\delta_{i,p}. \tag{2–17}$$

The actor update law for the p^{th} mode $\dot{\widehat{W}}_{a,p} \in \mathbb{R}^L$ is defined as

$$\dot{\widehat{W}}_{a,p} \triangleq -\eta_{a1,p}\left(\widehat{W}_{a,p} - \widehat{W}_{c,p}\right) - \eta_{a2,p}\widehat{W}_{a,p} + \eta_{c,p}\frac{1}{N_p}\sum_{i=1}^{N_p}\frac{G_{\phi i,p}^\top\widehat{W}_{a,p}\omega_{i,p}^\top}{4\rho_{i,p}}\widehat{W}_{c,p}. \tag{2–18}$$

The least-squares gain matrix update law of the p^{th} mode $\dot{\Gamma}_p \in \mathbb{R}^{L\times L}$ is defined as

$$\dot{\Gamma}_p \triangleq \left(\lambda_p \Gamma_p - \frac{\eta_{c,p}\Gamma_p}{N_p}\sum_{i=1}^{N_p}\frac{\omega_{i,p}\omega_{i,p}^\top \Gamma_p}{\rho_{i,p}^2}\right)\cdot \mathbf{1}_{\left\{\underline{\Gamma}_p \le \|\Gamma_p\| \le \overline{\Gamma}_p\right\}}, \tag{2–19}$$

where $\mathbf{1}_{\{\cdot\}}$ denotes the indicator function. Using (2–19) ensures that each $\underline{\Gamma}_p \le \|\Gamma_p\| \le \overline{\Gamma}_p$ for all $t \in \mathbb{R}_{>0}$. The on-trajectory points can be included in the weight update laws, such as in [92], but to focus the Lyapunov-based analysis, only off-trajectory BE extrapolation is performed.

The update laws in (2–17)-(2–19) are always active for each subsystem regardless of a subsystem's activity or inactivity. Hence, the update laws will update each subsystem p's weight estimates and least-squares gain matrix even if subsystem p is not active. Since the update laws are always learning for each subsystem, convergence of the states of each subsystem can be proven concurrently.

2.5 Stability Analysis

2.5.1 Subsystem Stability Analysis

To facilitate the stability analysis, a concatenated state $z \in \mathbb{R}^{n+2L|\mathcal{P}|+s}$ is defined as $z \triangleq \left[x^\top, \widetilde{W}_{c,1}^\top, \ldots, \widetilde{W}_{c,p}^\top, \widetilde{W}_{a,1}^\top, \ldots, \widetilde{W}_{a,p}^\top, \tilde{\theta}^\top\right]^\top$, and the candidate Lyapunov function $V_{L,p} : \mathbb{R}^{n+2L|\mathcal{P}|+s} \to \mathbb{R}_{\ge 0}$ is defined as

$$V_{L,p}(z) \triangleq V_p^*(x) + \frac{1}{2}\sum_{p\in\mathcal{P}}\widetilde{W}_{c,p}^\top \Gamma_p^{-1}\widetilde{W}_{c,p} + \frac{1}{2}\sum_{p\in\mathcal{P}}\widetilde{W}_{a,p}^\top \widetilde{W}_{a,p} + \frac{|\mathcal{P}|}{2}\tilde{\theta}^\top \Gamma_\theta^{-1}\tilde{\theta}. \tag{2–20}$$

According to [101, Lemma 4.3], (2–20) can generally be bounded as $\alpha_{1,p}(\|z\|) \le V_{L,p}(z) \le \alpha_{2,p}(\|z\|)$ using class $\mathcal{K}$ functions $\alpha_{1,p}, \alpha_{2,p} : \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$. The normalized regressors $\frac{\omega_p}{\rho_p}$ and $\frac{\omega_{i,p}}{\rho_{i,p}}$ are bounded as $\sup_{t\in\mathbb{R}_{\ge 0}}\left\|\frac{\omega_p}{\rho_p}\right\| \le \frac{1}{2\sqrt{\nu_p \underline{\Gamma}_p}}$ and $\sup_{t\in\mathbb{R}_{\ge 0}}\left\|\frac{\omega_{i,p}}{\rho_{i,p}}\right\| \le \frac{1}{2\sqrt{\nu_p \underline{\Gamma}_p}}$ for all $x \in \Omega$ and $x_i \in \Omega$, respectively. The function $G_{R,p}$ is bounded as $\sup_{x\in\Omega}\|G_{R,p}\| \le \overline{G}_p^2 \lambda_{\max}\left\{R_p^{-1}\right\}$, $G_{\phi,p}$ is bounded as $\sup_{x\in\Omega}\|G_{\phi,p}\| \le \left(\overline{\nabla\phi G_p}\right)^2 \lambda_{\max}\left\{R_p^{-1}\right\}$, and $Y(x)$ is bounded as $\sup_{x\in\Omega}\|Y(x)\| \le \overline{Y}$. To facilitate the subsequent analysis, define $r \in \mathbb{R}_{>0}$ to be the radius of a compact ball $\mathcal{B}_r \in \mathbb{R}^{n+2L|\mathcal{P}|+s}$ centered at the origin.

Theorem 2.1. *Let $x(\cdot)$ denote the trajectory of the p^{th} subsystem for a fixed p. Provided the control policy in* (2–10) *is used, the weight update laws in* (2–17)-(2–19) *are implemented, Assumptions* (2.1)-(2.6) *hold, and the conditions*

$$\eta_{a1,p} + \eta_{a2,p} \geq \frac{5}{4\sqrt{\nu_p \underline{\Gamma}_p}} \eta_{c2,p} \overline{WG_{\phi,p}} \tag{2–21}$$

$$\underline{c}_p \geq 3\frac{\eta_{a1,p}}{\eta_{c2,p}} + \frac{3\eta_{c,p}^2 \overline{W}^2}{4\eta_{c,p}\nu_p \underline{\Gamma}_p} \left(\frac{\overline{\nabla \phi}^2 \overline{Y}^2}{k_{ICL}\underline{\mathcal{Y}}} + \frac{5\overline{G_{\phi,p}}^2}{16\left(\eta_{a1,p} + \eta_{a2,p}\right)} \right) \tag{2–22}$$

$$v_{L,p}^{-1}\left(L_p\right) < \alpha_{2,p}^{-1}\left(\alpha_{1,p}\left(r\right)\right) \tag{2–23}$$

are satisfied for each individual subsystem, where L_p is a positive constant that depends on the NN bounding constants in Assumption 2.4, then the state x, every critic weight estimate error $\widetilde{W}_{c,p}$ $\forall p \in \mathcal{P}$, every actor weight estimate error $\widetilde{W}_{a,p}$ $\forall p \in \mathcal{P}$, and the parameter estimation error $\tilde{\theta}$ are UUB. Hence, each control policy $\hat{u}_p$ converges to a neighborhood of its respective optimal control policy u_p^*.

Proof. Using the HJB equation in (2–5), the BE in (2–16), the gain conditions in (2–21) and (2–22), and the weight update laws in (2–17)-(2–19), the time derivative of (2–20) can be bounded as

$$\dot{V}_{L,p} \leq -v_{L,p}\left(\|z\|\right) \;\; \forall \|z\| \geq v_{L,p}^{-1}\left(L_p\right) \tag{2–24}$$

for all $p \in \mathcal{P}$ and $t \in \mathbb{R}_{>0}$, where $v_{L,p} \triangleq \frac{1}{2}\underline{q}_p\left(\|x\|\right) + \sum_{p\in\mathcal{P}} \left[\frac{1}{12}\eta_{c,p}\underline{c}_p \left\| \widetilde{W}_{c,p} \right\|^2 + \frac{1}{20}\left(\eta_{a1,p} + \eta_{a2,p}\right) \left\| \widetilde{W}_{a,p} \right\|^2 \right] + \frac{|\mathcal{P}|}{4} k_{ICL}\underline{\mathcal{Y}} \left\| \tilde{\theta} \right\|^2$. Using (2–24), $v_{L,p}\left(\|z\|\right)$, and (2–23), [137, Theorem 4.18] can be invoked to conclude that z is UUB such that $\limsup_{t\to\infty} \|z\| \leq \alpha_{1,p}^{-1}\left(\alpha_{2,p}\left(v_{L,p}^{-1}\left(L_p\right)\right)\right)$ and the control policy $\hat{u}_p$ converges to a neighborhood of the optimal control policy u_p^*. Since $z \in \mathcal{L}_\infty$, it follows that $x, \widetilde{W}_{c,1}, ..., \widetilde{W}_{c,|\mathcal{P}|}, \widetilde{W}_{a,1}, ..., \widetilde{W}_{a,|\mathcal{P}|}, \tilde{\theta} \in \mathcal{L}_\infty$; hence, $x, \widehat{W}_{c,1}, \ldots, \widehat{W}_{c,|\mathcal{P}|}, \widehat{W}_{a,1}, \ldots, \widehat{W}_{a,|\mathcal{P}|}, \hat{\theta} \in \mathcal{L}_\infty$ and $u \in \mathcal{L}_\infty$. Additionally, every trajectory z that is initialized in the ball $\mathcal{B}_r$ is

bounded such that $z \in \mathcal{B}_r, \forall t \in \mathbb{R}_{\geq 0}, \forall p \in \mathcal{P}$. Since $z \in \mathcal{B}_r$, the states $x, \widetilde{W}_{c,1}, ..., \widetilde{W}_{c,|\mathcal{P}|}, \widetilde{W}_{a,1}, ..., \widetilde{W}_{a,|\mathcal{P}|}, \tilde{\theta}$ similarly lie in a compact set. □

Remark 2.3. See [92] for insight into satisfying the gain conditions in (2–21) and (2–22). See [92, Algorithm 1] for insight into selecting the size of the compact set Ω.

2.5.2 Switched UUB Stability Analysis

Since the unknown optimal value function $V_p^*(x)$ in (2–20) is different for each subsystem, (2–20) is not a common Lyapunov function. The previous theorem proves stability of the individual subsystems, but not stability of the overall switched system. The Lyapunov function for the switched system may instantaneously increase due to the increase in the optimal value function and the real-time updates of the weights. Due to switching between multiple Lyapunov functions, a dwell-time analysis is necessary; therefore [138, Theorem 5.2] is used to prove convergence of the overall switched system [105, Ch. 3].

2.6 Simulations

To demonstrate the effectiveness of the developed ADP technique, a simulation is performed on a control-affine nonlinear dynamical system with a two dimensional state $x = [x_1, x_2]^\top$. The control objective is to minimize each cost in (2–2) and switch between three controllers to ensure that the cost in (2–3) is lower than the cost in (2–2). There are three different reward functions and, thus, three subsystems such that $\mathcal{P} = \{1, 2, 3\}$. For value function approximation, the basis function for every subsystem is selected as $\phi = [x_1^2, x_1 x_2, x_2^2]$. The initial conditions for the subsystems are $x(0) = [-15, 15]^\top$, $\Gamma(0) = 250 * \mathbf{1}_{3\times 1}$, $\widehat{W}_{c,1}(0) = \widehat{W}_{a,1}(0) = 0.5 * \mathbf{1}_{3\times 1}$, $\widehat{W}_{c,2}(0) = \widehat{W}_{a,2}(0) = 0.05 * \mathbf{1}_{3\times 1}$, and $\widehat{W}_{c,3}(0) = \widehat{W}_{a,3}(0) = 0.1 * \mathbf{1}_{3\times 1}$. The system dynamics in (2–1) are

$$f = \begin{bmatrix} x_1 & x_2 & 0 & 0 \\ 0 & 0 & x_1 & x_2\left(1 - (cos(2x_1) + 2)^2\right) \end{bmatrix} \theta, \qquad g = \begin{bmatrix} 0 \\ cos(2x_1) + 2 \end{bmatrix} \quad (2\text{–}25)$$

where θ =$[-1, 1, -0.5, -0.5]^\top$ is the vector of unknown parameters. The simulation parameters are selected as $\eta_{c,p} = 10$, $\eta_{a1,p} = 15$, $\eta_{a2,p} = 0.1$, $\overline{\Gamma}_p = 1000$, $\underline{\Gamma}_p = 1$, $\lambda_p = 0.1$, $\nu_p = 0.2$, and $N_p = 10\ \forall p \in \mathcal{P}$. The cost functions for each subsystem are

$$J_1(x, u) = \int_{t_0}^{\infty} Q_1(x) + u^\top R_1 u\, d\tau, \tag{2–26}$$

$$J_2(x, u) = \int_{t_0}^{\infty} Q_2(x) + u^\top R_2 u\, d\tau, \tag{2–27}$$

$$J_3(x, u) = \int_{t_0}^{\infty} Q_3(x) + u^\top R_3 u\, d\tau, \tag{2–28}$$

where $Q_1 = 0.1x^\top \left(xx^\top\right) x$, $Q_2 = x^\top \mathrm{diag}\left([5,\ 1]\right) x$, $Q_3 = x^\top \mathrm{diag}\left([1,\ 12]\right) x$, $R_1 = 1$, $R_2 = 5$, and $R_3 = 5$.

Figure 2-2 shows that the developed HRL method enables the system states to converge to the origin while switching between subsystems with different cost functions. The dwell time is selected as 0.25 seconds. In this example, the switching instances occur at approximately 0.25 and 1.83 seconds. Figure 2-2 illustrates that the state converges as the system switches to the controller with the lowest-valued approximate cost-to-go. Specifically, after the first switching instance, the convergence rate of x_2 significantly improves. Because of the different cost functions in (2–26)-(2–28), it is expected that each transient behavior, generally, will differ.

Figure 2-3 shows the performance of four subsystems: Controller 1, Controller 2, Controller 3, and the HRL controller. The results illustrate that the HRL controller, which switches to the controller with the lowest-valued approximated cost-to-go in real-time, results in the fastest state convergence. Therefore, it is shown that the controller with the HRL agent yields more desirable behavior for the unknown system than the controllers without hierarchical agents.

Table 2-1 compares the four methods. The costs of Controllers 1, 2, and 3 are summed continuously throughout the duration of the simulation. The cost of the HRL

controller is calculated by summing the cost of each controller while active. The HRL controller has a lower total cost and 99% rise time in comparison to Controllers 1-3. Excluding the developed HRL controller, Controller 2 has the lowest cost, and Controller 1 has the fastest rise time. By switching between these control policies, the developed HRL controller captures the positive qualities of both controllers and outperforms them using these two metrics. The HRL controller yields a total cost that is 37% less than that of Controller 2. The HRL controller also yields a rise time that is 30% faster than that of Controller 1. The strengths of the HRL controller compared to the individual sub-controllers are shown in Table 2-1 and are illustrated in Figure 2-3.

2.7 Concluding Remarks

In this chapter, supervisory control is implemented within ADP to develop a supervised switching signal. The method introduced in this chapter presents a way to intelligently switch between controllers using a hierarchy to optimize a certain performance index. A supervisory switching policy is used to switch between the control policy with the least approximated cost-to-go in real-time. Stability of each subsystem is proven via a Lyapunov-based stability analysis. The overall switched system is proven to be stable in the sense that the system states converge to a neighborhood of the origin and the applied policy converges to a neighborhood of the selected optimal policy. Simulation results are presented to show that implementing the developed HRL controller yields a total cost of 37% less than the total cost of implementing one ADP sub-controller and an improved rise time.

The developed technique will be significant for problems that require more than one cost function to achieve the desired control objective. This chapter provides inroads for ADP to become more integrated into switched system/hybrid control problems. Additional, this chapter introduces a hierarchical framework that can be implemented in a multi-agent control scenario. Using the framework subsequently introduced in Chapter

3, HRL can be used to switch between controlling different pursuing agents depending on the behavior of the evading agent(s) to achieve the desired objective.

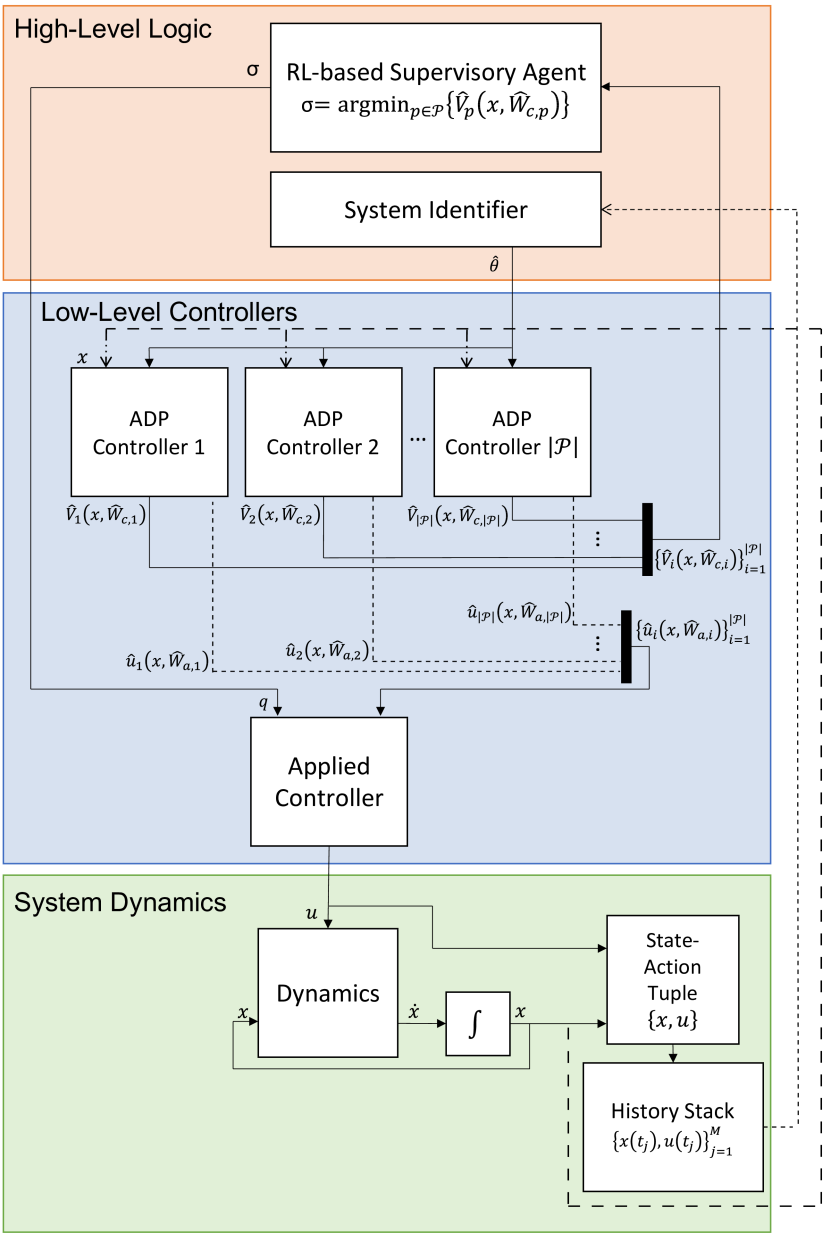

Figure 2-1. The high-level logic in the hierarchical supervisory control architecture contains the RL-based supervisory agent and the system identifier. The supervisory agent evaluates the family of $\widehat{V}_p$s and outputs the number of the subsystem with the lowest value function approximation. The system identifier approximates the uncertain model parameters; these parameter estimates are used to update the actor and critic weight estimates. Each ADP controller contains a different cost function, and the objective is to minimize each subsystem's respective cost-to-go. Together, the high-level logic and low-level controllers select the control input with the lowest approximated cost-to-go. The selected controller in (2–12) is applied to the dynamical system in (2–1). Then history stack data is provided to the high-level system identifier, the new state is provided to the low-level ADP controllers, and the policy in (2–12) is evaluated again.

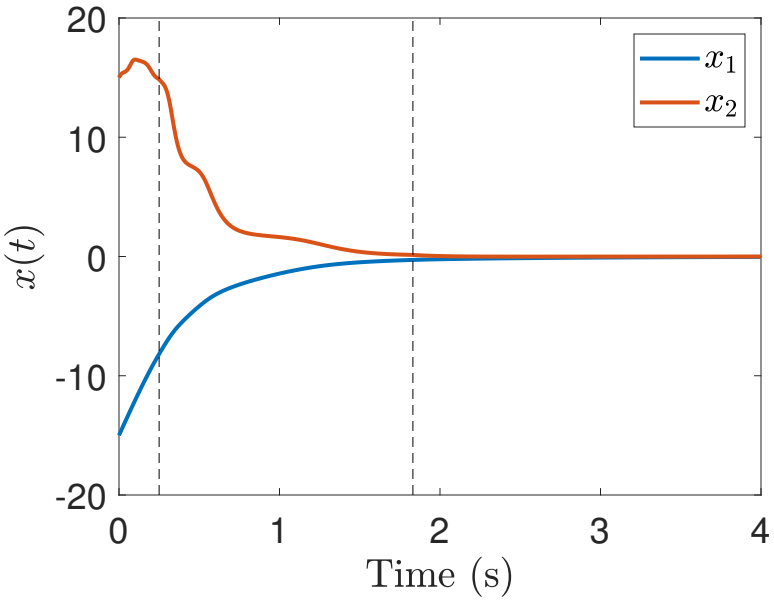

Figure 2-2. State trajectory for the two-state system while using the HRL controller. The black dashed lines represent the time at which the system switches to the controller with the lowest-valued approximate cost-to-go.

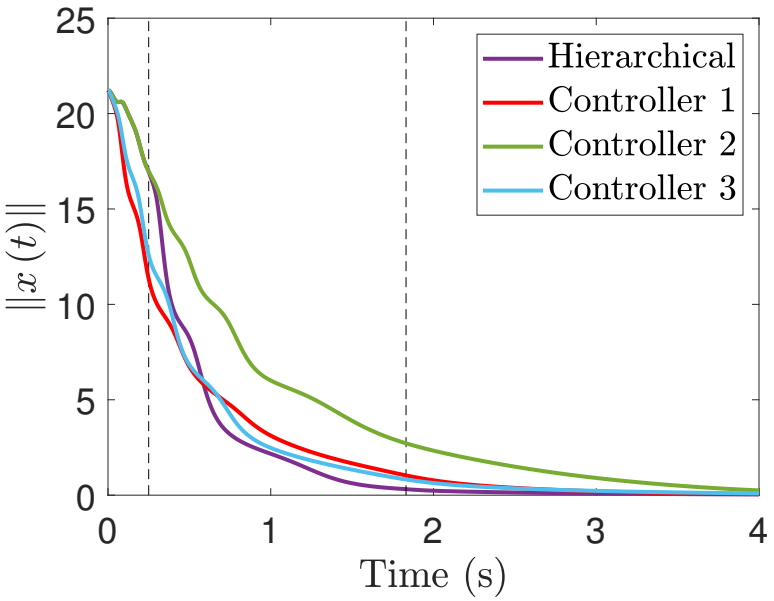

Figure 2-3. Comparison of the state convergence of three controllers and one HRL switching controller. The state $\|x\|$ converges in less time while using the developed HRL controller in comparison to controllers 1-3.

Table 2-1. Simulation Performance Metrics

Controller	Total Cost	99% Rise Time (s)
HRL Controller	1073	2.08
Controller 1	2683	2.97
Controller 2	1701	4.11
Controller 3	1940	3.07

CHAPTER 3
APPROXIMATE OPTIMAL INDIRECT REGULATION OF AN UNKNOWN AGENT WITH A LYAPUNOV-BASED DEEP NEURAL NEWORK

While ICL can facilitate system identification to update linear parameterized estimates in Chapter 2, and ICL has been used for system identification to update SNNs, ICL has never been used for more advanced function approximation methods such as DNNs. In this chapter and the work in [108], we build on the concepts in Chapter 2 to explore another class of problems known as indirect control problems where the states of a dynamic system are regulated by an influencing agent through an interaction dynamic. Specifically, the indirect herding problem is considered where a pursuing agent is tasked with intercepting and regulating and evading agent to a desired goal location. An approximate optimal policy is developed for a pursuing agent to indirectly regulate an evading agent coupled by an unknown interaction dynamic. ADP is used to design a controller for the pursuing agent to optimally influence the evading agent to a goal location. Since the interaction dynamic between the agents is unknown, ICL is used to update an Lb-DNN to facilitate sustained learning and system identification. A Lyapunov-based stability analysis is used to show UUB convergence. Comparative simulations are provided which show that the DNN outperforms the SNN.

3.1 Problem Formulation

The problem is formulated as a pursuing agent tasked with optimally intercepting and escorting an uncooperative evading agent to a desired goal state using unknown interaction dynamics between the pursuer and evader. The evader dynamics are

$$\dot{z} = f\left(z, \eta\right), \tag{3–1}$$

where $z : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$ denotes the state of the evader, $\eta : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$ denotes the state of the pursuer, $t_0 \in \mathbb{R}_{\geq 0}$ denotes the initial time, and $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ denotes the non-affine, unknown locally Lipschitz interaction function. While the evader dynamics in (3–1) cannot be directly controlled, the evader can be influenced through interaction with

the pursuer, which is directly controllable. The pursuer dynamics are

$$\dot{\eta} = h(z, \eta) + g(\eta) u, \tag{3–2}$$

where $h : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ denotes an unknown locally Lipschitz function representing the pursuer drift dynamics, $g : \mathbb{R}^n \to \mathbb{R}^{n \times m_\eta}$ denotes the known control effectiveness matrix, and $u : \mathbb{R}_{\geq t_0} \to \mathbb{R}^{m_\eta}$ is the pursuer's control input.

Assumption 3.1. There exist class $\mathcal{K}$ functions $\overline{\alpha}_1, \overline{\alpha}_2 \in \mathbb{R}_{\geq 0}$ that allow the uncertain dynamics in (3–1) to be bounded as $\|f(z, \eta)\| \leq \overline{\alpha}_1(\|z - \eta\|) + \overline{\alpha}_2(\|z - z_g\|)$, where $z_g \in \mathbb{R}^n$ denotes a fixed goal location [112].

Assumption 3.2. The control effectiveness matrix $g(\eta)$ is bounded and full column rank for all $\eta \in \mathbb{R}^n$, and $g^+ \in \mathbb{R}^{m_\eta \times n}$ is a locally Lipschitz and bounded pseudo inverse defined as $g^+ \triangleq \left(g^\top g\right)^{-1} g^\top$ [88].

To quantify the control objective, a regulation error denoted by $e_z : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$ is defined as

$$e_z \triangleq z - z_g, \tag{3–3}$$

where $z_g \in \mathbb{R}^n$ denotes a fixed user-defined goal location that is only known to the pursuer. It is not possible to directly control the error in (3–3). To address this, a backstepping formulation is used to design a virtual desired state that enables the pursuer to indirectly minimize (3–3) by tracking a virtual desired state denoted by $\eta_d : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$. Traditional backstepping cannot be used due to the nonlinear relationship in the dynamics; hence, additional error system development is motivated by backstepping approaches. To quantify the pursuer's ability to track the virtual desired state, an auxiliary error $e_\eta : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$ is defined as

$$e_\eta \triangleq \eta - \eta_d. \tag{3–4}$$

To quantify the virtual desired state objective, an additional auxiliary error $e_d : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$ is defined as

$$e_d \triangleq \eta_d - z_g - k_d e_z, \tag{3–5}$$

where $k_d \in \mathbb{R}$ denotes a positive control gain. The time derivative of the virtual desired state η_d is designed as $\dot{\eta}_d \triangleq \mu_d$, where $\mu_d : \mathbb{R}_{\geq t_0} \to \mathbb{R}^n$ is the subsequently designed virtual input that minimizes (3–5). To facilitate the minimization of (3–3)-(3–5), let $x \triangleq \left[e_z^\top, e_d^\top, e_\eta^\top\right]^\top$ and $x_d \triangleq \left[e_z^\top, e_d^\top, 0_{1\times n}\right]^\top$ denote the concatenated state and desired concatenated state, respectively. Additionally, the mappings $s_1, s_2 : \mathbb{R}^{3n} \to \mathbb{R}^n$ are defined as $s_1(x) \triangleq e_z + z_g$ and $s_2(x) \triangleq e_\eta + e_d + k_d e_z + z_g$. Using the error systems in (3–3)-(3–5), the evader and pursuer states are represented as $z = s_1(x)$ and $\eta = s_2(x)$, respectively.

Following the problem formulation in [112], a composite autonomous error system can be written as

$$\dot{x} = F(x) + G(x)\mu, \tag{3–6}$$

where $\mu \triangleq \begin{bmatrix} \mu_\eta^\top & \mu_d^\top \end{bmatrix}^\top \in \mathbb{R}^m$ is the total vector of control policies with $m = m_\eta + n$, where $\mu_\eta : \mathbb{R}_{\geq t_0} \to \mathbb{R}^{m_\eta}$ is defined as $\mu_\eta \triangleq u - u_d$, $u_d : \mathbb{R}_{\geq t_0} \to \mathbb{R}^{m_\eta}$ denotes a desired input defined as $u_d \triangleq g^+(\eta_d)(\mu_d - h(z, \eta_d))$ where locally Lipschitz pseudo inverse $g^+ : \mathbb{R}^n \to \mathbb{R}^{m_\eta \times n}$ is defined as $g^+ \triangleq \left(g^\top g\right)^{-1} g^\top$, and $F : \mathbb{R}^{3n} \to \mathbb{R}^{3n}$ and $G : \mathbb{R}^{3n} \to \mathbb{R}^{3n\times m}$ are defined as

$$F(x) \triangleq \begin{bmatrix} f(s_1(x), s_2(x)) \\ -k_d f(s_1(x), s_2(x)) \\ h(s_1(x), s_2(x)) - F_{sd}(x) \end{bmatrix},$$

and

$$G(x) \triangleq \begin{bmatrix} 0_{n\times m_\eta} & 0_{n\times n} \\ 0_{n\times m_\eta} & I_n \\ g(s_2(x)) & G_{sd}(x) \end{bmatrix},$$

where $F_{sd}(x) \triangleq g(s_2(x))\, g^+(s_2(x_d))\, h(s_1(x), s_2(x_d))$, and $G_{sd}(x) \triangleq g(s_2(x))\, g^+(s_2(x_d)) - I_n$. The pursuer's objective is achieved if $\eta \to \eta_d$ and $z, \eta_d \to z_g$; hence, e_z, e_η, and $e_d \to 0$.

The goal is to formulate an optimal control problem to regulate the states based on a given cost function. To minimize the errors in (3–3)-(3–5), μ_d and μ_η are designed to minimize the cost function

$$J(x, \mu) \triangleq \int_{t_0}^{\infty} Q(x) + P(x) + \mu^\top R\mu \; d\tau, \tag{3–7}$$

where $Q : \mathbb{R}^{3n} \to \mathbb{R}_{\geq 0}$ is a user-defined PD function that satisfies $\underline{q}\|x\|^2 \leq Q(x) \leq \overline{q}\|x\|^2$ for all $x \in \mathbb{R}^{3n}$, where $\underline{q}, \overline{q} \in \mathbb{R}_{>0}$, $R \triangleq$ blkdiag $\{R_\eta, R_d\}$, $R_\eta \in \mathbb{R}^{m_\eta \times m_\eta}$ and $R_d \in \mathbb{R}^{n\times n}$ are user-defined PD symmetric cost matrices, and $P : \mathbb{R}^{3n} \to \mathbb{R}$ is a positive semi-definite (PSD) user-defined penalty function described in [112].

Following the standard actor-critic-based approximate optimal control framework (see [101], [135]) in [112], the optimal value function approximation $\widehat{V} : \mathbb{R}^{3n} \times \mathbb{R}^L \to \mathbb{R}$ is defined as

$$\widehat{V}\left(x, \widehat{W}_c\right) = \widehat{W}_c^\top \sigma(x), \tag{3–8}$$

where $\widehat{W}_c \in \mathbb{R}^L$ is the critic weight estimate, and $\sigma : \mathbb{R}^{3n} \to \mathbb{R}^L$ is a user-selected bounded vector of basis functions. The control objective is to determine an approximation of the optimal control policy $\widehat{\mu} : \mathbb{R}^{3n} \times \mathbb{R}^L \to \mathbb{R}^m$, defined as

$$\widehat{\mu}\left(x, \widehat{W}_a\right) = -\frac{1}{2} R^{-1} G(x)^\top \nabla\sigma(x)^\top \widehat{W}_a, \tag{3–9}$$

where $\widehat{W}_a \in \mathbb{R}^L$ is the actor weight estimate, to minimize the cost given in (3–7). Minimizing this cost ensures that the errors in (3–3)-(3–5) are regulated to zero.

3.2 System Identification

A challenge for the control objective is that the approximate optimal control formulation requires the dynamic model of the pursuer and the evader. Since the interaction dynamics and pursuer dynamics are unknown, an approximation of the composite dynamics $F(x)$ must be used to approximate the solution to the HJB equation. The interaction dynamics between the pursuer and evader in (3–1) must be estimated using data collected online and in real-time to achieve the control objective since interaction data will often be unavailable *a priori*. The result in [112] estimated $F(x)$ online using a single layer NN and CL; however, recent evidence has shown that DNNs can learn more complex features and improve function approximation performance [139]. The recent results in [113] and [140] demonstrated a novel method for estimating dynamics online using a multi-timescale Lb-DNN framework with CL for system identification and control. Building on the previous results, this section develops an advanced ICL-based multi-timescale Lb-DNN framework.

The ICL-based multi-timescale learning framework approximates functions online by pairing a Lb-ICL adaptive update law for the output-layer weights of a DNN with a concurrent to real-time iterative ICL batch update for the inner-layer features of the DNN. Specifically, data is collected online in batches and each batch iteratively updates the inner-layer features of the Lb-DNN concurrent to real-time control using integral history stack data in a user-defined loss function and an optimizer such as Adam [119]. Since the inner-layer features are updated concurrent to real-time, but not in real-time like the output-layer weights, the inner-layer features actively used by the controller are iteratively switched to the most recently updated inner-layer features after a batch update.

Motivated by improved function approximation, (3–1) and (3–2) can be stacked and represented as

$$\dot{\breve{x}} = \phi\left(\Phi\left(x\right)\right)\theta + \epsilon_{\theta}\left(x\right) + \breve{G}\left(x, u\right), \tag{3–10}$$

where the concatenated state derivative vector is defined as $\dot{\breve{x}} \triangleq \left[k_d \dot{z} \;\; \dot{\eta}\right]^\top \in \mathbb{R}^{2\times n}$, and $\breve{G}(x,u) \triangleq \begin{bmatrix} 0_{n\times 1} & g(x)\,u \end{bmatrix}^\top \in \mathbb{R}^{2\times n}$. To streamline the subsequent development, a stacked matrix representation is used rather than a stacked vector representation. The drift dynamics are approximated on a compact set $\mathcal{C} \subset \mathbb{R}^n$ with a DNN where $\theta \triangleq \begin{bmatrix} \theta_z^\top & \theta_\eta^\top \end{bmatrix}^\top \in \mathbb{R}^{p\times n}$ denotes an unknown bounded ideal output-layer weight matrix with the subscripts z and η representing the evader and pursuer dynamics, respectively, and $p = p_z + p_\eta$ is the total number of rows of θ. Additionally, $\phi(\Phi(x)) \triangleq \begin{bmatrix} \phi_z^\top(\Phi_z(x)) & 0_{1\times p_\eta} \\ 0_{1\times p_z} & \phi_\eta^\top(\Phi_\eta(x)) \end{bmatrix}$ where $\phi : \mathbb{R}^{2p} \to \mathbb{R}^{2\times p}$ denotes the user-defined basis functions and $\Phi(x) : \mathbb{R}^{3n} \to \mathbb{R}^{2p}$ denotes a function that represents the ideal DNN inner-layer features as $\Phi \triangleq \begin{bmatrix} \Phi_z^\top & \Phi_\eta^\top \end{bmatrix}^\top$, and $\epsilon_\theta(x) : \mathbb{R}^{3n} \to \mathbb{R}^{2\times n}$ denotes the function approximation errors. The i^{th} DNN-based estimate of the system dynamics is defined as

$$\hat{\dot{\breve{x}}}_i = \phi\left(\widehat{\Phi}_i(x)\right)\widehat{\theta} + \breve{G}(x,u), \tag{3–11}$$

where $\widehat{\theta} \in \mathbb{R}^{p\times n}$ is the output-layer ideal weight matrix θ estimate, and $\widehat{\Phi}_i : \mathbb{R}^{3n} \to \mathbb{R}^{2p}$ is the i^{th} iteration selection of the inner features consisting of estimated inner-layer weights and user-selected activation functions.

Assumption 3.3. There is a constant weight matrix θ and known positive constants $\overline{\theta}$, $\overline{\phi}$, $\overline{\nabla_x \phi}$, $\overline{\epsilon_\theta}$, and $\overline{\nabla_x \epsilon_\theta} \in \mathbb{R}_{\geq 0}$, such that $\|\theta\| \leq \overline{\theta}$, $\sup_{x\in\mathcal{C}} \|\phi(\cdot)\| \leq \overline{\phi}$, $\sup_{x\in\mathcal{C}} \|\nabla_x \phi(x)\| \leq \overline{\nabla_x \phi}$, $\sup_{x\in\mathcal{C}} \|\epsilon_\theta(x)\| \leq \overline{\epsilon_\theta}$, and $\sup_{x\in\mathcal{C}} \|\nabla_x \epsilon_\theta(x)\| \leq \overline{\nabla_x \epsilon_\theta}$ [141, Ch. 4].

Assumption 3.4. The inner-layer features selection of $\widehat{\Phi}_i$ ensures that $\Phi(x) - \widehat{\Phi}_i(x) \leq \tilde{\Phi}_i(x)$, where $\tilde{\Phi}_i : \mathbb{R}^{3n} \to \mathbb{R}^{2p}$ is the function approximation error of the i^{th} iteration inner-layer Lb-DNN features, 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, $\left\|\phi(\Phi(x)) - \phi\left(\widehat{\Phi}_i(x)\right)\right\| \leq \overline{\nabla_x \phi}\,\overline{\tilde{\Phi}}$ [113].

Unlike the result in [113], which uses CL to learn the unknown ideal weights of the DNN, this result uses an ICL-based weight update policy. Following the ICL strategy in [136], let $\Delta t_\theta \in \mathbb{R}_{>0}$ be the time window of integration, where the integral of (3–10)

at time $t_j \in [\Delta t_\theta, t]$ can be represented as $\Delta \breve{x}_j = \breve{x}(t_j) - \breve{x}(t_j - \Delta t_\theta) = \varphi_j \theta + \mathcal{E}_j + \mathcal{G}_j$ where $\varphi_j = \varphi\left(\widehat{\Phi}_i(t_j)\right) \triangleq \int_{t_j - \Delta t_\theta}^{t_j} \phi\left(\widehat{\Phi}_i(x(\tau))\right) d\tau$, $\mathcal{E}_j = \mathcal{E}(t_j) \triangleq \int_{t_j - \Delta t_\theta}^{t_j} \epsilon_\theta(x(\tau)) d\tau$, and $\mathcal{G}_j = \mathcal{G}(t_j) \triangleq \int_{t_j - \Delta t_\theta}^{t_j} \breve{G}(x(\tau), u(\tau)) d\tau$. An ICL-based parameter estimate update law is designed as

$$\dot{\widehat{\theta}}(t) = k_\theta \Gamma_\theta \sum_{j=1}^{M} \varphi_j^\top \left(\Delta \breve{x}_j - \mathcal{G}_j - \varphi_j \widehat{\theta}\right), \tag{3–12}$$

where $k_\theta, \Gamma_\theta \in \mathbb{R}_{>0}$ are update gains, and $M \in \mathbb{Z}_{>0}$ is the amount of data points saved for the history stack.

Remark 3.1. The i^{th} approximation of Φ is updated with the $i + 1\text{th}$ batch optimization using the ICL history stack data in the loss function

$$\mathcal{L}_{i+1} = \frac{1}{M} \sum_{j=1}^{M} \left\| \Delta \breve{x}_j - \mathcal{G}_j - \varphi_j \widehat{\theta} \right\|^2 \tag{3–13}$$

and using Adam for the offline training optimization method. The estimates $\widehat{\Phi}_i$ are not computed *a priori*. Concurrent to real-time learning, input-output data is saved in a history stack, and then $\widehat{\Phi}$ is recomputed during the batch update.

Assumption 3.5. There exists $T_1 \in \mathbb{R}_{>0}$ such that $T_1 > \Delta t_\theta$, and there exists a constant $\lambda_1 \in \mathbb{R}_{>0}$ that facilitates $\lambda_1 I_p \leq \sum_{j=1}^{M} \varphi_j^\top \varphi_j$, $\forall t \geq T_1$ [136].

3.3 Online Learning

3.3.1 Bellman Error

The optimal value function $V^* : \mathbb{R}^{3n} \to \mathbb{R}_{\geq 0}$ and optimal control policy $\mu^* : \mathbb{R}^{3n} \to \mathbb{R}^m$ satisfy the HJB equation

$$0 = \nabla V^*(x)(F + G\mu^*) + Q(x) + P(x) + \mu^{*\top} R \mu^*, \tag{3–14}$$

where $V^*(0) = 0$. While (3–14) represents the HJB equation under optimal conditions, substituting the approximate terms from (3–8), (3–9), and (3–11), yields the BE

$$\delta\left(x,\widehat{\theta},\widehat{W}_c,\widehat{W}_a\right)\triangleq Q\left(x\right)+P\left(x\right)+\hat{\mu}^{\top}R\hat{\mu}+\nabla\widehat{V}\left(x,\widehat{W}_c\right)\left(\widehat{F}_i\left(x,\widehat{\theta}\right)+G\left(x\right)\widehat{\mu}\left(x,\widehat{W}_a\right)\right), \tag{3–15}$$

where

$$\widehat{F}_i\left(x,\widehat{\theta}\right)\triangleq\left[\left(\frac{1}{k_d}\widehat{\theta}_z^{\top}\phi_z\left(\widehat{\Phi}_{z,i}\left(x\right)\right)\right)^{\top},\left(-\widehat{\theta}_z^{\top}\phi_z\left(\widehat{\Phi}_{z,i}\left(x\right)\right)\right)^{\top},\right.$$
$$\left.\left(\widehat{\theta}_\eta^{\top}\phi_\eta\left(\widehat{\Phi}_{\eta,i}\left(x\right)\right)\right)^{\top}-\left(g\left(x_\eta\right)g^{+}\left(\eta_d\right)\widehat{\theta}_\eta^{\top}\phi_\eta\left(\widehat{\Phi}_{\eta,i}\left(x_d\right)\right)\right)^{\top}\right]^{\top}$$

and $\widehat{\mu}\left(x,\widehat{W}_a\right)\triangleq\left[\widehat{\mu}_\eta^{\top}\left(x,\widehat{W}_a\right),\widehat{\mu}_d^{\top}\left(x,\widehat{W}_a\right)\right]^{\top}$ from (3–9). The pursuer controller is $\widehat{u}\left(x,\widehat{\theta},\widehat{W}_a\right)\triangleq\widehat{\mu}_\eta\left(x,\widehat{W}_a\right)+\widehat{u}_d\left(x,\widehat{\theta},\widehat{W}_a\right)$, where $\widehat{u}_d\left(x,\widehat{\theta},\widehat{W}_a\right)\triangleq g^{+}\left(\eta_d\right)\left(\widehat{\mu}_d\left(x,\widehat{W}_a\right)-\widehat{\theta}_\eta^{T}\phi_\eta\left(\widehat{\Phi}_{\eta,i}\left(x_d\right)\right)\right)$. To facilitate the subsequent stability analysis, the BE can also be expressed in terms of the error $\widetilde{W}_c\triangleq W-\widehat{W}_c$ and $\widetilde{W}_a\triangleq W-\widehat{W}_a$. Subtracting (3–15) from (3–14) and, substituting (3–8) and (3–9), the BE in (3–15) can be rewritten as

$$\delta=-\omega^{\top}\widetilde{W}_c+\frac{1}{4}\widetilde{W}_a^{\top}G_\sigma\widetilde{W}_a-W^{\top}\nabla\sigma\widetilde{F}_i+O \tag{3–16}$$

where $\omega\triangleq\nabla\sigma\left(\widehat{F}_i+G\widehat{\mu}\right)$, $\widetilde{F}_i\triangleq F-\widehat{F}_i$, $G_\sigma\triangleq\nabla\sigma G_R\nabla\sigma^{\top}$, $G_R\triangleq GR^{-1}G^{\top}$, and O is uniformly bounded over the compact set Ω.

As explained in [92], the user-selected state x_e can be used to evaluate the BE in (3–15) at off-trajectory points within the state space Ω. The extrapolated BEs are evaluated as $\delta_e\triangleq\delta\left(x_e,\widehat{\theta},\widehat{W}_c,\widehat{W}_a\right)$.

3.3.2 Actor and Critic Weight Update Laws

The on and off-trajectory BEs are used in the subsequently defined adaptive update laws to improve the actor and critic weight approximations online. The critic weight update law is defined as

$$\dot{\widehat{W}}_c\triangleq-\Gamma_c\Big(k_{c1}\frac{\omega}{\rho^2}\delta+\frac{k_{c2}}{N}\sum_{e=1}^{N}\frac{\omega_e}{\rho_e^2}\delta_e\Big), \tag{3–17}$$

and the least-squares gain matrix update law is defined as

$$\dot{\Gamma}_c \triangleq \beta_c \Gamma_c - \Gamma_c k_{c1} \frac{\omega \omega^\top}{\rho^2} \Gamma_c - \Gamma_c \frac{k_{c2}}{N} \sum_{e=1}^{N} \frac{\omega_e \omega_e^\top}{\rho_e^2} \Gamma_c, \tag{3–18}$$

where $\rho \triangleq 1 + \gamma_1 \omega^\top \omega$, $\rho_e \triangleq 1 + \gamma_1 \omega_e^\top \omega_e$, $\omega_e \triangleq \omega\left(\delta_e, \hat{\theta}, \widehat{W}_a\right)$, and $k_{c1}, k_{c2}, \gamma_1, \beta_c \in \mathbb{R}_{>0}$ are user-defined learning gains. The actor weight update law is defined as

$$\dot{\widehat{W}}_a \triangleq -K_a k_{a1}\left(\widehat{W}_a - \widehat{W}_c\right) + K_a \frac{k_{c1}}{4} G_\sigma^\top \widehat{W}_a \frac{\omega^\top}{\rho^2} \widehat{W}_c - K_a k_{a2} \widehat{W}_a + K_a \frac{k_{c2}}{4N} \sum_{e=1}^{N} G_{\sigma e}^\top \widehat{W}_a \frac{\omega_e^\top}{\rho_e^2} \widehat{W}_c, \tag{3–19}$$

where $k_{a1}, k_{a2} \in \mathbb{R}_{\geq 0}$ are user-defined learning gains, and $K_a \in \mathbb{R}^{L \times L}$ is a user-defined positive-definite symmetric matrix.

Assumption 3.6. There exist constants $T_2, \underline{c}_1, \underline{c}_2, \underline{c}_3 \in \mathbb{R}_{\geq 0}$ such that

$$\begin{aligned}
\underline{c}_1 I_L &\leq \inf_{t \in \mathbb{R}_{\geq t_0}} \frac{1}{N} \sum_{e=1}^{N} \frac{\omega_e \omega_e^\top}{\rho_e^2}, \\
\underline{c}_2 I_L &\leq \int_t^{t+T_2} \left(\frac{1}{N} \sum_{e=1}^{N} \frac{\omega_e(\tau)\, \omega_e^\top(\tau)}{\rho_e^2(\tau)} \right) d\tau, \ \forall t \in \mathbb{R}_{\geq t_0}, \\
\underline{c}_3 I_L &\leq \int_t^{t+T_2} \left(\frac{\omega(\tau)\, \omega^\top(\tau)}{\rho^2(\tau)} \right) d\tau, \ \forall t \in \mathbb{R}_{\geq t_0},
\end{aligned}$$

where T_2 and at least one of the constants $\underline{c}_1$, $\underline{c}_2$, or $\underline{c}_3$ is strictly positive [142].

Remark 3.2. See [112] for insight into Assumption (3.6).

3.4 Stability Analysis

Let $B_\zeta \subset \mathbb{R}^{3n+2L+np}$ represent a closed ball with a radius $\zeta \in \mathbb{R}_{>0}$ centered at the origin. Let $Z \in \mathbb{R}^{3n+2L+np}$ denote a concatenated state vector defined as $Z_L \triangleq \left[\begin{array}{cccc} x^\top, & \widetilde{W}_c^\top, & \widetilde{W}_a^\top, & Z_\theta^\top \end{array} \right]^\top$ where $Z_\theta = \text{vec}\left(\tilde{\theta}\right)$ and $\tilde{\theta} \triangleq \theta - \hat{\theta}$. Let $V_L : \mathbb{R}^{3n+2L+np} \times \mathbb{R}_{\geq t_0} \to \mathbb{R}$ denote a candidate Lyapunov function defined as

$$V_L(Z_L, t) \triangleq V^*(x, t) + \frac{1}{2} \widetilde{W}_c^\top \Gamma_c^{-1}(t)\, \widetilde{W}_c + \frac{1}{2} \widetilde{W}_a^\top K_a^{-1} \widetilde{W}_a + V_\theta(Z_\theta, t), \tag{3–20}$$

where $V_\theta\left(Z_\theta, t\right) \triangleq \frac{1}{2}\text{tr}\left(\tilde{\theta}^\top \Gamma_\theta^{-1}\left(t\right)\tilde{\theta}\right)$, that can be bounded by class $\mathcal{K}$ functions $\underline{v}_l, \overline{v}_l : \mathbb{R} \to \mathbb{R}_{\geq 0}$ as

$$\underline{v}_l\left(\|Z_L\|\right) \leq V_L\left(Z_L, t\right) \leq \overline{v}_l\left(\|Z_L\|\right) \tag{3–21}$$

for all $t \in \mathbb{R}_{\geq t_0}$ where $Z_L \in \mathbb{R}^{3n+2L+np}$. The sufficient conditions for ultimate boundedness of Z are derived based on the subsequent analysis as

$$k_d \geq 1, \qquad \lambda_{\min}\left\{H\right\} > 0, \qquad \sqrt{\frac{l}{\kappa}} \leq \underline{v}_l^{-1}\left(\overline{v}_l\left(\zeta\right)\right), \tag{3–22}$$

where

$$\kappa \triangleq \min\left\{\frac{1}{2}\underline{q}, \frac{1}{4}k_\theta \lambda_{\min}\left[\Sigma_\theta\right], \frac{1}{6}k_{c2}\underline{c}, \frac{1}{6}\left(k_{a1}+k_{a2}\right)\right\},$$

$$\underline{c} \triangleq \left(\frac{\beta_c}{2k_{c2}\overline{\Gamma}_c} + \frac{\underline{c}_1}{2}\right),$$

$$\varphi_a \triangleq \frac{\left(k_{c1}+k_{c2}\right)}{4}\overline{\|G_\sigma\|}\frac{k_\rho}{\sqrt{\gamma_1}}\|W\| + \frac{1}{2}\frac{1}{\lambda_{\min}\left\{K_a\right\}}\overline{\|\nabla W G_R \nabla\sigma^\top\|},$$

$$\Sigma_\theta \triangleq \left[\sum_{j=1}^{M}\varphi_j^\top \varphi_j\right],$$

$$\varphi_{ac} \triangleq k_{a1} + \frac{k_{c1}+k_{c2}}{4}\overline{\|G_\sigma\|\|W\|}\frac{k_\rho}{\sqrt{\gamma_1}} + \frac{1}{2}\frac{1}{\underline{\Gamma}_c}\overline{\|\nabla W\|\|G_R\|\|\nabla\sigma^\top\|},$$

$$\varphi_{c\theta} \triangleq \left(k_{c1}+k_{c2}\right)\frac{k_\rho}{\sqrt{\gamma_1}}\left(\overline{\|W^\top\|\|\nabla\sigma\|\|\phi\|}\left(1 + \frac{1}{k_d} + \overline{\|g\|\|g^+\|}\right)\right),$$

and $l \in \mathbb{R}_{>0}$ is a constant that depends on the bounded NN constants.

In contrast to the result in [112], the multi-timescale Lb-DNN identifier introduces piecewise-in-time discontinuities in the dynamics which complicates the stability analysis in the sense that common actor-critic methods cannot be readily applied in the stability analysis of the closed-loop system. The following theorem contains a Lyapunov-like

stability analysis which considers functions containing discontinuities that are piecewise continuous in time.

Theorem 3.1. *Provided all assumptions are satisfied, and conditions in (3–22) are met, then the error state x, the critic weight estimate error $\widetilde{W}_c$, the actor weight estimate error $\widetilde{W}_a$, and the parameter estimation error $\tilde{\theta}$ are UUB. Hence, the approximate control policy $\widehat{\mu}$ converges to a neighborhood of the optimal control policy μ^*.*

Proof. Taking the time derivative of (3–20), and substituting (3–14), $\dot{V}^*(x) = \nabla V^*(F(x)+G(x)\mu)$, $\dot{\widetilde{W}}_c \triangleq \dot{W} - \dot{\widehat{W}}_c$, $\dot{\widetilde{W}}_a \triangleq \dot{W} - \dot{\widehat{W}}_a$, and $\dot{W} \triangleq \nabla W(x)(F(x)+G(x)\mu)$ yields

$$\dot{V}_L = \nabla V^*(F+G\mu) + \dot{V}_\theta(Z_\theta) - \frac{1}{2}\widetilde{W}_c^\top\left(\Gamma_c^{-1}\dot{\Gamma}_c\Gamma_c^{-1}\right)\widetilde{W}_c + \widetilde{W}_c^\top\Gamma_c^{-1}\left(\nabla W(F+G\mu) - \dot{\widehat{W}}_c\right) + \widetilde{W}_a^\top K_a^{-1}\left(\nabla W(F+G\mu) - \dot{\widehat{W}}_a\right).$$

Using (3–15), the update laws in (3–12) and (3–17)-(3–19), $\widehat{W}_a = W - \widetilde{W}_a$, $\widehat{W}_c = W - \widetilde{W}_c$, Assumptions 3.3-3.6, and implementing bounding and completing the square yields $\dot{V}_L \leq -\kappa\|Z_L\|^2 - \kappa\|Z_L\|^2 + l - Z_v^\top H Z_v$, where $Z_v \triangleq \left[\left\|\widetilde{W}_a\right\|, \left\|\widetilde{W}_c\right\|, \|Z_\theta\|\right]^\top$. Specifically, Assumption 3.4 is used to bound the system identification parameter estimation term $\dot{V}_\theta$ in the Lyapunov function. Provided the sufficient conditions in (3–22) are met, then $\dot{V}_L$ can be bounded as

$$\dot{V}_L \leq -\kappa\|Z_L\|^2, \ \forall \|Z_L\| \geq \sqrt{\frac{l}{\kappa}} > 0. \tag{3–23}$$

As a result of the discontinuities in the update laws in (3–12) and (3–17)-(3–19) being piecewise continuous in time, and by using (3–22) and (3–23), [137, Theorem 4.18] can be enforced to conclude that Z_L is UUB such that $\|Z_L\| \leq \underline{v}^{-1}\left(\overline{v}\left(\sqrt{\frac{l}{\kappa}}\right)\right)$ and $\widehat{\mu}$ converges to a neighborhood around the optimal policy μ^*. Since $Z_L \in \mathcal{L}_\infty$, then x, $\widetilde{W}_c$, $\widetilde{W}_a$, $\tilde{\theta} \in \mathcal{L}_\infty$ and thus $\mu \in \mathcal{L}_\infty$. Moreover, since $x \in \mathcal{L}_\infty$, and since W is a continuous function of x, it follows that $W(x) \in \mathcal{L}_\infty$. Furthermore, since $x \in \mathcal{L}_\infty$, then

$e_\eta,\ e_z,\ e_d \in \mathcal{L}_\infty$. Using (3–3)-(3–5), $z \in \mathcal{L}_\infty$, and $\eta_d \in \mathcal{L}_\infty$; hence, $\eta,\ (z-\eta) \in \mathcal{L}_\infty$ follows. Lastly, since $\eta_d,\ \mu,\ g^+,\ \tilde{\theta} \in \mathcal{L}_\infty$, it follows that $\widehat{\theta},\ u_d \in \mathcal{L}_\infty$ and $u \in \mathcal{L}_\infty$. □

3.5 Simulations

An example scenario is simulated to illustrate the performance of the developed ICL-Lb-DNN ADP architecture where an evader and pursuer are uniformly randomly placed in a 1000×1000 unit area with the goal of position control ($n = 2$). The goal region is set to a uniformly random location within a 100 unit radius of the pursuer while the evader is uniformly randomly initialized at least 500 units from the goal region. The pursuer must therefore leave the goal area to catch the evader, learn the interaction dynamics in real-time using the deep ICL learning architecture, and approximate the optimal influencing policy using ADP. The typical performance of the architecture in simulation is shown in Figure 3-1 to indirectly control the position of the evader, where the pursuer is initially in the top-right (blue circle with white plus) near the goal (orange circle) and the evader is initially in the bottom-left (orange circle with white plus).

Without loss of generality, the dynamics for the pursuer were $h(z,\eta) = 0_{2\times 1}$ and $g(\eta) = I_{2\times 2}$. The evader dynamics were $f(z,\eta) = (z-\eta)\exp\left(-\frac{1}{20{,}000}(z-\eta)^\top (z-\eta)\right)$. In the simulation, the ICL-DNN function approximation was implemented using PyTorch, and all the history stack data was collected online in real-time (approximately 45 Hz). The ICL-Lb-DNN and the history stack remained on the graphics card for optimization using a maximum of approximately 1GB of memory. At each time step, the data was added to the history stack which was a sliding buffer containing the most recent second of data ($\Delta t_\theta = 1.0$ second). At each time step the integrals of the data were approximated using the trapezoidal rule to update the output weights using (3–12) and update the DNN inner-layer features using the loss discussed in Remark 3.1, where a single optimizer step was performed for each simulation step on a batch of integral data from the history stack using Adam with a linearly annealing learning rate (initialized to 0.001 and linearly decayed to 0.0001). To enable online optimization, the DNN was

constrained to 3 inner layers, each with 64 neurons and hyperbolic tangent activation functions while the final layer had 64 output weights. The output weights and inner-layer weights were randomly initialized using a zero mean and standard deviation of 0.01 ($\hat{\theta}_z(0) \sim \mathcal{N}(0, 0.01)$) with $\Gamma_\theta = 0.1$ and $k_\theta = 1.0$. The function approximation results in Figure 3-2 show that in the 35 second simulation, the ICL-DNN function approximation converges to within 10% of the true value of the nonlinear interaction dynamics while the loss converges to 0.04. Additionally, the SNN from [112], with 256 output weights, converges to within 50% of the true value of the dynamics demonstrating the DNN outperforms the SNN used in [112].

The efficient State following (StaF) kernels method from [142] was used to approximate the optimal policy online in real-time (approximately 45 Hz) while simultaneously estimating the dynamics using the ICL-DNN function approximation. The value function was approximated using 7 StaF kernels $\sigma(x, c(x)) = \begin{bmatrix} \sigma_1(x, c_1(x)) & \dots & \sigma_7(x, c_7(x)) \end{bmatrix}^\top$ where each kernel $\sigma_q(x(t), c_q(x(t))) = \frac{x^\top(t) c_q(x(t))}{\|x(0)\|^2}$, $c_q(x(t)) = x(t) + \|x(0)\| \nu(x(t)) d_q$, $\nu(x(t)) = \frac{x^\top(t)x(t) + 0.01\|x(0)\|^2}{\|x(0)\|^2 + x^\top(t)x(t)}$, and d_q are the vertices of a 6-simplex. The actor and critic weights were initialized as $\widehat{W}_a(0) = 1_7$ and $\widehat{W}_c(0) = 2\widehat{W}_a(0)$ while $\Gamma_c(0) = 5I_{7\times 7}$. The gains used to update the weights were selected as $k_{c1} = 0.9$, $k_{c2} = 0.1$, $K_a = 1.0$, $k_{a1} = 0.25$, $k_{a2} = 0.005$, $\beta_c = 0.001$, $\gamma_1 = 0.75$, and $N = 10$ extrapolation points were selected within a radius of $\nu(x)$ of x. The cost and control gains selected were $Q = 0.0001 I_{6\times 6}$, $R = 0.01$, and $k_d = 1.3$. Using the selected gains resulted in excellent tracking performance as shown by the tracking errors in Figure 3-3 where the tracking error $e_z \to 0$. These results demonstrate that the DNN-ICL-based ADP architecture is an excellent approach for real-time approximation of the optimal policy when dynamics are unknown and highly nonlinear.

3.6 Concluding Remarks

This chapter presents the first result where ICL is used to update a DNN, specifically an ICL-based adaptive update law is used to update output-layer weights in

real-time and integrated datasets in the loss function are used to batch update the inner-layer features concurrent to real-time. A deep ICL-based implementation of ADP is presented to achieve an approximate optimal online solution to the indirect regulation herding problem for unknown agents. An ICL-based system identifier is facilitated by an Lb-DNN to estimate the unknown interaction dynamic between the pursuer and evader. A Lyapunov-based analysis is provided to prove UUB convergence of the evader to the desired goal location known by the pursuer. The simulation shows that the pursuer is able to intercept and regulate the evader towards the desired goal location and that the DNN system identifier outperforms the SNN system identifier.

This chapter opens the door for the unknown interaction dynamics of multiple agents to be learned online via deep learning. Future work consists of extending this result to problems with multiple pursuers, multiple evaders, or both. Additional future ideas consist of extensions to problems with more complex dynamical agent interaction such as the evader optimally evading the pursuer or an asymmetric interaction between the pursuer and evader.

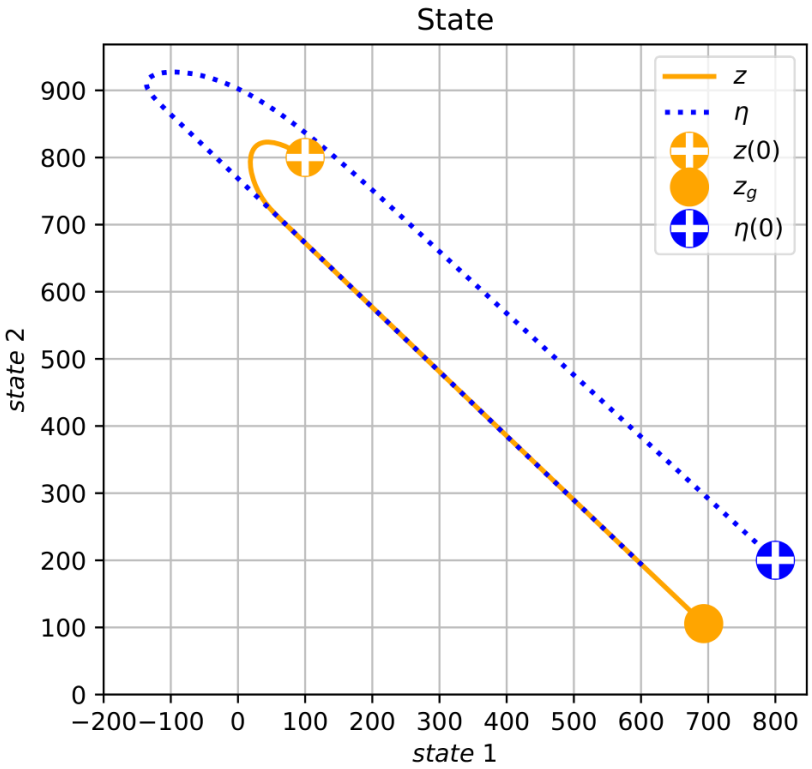

Figure 3-1. Simulation example where the pursuer is initialized in the bottom-right (blue circle with white plus), the goal region is to the left of the pursuer (orange circle), and the evader is initialized in the top-left (orange circle with white plus). The pursuer trajectory and evader trajectory over the experiment are shown in blue and orange, respectively. Simulation shows evader initially flees towards top-left; however, the pursuer approximates the interaction dynamics and optimal policy in real-time and quickly escorts the evader to the goal region.

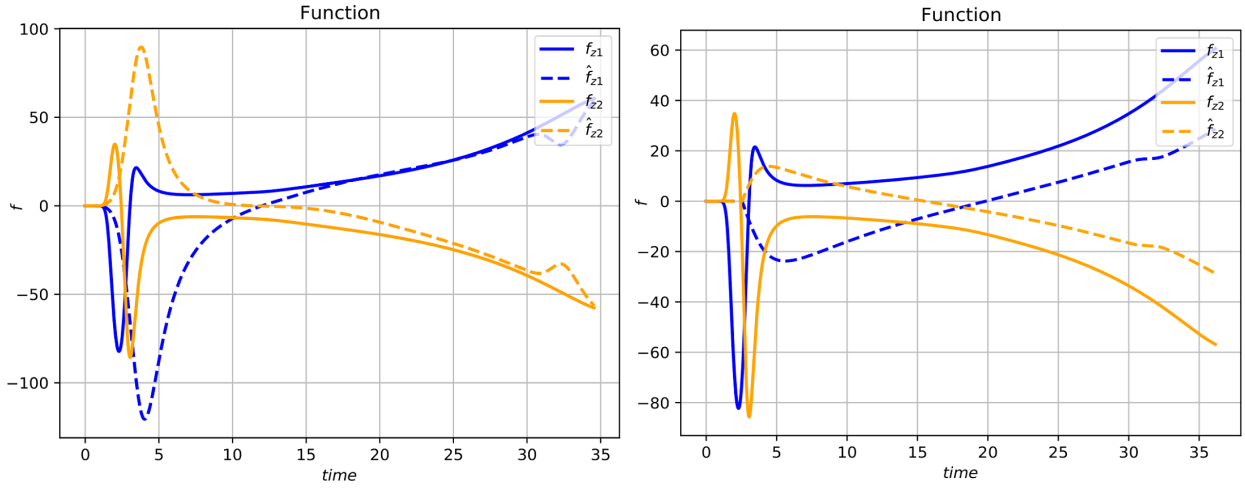

Figure 3-2. Function approximation where the true values are shown in solid lines and the estimated values are shown in dashed lines. The ICL-DNN estimates quickly converged near the true values using the data collected online. The left figure shows the ICL-DNN approximation and right figure shows the ICL-SNN approximation demonstrating that the ICL-DNN outperforms the ICL-SNN.

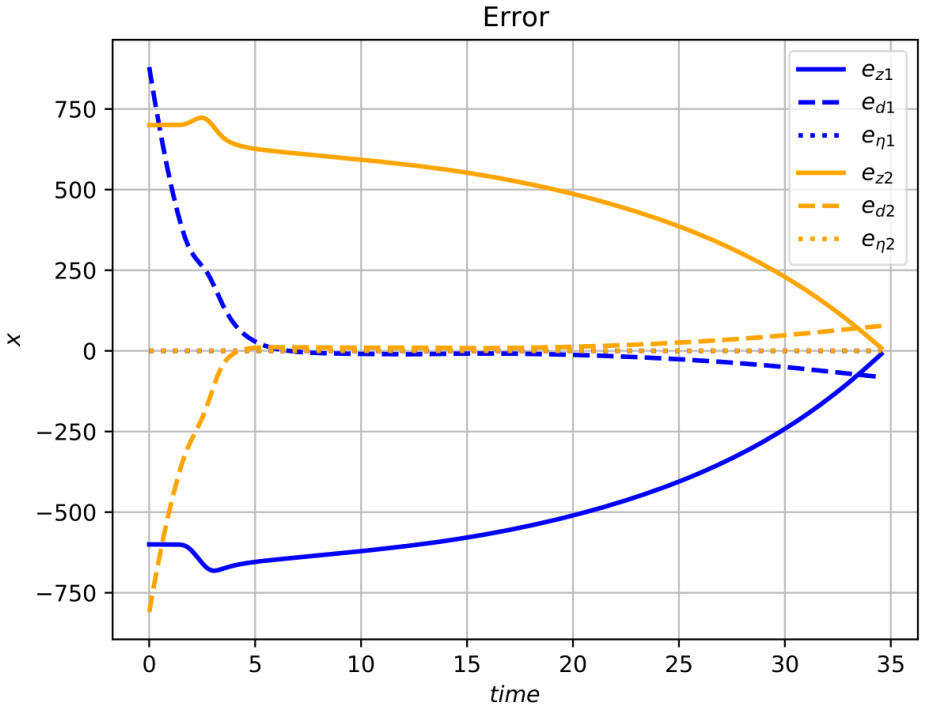

Figure 3-3. The evader tracking error steadily decays after the evader initially flees. The auxiliary errors also converge to a small radius of the goal.

CHAPTER 4
LYAPUNOV-BASED DEEP REINFORCEMENT LEARNING FOR APPROXIMATE OPTIMAL CONTROL

Chapter 3 used a DNN for system identification and an SNN for value function approximation. Building on the results in Chapter 3, in this chapter a DNN is used for value function approximation for the first time. Both the unknown drift dynamics and the unknown optimal value function are approximated online via Lb-DNNs. This chapter approximates an online solution to the infinite-horizon optimal tracking problem for control-affine continuous-time nonlinear systems. Similar to Chapter 3, the Lb-DNNs operate on a multi-timescale where the output-layer weights are updated online in real-time and the inner-layer features are updated concurrent to online execution via batch updates. In this chapter the output-layer weight update laws are formulated motivated by a Lyapunov-based stability analysis, and the inner-layer features use optimization algorithms for batch updates. The approximation of the optimal control policy is proven to converge to a neighborhood of the optimal control policy, and UUB stability of the states is proven via a Lyapunov-based stability analysis. The comparative simulation results show that the deep value function approximation method results in 95.04% improvement in BE minimization and 5.06% faster convergence.

4.1 Background Information

Consider the class of nonlinear control-affine systems introduced in (2–1) and Assumptions 2.1 and 2.2. The tracking error is defined as $e \triangleq x - x_d$ where $x_d \in \mathbb{R}^n$ denotes a time-varying continuously differentiable desired state trajectory. The following assumptions facilitate the development of the approximately optimal tracking controller [93].

Assumption 4.1. The desired trajectory is bounded from above by an unknown positive constant $\overline{x_d} \in \mathbb{R}$ such that $\sup_{t \in \mathbb{R}_{\geq 0}} \|x_d\| \leq \overline{x_d}$.

Assumption 4.2. There exists a locally Lipschitz function $h_d : \mathbb{R}^n \to \mathbb{R}^n$, such that $h_d(x_d) \triangleq \dot{x}_d$ and $g^+(x_d) g(x_d) (h_d(x_d) - f(x_d)) = h_d(x_d) - f(x_d) \; \forall t \in \mathbb{R}_{\geq 0}$,

where $g^{+}:\mathbb{R}^{n}\rightarrow\mathbb{R}^{m\times n}$ is defined as $g^{+}(x)\triangleq\left(g^{\top}(x)\,g(x)\right)^{-1}g^{\top}(x)$. It follows that $\sup_{t\in\mathbb{R}_{\geq 0}}\left\|g^{+}(x_{d})\right\|\leq\overline{g_{d}^{+}}$.

Remark 4.1. Assumption 4.1 and Assumption 4.2 are classical assumptions used to transform a time-varying tracking problem to a time-invariant optimal control problem. The user selection of x_d and h_d can satisfy Assumptions 4.1 and 4.2. [93]

Leveraging the technique in [93] to transform the time-varying tracking problem into a time-invariant infinite horizon regulation problem, the nonlinear control-affine dynamics are rewritten as

$$\dot{\zeta}=F(\zeta)+G(\zeta)\,\mu, \tag{4–1}$$

where $\zeta\triangleq\left[e^{\top},x_{d}^{\top}\right]^{\top}$ is a concatenated state vector, $F:\mathbb{R}^{2n}\rightarrow\mathbb{R}^{2n}$ is defined as

$$F(\zeta)\triangleq\begin{bmatrix}f(e+x_{d})-h_{d}(x_{d})+g(e+x_{d})\,u_{d}(x_{d})\\ h_{d}(x_{d})\end{bmatrix}, \tag{4–2}$$

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

$$G(\zeta)\triangleq\left[g(e+x_{d})^{\top},\mathbf{0}_{m\times n}\right]^{\top}, \tag{4–3}$$

$\mu\triangleq u-u_{d}(x_{d})$ is the transient component of the controller, and $u_{d}:\mathbb{R}^{n}\rightarrow\mathbb{R}^{m}$ is the trajectory tracking component of the controller defined as

$$u_{d}(x_{d})\triangleq g^{+}(x_{d})\left(h_{d}(x_{d})-f(x_{d})\right).$$

From Assumption 4.2, it follows that $\sup_{t\in\mathbb{R}_{\geq 0}}\left\|g^{+}(x_{d})\right\|\leq\overline{g_{d}^{+}}$. Since the drift dynamics f are unknown, u_d cannot be calculated and thus must be approximated. The approximation of the trajectory tracking component $\widehat{u}_{d}$ is subsequently defined in Section 4.4. The action space for μ is $U\subset\mathbb{R}^{m}$.

4.2 Deep System Identification

This section uses the multi-timescale deep system identification formulation in Section 3.2 to approximate the drift dynamics with a DNN as

$$f(x) = \theta^{\top}\phi(\Phi(x)) + \epsilon_{\theta}(x), \tag{4–4}$$

and the i^{th} DNN estimate of the drift dynamics as

$$\hat{f}_i\left(x, \hat{\theta}\right) = \hat{\theta}^{\top}\phi\left(\widehat{\Phi}_i(x)\right). \tag{4–5}$$

Assumptions 3.3 and 3.4 contain the bounds on the NN parameters. Unlike Chapter 3, in this chapter CL is used to update the output-layer weights and inner-layer features.

Assumption 4.3. A history stack consisting of input-output data pairs $\{\dot{x}_j, x_j, u_j\}_{j=1}^{M}$ is collected online where $\dot{x}_j \triangleq f(x_j) + g(x_j)u_j$. In some cases the system identification history stack may be available a priori, but it is not necessary [93]. There exists a constant $\underline{\lambda}$ such that the history stack satisfies the inequality $\lambda_{\min}\left(\sum_{j=1}^{M}\phi\left(\widehat{\Phi}_i(x_j)\right)\phi\left(\widehat{\Phi}_i(x_j)\right)^{\top}\right) > \underline{\lambda}$.

Remark 4.2. The selected data must be sufficiently exciting to satisfy the inequality in Assumption 4.3.

Input-output data stored in the CL history stack can be collected online and can simultaneously update the output-layer weights and inner-layer features of the system identification DNN on a multi-timescale. Specifically, the CL history stacks consisting of $\{\dot{x}(t), x(t), u(t)\}$ can update $\hat{\theta}$ in real-time and update $\widehat{\Phi}_i(x)$ from i to $i+1$ iteratively.

4.2.1 Output-Layer Weight Updates

The output-layer DNN weights $\hat{\theta}$ are updated according to the CL-based update law

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

where $\Gamma_{\theta} \in \mathbb{R}^{p\times p}$ is a user-selected constant.

4.2.2 Inner-Layer Feature Updates

Motivated by the subsequent analysis, the output-layer weights of the DNNs are updated online in real-time using the aforementioned adaptive update law in (4–6); however, updating the inner-layer features may improve function approximation. Therefore, using the CL history stack, the inner-layer features are updated concurrent to task execution (but slower than real-time) using an optimization technique to minimize the mean squared error based on the saved data $\{\dot{x}(t), x(t), u(t)\}$. The $i+1^{\text{th}}$ batch optimization is used to update the i^{th} approximation of Φ through the loss function

$$\mathcal{L}_{i+1} = \frac{1}{M}\sum_{j=1}^{M}\left\|\dot{x}_j - g_j(x_j)\,u_j - \hat{\theta}^{\top}\phi\left(\widehat{\Phi}_i(x_j)\right)\right\|^2.$$

4.3 Control Objective

The control objective can be satisfied by solving the infinite-horizon optimal tracking problem, i.e. finding a control policy μ that minimizes the cost functional

$$J(\zeta,\mu) = \int_0^{\infty} \overline{Q}(\zeta(\tau)) + \mu(\tau)^{\top} R\mu(\tau)\; d\tau, \tag{4–7}$$

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

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

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

$$0 = \nabla_{\zeta} V^*(\zeta)\left(F(\zeta) + G(\zeta)\mu^*\right) + \overline{Q}(\zeta) + \mu^*(\zeta)^{\top} R\mu^*(\zeta), \tag{4–8}$$

the solution to which is the analytical optimal control policy

$$\mu^{*}\left(\zeta\right)=-\frac{1}{2}R^{-1}G\left(\zeta\right)^{\top}\left(\nabla_{\zeta}V^{*}\left(\zeta\right)\right)^{\top}.$$

4.4 Deep Value Function Approximation

Simultaneous to DNN-based system identification, the unknown optimal value function is approximated with an additional Lb-DNN. Motivated to improve value function approximation, this section presents the problem formulation for applying the multi-timescale Lb-DNN method to approximate the optimal value function. Leveraging results based on the universal function approximation property of NNs, let $\zeta \in \Omega \subset \mathbb{R}^{2n}$ be a compact domain of the state space. Theorem 4.1 proves that if ζ is initialized within a compact set, then it will remain in a compact set. Refer to [101, Algorithm A.2] for discussion on establishing the size of compact sets. The optimal value function V^{*} can be approximated on Ω with a DNN as

$$V^{*}\left(\zeta\right)=W^{\top}\psi\left(\Psi\left(\zeta\right)\right)+\epsilon_{v}\left(\zeta\right), \tag{4–9}$$

where $W \in \mathbb{R}^{L}$ is a vector of ideal constant output-layer weights, $\psi : \mathbb{R}^{h} \to \mathbb{R}^{L}$ is a vector of user-selected activation functions, $\Psi : \mathbb{R}^{2n} \to \mathbb{R}^{h}$ is a function that represents the ideal DNN inner-layer features, and $\epsilon_{v} : \mathbb{R}^{2n} \to \mathbb{R}$ is the function approximation error. Based on the approximation in (4–9), the optimal control policy $\mu^{*} : \mathbb{R}^{2n} \to \mathbb{R}^{m}$ is

$$\mu^{*}\left(\zeta\right)=-\frac{1}{2}R^{-1}G\left(\zeta\right)^{\top}\cdot\left(W^{\top}\nabla_{\zeta}\psi\left(\Psi\left(\zeta\right)\right)+\nabla_{\zeta}\epsilon_{v}\left(\zeta\right)\right)^{\top}. \tag{4–10}$$

If $V^{*}\left(\zeta\right)$ is known, then it can be substituted into (4–10) to obtain an optimal control policy; however, the ideal output-layer weights W and ideal inner-layer features Ψ are unknown *a priori*. Leveraging an actor-critic-based framework, the optimal value function in (4–9) can be approximated with an estimate of the ideal weights $\widehat{W}_{c} \in \mathbb{R}^{L}$. The

optimal value function approximation $\widehat{V} : \mathbb{R}^{2n} \times \mathbb{R}^L \to \mathbb{R}$ can then be defined as

$$\widehat{V}\left(\zeta, \widehat{W}_c\right) \triangleq \widehat{W}_c^{\top} \psi\left(\widehat{\Psi}_k\left(\zeta\right)\right), \tag{4–11}$$

where $\widehat{\Psi}_k : \mathbb{R}^{2n} \to \mathbb{R}^h$ is an approximation of the ideal DNN inner-layer features Ψ. The subscript $k \in \mathbb{N}_{\geq 0}$ refers to the index of the inner-layer features, which increases when the new set of inner-layer features are implemented (i.e., when the DNN training is complete). Furthermore, the optimal control policy in (4–10) can be approximated with an estimate of the ideal weights $\widehat{W}_a \in \mathbb{R}^L$, resulting in the optimal control policy approximation $\hat{\mu} : \mathbb{R}^{2n} \times \mathbb{R}^L \to \mathbb{R}^m$ defined as

$$\hat{\mu}\left(\zeta, \widehat{W}_a\right) \triangleq -\frac{1}{2} R^{-1} G\left(\zeta\right)^{\top} \cdot \left(\widehat{W}_a^{\top} \nabla_{\zeta} \psi\left(\widehat{\Psi}_k\left(\zeta\right)\right)\right)^{\top}. \tag{4–12}$$

Assumption 4.4. There exist constant weights W and positive constants $\overline{W}$, $\overline{\psi}$, $\overline{\nabla_{\zeta}\psi}$, $\overline{\epsilon_v}$, and $\overline{\nabla_{\zeta}\epsilon_v} \in \mathbb{R}_{\geq 0}$, such that $\|W\| \leq \overline{W}$, $\sup_{\zeta \in \Omega} \|\psi\left(\cdot\right)\| \leq \overline{\psi}$, $\sup_{\zeta \in \Omega} \|\nabla_{\zeta}\psi\left(\cdot\right)\| \leq \overline{\nabla_{\zeta}\psi}$, $\sup_{\zeta \in \Omega} \|\epsilon_v\left(\zeta\right)\| \leq \overline{\epsilon_v}$, and $\sup_{\zeta \in \Omega} \|\nabla_{\zeta}\epsilon_v\left(\zeta\right)\| \leq \overline{\nabla_{\zeta}\epsilon_v}$ [141, Ch. 4].

Assumption 4.5. The k^{th} user-selected inner-layer features $\hat{\Psi}_k$ are selected such that $\psi\left(\hat{\Psi}_k\left(\zeta\right)\right) + \epsilon_{v,k}\left(\zeta\right) = \psi\left(\Psi\left(\zeta\right)\right)$, where $\epsilon_{v,k} : \mathbb{R}^{2n} \to \mathbb{R}^L$ is an approximation error of the ideal and k^{th}user-selected inner-layer features, and $\nabla_{\zeta}\psi\left(\hat{\Psi}_k\left(\zeta\right)\right) + \nabla_{\zeta}\epsilon_{v,k}\left(\zeta\right) = \nabla_{\zeta}\psi\left(\Psi\left(\zeta\right)\right)$. Furthermore, $\sup_{\zeta \in \Omega,\, k \in \mathbb{N}} \|\epsilon_{v,k}\left(\zeta\right)\| \leq \overline{\epsilon_v}$ and $\sup_{\zeta \in \otimes,\, k \in \mathbb{N}} \|\nabla_{\zeta}\epsilon_{v,k}\left(\zeta\right)\| \leq \overline{\nabla_{\zeta}\epsilon_v}$, where $\overline{\epsilon_v}, \overline{\nabla_{\zeta}\epsilon_v} \in \mathbb{R}_{\geq 0}$ are constants for all k iterations.

Recall, the trajectory tracking component of the controller u_d is not known *a priori* due to unknown drift dynamics. An approximation of the trajectory tracking component $\widehat{u}_d : \mathbb{R}^n \times \mathbb{R}^{p \times n} \to \mathbb{R}^m$ is defined as $\widehat{u}_d\left(x_d, \hat{\theta}\right) \triangleq g^{+}\left(x_d\right)\left(h_d\left(x_d\right) - \hat{f}_i\left(x_d, \hat{\theta}\right)\right)$; hence, the control policy u applied to the dynamical system $\dot{x} = f\left(x\right) + g\left(x\right)u$ is [93]

$$u \triangleq \hat{\mu}\left(\zeta, \widehat{W}_a\right) + \widehat{u}_d\left(x_d, \hat{\theta}\right). \tag{4–13}$$

4.5 Bellman Error

The left-hand side of the HJB in (4–8) is equal to zero under optimal conditions. However, substituting (4–11), (4–12), and the approximated drift dynamics $\hat{f}_i\left(x,\hat{\theta}\right)$ into (4–8) yields a residual term $\hat{\delta}:\mathbb{R}^{2n}\times\mathbb{R}^{p\times n}\times\mathbb{R}^{L}\times\mathbb{R}^{L}\to\mathbb{R}$ known as the BE. The BE is defined as

$$\hat{\delta}\left(\zeta,\hat{\theta},\widehat{W}_c,\widehat{W}_a\right)\triangleq\hat{\mu}\left(\zeta,\widehat{W}_a\right)^{\top}R\hat{\mu}\left(\zeta,\widehat{W}_a\right)+\overline{Q}\left(\zeta\right)+\nabla_{\zeta}\widehat{V}\left(\zeta,\widehat{W}_c\right)\left(\widehat{F}_i\left(\zeta,\hat{\theta}\right)+G\left(\zeta\right)\hat{\mu}\left(\zeta,\widehat{W}_a\right)\right),\quad(4\text{–}14)$$

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

$$\widehat{F}_i\left(\zeta,\hat{\theta}\right)\triangleq\left[\hat{f}_i\left(e+x_d,\hat{\theta}\right)^{\top}-h_d\left(x_d\right)^{\top}+u_d\left(x_d,\hat{\theta}\right)^{\top}g\left(e+x_d\right)^{\top},h_d\left(x_d\right)^{\top}\right]^{\top}.\quad(4\text{–}15)$$

The BE in (4–14) is an indirect measure of the suboptimality of the value function approximation which defines the error between the weight estimates and the unknown ideal values as $\widetilde{W}_c\triangleq W-\widehat{W}_c$ and $\widetilde{W}_a\triangleq W-\widehat{W}_a$. Substituting (4–9) and (4–10) into (4–8) and subtracting the BE in (4–14) yields an analytical form of the BE, given as

$$\hat{\delta}=-\hat{\omega}^{T}\widetilde{W}_c+\frac{1}{4}\widetilde{W}_a^{T}G_{\hat{\psi}}\widetilde{W}_a^{T}-W^{T}\left(\nabla_{\zeta}\psi\left(\widehat{\Psi}_k\left(\zeta\right)\right)\right)\left(F-\widehat{F}_i\right)+O\left(\zeta\right),\quad(4\text{–}16)$$

where

$$\hat{\omega}\left(\zeta,\widehat{W}_a,\hat{\theta}\right)\triangleq\nabla_{\zeta}\psi\left(\widehat{\Psi}_k\left(\zeta\right)\right)\left(\widehat{F}_i\left(\zeta,\hat{\theta}\right)+G\left(\zeta\right)\hat{\mu}\left(\zeta,\widehat{W}_a\right)\right),$$

$$O\left(\zeta\right)\triangleq\frac{1}{2}W^{\top}\nabla_{\zeta}\psi\left(\Psi\left(\zeta\right)\right)G_R\nabla_{\zeta}\epsilon_v\left(\zeta\right)^{\top}+\frac{1}{4}G_{\epsilon}-W^{\top}\nabla_{\zeta}\epsilon_v\left(\zeta\right)F\left(\zeta\right)+\frac{1}{4}W^{\top}G_{\psi}W-\frac{1}{4}W^{\top}G_{\hat{\psi}}W-\nabla_{\zeta}\epsilon_v\left(\zeta\right)F\left(\zeta\right),$$

$$G_R\left(\zeta\right)\triangleq G\left(\zeta\right)R^{-1}G\left(\zeta\right)^{\top},$$

$$G_{\psi}\left(\zeta\right)\triangleq\nabla_{\zeta}\psi\left(\Psi\left(\zeta\right)\right)G_R\left(\zeta\right)\nabla_{\zeta}\psi\left(\Psi\left(\zeta\right)\right)^{\top},$$

$$G_{\hat{\psi}}(\zeta) \triangleq \nabla_{\zeta}\psi\left(\widehat{\Psi}_k(\zeta)\right) G_R(\zeta) \nabla_{\zeta}\psi\left(\widehat{\Psi}_k(\zeta)\right)^{\top},$$

and

$$G_{\epsilon}(\zeta) \triangleq \nabla_{\zeta}\epsilon_v(\zeta) G(\zeta) \nabla_{\zeta}\epsilon_v(\zeta)^{\top}.$$

4.5.1 Bellman Error Extrapolation

The BE is extrapolated from the user-defined, off-policy trajectories $\{\zeta_e : \zeta_e \in \Omega\}_{e=1}^{N}$ set by the user, where $N \in \mathbb{N}$ denotes a user-specified number of extrapolated trajectories in the compact set Ω. The extrapolated trajectories $\zeta_e \in \Omega$ are used to evaluate the BE in (4–14) such that $\hat{\delta}_e \triangleq \hat{\delta}\left(\zeta_e, \hat{\theta}, \widehat{W}_c, \widehat{W}_a\right)$. The data stacks corresponding to Ω are represented as $(\Sigma_c, \Sigma_a, \Sigma_{\Gamma})$ such that $\Sigma_c \triangleq \frac{1}{N}\sum_{e=1}^{N} \frac{\omega_e}{\rho_e}\hat{\delta}_e$, $\Sigma_a \triangleq \frac{1}{N}\sum_{e=1}^{N} \frac{G_{\hat{\psi}e}^{\top}\widehat{W}_a\omega_e^{\top}}{4\rho_e}$, and $\Sigma_{\Gamma} \triangleq \frac{1}{N}\sum_{e=1}^{N} \frac{\omega_e\omega_e^{\top}}{\rho_e^2}$, where $\omega_e \triangleq \omega\left(\zeta_e, \hat{\theta}, \widehat{W}_a\right)$, and $\rho_e = \rho\left(\zeta_e, \hat{\theta}, \widehat{W}_a\right) = 1 + \nu\omega_e^{\top}\Gamma\omega_e$ containing $\nu \in \mathbb{R}_{>0}$ as a user-defined gain and $\Gamma : \mathbb{R}^{L\times L}$ as a user-defined learning gain. The normalized regressors $\frac{\omega}{\rho}$ and $\frac{\omega_e}{\rho_e}$ can be bounded as $\sup_{t\in\mathbb{R}_{\geq 0}}\left\|\frac{\omega}{\rho}\right\| \leq \frac{1}{2\sqrt{\nu\underline{\Gamma}}}$ and $\sup_{t\in\mathbb{R}_{\geq 0}}\left\|\frac{\omega_e}{\rho_e}\right\| \leq \frac{1}{2\sqrt{\nu\underline{\Gamma}}}$ for all $\zeta \in \Omega$ and $\zeta_e \in \Omega$, respectively. From Assumption **??** it follows that $0 < \|G(\zeta)\| \leq \overline{G}$, where $\overline{G} \in \mathbb{R}_{>0}$. The matrices G_R and $G_{\hat{\psi}}$ can be bounded as $\sup_{\zeta\in\Omega}\|G_R\| \leq \lambda_{\max}\{R^{-1}\}\overline{G}^2 \triangleq \overline{G_R}$ and $\sup_{\zeta\in\Omega_k}\left\|G_{\hat{\psi}}\right\| \leq \left(\overline{\nabla_{\zeta}\psi G}\right)^2 \lambda_{\max}\{R^{-1}\} \triangleq \overline{G_{\hat{\psi}}}$, respectively, where $\lambda_{\max}\{\cdot\}$ denotes the maximum eigenvalue. The expression for the BE in (4–14) and (4–16) are equivalent, but (4–14) is used in implementation, and (4–16) is used in the subsequent Lyapunov-based stability analysis.

Assumption 4.6. On the compact set Ω, there exists a finite set of off-policy 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 $t \in \mathbb{R}_{\geq 0}$, where $\underline{c}$ is a constant scalar lower bound of the value of each input-output data pairs minimum eigenvalues [92].

Remark 4.3. Assumption 4.6 can be verified online and heuristically satisfied by selecting a greater amount BE extrapolation points than amount of neurons in ψ such that $N \gg L$ [92].

4.6 DNN Value Function Update Laws

Similar to the CL history stack used to update the system identifier, BE extrapolation is used to update the value function approximation by treating the BE as the cost to be minimized. The data stacks consisting of $\{\Sigma_c, \Sigma_a, \Sigma_\Gamma\}$ are collected online and simultaneously update the output-layer weights and inner-layer features of the DNN. Specifically the BE extrapolation data stacks can update $\widehat{W}_c$ and $\widehat{W}_a$ in real-time and update $\widehat{\Psi}_k(\zeta)$ from k to $k+1$ iteratively.

4.6.1 Actor-Critic Output-Layer Weight Updates

The actor and critic weights are updated using the instantaneous BE $\hat{\delta}$ and extrapolated BEs $\hat{\delta}_e$. In the subsequent update laws, $\eta_{c1}, \eta_{c2}, \eta_{a1}, \eta_{a2}, \lambda \in \mathbb{R}$ are positive constant learning gains.

The critic weight update policy $\dot{\widehat{W}}_c \in \mathbb{R}$ is defined as

$$\dot{\widehat{W}}_c \triangleq -\eta_{c1}\Gamma\frac{\hat{\omega}}{\rho}\hat{\delta} - \eta_{c2}\Gamma\Sigma_c, \tag{4–17}$$

and the actor weight update policy $\dot{\widehat{W}}_a$ is defined as

$$\dot{\widehat{W}}_a \triangleq -\eta_{a1}\left(\widehat{W}_a - \widehat{W}_c\right) - \eta_{a2}\widehat{W}_a + \eta_{c1}\frac{G_{\hat{\psi}}^\top \widehat{W}_a \hat{\omega}^\top}{4\rho}\widehat{W}_c + \eta_{c2}\Sigma_a\widehat{W}_c. \tag{4–18}$$

The least-squares gain update policy $\dot{\Gamma} : \mathbb{R}^{L\times L}$ is defined as

$$\dot{\Gamma} \triangleq \left(\lambda\Gamma - \eta_{c1}\frac{\Gamma\hat{\omega}\hat{\omega}^\top\Gamma}{\rho^2} - \eta_{c2}\Gamma\Sigma_\Gamma\Gamma\right)\mathbf{1}_{\left\{\underline{\Gamma}\leq\|\Gamma\|\leq\overline{\Gamma}\right\}}, \tag{4–19}$$

where $\mathbf{1}_{\{\cdot\}}$ denotes the indicator function used to ensure that $\overline{\Gamma} \leq \|\Gamma\| \leq \underline{\Gamma}$ for all $t \in \mathbb{R}_{>0}$.

4.6.2 Inner-Layer Feature Updates

Offline function approximation methods are used to update the inner-layer DNN weight estimates simultaneous to the online execution using a random set of $N_\delta \in \mathbb{Z}$ states from the saved history of states. The inner-layer features are updated periodically using an optimization technique to minimize the normalized mean squared error of the BE. The $k+1^{\text{th}}$ batch optimization is used to update the k^{th} approximation of $\Psi(\zeta)$

through the loss function

$$\mathcal{L}_{\delta,k+1} = \frac{1}{N_\delta} \sum_{e=1}^{N_\delta} \frac{\hat{\delta}_e^2}{\rho_e} \left(1 - w_e\left(\pi_e, \hat{\mu}_e\right)\right) + \left(\pi_e - \hat{\mu}_e\right)^2 w_e\left(\pi_e, \hat{\mu}_e\right), \tag{4–20}$$

where $\pi_e \in \mathbb{R}^m$ is any exploration policy (e.g. StaF policy [143]) and $w_e : \mathbb{R}^m \times \mathbb{R}^m \to [0,1]$ is the dynamic confidence weighting assigned to the exploration policy that considers the confidence of the current policy compared to the confidence of the exploration policy. The use of an exploration policy is to accelerate the training of the value function DNN. A general exploration policy is used, and a specific example is provided in [108] and used as an example application in Section 4.8.

4.7 Stability Analysis

Because the optimal value function V^* is PSD, the optimal value function is not a valid Lyapunov function. However, the optimal value function can be represented in a nonautonomous form, denoted as $V_{na}^* : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ and defined as $V_{na}^*(e,t) \triangleq V^*(\zeta)$, that is PD and decrescent where $V_{na}^*(0,t) = 0$ [88]. Class $\mathcal{K}_\infty$ functions $\underline{v}, \overline{v} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ exist that bound $\underline{v}(\|e\|) \leq V_{na}^*(e,t) \leq \overline{v}(\|e\|)\ \forall e \in \mathbb{R}^n, t \in \mathbb{R}_{\geq 0}$, therefore ensuring $V_{na}^*(e,t)$ is a valid Lyapunov function. Let $Z \in \mathbb{R}^{n+2L+pn}$ be a concatenated state defined as $Z \triangleq \left[e^\top, \widetilde{W}_c^\top, \widetilde{W}_a^\top, \operatorname{vec}\left(\tilde{\theta}\right)^\top\right]^\top$ where $\tilde{\theta} \triangleq \theta - \hat{\theta}$. Let $V_L : \mathbb{R}^{n+2L+pn} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ be a candidate Lyapunov function defined as

$$V_L(Z,t) \triangleq V_{na}^*(e,t) + \frac{1}{2}\widetilde{W}_c^\top \Gamma(t)^{-1} \widetilde{W}_c + \frac{1}{2}\widetilde{W}_a^\top \widetilde{W}_a + \frac{1}{2}\text{tr}\left(\tilde{\theta}^\top \Gamma_\theta^{-1} \tilde{\theta}\right). \tag{4–21}$$

According to [137, Lemma 4.3], and the properties of $V_{na}^*(e,t)$, there exist class $\mathcal{K}_\infty$ functions $\alpha_1, \alpha_2 : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that bound (4–21) as $\alpha_1(\|Z\|) \leq V_L(Z,t) \leq \alpha_2(\|Z\|)$. See [92, Algorithm 1] for insight into selecting the subsequently defined gain conditions.

Theorem 4.1. *Given the dynamics in (4–1), provided that Assumptions 4.1-4.6, as well as 2.1 and 2.2, hold, the weight update laws in (4–17)-(4–19) are implemented, and the*

conditions

$$\eta_{a1} + \eta_{a2} \geq \frac{\eta_{c1} + \eta_{c2}}{\sqrt{\nu \underline{\Gamma}}} \overline{W}\, \overline{G_{\hat{\psi}}}, \tag{4–22}$$

$$\underline{c} \geq 3\frac{\eta_{a1}}{\eta_{c2}} + \frac{3\left(\eta_{c1} + \eta_{c2}\right)^2 \overline{W}^2 \overline{G_{\hat{\psi}}}^2}{16\nu\underline{\Gamma}\eta_{c2}\left(\eta_{a1} + \eta_{a2}\right)} + \frac{9\left(\eta_{c1} + \eta_{c2}\right)^2 \overline{W}^2 \overline{\nabla\psi}^2 \overline{\phi}^2 \left(1 + \overline{g}\left(x\right)\overline{g^+}\right)^2}{8\nu\underline{\Gamma}\eta_{c2}k_\theta \underline{\lambda}}, \tag{4–23}$$

$$v_L^{-1}\left(l\right) < \alpha_2^{-1}\left(\alpha_1\left(r\right)\right), \tag{4–24}$$

are satisfied, where v_L is a subsequently defined PD function, l is a positive constant depending on the NN bounding constants in [113, Assumptions 6 & 7] and Assumptions 4.4 and 4.5 and $r \in \mathbb{R}_{>0}$ is the radius of the compact ball $\chi \subset \mathbb{R}^{n+2L+pn}$ centered at the origin, then the tracking error e, weight estimation errors $\widetilde{W}_c$ and $\widetilde{W}_a$, and the output-layer weight matrix error $\tilde{\theta}$ are UUB, and therefore the control policy $\widehat{u}$ converges to a neighborhood of the optimal control policy u^*.

Proof. Using the class of dynamics in (4–1), the fact that $V_{na}^*\left(e,t\right) \triangleq V^*\left(\zeta\right)$, $\dot{V}^*\left(\zeta\right) = \nabla_\zeta V^*\left(\zeta\right) * F\left(\zeta\right) + G\left(\zeta\right)\mu$, (4–6), (4–17)-(4–19), Young's inequality, nonlinear damping, Assumptions 4.3 and 4.6, and substituting the sufficient gain conditions in (4–22) and (4–23) yields

$$\dot{V}_L \leq -v_L\left(\left\|Z\right\|\right) \ \forall \alpha_2^{-1}\left(\alpha_1\left(r\right)\right) \geq \left\|Z\right\| \geq v_L^{-1}\left(l\right), \tag{4–25}$$

for all $t \in \mathbb{R}_{\geq 0}$, where $v_L\left(\left\|Z\right\|\right) \triangleq \frac{1}{2}\underline{q}\left(e\right) + \frac{1}{12}\eta_{c2}\underline{c}\left\|\widetilde{W}_c\right\|^2 + \frac{1}{16}\left(\eta_{a1} + \eta_{a2}\right)\left\|\widetilde{W}_a\right\|^2 + \frac{1}{6}k_\theta\underline{\lambda}\left\|\mathrm{vec}\left(\tilde{\theta}\right)\right\|^2$.

Since the update laws in (4–6) and (4–17)-(4–19) contain discontinuities that are piecewise continuous in time, and (4–21) is a common Lyapunov function across each DNN iteration i and k, [137, Theorem 4.18] can be invoked to conclude that Z is UUB such that $\limsup_{t\to\infty}\left\|Z\left(t\right)\right\| \leq \alpha_1^{-1}\left(\alpha_2\left(\nu_l^{-1}\left(l\right)\right)\right)$ and $\hat{\mu}$ converges to a neighborhood

of the optimal policy μ^*. Since $Z \in \mathcal{L}_\infty$, then $e, \widetilde{W}_c, \widetilde{W}_a, \tilde{\theta} \in \mathcal{L}_\infty$, and it follows that, $x, \widehat{W}_c, \widehat{W}_a, \hat{\theta} \in \mathcal{L}_\infty$ and $\mu \in \mathcal{L}_\infty$.

Using (4–25), every trajectory $Z(t)$ that satisfies the initial condition $\|Z(0)\| < \alpha_2^{-1}(\alpha_1(r))$ can be shown to be bounded for all $t \in \mathbb{R}_{\geq 0}$, by invoking the result in [137, Theorem 4.18], such that $Z \in \chi \, \forall t \in \mathbb{R}_{\geq 0}$. Since $Z \in \chi$, then the states $e^\top, \widetilde{W}_c^\top, \widetilde{W}_a^\top, \tilde{\theta}^\top$ lie on compact sets. Additionally, since $x_d \leq \overline{x_d}$, it follows that $x \in \mathcal{C}$ and $\zeta \in \Omega$ where $\mathcal{C}$ is the compact set that facilitates the DNN-based system identification, and Ω is the compact set that facilitates the DNN-based value function approximation. □

4.8 Simulation Example

The performance of the developed deep reinforcement learning framework is examined for the application of approximate optimal indirect herding with unknown interaction dynamics [108], [112]. Similar to [108], a DNN is used to estimate the interaction dynamics between the pursuing and evading agents; however, in this result we demonstrate the benefits of using the additional separate DNN to estimate the value function. Additionally, the developed method is compared to the method in [108], using CL for the dynamics estimate, as a baseline.

The exploration policy used in the simulation environment is the policy from [108] where the confidence weighting w_e, defined in (4–20), is derived from the approximate BE in (4–14) and the approximate BE from [108] as $w \triangleq w_\pi w_\mu$, where $w_\pi \triangleq \frac{1}{1+k_\pi|\delta(\pi)|}$, $w_\mu \triangleq \frac{k_\mu|\delta(\mu)|}{1+k_\mu|\delta(\mu)|}$, $\delta(\pi)$ is the BE of the exploration policy, $k_\pi = 0.01$ is a gain for the exploration policy, $\delta(\mu)$ is the BE of DNN policy, and $k_\mu = 0.01$ is a gain for the DNN policy. The confidence weighting is used since the BE is a measure of the optimality of each policy, therefore as the DNN policy improves, the weight of the exploration policy decreases to minimize the bias of an inaccurate exploration policy. Specifically, this is summarized in the following effects: $|\delta(\mu)| \to 0$ then $w \to 0$, $|\delta(\mu)| >> 0$ and $|\delta(\pi)| \to 0$ then $w \to 1$, and $|\delta(\pi)| >> 0$ then $w \to 0$.

The dynamics DNN and value function DNN are both initialized with small random weights with a uniform distribution of $-\sqrt{k}$ to $\sqrt{k}$, where k=1/layer size, and use a residual network architecture with a $\tanh(\cdot)$ output activation function for the dynamics DNN and a $(\cdot)^2 + \log(\cosh(\cdot))$ activation function for the value function DNN. The dynamics DNN consists of an input layer mapping the state size to the hidden layer size, 5 hidden layers of size 64, and an output layer of size 32 with the $\tanh(*)$ activation. The value DNN also consists of an input layer mapping the input size to the hidden layer size, 2 hidden layers of size 64, an output layer of size 32 through the $\log(\cosh(\cdot))$, and a skip connection from the input layer mapped to the output layer size through the $(\cdot)^2$. The output weights are initialized as $\widehat{W}_a = 1_L$, $\widehat{W}_c = 2\widehat{W}_a$, $\Gamma = 5I_{L\times L}$, and $\hat{\theta} \sim \mathcal{N}(0, 1)$. The gains are initialized $\eta_{c1} = \eta_{c2} = 0.01$, $\eta_{a1} = 0.25$, $\eta_{a2} = 0.005$, $\lambda = 0.001$, with $k_\theta = 1$ and $\Gamma_\theta = 10$. The batch sizes are chosen to be $N_\delta = 8$, $M = 20$, and the extrapolation stack size size is $N = 16$. The cost parameters are $Q = 0.01I_{6\times 6}$ and $R = 0.01$.

Figure 4-1 shows the function approximation error from using the DNN to identify the dynamics. The DNN estimate of the dynamics is used in the subsequently shown BE to facilitate model-based deep reinforcement learning. The training for both policies consisted of simulating four different initial conditions around the goal region, the top-right, top-left, bottom-left, and bottom-right locations around the goal, where in all scenarios the dynamics model and value function estimate for both the exploration and DNN policies were updated online (both policies used the same dynamics model). The final simulation results shown are for a fifth scenario as shown in Figure 4-2, where the position of the pursuing agent is represented with a blue circle with a white plus, the position of the evading agent is represented with an orange circle with a white plus, and the goal location is represented with a solid orange circle.

Figure 4-3 compares the BE approximation of the developed method to the method in [108]. Since the BE is used as an indirect measure of optimality, it is shown that the DNN outperforms the policy from [108] for value function approximation using the

BE as a metric of performance. The BE starts very large with the baseline method and converges at about 3 seconds, whereas the BE starts small with the DNN and converges in less than 2 seconds. Using the deep value function approximation method results in 95.04% improvement in BE minimization in terms of mean absolute BE.

Using the selected gains results in excellent tracking performance as shown by the norm tracking error in Figure 4-4. Recall that the state x is a concatenated vector of the errors. Using the developed deep value function approximation method results in a similar norm tracking error mean but 5.06% faster convergence. The baseline method from [108] takes more than 4 seconds to regulate the evading agent to the goal location, while the DNN method takes less than 4 seconds. Figure 4-5 compares the norm inputs between the DNN method and the baseline method which have similar policies with the DNN method having a slightly larger mean norm input which resulted in 5.06% faster convergence with a 95.04% lower BE.

4.9 Concluding Remarks

Lb-DNN function approximation of the system dynamics has been used in Chapter 3, but Lb-DNN value function approximation (policy evaluation) has never been investigated before. This chapter develops a framework for simultaneous Lb-DNN function approximation of the system dynamics and the value function online. A multi-timescale Lb-DNN control method is implemented to update the output-layer DNN weights online via real-time adaptive weight update laws and the inner-layer features via optimization loss functions concurrent to online execution. This framework leverages CL data for the system identification DNN and BE data for the value function DNN. The Lyapunov-based stability analysis proves the states are UUB, and the applied control policy is approximated to within a neighborhood of the optimal control policy. The presented simulation results show that the deep value function approximation method results in 95.04% improvement in BE minimization and 5.06% faster convergence.

This chapter introduces the first result of Lyapunov-based deep reinforcement learning, specifically Lyapunov-based deep policy evaluation. Future work consists of implementing Lyapunov-based deep policy evaluation using a DNN that updates the weights in all of the layers with real-time adaptive update laws using the adaptive DNN analysis subsequently introduced in Chapter 5.

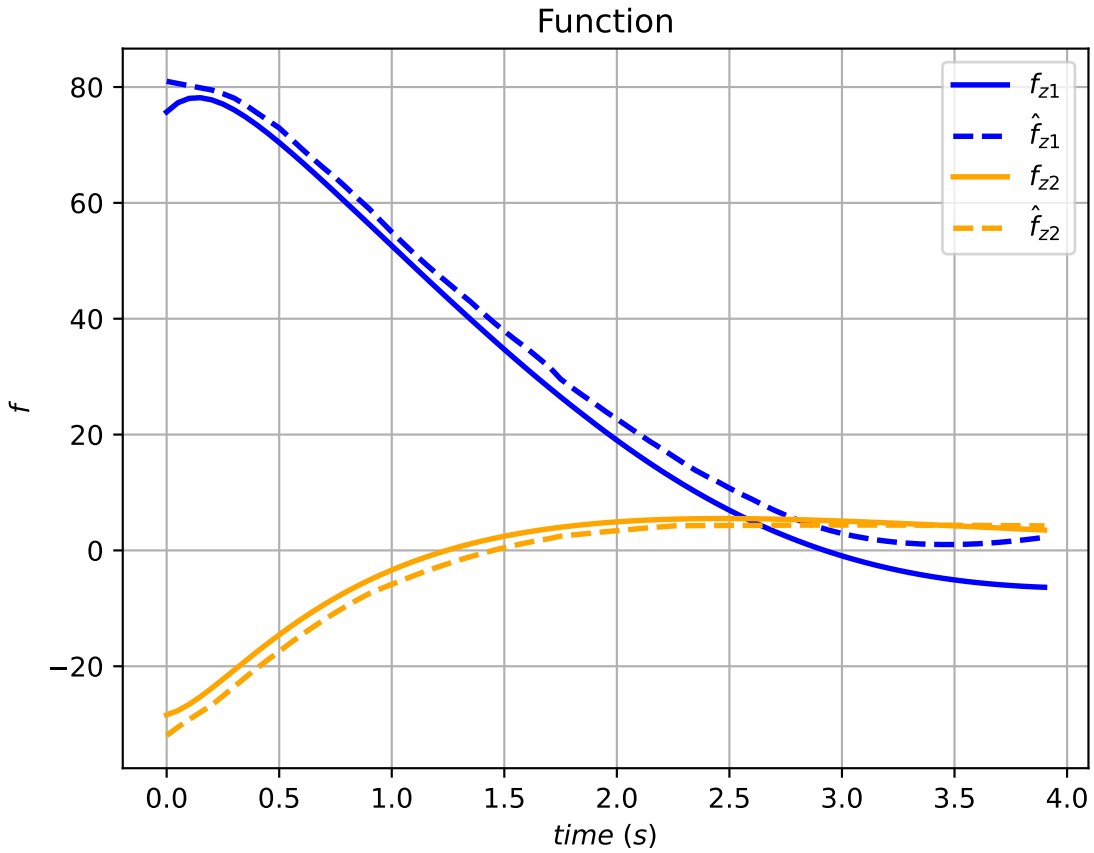

Figure 4-1. Function approximation of the dynamics with the DNN system identifier. The solid lines represent the true values of the dynamics and the dashed lines represent the DNN approximation of the dynamics

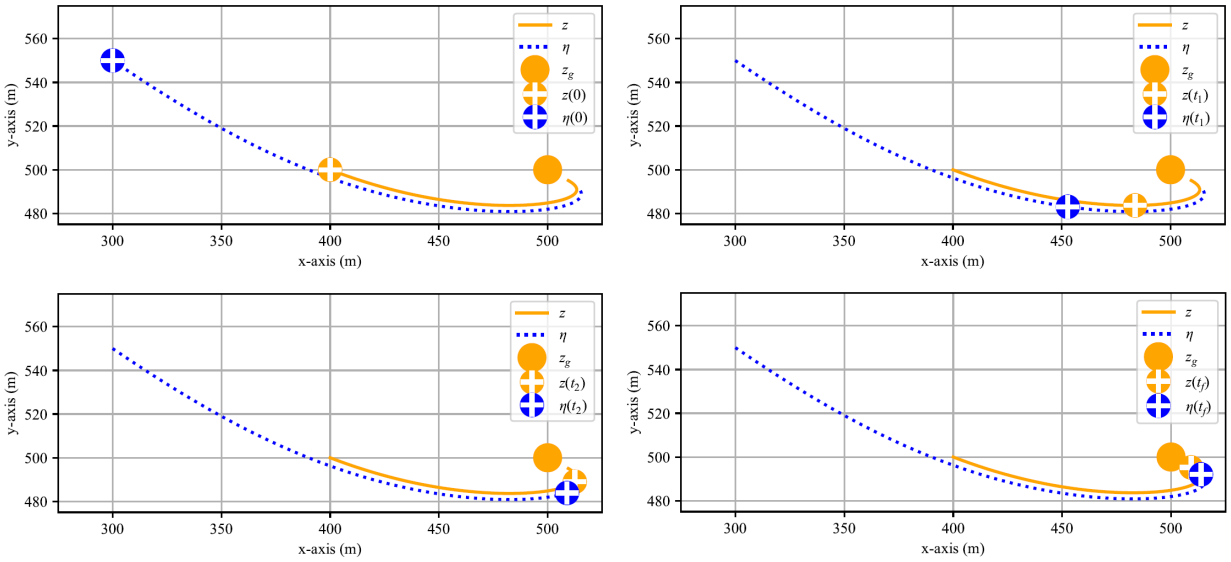

Figure 4-2. Simulation example showing the positions of the pursuing agent, the evading agent, and the goal location. The trajectories for each agent are shown in their respective colors. The simulation shows the pursuing agent escorting the evading agent to the goal location at four different instances in time during the simulation.

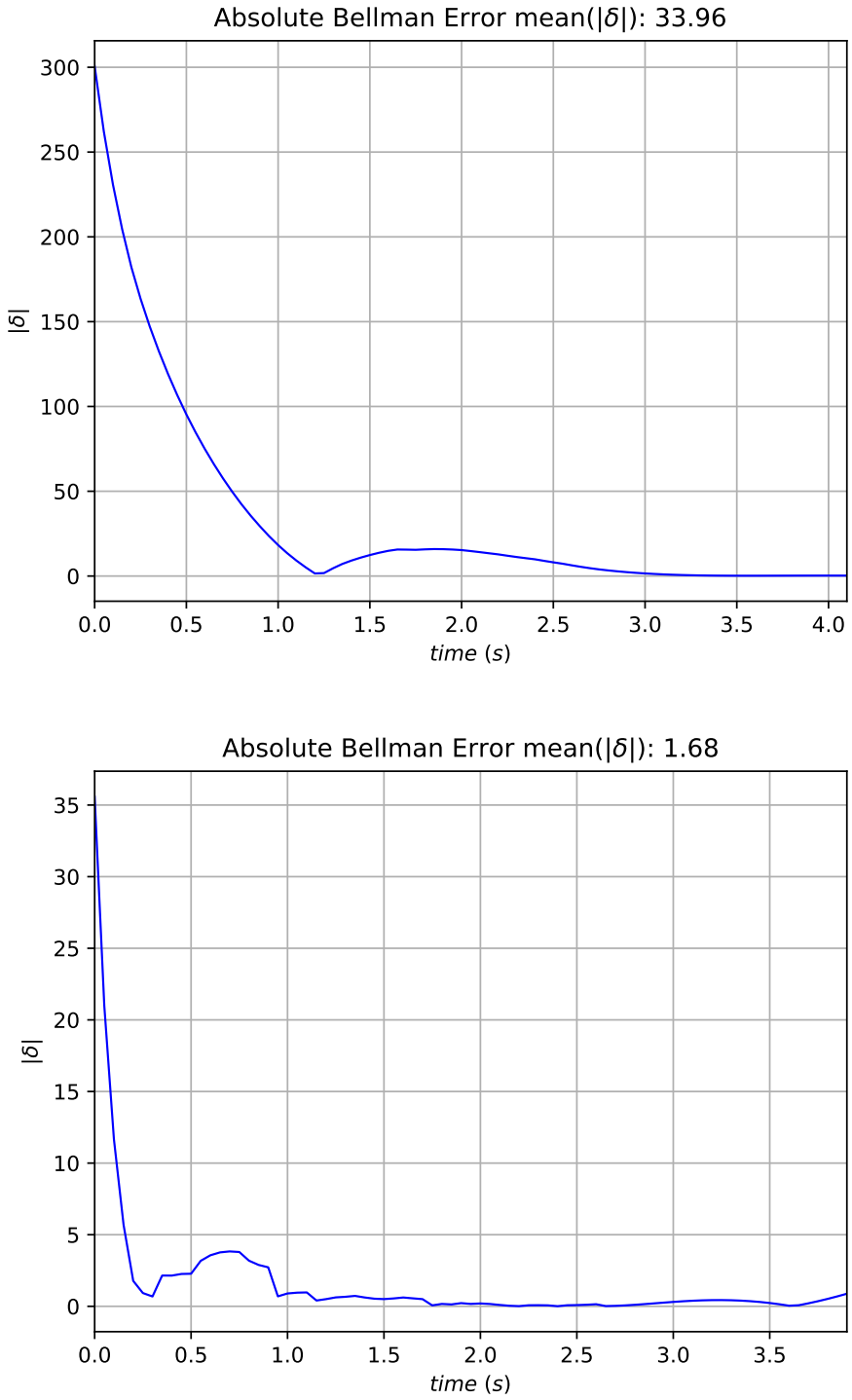

Figure 4-3. Comparative plots of the BE convergence for the baseline method (TOP) compared to the developed deep value function approximation (BOTTOM). The developed deep value function approximation method achieved significantly better BE.

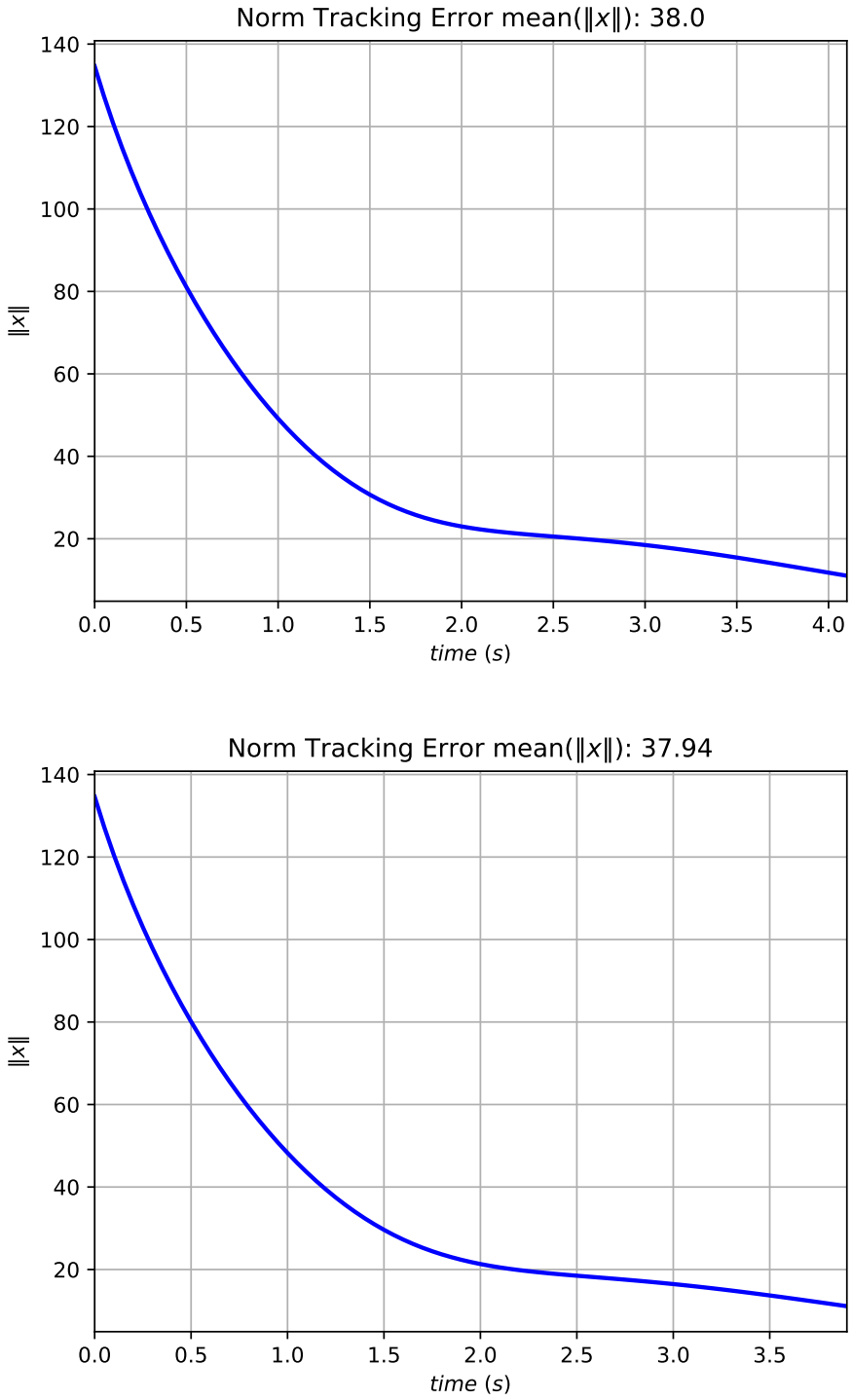

Figure 4-4. Comparative plots of the norm of the state error convergence x for the baseline method (TOP) compared to the developed deep value function approximation (BOTTOM). Both methods resulted in similar mean norm error.

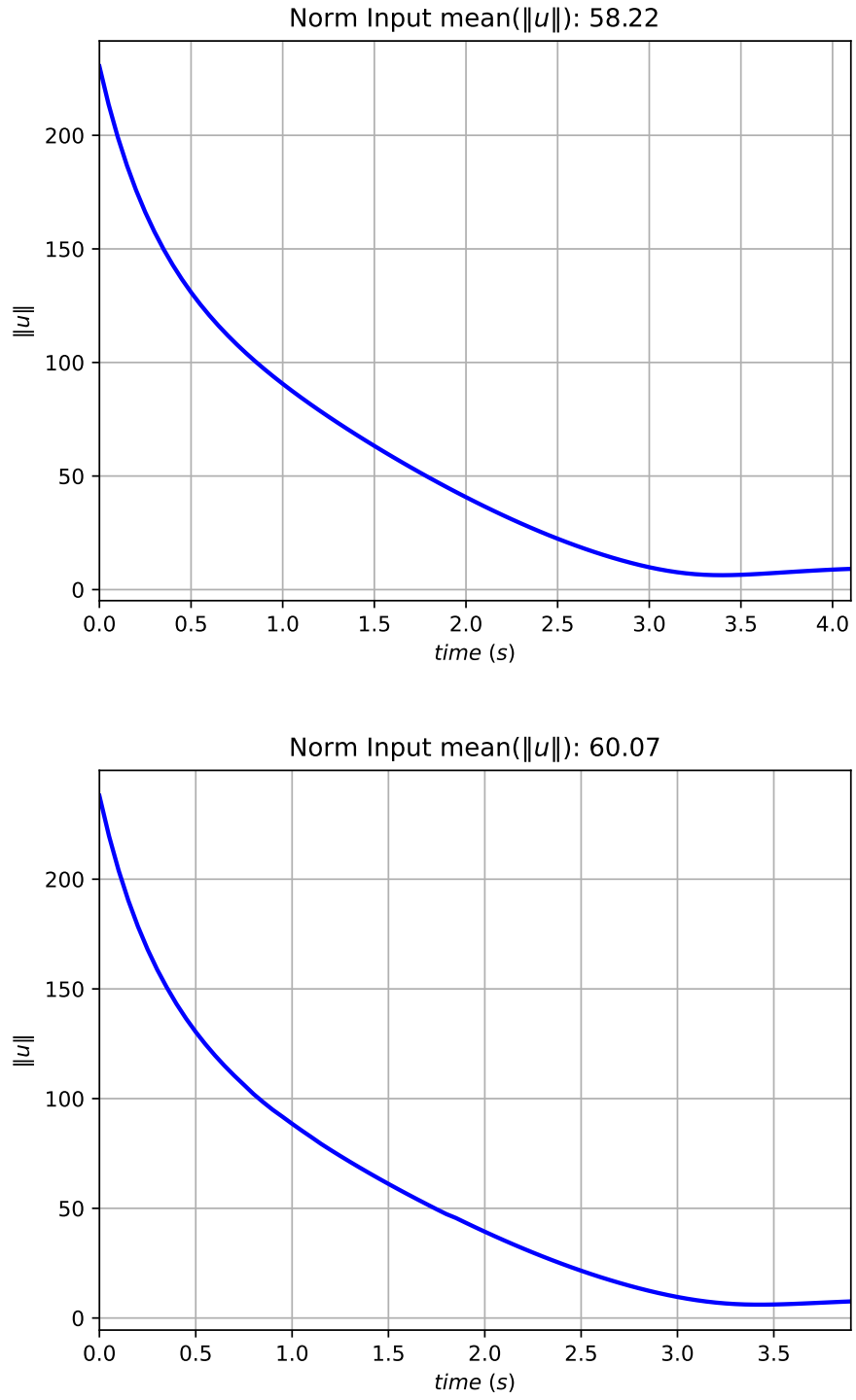

Figure 4-5. Comparative plots of the norm of the input for the baseline method (TOP) compared to the developed deep value function approximation (BOTTOM). Both methods resulted in similar mean norm input.

CHAPTER 5
LYAPUNOV-BASED ADAPTIVE DEEP LEARNING FOR APPROXIMATE DYNAMIC PROGRAMMING

The DNNs in Chapters 3 and 4 were updated using a multi-timescale approach where the output-layer weights are updated using a Lyapunov-based adaptive update law in real-time and the inner-layer weights are updated using iterative batch updates concurrent to real-time. Advancing the DNNs in Chapters 3 and 4, in this chapter an adaptive DNN is used where the weights in all of the layers of the DNN are updated using adaptive update laws in real-time. ADP is performed using an Lb-DNN adaptive identifier to approximate the unknown drift dynamics. Provided the Jacobian of the Lb-DNN satisfies the PE condition, the Lb-DNN is shown to be exponentially convergent to a neighborhood of the DNN weight estimation error, and the control policy is shown to converge to a neighborhood of the optimal control policy. Simulation results show that the Lb-DNN yields 65.73% improved RMS regulation error, 31.82% improved RMS controller error, and 78.97% improved RMS function approximation error in comparison to the previously developed multi-timescale DNN.

5.1 Background DNN Information

DNNs are known to approximate any given continuous function on a compact set, based on the universal function approximation theorem [144]. Although various DNN architectures can be used, a fully-connected DNN is described here as an example. Let $\sigma \in \mathbb{R}^{L_{\text{in}}}$ denote the DNN input with size $L_{\text{in}} \in \mathbb{Z}_{>0}$, and $\theta \in \mathbb{R}^{p}$ denote the vector of DNN parameters (i.e., weights and bias terms) with size $p \in \mathbb{Z}_{>0}$. Then, a fully-connected feedforward DNN $\Phi(\sigma, \theta)$ with output size $L_{\text{out}} \in \mathbb{Z}_{>0}$ is defined using a recursive relation $\Phi_j \in \mathbb{R}^{L_{j+1}}$ modeled as

$$\Phi_j \triangleq \begin{cases} V_j^{\top} \phi_j \left(\Phi_{j-1} \right), & j \in \{1, \ldots, k\}, \\ V_j^{\top} \sigma_a, & j = 0, \end{cases} \tag{5–1}$$

where $\Phi(\sigma,\theta) = \Phi_k$, and $\sigma_a \triangleq \begin{bmatrix} \sigma^\top & 1 \end{bmatrix}^\top$ denotes the augmented input that accounts for the bias terms, $k \in \mathbb{Z}_{>0}$ denotes the total number of hidden layers, $V_j \in \mathbb{R}^{L_j \times L_{j+1}}$ denotes the matrix of weights and biases, $L_j \in \mathbb{Z}_{>0}$ denotes the number of nodes in the j^{th} layer for all $j \in \{0,\dots,k\}$ with $L_0 \triangleq L_{\text{in}} + 1$ and $L_{k+1} = L_{\text{out}}$. The vector of smooth activation functions is denoted by $\phi_j : \mathbb{R}^{L_j} \to \mathbb{R}^{L_j}$ for all $j \in \{1,\dots,k\}$. If the DNN involves multiple types of activation functions at each layer, then ϕ_j may be represented as $\phi_j \triangleq \begin{bmatrix} \varsigma_{j,1} & \dots & \varsigma_{j,L_j-1} & 1 \end{bmatrix}^\top$, where $\varsigma_{j,p} : \mathbb{R} \to \mathbb{R}$ denotes the activation function at the p^{th} node of the j^{th} layer. For the DNN architecture in (5–1), the vector of DNN weights is $\theta \triangleq \begin{bmatrix} \text{vec}(V_0)^\top & \dots & \text{vec}(V_k)^\top \end{bmatrix}^\top$ with size $p = \Sigma_{j=0}^{k} L_j L_{j+1}$. The Jacobian of the activation function vector at the j^{th} layer is denoted by $\phi_j' : \mathbb{R}^{L_j} \to \mathbb{R}^{L_j \times L_j}$, and $\phi_j'(y) \triangleq \frac{\partial}{\partial z}\phi_j(z)\big|_{z=y}$, $\forall y \in \mathbb{R}^{L_j}$. Let the Jacobian of the DNN with respect to the weights be denoted by $\Phi'(\sigma,\theta) \triangleq \frac{\partial}{\partial \theta}\Phi(\sigma,\theta)$, which can be represented using $\Phi'(\sigma,\theta) = \begin{bmatrix} \Phi_0', & \Phi_1', & \dots, & \Phi_k' \end{bmatrix}$, where $\Phi_j' \triangleq \frac{\partial}{\partial \text{vec}(V_j)}\Phi(\sigma,\theta)$ for all $j \in \{0,\dots,k\}$. Then, using (5–1) and the property of the vectorization operator in (1–2) yields

$$\Phi_0' = \left(\prod_{l=1}^{\overset{\curvearrowleft}{k}} \widehat{V}_l^\top \phi_l'(\Phi_{l-1}) \right) (I_{L_1} \otimes \sigma_a^\top), \tag{5–2}$$

and

$$\Phi_j' = \left(\prod_{l=j+1}^{\overset{\curvearrowleft}{k}} \widehat{V}_l^\top \phi_l'(\Phi_{l-1}) \right) \left(I_{L_{j+1}} \otimes \phi_j^\top(\Phi_{j-1}) \right), \tag{5–3}$$

for all $j \in \{1,\dots,k\}$.

5.2 Problem Formulation

This chapter uses the dynamics in (2–1) and Assumptions 2.1 and 2.2.

Assumption 5.1. The function f is $\mathcal{C}^2$

The control objective is to solve the infinite horizon optimal regulation problem online, i.e. find an optimal control policy u that minimizes the cost function

$$J(x,u) = \int_0^\infty Q(x(\tau)) + u(\tau)^\top Ru(\tau)\ d\tau. \tag{5–4}$$

In (5–4), $Q : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is a PD cost function where Q satisfies $\underline{q}(\|x\|) \leq Q(x) \leq \overline{q}(\|x\|)$ for $\underline{q}, \overline{q} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, and $R \in \mathbb{R}^{m\times m}$ is a user-defined constant PD symmetric cost matrix.

The cost-to-go (i.e. the infinite horizon value function) $V^* : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is defined as

$$V^*(x) \triangleq \min_{u\in U} \int_t^\infty Q(x(\tau)) + u(\tau)^\top Ru(\tau)\ d\tau, \tag{5–5}$$

where $U \subseteq \mathbb{R}$ is the action space for u.

A major roadblock in finding the approximately optimal control policy is that the drift dynamics f are unknown and involve complex nonlinearities. The following section provides a method for identifying the unknown drift dynamics in real-time using DNNs.

5.3 System Identification

DNNs are known to be effective at approximating unknown nonlinear functions such as the drift dynamics f. Previous results in [108, 113, 115, 116], Chapter 3, and Chapter 4 have used DNNs for system identification in the approximate optimal control problem. However, in those results, the inner-layer weights of the DNN were updated concurrent to real-time in batches using offline training techniques. Traditional offline training techniques require large amounts of data and do not account for disturbances and uncertainties in real-time. In this chapter, the system dynamics are identified using an Lb-DNN with real-time weight adaptation laws for all layers of the DNN. The developed DNN weight estimates are shown to approximately converge to their true values under an explicit PE condition, unlike the previously cited results.

5.3.1 Dynamics Estimate

Let $\Phi : \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}^n$ denote a generalized DNN defined in the Appendix where p represents the total number of DNN weights. DNNs are known to approximate continuous functions on a compact set using the Universal Function Approximation theorem [144]. The subsequent stability analysis guarantees that if x is initialized in an appropriately-sized subset of Ω, then it will stay in Ω. The drift dynamics can be approximated with a DNN on a compact set $\Omega \subset \mathbb{R}^n$ as

$$f(x) = \Phi(x, \theta^*) + \varepsilon(x) \tag{5–6}$$

where $\varepsilon : \mathbb{R}^n \rightarrow \mathbb{R}^n$ denotes an unknown function approximation error that can be bounded as $\sup_{x\in\Omega} \|\varepsilon(x)\| \leq \overline{\varepsilon}$, and $\theta^* \in \mathbb{R}^p$ denotes ideal weights such that $\sup_{x\in\Omega} \|f(x) - \Phi(x, \theta^*)\| \leq \overline{\varepsilon}$. An estimate of the dynamics is represented as $\Phi\left(x, \hat{\theta}\right)$ where $\hat{\theta} \in \mathbb{R}^p$ is the subsequently designed adaptive estimate of the ideal DNN weights θ^*. Various different architectures such as fully-connected DNNs, ResNets, LSTMs found in [125], [126], and [128] respectively, can be used in the system identifier.

Assumption 5.2. There exists a known constant $\overline{\theta} \in \mathbb{R}_{>0}$ such that the unknown ideal weights can be bounded as $\|\theta^*\| \leq \overline{\theta}$.

Real-time system identifiers typically use an identification error as feedback. However, the identification error of the dynamics cannot be directly evaluated due to the absence of state-derivative information. Results in [108, 115, 116] and Chapter 3 use integrators to avoid requiring state-derivative information while identifying the output-layer weights. However, integrators do not assist in identifying the inner-layer weights due to the nonlinear parameterization of DNNs. To overcome these challenges, a RISE-based dynamics observer is used to obtain an instantaneous second estimate of the dynamics [145]. The subsequent RISE-based dynamics observer is capable of exponentially identifying uncertainty and disturbances in the function and is designed as

$$\dot{\hat{x}} = \hat{f} + gu + \alpha_1 \tilde{x} \tag{5–7}$$

$$\dot{\hat{f}} = \tilde{x} + k_f \left(\dot{\tilde{x}} + \alpha_1 \tilde{x} \right) + \beta_f \text{sgn}\left(\tilde{x} \right) \tag{5–8}$$

where $\hat{x}, \hat{f} \in \mathbb{R}^n$ are the observer estimates of x and f, respectively, $\tilde{x}, \tilde{f} \in \mathbb{R}^n$ are the observer errors $\tilde{x} \triangleq x - \hat{x}$ and $\tilde{f} \triangleq f(x) - \hat{f}$, respectively, and $\alpha_1, k_f, \beta_f \in \mathbb{R}_{>0}$ denote constant observer gains. The observer error $\tilde{x}$ is known because x and $\hat{x}$ are known. However, since $\dot{\tilde{x}}$ is unknown, (5–8) can be implemented by integrating both sides and using the relation $\int_0^t \dot{\tilde{x}}\left(\tau\right) d\tau = \tilde{x}(t) - \tilde{x}(0)$ to obtain $\hat{f}(t) = \hat{f}(0) + k_f \tilde{x}(t) - k_f \tilde{x}(0) + \int_0^t \left[(k_f \alpha_1 + 1) \tilde{x}(\tau) + \beta_f \text{sgn}\left(\tilde{x}\left(\tau\right) \right) \right] d\tau$ which is a solution to (5–8). Taking the derivative of $\tilde{x}$ and substituting (5–7) yields

$$\dot{\tilde{x}} = \tilde{f} - \alpha_1 \tilde{x}. \tag{5–9}$$

Additionally, taking the derivative of $\tilde{f}$ and substituting (5–8) and (5–9) yields

$$\dot{\tilde{f}} = \dot{f} - \tilde{x} - k_f \tilde{f} - \beta_f \text{sgn}\left(\tilde{x} \right), \tag{5–10}$$

where $\dot{f} \triangleq \frac{\partial f}{\partial x} \dot{x}$. An identification error $E \in \mathbb{R}^n$ based on the observer estimate of f and the DNN estimate of f is calculated as

$$E = \hat{f} - \Phi\left(x, \hat{\theta} \right) \tag{5–11}$$

and is used to update the weights of the DNN in real-time.

Remark 5.1. Although a RISE-based observer is capable of producing an instantaneous estimate of the drift dynamics by essentially acting as a state-derivative estimator, the subsequent control development requires extrapolation of the dynamics to unexplored areas of the state space that can only be achieved using the identified DNN.

5.3.2 Adaptation Laws

To facilitate the subsequent analysis, the least squares adaptive update law is designed as

$$\dot{\hat{\theta}} = \Gamma_\theta \Phi'^{\top} \left(x, \hat{\theta} \right) E, \tag{5–12}$$

where the Jacobian $\Phi'\left(x,\hat{\theta}\right) \in \mathbb{R}^{n\times p}$ is calculated using (5–2) and (5–3), and the term $\Gamma_\theta \in \mathbb{R}^{p\times p}$ denotes a symmetric, positive-definite (PD) time-varying least squares adaptation gain matrix that is a solution to [146, Eqns. (16) and (17)]

$$\frac{d}{dt}\Gamma_\theta^{-1} = -\beta(t)\Gamma_\theta^{-1} + \Phi'^{\top}\left(x,\hat{\theta}\right)\Phi'\left(x,\hat{\theta}\right), \tag{5–13}$$

where the bounded-gain time-varying forgetting factor $\beta : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is designed as

$$\beta(t) \triangleq \beta_0\left(1 - \frac{\lambda_{\max}\{\Gamma_\theta\}}{\kappa_0}\right), \tag{5–14}$$

where $\beta_0, \kappa_0 \in \mathbb{R}_{>0}$ are user-selected constants that denote the maximum forgetting rate and the prescribed bound on $\lambda_{\max}\{\Gamma_\theta\}$, respectively. The adaptation gain in (5–13) is initialized to be PD such that $\lambda_{\max}\{\Gamma_\theta(0)\} < \kappa_0$, and it can be shown that $\Gamma_\theta(t)$ remains PD for all $t \in \mathbb{R}_{\geq 0}$ [146]. The term $\beta(t)$ can be lower bounded as $\beta \geq \beta_1$, where $\beta_1 \in \mathbb{R}_{\geq 0}$ is a constant which satisfies the properties stated in the subsequent remark.

If $\Phi'\left(x,\hat{\theta}\right)$ satisfies the PE condition, i.e., there exists constants $\varphi_1, \varphi_2 \in \mathbb{R}_{>0}$ for all $t_1 \in \mathbb{R}_{\geq 0}$ and some $T \in \mathbb{R}_{>0}$ such that $\varphi_1 I_p \leq \int_{t_1}^{t_1+T} \Phi'^{\top}\left(x(\tau),\hat{\theta}(\tau)\right)\Phi'\left(x(\tau),\hat{\theta}(\tau)\right)d\tau \leq \varphi_2 I_p$, then it can be shown that $\beta_1 > 0$ [146, Sec. 4.2].

Remark 5.2. The PE condition requires the Jacobian Φ' to be sufficiently rich, which yields sufficient exploration of the state-space. Under this condition, the weight estimates are shown to converge to a neighborhood of their ideal values. As a result, the DNN can generalize beyond the explored trajectory, thus allowing extrapolation in the subsequent control development.

Analyzing the convergence properties of the adaptive update law in (5–12) is challenging due to the nested nonlinear parameterization of the DNN. To address this challenge, a first-order Taylor series approximation is used. A first-order Taylor series approximation is introduced to overcome the challenges of the nested nonlinear parameterization introduced by the DNN. Applying a first-order Taylor series approximation to

the generalized DNN illustrated in the Appendix yields

$$\Phi(x,\theta^*) - \Phi\left(x,\hat{\theta}\right) = \Phi'\left(x,\hat{\theta}\right)\tilde{\theta} + \mathcal{O}\left(\left\|\tilde{\theta}\right\|^2\right), \tag{5–15}$$

where $\tilde{\theta} \in \mathbb{R}^p$ denotes the parameter estimation error $\tilde{\theta} \triangleq \theta - \hat{\theta}$, and $\mathcal{O}\left(\left\|\tilde{\theta}\right\|^2\right)$ denotes the higher-order terms. By adding and subtracting f and substituting (5–6) and (5–15), the identification error E can be rewritten as

$$E = -\tilde{f} + \Phi'\left(x,\hat{\theta}\right)\tilde{\theta} + \Delta, \tag{5–16}$$

where $\Delta \triangleq \mathcal{O}\left(\left\|\tilde{\theta}\right\|^2\right) + \varepsilon(x)$. Since $\dot{\tilde{\theta}} \triangleq -\dot{\hat{\theta}}$, by substituting (5–12) and (5–16) the time derivative of $\tilde{\theta}$ is calculated as

$$\dot{\tilde{\theta}} = -\Gamma_\theta \Phi'^{\top}\left(x,\hat{\theta}\right)\Phi'\left(x,\hat{\theta}\right)\tilde{\theta} + \Gamma_\theta \Phi'^{\top}\left(x,\hat{\theta}\right)\tilde{f} - \Gamma_\theta \Phi'^{\top}\left(x,\hat{\theta}\right)\Delta. \tag{5–17}$$

5.3.3 Stability Analysis

To achieve exponential convergence, a P-function is included in the subsequent Lyapunov analysis in addition to the typical sum of norm squared error terms [147]. The P-function is designed as

$$P \triangleq \beta \left\|\tilde{x}\right\|_1 - \tilde{x}^\top \dot{f} + \mathrm{e}^{-\lambda_P t} * \left(\tilde{x}^\top \ddot{f}\right) + \mathrm{e}^{-\lambda_P t} * \left((\alpha_1 - \lambda_P)\left(\beta \left\|\tilde{x}\right\|_1 - \tilde{x}^\top \dot{f}\right)\right), \tag{5–18}$$

where $\lambda_p \in \mathbb{R}_{>0}$ is a user-selected constant, and $\ddot{f} \triangleq \dot{x}^\top \left(\frac{\partial^2 f}{\partial x^2}\right)\dot{x} + \frac{\partial f}{\partial x}\ddot{x}$. The convolutional integral $\mathrm{e}^{-\lambda_P t} * q = \int_{t_0}^{t} \mathrm{e}^{-\lambda_P (t-\sigma)} q(\sigma) d\sigma$ is denoted by '$*$' for any given $q : [t_0, \infty) \to \mathbb{R}$, and can be verified using the Leibniz rule that $\frac{d}{dt}\left(\int_{t_0}^{t} \mathrm{e}^{-\lambda_P (t-\sigma)} q(\sigma) d\sigma\right) = q(t) - \lambda_P \int_{t_0}^{t} \mathrm{e}^{-\lambda_P (t-\sigma)} q(\sigma) d\sigma$. The convolutional integral therefore satisfies the property $\frac{d}{dt}\left(\mathrm{e}^{-\lambda_P t} * q\right) = q(t) - \lambda_P \mathrm{e}^{-\lambda_P t} * q$. The mapping $t \mapsto \left\|e(t)\right\|_1$ is differentiable for almost all time since $t \mapsto e(t)$ is absolutely continuous and $\left\|\cdot\right\|_1$ is globally Lipschitz; hence, the use of the chain rule in [148, Theorem 2.2] yields $\frac{d}{dt}\left(\left\|e\right\|_1\right) \overset{a.a.t.}{\in} K[\mathrm{sgn}](e)$. Taking the time-derivative of (5–18), using Leibniz's rule, and substituting (5–9) and (5–18) results

in

$$\dot{P} \overset{a.a.t.}{\in} -\lambda_p P + \tilde{f}^\top \left(\beta_f \text{sgn}\left(\tilde{x}\right) - \dot{f} \right).$$

A more detailed derivation can be found in [147, Proof of Lemma 3].

Let $z_\theta \triangleq \begin{bmatrix} \tilde{x}^\top & \tilde{f}^\top & \tilde{\theta}^\top & \sqrt{2P} \end{bmatrix}^\top \in \mathbb{R}^{2n+p+1}$ denote the concatenated state. The candidate Lyapunov function $V_\theta : \mathbb{R}^{2n+p+1} \to \mathbb{R}$ is defined as

$$V_\theta\left(z_\theta\right) = \frac{1}{2}\tilde{x}^\top \tilde{x} + \frac{1}{2}\tilde{f}^\top \tilde{f} + \frac{1}{2}\tilde{\theta}^\top \Gamma_\theta^{-1} \tilde{\theta} + P. \tag{5–19}$$

The Lyapunov function is bounded as

$$\lambda_1 \left\| z_\theta \right\|^2 \leq V_\theta(z_\theta) \leq \lambda_2 \left\| z_\theta \right\|^2, \tag{5–20}$$

where $\lambda_1 \triangleq \min\{\frac{1}{2}, \frac{1}{2\lambda_{\max}\{\Gamma_\theta\}}\}$ and $\lambda_2 \triangleq \max\{\frac{1}{2}, \frac{1}{2\lambda_{\min}\{\Gamma_\theta\}}\}$. Consider the compact domain $\mathcal{D} \triangleq \{\zeta \in \mathbb{R}^{4n+p} : \|\zeta\| \leq \chi\}$ where $\chi \in \mathbb{R}_{>0}$ is a bounding constant. The subsequent analysis shows that the concatenated state $z_\theta(t) \in \mathcal{D}$ for all $t \in \mathbb{R}_{\geq 0}$ if z is initialized in the set $\mathcal{S} \triangleq \left\{\zeta \in \mathbb{R}^{2n+p+1} : \|\zeta\| \in \sqrt{\frac{\lambda_1}{\lambda_2}\chi^2 - \frac{C}{\lambda_3}}\right\}$ in the subsequent stability analysis. Using [147, Lemma 4] it can be shown that $P \geq 0$ if the gain conditions

$$\alpha > \lambda_P \tag{5–21}$$

and

$$\beta > \Upsilon_1 + \frac{\Upsilon_2}{\alpha - \lambda_P} \tag{5–22}$$

are satisfied, where bounds $\left\|\dot{f}\right\| \leq \Upsilon_1$ and $\left\|\ddot{f}\right\| \leq \Upsilon_2$ hold where Υ_1 and Υ_2 are bounding constants based on Assumption 5.1 and the fact that $\dot{x}$ and $\ddot{x}$ are bounded when $z \in \mathcal{D}$.

Theorem 5.1. *Provided Assumptions 5.1 and 5.2 and the gain conditions in (5–21), (5–22), and (5–25) are satisfied, the adaptive update laws in (5–12) and (5–13) ensure*

that the estimation errors defined in z_θ are uniformly ultimately bounded (UUB) such that

$$\|z_\theta(t)\| \leq \sqrt{\frac{\lambda_2}{\lambda_1}\|z_\theta(0)\|^2 \mathrm{e}^{-\frac{\lambda_3}{\lambda_2}t} + \frac{\lambda_2 C}{\lambda_1\lambda_3}\left(1-\mathrm{e}^{-\frac{\lambda_3}{\lambda_2}t}\right)}.$$

Proof. Taking the time derivative of (5–19) yields

$$\dot{V}_\theta \overset{a.a.t.}{\in} \tilde{x}^\top\dot{\tilde{x}} + \tilde{f}^\top\dot{\tilde{f}} + \tilde{\theta}^\top\Gamma_\theta^{-1}\dot{\tilde{\theta}} + \frac{1}{2}\tilde{\theta}^\top\left(\frac{d}{dt}\Gamma_\theta^{-1}\right)\tilde{\theta} - \lambda_p P + \tilde{f}^\top\left(\beta_f \mathsf{sgn}\left(\tilde{x}\right) - \dot{f}\right). \tag{5–23}$$

By substituting (5–9), (5–10), and (5–17), and cancelling coupling terms, (5–23) can be upper bounded as

$$\begin{aligned}\dot{V}_\theta &\overset{a.a.t.}{\leq} -\alpha_1\tilde{x}^2 - k_f\left\|\tilde{f}\right\|^2 - \frac{\beta(t)}{2}\tilde{\theta}^\top\tilde{\theta} - \lambda_p P \\ &\quad - \tilde{\theta}^\top\left(\frac{1}{2}\Phi'^\top\left(x,\hat{\theta}\right)\Phi'\left(x,\hat{\theta}\right)\right)\tilde{\theta} + \tilde{\theta}^\top\Phi'^\top\left(x,\hat{\theta}\right)\left(\tilde{f}-\Delta\right).\end{aligned}$$

The parameter estimation error can be bounded as $||\tilde{\theta}|| \leq \chi$ when $z \in \mathcal{D}$. Additionally, since f and Φ are continuously differentiable, the bounds $\|\Delta\| \leq \gamma_1$ and $\left\|\Phi'\left(x,\hat{\theta}\right)\right\| \leq \gamma_2$ hold when $z_\theta \in \mathcal{D}$, where $\gamma_1, \gamma_2 \in \mathbb{R}_{>0}$ denote bounding constants. Therefore, using Young's inequality yields the bound $\tilde{\theta}^\top\Phi'^\top\left(x,\hat{\theta}\right)\left(\tilde{f}-\Delta\right) \leq \gamma_2\left\|\tilde{\theta}\right\|^2 + \frac{\gamma_2}{2}\left\|\tilde{f}\right\|^2 + \frac{\gamma_2\gamma_1^2}{2}$. As a result, $\dot{V}_\theta$ can further be upper-bounded as

$$\dot{V}_\theta \overset{a.a.t.}{\leq} -\lambda_3\|z\|^2 + C - \frac{1}{2}\tilde{\theta}^\top\Phi'^\top\left(x,\hat{\theta}\right)\Phi'\left(x,\hat{\theta}\right)\tilde{\theta}, \tag{5–24}$$

when $z_\theta \in \mathcal{D}$, where $\lambda_3 \triangleq \min\{\alpha_1, k_f - \frac{\gamma_2}{2}, \frac{\beta_1}{2} - \gamma_2, \lambda_P\}$ and $C \triangleq \frac{\gamma_2\gamma_1^2}{2}$. Using (5–20) and (5–24), when the gain condition

$$\lambda_3 \;>\; 0 \tag{5–25}$$

is satisfied, $\dot{V}$ can be upper-bounded as

$$\dot{V}_\theta \overset{a.a.t.}{\leq} -\frac{\lambda_3}{\lambda_2}V_\theta + C, \tag{5–26}$$

when $z_\theta \in \mathcal{D}$. Solving the differential inequality in (5–26) yields

$$V_\theta\left(z(t)\right) \leq V_\theta\left(z(0)\right) \mathrm{e}^{-\frac{\lambda_3}{\lambda_2}t} + \frac{\lambda_2 C}{\lambda_3}\left(1 - \mathrm{e}^{-\frac{\lambda_3}{\lambda_2}t}\right),$$

when $z_\theta \in \mathcal{D}$, and applying (5–20) yields the bound

$$\|z_\theta(t)\| \leq \sqrt{\frac{\lambda_2}{\lambda_1}\|z_\theta(0)\|^2 \mathrm{e}^{-\frac{\lambda_3}{\lambda_2}t} + \frac{\lambda_2 C}{\lambda_1\lambda_3}\left(1 - \mathrm{e}^{-\frac{\lambda_3}{\lambda_2}t}\right)}, \tag{5–27}$$

when $z_\theta \in \mathcal{D}$. To guarantee $z_\theta(t) \in \mathcal{D}$ for all $t \in \mathbb{R}_{\geq 0}$, (5–27) can be upper bounded as $\|z_\theta(t)\| \leq \sqrt{\frac{\lambda_2}{\lambda_1}\|z_\theta(0)\|^2 + \frac{\lambda_2 c}{\lambda_1\lambda_3}}$ for all $t \in \mathbb{R}_{\geq 0}$. Due to the fact that $\mathcal{D} \triangleq \{\zeta \in \mathbb{R}^{4n+p} : \|\zeta\| \leq \chi\}$, the relation $z_\theta(t) \in \mathcal{D}$ holds if $\sqrt{\frac{\lambda_2}{\lambda_1}\|z_\theta(0)\|^2 + \frac{\lambda_2 c}{\lambda_1\lambda_3}} \leq \chi$, which is achieved if $\|z_\theta(0)\| \leq \sqrt{\frac{\lambda_1}{\lambda_2}\chi^2 - \frac{c}{\lambda_3}}$, i.e., $z_\theta(0) \in S$; in this case, (5–27) holds for all $t \in [0, \infty)$. □

5.4 Approximate Optimal Control

The optimal value function in (5–5) is the solution to the corresponding HJB equation

$$0 = \nabla V^*(x)\left(f(x) + g(x)u^*\right) + Q(x) + u^*(x)^\top R u^*(x), \tag{5–28}$$

with the condition $V^*(0) = 0$ and where $u^* : \mathbb{R}^n \to \mathbb{R}^m$ is the optimal control policy. Taking the partial derivative of (5–28) with respect to the minimizing argument $u^*(x)$, setting it equal to zero, and solving for $u^*(x)$ results in the optimal control policy

$$u^*(x) = -\frac{1}{2}R^{-1}g(x)^\top\left(\nabla V^*(x)\right)^\top. \tag{5–29}$$

Assumption 5.3. The optimal value function V^* is continuously differentiable [92].

5.4.1 Value Function Approximation

The optimal value function is generally unknown for nonlinear systems. To solve for the optimal control policy in (5–29), the optimal value function can be approximated with

a NN in a compact set $\Omega \subset \mathbb{R}^n$ using the Universal Function Approximation Theorem as

$$V^*(x) = W^\top \phi(x) + \epsilon(x) \ \forall x \in \Omega, \tag{5–30}$$

where $W \in \mathbb{R}^L$ is a vector of unknown weights, $\phi : \mathbb{R}^n \to \mathbb{R}^L$ is a user-defined vector of basis functions, and $\epsilon : \mathbb{R}^n \to \mathbb{R}$ is the bounded function reconstruction error. Substituting (5–30) into (5–31), the optimal control policy in (5–29) can be approximated with a NN as

$$u^*(x) = -\frac{1}{2} R^{-1} g(x)^\top \left(\nabla \phi(x)^\top W + \nabla \epsilon(x)^\top \right). \tag{5–31}$$

Assumption 5.4. There exists a set of known positive constants $\overline{W}, \overline{\phi}, \overline{\nabla \phi}, \overline{\epsilon}, \overline{\nabla \epsilon} \in \mathbb{R}_{>0}$ such that $\sup \|W\| \leq \overline{W}$, $\sup_{x \in \Omega,} \|\phi(x)\| \leq \overline{\phi}$, $\sup_{x \in \Omega} \|\nabla \phi(x)\| \leq \overline{\nabla \phi}$, $\sup_{x \in \Omega} \|\epsilon(x)\| \leq \overline{\epsilon}$, and $\sup_{x \in \Omega,} \|\nabla \epsilon(x)\| \leq \overline{\nabla \epsilon}$ [135, Assumptions 9.1.c-e].

The ideal weights W in (5–30) and (5–31) are unknown *a priori*. In this chapter, an actor-critic NN architecture is used where actor and critic weight estimates are used to approximate W. The critic weight estimate vector $\hat{W}_c \in \mathbb{R}^L$ is used to approximate (5–30), resulting in the optimal value function estimate $\widehat{V} : \mathbb{R}^n \times \mathbb{R}^L \to \mathbb{R}$, defined as

$$\widehat{V}\left(x, \widehat{W}_c\right) \triangleq \widehat{W}_c^\top \phi(x). \tag{5–32}$$

The actor weight estimate vector $\widehat{W}_a \in \mathbb{R}^L$ is used to approximate (5–31), resulting in the optimal control policy estimate $\hat{u} : \mathbb{R}^n \times \mathbb{R}^L \to \mathbb{R}^m$, defined as

$$\hat{u}\left(x, \widehat{W}_a\right) \triangleq -\frac{1}{2} R^{-1} g(x)^\top \left(\nabla \phi(x)^\top \widehat{W}_a \right). \tag{5–33}$$

5.4.2 Bellman Error

The error resulting from approximating the system dynamics, the optimal value function, and the optimal control input introduces an error in the HJB equation in (5–28). This error, termed the BE, is representative of the performance of the developed method, and is used to update the actor-critic weights in the subsequent development. Replacing the drift dynamics f with the estimate $\Phi\left(x, \hat{\theta}\right)$, the optimal value function

$V^*(x)$ with the estimate $\widehat{V}\left(x, \widehat{W}_c\right)$, and the optimal control policy $u^*(x)$ with the estimate $\hat{u}\left(x, \widehat{W}_a\right)$ in (5–28) results in the BE $\hat{\delta} : \mathbb{R}^n \times \mathbb{R}^L \times \mathbb{R}^L \to \mathbb{R}$ defined as

$$\hat{\delta}\left(x, \widehat{W}_c, \widehat{W}_a\right) \triangleq Q(x) + \hat{u}\left(x, \widehat{W}_a\right)^\top R \hat{u}\left(x, \widehat{W}_a\right) + \nabla \widehat{V}\left(x, \widehat{W}_c\right)\left(\Phi\left(x, \hat{\theta}\right) + g(x)\,\hat{u}\left(x, \widehat{W}_a\right)\right). \tag{5–34}$$

The BE represents the difference between the actor and critic weight approximations and their ideal weight values. While (5–34) is used for implementation, to facilitate the subsequent stability analysis, (5–34) can be rewritten in terms of the weight approximation errors $\widetilde{W}_c \triangleq W - \widehat{W}_c$ and $\widetilde{W}_a \triangleq W - \widehat{W}_a$. Subtracting (5–28) from (5–34) and substituting (5–30)-(5–31), the analytical form of the BE in (5–34) can be expressed as

$$\hat{\delta}\left(x, \widehat{W}_c, \widehat{W}_a\right) = -\omega^\top \widetilde{W}_c + \frac{1}{4}\widetilde{W}_a^\top G_\phi(x)\,\widetilde{W}_a + O(x), \tag{5–35}$$

where the Bellman regressor $\omega : \mathbb{R}^n \times \mathbb{R}^L \to \mathbb{R}^n$ is $\omega\left(x, \widehat{W}_a\right) \triangleq \nabla\phi(x)\left(\Phi\left(x, \hat{\theta}\right) + g(x)\,\hat{u}\left(x, \widehat{W}_a\right)\right)$ and $O(x) \triangleq \frac{1}{2}\nabla\epsilon(x)\,G_R\nabla\phi(x)^\top W + \frac{1}{4}G_\epsilon - \nabla\epsilon(x)\,f(x)$, where $G_R(x) \triangleq g(x)\,R^{-1}g(x)^\top$, $G_\phi(x) \triangleq \nabla\phi(x)\,G_R(x)\,\nabla\phi(x)^\top$, and $G_\epsilon(x) \triangleq \nabla\epsilon(x)\,G_R(x)\,\nabla\epsilon(x)^\top$. The function G_R is bounded as $\sup_{x\in\Omega}\|G_R\| \leq \overline{g}^2\lambda_{\max}\{R^{-1}\} \triangleq \overline{G_R}$, and G_ϕ is bounded as $\sup_{x\in\Omega}\|G_\phi\| \leq \left(\overline{\nabla\phi}\overline{g}\right)^2\lambda_{\max}\{R^{-1}\} \triangleq \overline{G}_\phi$.

The BE in (5–34) can be evaluated at any user-defined point in the state space using a user-selected state x_i, the critic weight estimate $\widehat{W}_c$, and the actor weight estimate $\widehat{W}_a$. Using the DNN system identifier and adaptive update laws developed in Section 5.3, experience can be simulated by extrapolating the BE over unexplored off-trajectory points in the state space via BE extrapolation. BE extrapolation uses the estimated dynamics to yield simultaneous exploration and exploitation providing simulation of experience and yielding faster policy learning. To gain experience for sufficient exploration, the BE is extrapolated to user-defined off-trajectory points $\{x_i : x_i \in \Omega\}_{i=1}^N$, where $N \in \mathbb{N}$ is a user-specified number of total extrapolation trajectories in the compact set Ω [92].

As the estimate of the identified dynamics becomes more accurate, the estimate of the control policy becomes more accurate.

5.4.3 Update Laws for Actor and Critic Weights

The experience gained along the state trajectory and from the extrapolated points is used to update the actor and critic weights simultaneously. In the subsequent adaptive weight update laws, $\eta_{c1},\ \eta_{c2},\ \eta_{a1},\ \eta_{a2},\ \lambda \in \mathbb{R}_{>0}$ are positive constant adaptation gains, $\rho = 1 + \nu\omega^\top\Gamma\omega$, $\rho_i = 1 + \nu\omega_i^\top\Gamma\omega_i$, $\nu \in \mathbb{R}_{>0}$ is a user-defined gain, $\Gamma \in \mathbb{R}^{L\times L}$ is a time-varying least-squares gain matrix, and $\underline{\Gamma},\ \overline{\Gamma} \in \mathbb{R}_{>0}$ denote lower and upper bounds for Γ. The normalized regressors $\frac{\omega}{\rho}$ and $\frac{\omega_i}{\rho_i}$ are bounded as $\sup_{t\in\mathbb{R}_{\geq 0}} \left\|\frac{\omega}{\rho}\right\| \leq \frac{1}{2\sqrt{\nu\underline{\Gamma}}}$ 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 $x \in \Omega$ and $x_i \in \Omega$, respectively. The critic update law $\dot{\widehat{W}}_c \in \mathbb{R}^L$ is defined as

$$\dot{\widehat{W}}_c \triangleq -\eta_{c1}\Gamma\frac{\omega}{\rho}\hat{\delta} - \eta_{c2}\Gamma\frac{1}{N}\sum_{i=1}^{N}\frac{\omega_i}{\rho_i}\delta_i. \tag{5–36}$$

The least-squares gain matrix update law $\dot{\Gamma} \in \mathbb{R}^{L\times L}$ is defined as

$$\dot{\Gamma} \triangleq \left(\lambda\Gamma - \eta_{c1}\frac{\Gamma\omega\omega^\top\Gamma}{\rho^2} - \frac{\eta_{c2}\Gamma}{N}\sum_{i=1}^{N}\frac{\omega_i\omega_i^\top\Gamma}{\rho_i^2}\right)\cdot \mathbf{1}_{\left\{\underline{\Gamma}\leq\|\Gamma\|\leq\overline{\Gamma}\right\}}, \tag{5–37}$$

where $\mathbf{1}_{\{\cdot\}}$ denotes the indicator function ensuring that $\underline{\Gamma} \leq \|\Gamma\| \leq \overline{\Gamma}$ for all $t \in \mathbb{R}_{>0}$. The actor update law $\dot{\widehat{W}}_a \in \mathbb{R}^L$ is defined as

$$\dot{\widehat{W}}_a \triangleq -\eta_{a1}\left(\widehat{W}_a - \widehat{W}_c\right) - \eta_{a2}\widehat{W}_a + \frac{\eta_{c1}G_\phi^\top\widehat{W}_a\omega^\top}{4\rho}\widehat{W}_c + \eta_{c2}\frac{1}{N}\sum_{i=1}^{N}\frac{G_{\phi i}^\top\widehat{W}_a\omega_i^\top}{4\rho_i}\widehat{W}_c. \tag{5–38}$$

The following assumption aids in the subsequent stability analysis by imposing a condition on sufficient richness of the Bellman regressor ω.

Assumption 5.5. On the compact set, Ω, a finite set of off-trajectory points $\{x_i : x_i \in \Omega\}_{i=1}^N$ are user-selected such that $0 < \underline{c} \triangleq \inf_{t\in\mathbb{R}_{\geq 0}} \lambda_{\min}\left\{\frac{1}{N}\sum_{i=1}^{N}\frac{\omega_i\omega_i^\top}{\rho_i^2}\right\}$, where $\underline{c}$ is a constant scalar lower bound of the value of each history stack's minimum eigenvalues [92].

5.5 Stability Analysis

To facilitate the stability analysis, let a concatenated state $z \in \mathbb{R}^{n+2L}$ be defined as $z \triangleq \left[x^\top, \widetilde{W}_c^\top, \widetilde{W}_a^\top\right]^\top$, and let the candidate Lyapunov function $V_L : \mathbb{R}^{n+2L} \to \mathbb{R}_{\geq 0}$ be defined as

$$V_L(z) \triangleq V^*(x) + \frac{1}{2}\widetilde{W}_c^\top \Gamma^{-1} \widetilde{W}_c + \frac{1}{2}\widetilde{W}_a^\top \widetilde{W}_a. \tag{5–39}$$

According to [101, Lemma 4.3], (5–39) can generally be bounded as

$$\underline{v_l}(\|z\|) \leq V_L(z) \leq \overline{v_l}(\|z\|) \tag{5–40}$$

using class $\mathcal{K}$ functions $\underline{v_l}, \overline{v_l} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. To facilitate the subsequent analysis, let $v_l(\|z\|) = \frac{1}{2}\underline{q}(\|x\|) + \frac{1}{12}\eta_{c2}\underline{c}\left\|\widetilde{W}_c\right\|^2 + \frac{1}{16}(\eta_{a1} + \eta_{a2})\left\|\widetilde{W}_a\right\|^2$ and define $r \in \mathbb{R}_{>0}$ to be the prescribed radius of a compact ball $\mathcal{B}_r \in \mathbb{R}^{n+2L}$ centered at the origin where the convergence of z is desired.

Theorem 5.2. *Taking the time derivative of (5–39) yields*

$$\dot{V}_L \overset{a.a.t.}{\leq} \nabla V^* \dot{x} - \widetilde{W}_c^\top \Gamma^{-1} \dot{\widehat{W}}_c - \widetilde{W}_a^\top \dot{\widehat{W}}_a - \frac{1}{2}\widetilde{W}_c^\top \Gamma^{-1} \dot{\Gamma} \Gamma^{-1} \widetilde{W}_c. \tag{5–41}$$

Provided the weight update laws in (5–36)-(5–38) are implemented, Assumptions 5.1-5.5, as well as 2.1 and 2.2, hold, and the conditions

$$\eta_{a1} + \eta_{a2} > \frac{1}{\sqrt{\nu \underline{\Gamma}}}(\eta_{c1} + \eta_{c2})\overline{WG_\phi} \tag{5–42}$$

$$\underline{c} > 3\frac{\eta_{a1}}{\eta_{c2}} + \frac{3(\eta_{c1} + \eta_{c2})^2 \overline{W}^2}{8\eta_{c2}\nu\underline{\Gamma}}\left(\frac{\overline{G_\phi}^2}{2(\eta_{a1} + \eta_{a2})}\right) \tag{5–43}$$

$$l < v_l\left(\overline{v_l}^{-1}\left(\underline{v_l}(r)\right)\right) \tag{5–44}$$

$$\|z(0)\| \leq \overline{v_l}^{-1}\left(\underline{v_l}(r)\right) \tag{5–45}$$

are satisfied, where l is a positive constant that depends on the NN bounding constants in Assumption 5.4, then x, $\widetilde{W}_c$, and $\widetilde{W}_a$ are UUB. Hence, each control policy $\hat{u}$ converges to a neighborhood of its respective optimal control policy u^*.

Proof. Using the HJB equation in (5–28), the BE in (5–35), the gain conditions in (5–42) and (5–43), and the weight update laws in (5–36)-(5–38), the time derivative of (5–39) can be bounded as

$$\dot{V}_L \leq -v_l\left(\|z\|\right) \;\; \forall \|z\| \geq v_l^{-1}\left(l\right) \tag{5–46}$$

for all $t \in \mathbb{R}_{>0}$. Using (5–40) and (5–46), [137, Theorem 4.18] can be invoked to conclude that every trajectory $z\left(t\right)$ that satisfies the initial condition $\|z\left(0\right)\| \leq \overline{v_l}^{-1}\left(\underline{v_l}\left(r\right)\right)$ is bounded for all $t \in \mathbb{R}$, z is UUB such that $\limsup_{t\to\infty}\|z\| \leq \underline{v_l}^{-1}\left(\overline{v_l}\left(v_l^{-1}\left(l\right)\right)\right)$, and the control policy $\hat{u}$ converges to a neighborhood of the optimal control policy u^*. Since $z \in \mathcal{L}_\infty$, it follows that $x, \widetilde{W}_c, \widetilde{W}_a \in \mathcal{L}_\infty$; hence, $x, \widehat{W}_c, \widehat{W}_a \in \mathcal{L}_\infty$ and $u \in \mathcal{L}_\infty$. Additionally, every trajectory z that is initialized in the ball $\mathcal{B}_r$ is bounded such that $z \in \mathcal{B}_r$, $\forall t \in \mathbb{R}_{\geq 0}$. Since $z \in \mathcal{B}_r$, the states $x, \widehat{W}_c, \widehat{W}_a$ similarly lie in a compact set. □

5.6 Simulations

To demonstrate the effectiveness of the developed ADP technique, comparative simulations are performed on a control-affine nonlinear dynamical system with a two dimensional state $x = \left[x_1, x_2\right]^\top$. The developed method results are compared with the multi-timescale DNN technique in [113] as the baseline. For the baseline method, the inner-layer weights are retrained and updated once online, and the mean squared error is used as the loss function for training.

For value function approximation, the basis function is selected as $\phi = \left[x_1^2, x_1x_2, x_2^2\right]$. The initial conditions for the system are $x\left(0\right) = \left[-5, 5\right]^\top$, $\Gamma\left(0\right) = 10 \cdot I_{3\times 3}$, and $\widehat{W}_c\left(0\right) =$

$\widehat{W}_a(0) = 0.1 \cdot \mathbf{1}_{3\times 1}$. The system dynamics are

$$f = \begin{bmatrix} x_1 & x_2 & 0 & 0 \\ 0 & 0 & x_1 & x_2\left(1 - (cos(2x_1) + 2)^2\right) \end{bmatrix} \theta, \qquad g = \begin{bmatrix} 0 \\ cos(2x_1) + 2 \end{bmatrix}$$

where θ =$[-1, 1, -0.5, -0.5]^\top$ [135]. The simulation parameters are selected as $\eta_{c1} = 0.005$, $\eta_{c2} = 0.1$, $\eta_{a1} = 15$, $\eta_{a2} = 0.1$, $\lambda = 0.4$, $\nu = 0.005$, $N = 100$, $\Gamma_\theta = 0.02 \cdot I_{L\times L}$, $\alpha_1 = 30$, $k_f = 100$, $\beta_f = 0.2$. The cost parameters in (5–4) are selected as $Q = x^T \text{diag}([.001,\ 3])\, x$ and $R = 1$. The implemented DNN contains 7 hidden layers with 7 neurons in each layer.

Figure 5-1 presents the state errors of the infinite horizon regulation problem. It is shown that using the developed system identifier to learn the dynamics in real-time successfully yields faster convergence of the system states. The state errors rapidly converged to steady state at approximately 5 seconds, whereas it took approximately 25 seconds for the state errors to converge with the baseline method.

Figure 5-2 shows the comparative plots of the RMS function approximation error norm with the developed and baseline method. The initial function approximation error is high with both methods because the DNN weights are randomly initialized. There is an initial overshoot in the DNN weight estimates due to the initial learning phase resulting from the exploration of the state-space by the system states. The initial overshoot is higher with the developed method since it updates all of the DNN weights, unlike the baseline method which updates only the output-layer weights. However, the simultaneous update on all weights results in a rapid function approximation error convergence with the developed method, i.e., within 5 seconds. In contrast, the baseline method did not yield function approximation error convergence for the duration of the simulation. The improved learning of the dynamics is beneficial to the ADP framework as it provides a more accurate model to be used in BE extrapolation which results in faster convergence to the optimal control policy as shown in Figure 5-3.

The aforementioned control objective is to find an optimal control policy u that minimizes the cost function. Figure 5-3 shows that the control policy is minimized faster in the developed method. Although the initial overshoot is higher, due to the initial learning phase, the developed method converged to steady state at approximately 3 seconds while the baseline method converged to steady state at approximately 15 seconds.

Table 5-1 provides a quantitative representation of the comparative simulations. The RMS regulation error, controller error, and function approximation error are shown decrease by 65.73%, 31.82%, and 78.97%, respectively, when using the developed method compared to the baseline. The steady-state RMS error values and the percent improvement values show that the developed method yields significant improvement.

5.7 Concluding Remarks

The developed method uses an adaptive Lb-DNN system identifier in conjunction with a RISE-based dynamics observer within an ADP framework. A least-squares continuous-time update law is used to update all layers of DNN weights online. The system identifier is used to obtain an estimate of the unknown system dynamics. Exponential convergence to a neighborhood of the DNN weight estimation error, provided the Jacobian of the DNN satisfies the PE condition, is shown via a Lyapunov-based stability analysis. The entire system is shown to be UUB such that the developed control policy is shown to converge to a neighborhood of the optimal control policy. Simulation results show that the adaptive DNN yields 65.73% improved RMS regulation error, 31.82% improved controller error, and 78.97% improved function approximation error in comparison to the previously developed multi-timescale DNN. For future work, the all-layer updates developed in this chapter can be combined with insight from Chapter 4 to develop all-layer update laws for the actor and critic weight estimates.

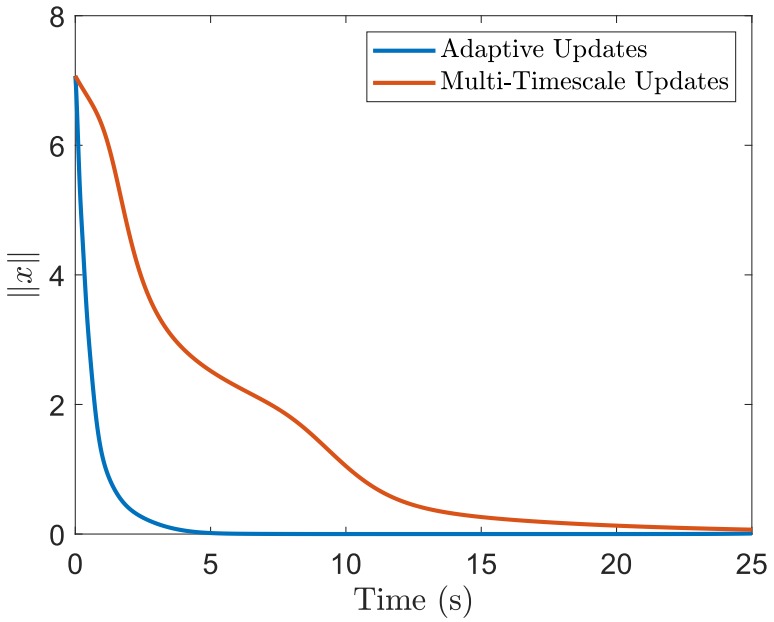

Figure 5-1. Comparative plots of the regulation error norms $\|x\|$ for the developed method consisting of adaptive updates of all the DNN layers compared to the previous method consisting of multi-timescale updates of the DNN.

Table 5-1. Performance Comparison

	Multi-timescale	Adaptive	% Decrease
$\|x\|_{\text{RMS}}$	2.265	0.800	65.73
$\|u\|_{\text{RMS}}$	1.525	1.043	31.82
$\left\|f(x,\dot{x}) - \Phi\left(X,\hat{\theta}\right)\right\|_{\text{RMS}}$	8.732	1.950	78.97

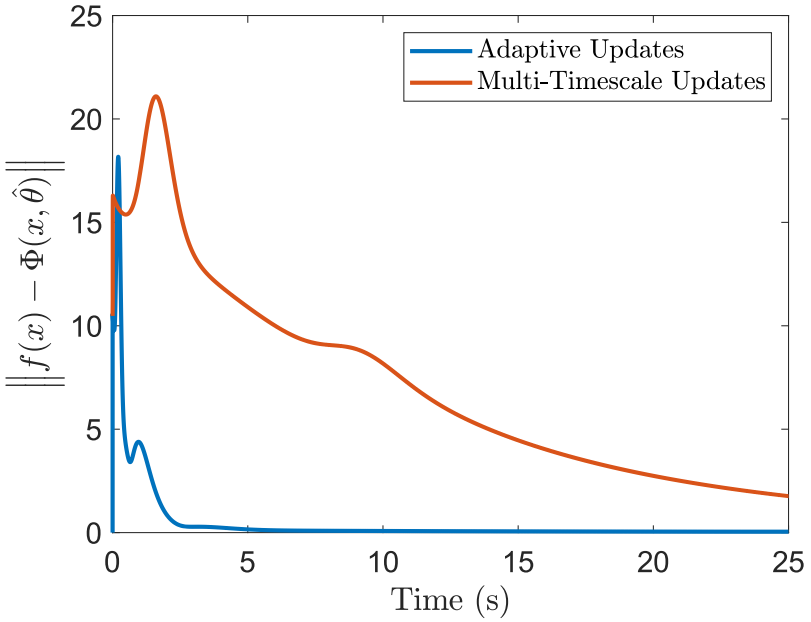

Figure 5-2. Comparative plots of the RMS function approximation error norm $\left\|f(x) - \Phi\left(X, \hat{\theta}\right)\right\|$ for the developed method consisting of adaptive updates of all the DNN layers compared to the previous method consisting of multi-timescale updates of the DNN.

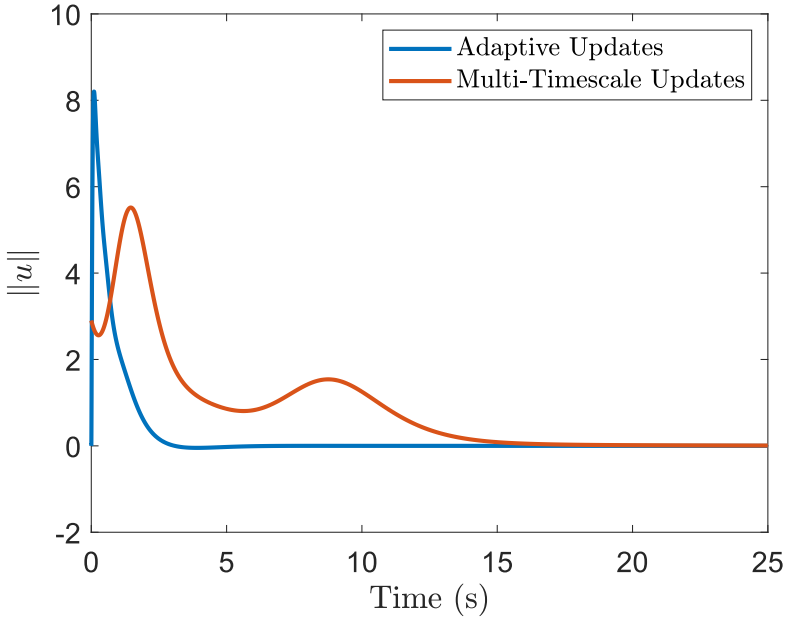

Figure 5-3. Comparative plots of the control input $\|u\|$ for the developed method consisting of adaptive updates of all the DNN layers compared to the previous method consisting of multi-timescale updates of the DNN.

CHAPTER 6
CONCLUSION

ADP is a powerful tool that leverages reinforcement learning and adaptive control techniques to solve optimal control problems. ADP can be used for learning parametric uncertainties and approximating optimal control policies in nonlinear dynamical systems. This dissertation develops methods that advance the state-of-the-art in ADP-based controllers in the presence of model uncertainty. First, linear parameterization is used to estimate unknown system dynamics while switching between controllers. Then, the system identification technique is advanced to using a multi-timescale DNN to estimate the unknown dynamics between a pursuing and evading agent to facilitate indirect herding to a desired goal location. Then, the value function approximation technique is advanced to using a multi-timescale DNN to approximate the optimal value function in conjunction with a multi-timescale DNN used to approximate the unknown dynamics. Then, the multi-timescale DNN is advanced to an adaptive DNN that updates all the weights in real-time to be used with a RISE-based dynamics observer to estimate unknown dynamics.

Chapter 2 provides an HRL technique to control switching between approximately optimal controllers. The hierarchical framework identifies which controller should be active at a given time and generates a switching signal indicating the most desirable switching pattern based on comparing multiple value function estimates. This chapter is impactful because previous results contained unsupervised switching, and now supervised switching can be used to achieve optimality by using a hierarchy to optimize a selected performance method. This technique is especially important in the presence of unknown dynamics where the control policy that minimizes the system cost is unknown *a priori*. UUB regulation of the system states to a neighborhood of the origin, and convergence of the approximate control policy to a neighborhood of the optimal control policy, are proven using a Lyapunov-based stability and dwell-time analysis.

Simulation results show that implementing the developed HRL controller yields a total cost that is 37% less than the total cost of implementing one ADP sub-controller and an improved rise time. Future work will consist of implementing the HRL framework for the optimal tracking problem.

Chapter 2 will be significant for problems that require more than one cost function to achieve the desired control objective. This chapter provides inroads for ADP become more integrated into switched system/hybrid controls problems Additionally, HRL can be used in the class of problems introduced in Chapter 3 to facilitate multi-agent herding. The HRL technique can be used to switch between different pursuers to regulate an evader or multiple evaders based on specific metrics (e.g. distance between agents) to the desired goal location.

Advancing the ICL result in Chapter 2, Chapter 3 develops a deep ICL-based implementation of ADP to achieve an approximate optimal online solution to the indirect regulation herding problem for unknown agents. The contribution of this chapter lies in the fact that this is the first time ICL has been used in a DNN system identifier framework, successfully removing the need to measure the state derivatives. The ICL-based system identifier is facilitated by an Lb-DNN to estimate the unknown interaction dynamic between the pursuer and evader. A Lyapunov-based analysis is provided to prove UUB convergence of the evader to the desired goal location known by the pursuer. The simulation shows that the pursuer is able to intercept and regulate the evader towards the desired goal location and that the Lb-DNN system identifier outperforms the SNN system identifier.

Future work for Chapter 3 consists of extending this result to problems with multiple pursuers, multiple evaders, or both. Additionally, since the target agent in this chapter is moving agnostically, future research directions will include more complex dynamical agent interactions such as the evader optimally evading the pursuer or an asymmetric interaction between the pursuer and evader where the evader is not just influenced

forward in a straight line. Lastly, this chapter can be extended to obstacle avoidance herding problems consisting of avoidance regions characterized by penalty functions or control barrier functions.

Using the multi-timescale DNN introduced in Chapter 3, Chapter 4 develops a framework for using DNN-based value function approximation within the ADP framework to solve the infinite-horizon optimal tracking problem online. In existing ADP literature, while DNNs have been used for online system identification, the optimal value function has only been approximated with a single-layer NN, therefore the contribution of this chapter is using a multi-timescale DNN to approximate the optimal value function in real-time in an attempt to achieve a more accurate approximation. The approximation of the optimal control policy is proven to converge to a neighborhood of the optimal control policy, and UUB stability of the states is proven via a Lyapunov-based stability analysis. The simulation results show that the deep value function approximation method results in 95.04% improvement in BE minimization and 5.06% faster convergence.

Improving upon the function approximation capabilities discussed in Chapters 3 and 4, Chapter 5 provides the first ADP result with Lyapunov-derived weight adaptation laws for each layer of a DNN online. The developed method uses an adaptive Lb-DNN system identifier in conjunction with a RISE-based dynamics observer within the ADP framework. A least-squares continuous-time update law is used to update all layers of DNN weights online. The system identifier is used to obtain an estimate of the unknown system dynamics. UUB convergence of the DNN weight estimation error, provided the Jacobian of the DNN satisfies the PE condition, is shown via a Lyapunov-based stability analysis. The entire system is shown to be UUB such that the developed control policy is shown to converge to a neighborhood of the optimal control policy. Simulation results show that the adaptive DNN yields 65.73% improved RMS regulation error, 31.82% improved controller error, and 78.97% improved function approximation error in comparison to the previously developed multi-timescale DNN.

While a multi-timescale DNN where the output-layer weights update in real-time is used for value function approximation in Chapter 4, and an adaptive DNN where the weights in all layers update in real-time is used for system identification in Chapter 5, the parametric estimate for value function approximation has never consisted of all layers updating in real-time. Because of the breakthroughs in Chapters 4 and 5, future work could consist of investigating the use of online adaptation for each layer of the DNN to approximate the optimal value function. Significant efforts will be required to combine the results in the aforementioned chapters. With the large influx of machine learning and reinforcement learning in the world over the past decades, the findings in this dissertation will have an impactful role in the future of control for autonomous systems.

REFERENCES

[1] J. Si, A. Barto, W. Powell, and D. Wunsch, eds., *Handbook of Learning and Approximate Dynamic Programming*. Wiley-IEEE Press, 2004.

[2] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multi-agent reinforcement learning," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 38, no. 2, pp. 156–172, 2008.

[3] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, 2010.

[4] M. P. Deisenroth, *Efficient reinforcement learning using Gaussian processes*. KIT Scientific Publishing, 2010.

[5] M. E. Harmon, L. C. Baird, and A. H. Klopf, "Reinforcement learning applied to a differential game," *Adaptive Behavior*, vol. 4, no. 1, pp. 3–28, 1995.

[6] J. Izawa, T. Kondo, and K. Ito, "Biological arm motion through reinforcement learning," *Biol Cybern*, vol. 91, no. 1, pp. 10–22, 2004.

[7] T. Jaakkola, S. Singh, and M. Jordan, "Reinforcement learning algorithm for partially observable markov decision problems," *Adv Neural Inf Process Syst*, pp. 345–352, 1995.

[8] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[9] S. Kalyanakrishnan and P. Stone, "Batch reinforcement learning in a complex domain," in *Proc. Int. Conf. Auton. Agents Multi-Agent Syst.*, (Honolulu, HI), pp. 650–657, 2007.

[10] T. Landelius, *Reinforcement learning and Distributed Local Model Synthesis*. PhD thesis, Linkoping University, Sweden, 1997.

[11] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, no. 3-4, pp. 293–321, 1992.

[12] M. Littman, "Value-function reinforcement learning in markov games," *Cogn. Syst. Res.*, vol. 2, no. 1, pp. 55–66, 2001.

[13] A. Mellouk, ed., *Advances in Reinforcement Learning*. InTech, 2011.

[14] A. Schwartz, "A reinforcement learning method for maximizing undiscounted rewards," in *Proc. Int. Conf. Mach. Learn.*, vol. 298, pp. 298–305, 1993.

[15] S. P. Singh, "Reinforcement learning with a hierarchy of abstract models," in *AAAI Natl. Conf. Artif. Intell.*, vol. 92, pp. 202–207, 1992.

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

[17] P. J. Werbos, "Back propagation: past and future," in *Proc. Int. Conf. Neural Netw.*, vol. 1, pp. 1343–1353, 1989.

[18] P. Werbos, "A menu of designs for reinforcement learning over time," *Neural Netw. for Control*, pp. 67–95, 1990.

[19] 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.

[20] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press, 1 ed., 1957.

[21] J. Y. Lee, J. B. Park, and Y. H. Choi, "On integral generalized policy iteration for continuous-time linear quadratic regulations," *Autom.*, vol. 50, pp. 475–489, 2014.

[22] Q. Lin, Q. Wei, and D. Liu, "A novel optimal tracking control scheme for a class of discrete-time nonlinear systems using generalised policy iteration adaptive dynamic programming algorithm," *International Journal of Systems Science*, vol. 48, pp. 525 – 534, 2017.

[23] D. Liu, Q. Wei, and P. Yan, "Generalized policy iteration adaptive dynamic programming for discrete-time nonlinear systems," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 45, no. 12, pp. 1577–1591, 2015.

[24] D. L. Vrabie and F. L. Lewis, "Generalized policy iteration for continuous-time systems," *2009 International Joint Conference on Neural Networks*, pp. 3224–3231, 2009.

[25] R. Howard, *Dynamic programming and Markov processes*. Technology Press of Massachusetts Institute of Technology (Cambridge), 1960.

[26] R. Leake and R. Liu, "Construction of suboptimal control sequences," *SIAM J. Control*, vol. 5, p. 54, 1967.

[27] S. Bradtke, B. Ydstie, and A. Barto, "Adaptive linear quadratic control using policy iteration," in *Proc. Am. Control Conf.*, pp. 3475–3479, IEEE, 1994.

[28] 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.

[29] S. K. Jha and S. Bhasin, "On-policy q-learning for adaptive optimal control," in *Proc. IEEE Symp. Adapt. Dyn. Progr. Reinf. Learn.*, pp. 1–6, Dec. 2014.

[30] M. Palanisamy, H. Modares, F. L. Lewis, and M. Aurangzeb, "Continuous-time q-learning for infinite-horizon discounted cost linear quadratic regulator problems," *IEEE Trans. Cybern.*, vol. 45, pp. 165–176, Feb. 2015.

[31] L. Baird, "Advantage updating," tech. rep., Wright Lab, Wright-Patterson Air Force Base, OH, 1993.

[32] C. J. C. H. Watkins, *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

[33] C. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, 1992.

[34] F. L. Lewis and V. L. Syrmos, *Optimal Control*. Wiley, 2 ed., 1995.

[35] E. Hofer and B. Tibken, "An iterative method for the finite-time bilinear-quadratic control problem," *J Optim Theory Appl*, vol. 57, no. 3, pp. 411–427, 1988.

[36] Z. Aganovic and Z. Gajic, "The successive approximation procedure for finite-time optimal control of bilinear systems," *IEEE Trans. Autom. Control*, vol. 39, no. 9, pp. 1932–1935, 1994.

[37] W. Cebuhar and V. Costanza, "Approximation procedures for the optimal control of bilinear and nonlinear systems," *Journal of Optimization Theory and Applications*, vol. 43, no. 4, pp. 615–627, 1984.

[38] O. Rosen and R. Luus, "Global optimization approach to nonlinear optimal control," *Journal of Optimization Theory and Applications*, vol. 73, no. 3, pp. 547–562, 1992.

[39] E. Al'Brekht, "On the optimal stabilization of nonlinear systems," *J. Appl. Math. Mech.*, vol. 25, no. 5, pp. 1254–1266, 1961.

[40] W. L. Garrard and J. M. Jordan, "Design of nonlinear automatic flight control systems," *Automatica*, vol. 13, no. 5, pp. 497–505, 1977.

[41] D. L. Lukes, "Optimal regulation of nonlinear dynamical systems," *SIAM J. Control*, vol. 7, no. 1, pp. 75–100, 1969.

[42] Y. Nishikawa, N. Sannomiya, and H. Itakura, "A method for suboptimal design of nonlinear feedback systems," *Automatica*, vol. 7, no. 6, pp. 703–712, 1971.

[43] I. C. Dolcetta, "On a discrete approximation of the hamilton-jacobi equation of dynamic programming," *Appl. Math. Optim.*, vol. 10, no. 1, pp. 367–377, 1983.

[44] M. Falcone and R. Ferretti, "Discrete time high-order schemes for viscosity solutions of hamilton-jacobi-bellman equations," *Numer. Math.*, vol. 67, no. 3, pp. 315–344, 1994.

[45] R. Gonzalez and E. Rofman, “On deterministic control problems: An approximation procedure for the optimal cost i. the stationary problem,” *SIAM J. Control Optim.*, vol. 23, no. 2, pp. 242–266, 1985.

[46] R. Gonzalez and E. Rofman, “On deterministic control problems: An approximation procedure for the optimal cost ii. the nonstationary case,” *SIAM J Control Optim*, vol. 23, no. 2, pp. 267–285, 1985.

[47] M. Falcone, “A numerical approach to the infinite horizon problem of deterministic control theory,” *Appl. Math. Optim.*, vol. 15, no. 1, pp. 1–13, 1987.

[48] R. W. Beard and T. W. Mclain, “Successive galerkin approximation algorithms for nonlinear optimal and robust control,” *Int. J. Control*, vol. 71, no. 5, pp. 717–743, 1998.

[49] R. W. Beard, G. N. Saridis, and J. T. Wen, “Approximate solutions to the time-invariant hamilton–jacobi–bellman equation,” *J Optim Theory Appl*, vol. 96, no. 3, pp. 589–626, 1998.

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

[51] D. Pandey and P. Pandey, “Approximate q-learning: An introduction,” in *Int. Conf. Mach. Learn. Comput*, pp. 317–320, IEEE, 2010.

[52] Y. Li, C. Yang, Z. Hou, Y. Feng, and C. Yin, “Data-driven approximate q-learning stabilization with optimality error bound analysis,” *Automatica*, vol. 103, pp. 435–442, 2019.

[53] V. Konda and J. Tsitsiklis, “On actor-critic algorithms,” *SIAM J. Control Optim.*, vol. 42, no. 4, pp. 1143–1166, 2004.

[54] R. J. Williams, “Toward a theory of reinforcement-learning connectionist systems,” Tech. Rep. NU-CCS-88-3, Northeastern University, College of Computer Science, 1988.

[55] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Mach. Learn*, vol. 8, no. 3, pp. 229–256, 1992.

[56] H. Kimura, K. Miyazaki, and S. Kobayashi, “Reinforcement learning in pomdps with function approximation,” in *Proc. Int. Conf. Mach. Learn.*, vol. 97, pp. 152–160, 1997.

[57] R. Sutton, “Learning to predict by the methods of temporal differences,” *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, 1988.

[58] K. S. Fu, “Learning control systems,” in *Comput. Inf. Sci. Collect. Pap. Learn. Adapt. Control Inf. Syst.* (J. T. Tou and R. H. Wilcox, eds.), (Washington), pp. 318–343, Spartan Books, 1964.

[59] K. S. Fu, “Learning control systems,” in *Advances in Information Systems Science: Volume 1* (J. T. Tou, ed.), pp. 251–292, Boston, MA: Springer US, 1969.

[60] B. Widrow, N. Gupta, and S. Maitra, “Punish/reward: Learning with a critic in adaptive threshold systems,” *IEEE Trans. Syst. Man Cybern.*, vol. 3, no. 5, pp. 455–465, 1973.

[61] H. Zhang, K. Zhang, G. Xiao, and H. Jiang, “Robust optimal control scheme for unknown constrained-input nonlinear systems via a plug-n-play event-sampled critic-only algorithm,” *IEEE Trans. Syst. Man and Cybern*, vol. 50, no. 9, pp. 3169–3180, 2019.

[62] H. Dong, X. Zhao, and H. Yang, “Reinforcement learning-based approximate optimal control for attitude reorientation under state constraints,” *IEEE Trans Control Syst Technol*, vol. 29, no. 4, pp. 1664–1673, 2020.

[63] I. H. Witten, “An adaptive optimal controller for discrete-time markov environments,” *Inf. control*, vol. 34, no. 4, pp. 286–295, 1977.

[64] A. Barto, R. Sutton, and C. Anderson, “Neuron-like adaptive elements that can solve difficult learning control problems,” *IEEE Trans. Syst. Man Cybern.*, vol. 13, no. 5, pp. 834–846, 1983.

[65] J. Murray, C. Cox, G. Lendaris, and R. Saeks, “Adaptive dynamic programming,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 32, no. 2, pp. 140–153, 2002.

[66] X. Liu and S. Balakrishnan, “Convergence analysis of adaptive critic based optimal control,” in *Proc. Am. Control Conf.*, vol. 3, 2000.

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

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

[69] 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.

[70] D. Vrabie and F. L. Lewis, “Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems,” *Neural Netw.*, vol. 22, no. 3, pp. 237–246, 2009.

[71] T. Bian, Y. Jiang, and Z.-P. Jiang, “Adaptive dynamic programming and optimal control of nonlinear nonaffine systems,” *Automatica*, vol. 50, no. 10, pp. 2624–2632, 2014.

[72] P. Werbos, “Beyond regression: new tools for prediction and analysis in the behavioral sciences,” *Ph. D. dissertation, Harvard University*, 1974.

[73] 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.

[74] Z. Chen and S. Jagannathan, “Generalized Hamilton-Jacobi-Bellman formulation -based neural network control of affine nonlinear discrete-time systems,” *IEEE Trans. Neural Netw.*, vol. 19, pp. 90–106, Jan. 2008.

[75] 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.

[76] Y. M. Park, M. S. Choi, and K. Y. Lee, “An optimal tracking neuro-controller for nonlinear dynamic systems,” *IEEE Trans. Neural Netw.*, vol. 7, no. 5, pp. 1099–1110, 1996.

[77] Y. Luo and M. Liang, “Approximate optimal tracking control for a class of discrete-time non-affine systems based on gdhp algorithm,” in *IWACI Int. Workshop Adv. Comput. Intell.*, pp. 143–149, 2011.

[78] D. Wang, D. Liu, and Q. Wei, “Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach,” *Neurocomputing*, vol. 78, no. 1, pp. 14–22, 2012.

[79] 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.

[80] T. Dierks and S. Jagannathan, “Online optimal control of affine nonlinear discrete-time systems with unknown internal dynamics by using time-based policy update,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1118–1129, 2012.

[81] 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, 2009.

[82] T. Dierks and S. Jagannathan, “Optimal control of affine nonlinear continuous-time systems,” in *Proc. Am. Control Conf.*, pp. 1568–1573, 2010.

[83] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, “Adaptive optimal control for continuous-time linear systems based on policy iteration,” *Automatica*, vol. 45, no. 2, pp. 477–484, 2009.

[84] 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.

[85] 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.

[86] J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci. U.S.A.*, vol. 81, no. 10, p. 3088, 1984.

[87] P. M. Patre, W. MacKunis, K. Kaiser, and W. E. Dixon, "Asymptotic tracking for uncertain dynamic systems via a multilayer neural network feedforward and RISE feedback control structure," *IEEE Trans. Autom. Control*, vol. 53, no. 9, pp. 2180–2185, 2008.

[88] 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.

[89] K. G. Vamvoudakis and F. L. Lewis, "Online synchronous policy iteration method for optimal control," in *Recent Advances in Intelligent Control Systems* (W. Yu, ed.), pp. 357–374, London, UK: Springer, 2009.

[90] 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.

[91] 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.

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

[93] 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.

[94] G. Chowdhary, *Concurrent Learning for Convergence in Adaptive Control without Persistency of Excitation*. PhD thesis, Georgia Institute of Technology, Dec. 2010.

[95] G. Battistelli, J. Hespanha, and P. Tesi, "Supervisory control of switched nonlinear systems," *Int. J. Adapt. Control Signal Process.*, vol. 26, no. 8, pp. 723–738, 2012.

[96] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corp., 1971.

[97] J. Raisch and S. D. O'Young, "Discrete approximation and supervisory control of continuous systems," *IEEE Trans. Autom. Control*, vol. 43, no. 4, pp. 569–573, 1998.

[98] V. Pantelic and M. Lawford, "Optimal supervisory control of probabilistic discrete event systems," *IEEE Trans. Autom. Control*, vol. 57, no. 5, pp. 1110–1124, 2012.

[99] S. van Dooren, A. Amstutz, and C. H. Onder, "A causal supervisory control strategy for optimal control of a heavy-duty diesel engine with scr aftertreatment," *Control. Eng. Pract.*, vol. 119, p. 104982, 2022.

[100] G. Jing, H. Bai, J. George, and A. Chakrabortty, "Model-free optimal control of linear multiagent systems via decomposition and hierarchical approximation," *IEEE Control Netw. Syst.*, vol. 8, no. 3, pp. 1069–1081, 2021.

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

[102] Y. Jiang and Z.-P. Jiang, *Robust Adaptive Dynamic Programming*. John Wiley & Sons, 2017.

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

[104] 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.

[105] D. Liberzon, *Switching in Systems and Control*. Birkhauser, 2003.

[106] M. Branicky, "Multiple Lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Trans. Autom. Control*, vol. 43, pp. 475–482, 1998.

[107] 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.

[108] W. Makumi, Z. Bell, and W. E. Dixon, "Approximate optimal indirect regulation of an unknown agent with a lyapunov-based deep neural network," *IEEE Control Syst. Lett.*, vol. 7, pp. 2773–2778, 2023.

[109] P. Kachroo, S. A. Shedied, J. S. Bay, and H. Vanlandingham, "Dynamic programming solution for a class of pursuit evasion problems: the herding problem," *IEEE Trans. Syst. Man Cybern.*, vol. 31, pp. 35–41, Feb. 2001.

[110] A. D. Khalafi and M. R. Toroghi, "Capture zone in the herding pursuit evasion games," *Appl. Math. Sci.*, vol. 5, no. 39, pp. 1935–1945, 2011.

[111] S. A. Shedied, "Optimal trajectory planning for the herding problem: a continuous time model," *Int. J. Mach. Learn. Cybern.*, vol. 4, no. 1, pp. 25–30, 2013.

[112] P. Deptula, Z. Bell, F. Zegers, R. Licitra, and W. E. Dixon, "Approximate optimal influence over an agent through an uncertain interaction dynamic," *Automatica*, vol. 134, pp. 1–13, Dec. 2021.

[113] M. Greene, Z. Bell, S. Nivison, and W. E. Dixon, "Deep neural network-based approximate optimal tracking for unknown nonlinear systems," *IEEE Trans. Autom. Control*, vol. 68, no. 5, pp. 3171–3177, 2023.

[114] 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, 2022.

[115] W. Makumi, Z. Bell, and W. E. Dixon, "Cooperative approximate optimal indirect regulation of uncooperative agents with lyapunov-based deep neural network," in *AIAA SciTech*, 2024.

[116] J. Philor, W. Makumi, Z. Bell, and W. E. Dixon, "Approximate optimal indirect control of an unknown agent within a dynamic environment using a lyapunov-based deep neural network," in *Proc. Am. Control Conf.*, 2024.

[117] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[118] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[119] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[120] W. Makumi, M. Greene, Z. Bell, S. Nivison, R. Kamalapurkar, and W. E. Dixon, "Hierarchical reinforcement learning-based supervisory control of unknown nonlinear systems," in *IFAC World Congr.*, 2023.

[121] W. Makumi, M. Greene, Z. Bell, B. Bialy, R. Kamalapurkar, and W. E. Dixon, "Hierarchical reinforcement learning and gain scheduling-based control of a hypersonic vehicle," in *AIAA SciTech*, 2023.

[122] 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.

[123] R. Kamalapurkar, J. R. Klotz, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for differential graphical games," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 1, pp. 423–433, 2018.

[124] N. Wang, Y. Gao, H. Zhao, and C. K. Ahn, "Reinforcement learning-based optimal tracking control of an unknown unmanned surface vehicle," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 3034–3045, 2020.

[125] O. Patil, D. Le, M. Greene, and W. E. Dixon, "Lyapunov-derived control and adaptive update laws for inner and outer layer weights of a deep neural network," *IEEE Control Syst Lett.*, vol. 6, pp. 1855–1860, 2022.

[126] O. S. Patil, D. M. Le, E. Griffis, and W. E. Dixon, "Deep residual neural network (ResNet)-based adaptive control: A Lyapunov-based approach," in *Proc. IEEE Conf. Decis. Control*, 2022.

[127] E. Griffis, O. Patil, W. Makumi, and W. E. Dixon, "Deep recurrent neural network-based observer for uncertain nonlinear systems," in *IFAC World Congr.*, 2023.

[128] E. Griffis, O. Patil, Z. Bell, and W. E. Dixon, "Lyapunov-based long short-term memory (Lb-LSTM) neural network-based control," *IEEE Control Syst. Lett.*, vol. 7, pp. 2976–2981, 2023.

[129] E. Griffis, O. Patil, R. Hart, and W. E. Dixon, "Lyapunov-based long short-term memory (Lb-LSTM) neural network-based adaptive observer," *IEEE Control Syst. Lett.*, vol. 8, pp. 97–102, 2024.

[130] R. Hart, E. Griffis, O. Patil, and W. E. Dixon, "Lyapunov-based physics-informed long short-term memory (LSTM) neural network-based adaptive control," *IEEE Control Syst. Lett.*, vol. 8, pp. 13–18, 2024.

[131] R. Hart, O. Patil, E. Griffis, and W. E. Dixon, "Deep Lyapunov-based physics-informed neural networks (DeLb-PINN) for adaptive control design," in *Proc. IEEE Conf. Decis. Control*, 2023.

[132] O. S. Patil, E. J. Griffis, W. A. Makumi, and W. E. Dixon, "Composite adaptive lyapunov-based deep neural network (lb-dnn) controller," *arXiv preprint arXiv:2311.13056*, 2023.

[133] D. S. Bernstein, *Matrix Mathematics*. Princeton university press, 2009.

[134] 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.

[135] 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.

[136] 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.

[137] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 3 ed., 2002.

[138] M. Greene, *Nonsmooth Data-Based Reinforcement Learning for Online Approximate Optimal COntrol*. PhD thesis, University of Florida, 2022.

[139] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[140] Z. Bell, R. Sun, K. Volle, P. Ganesh, S. Nivison, and W. E. Dixon, “Target tracking subject to intermittent measurements using attention deep neural networks,” *IEEE Control Syst. Lett.*, vol. 7, pp. 379–384, 2023.

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

[142] 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.

[143] J. A. Rosenfeld, R. Kamalapurkar, and W. E. Dixon, “The state following (staf) approximation method,” *IEEE Trans. on Neural Netw. Learn. Syst.*, vol. 30, pp. 1716–1730, June 2019.

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

[145] A. Isaly, O. Patil, H. Sweatland, R. Sanfelice, and W. E. Dixon, “Adaptive safety with a rise-based disturbance observer,” *IEEE Trans. Autom. Control*, 2024.

[146] J. J. Slotine and W. Li, “Composite adaptive control of robot manipulators,” *Automatica*, vol. 25, pp. 509–519, July 1989.

[147] O. Patil, A. Isaly, B. Xian, and W. E. Dixon, “Exponential stability with RISE controllers,” *IEEE Control Syst. Lett.*, vol. 6, pp. 1592–1597, 2022.

[148] D. Shevitz and B. Paden, “Lyapunov stability theory of nonsmooth systems,” *IEEE Trans. Autom. Control*, vol. 39 no. 9, pp. 1910–1914, 1994.

BIOGRAPHICAL SKETCH

Wanjiku Aprile Makumi was born in Raleigh, North Carolina in 1997. She received her Bachelor of Science (B.S.) degree from the Joint Department of Biomedical Engineering at North Carolina State University and the University of North Carolina at Chapel Hill. In August 2020, Wanjiku joined the Nonlinear Controls and Robotics Laboratory at the University of Florida under the supervision of Dr. Warren Dixon to pursue her Ph.D. She received her Master of Science (M.S) degree in mechanical engineering in December 2021. She received the Science, Mathematics, and Research for Transformation (SMART) Scholarship in 2023. She received her Ph.D. in aerospace engineering in May 2024. Wanjiku's research focuses on using adaptive control, reinforcement learning, and deep learning to study Lyapunov-based control of nonlinear and uncertain dynamical systems.