Continuous-Time Reinforcement Learning:

New Design Algorithms with Theoretical Insights and Performance Guarantees

by

Brent A. Wallace

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved December 2023 by the
Graduate Supervisory Committee:

Jennie Si, Chair
Spring M. Berman
Dimitri P. Bertsekas
Konstantinos S. Tsakalis

ARIZONA STATE UNIVERSITY

May 2024

ABSTRACT

This dissertation discusses continuous-time reinforcement learning (CT-RL) for control of affine nonlinear systems. Continuous-time nonlinear optimal control problems hold great promise in real-world applications. After decades of development, reinforcement learning (RL) has achieved some of the greatest successes as a general nonlinear control design method. Yet as RL control has developed, CT-RL results have greatly lagged their discrete-time RL (DT-RL) counterparts, especially in regards to real-world applications. Current CT-RL algorithms generally fall into two classes: adaptive dynamic programming (ADP), and actor-critic deep RL (DRL). The first school of ADP methods features elegant theoretical results stemming from adaptive and optimal control. Yet, they have not been shown effectively synthesizing meaningful controllers. The second school of DRL has shown impressive learning solutions, yet theoretical guarantees are still to be developed. A substantive analysis uncovering the quantitative causes of the fundamental gap between CT and DT remains to be conducted.

Thus, this work develops a first-of-its kind quantitative evaluation framework to diagnose the performance limitations of the leading CT-RL methods. This dissertation also introduces a suite of new CT-RL algorithms which offers both theoretical and synthesis guarantees. The proposed design approach relies on three important factors. First, for physical systems that feature physically-motivated dynamical partitions into distinct loops, the proposed decentralization method breaks the optimal control problem into smaller subproblems. Second, the work introduces a new excitation framework to improve persistence of excitation (PE) and numerical conditioning via classical input/output insights. Third, the method scales the learning problem via design-motivated invertible transformations of the system state variables in order to modulate the algorithm learning regression for further increases in numerical

stability. This dissertation introduces a suite of (decentralized) excitable integral reinforcement learning (EIRL) algorithms implementing these paradigms. It rigorously proves convergence, optimality, and closed-loop stability guarantees of the proposed methods, which are demonstrated in comprehensive comparative studies with the leading methods in ADP on a significant application problem of controlling an unstable, nonminimum phase hypersonic vehicle (HSV). It also conducts comprehensive comparative studies with the leading DRL methods on three state-of-the-art (SOTA) environments, revealing new performance/design insights.

To my parents Anita & John Wallace,

and

my brother Ben Wallace

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Jennie Si who has guided and supported me immensely throughout my journey in graduate school. She has taught me so much, both technical and non-technical, about how to be a successful researcher and educator. She has also inspired my aspirations to one day follow in her footsteps as a professor. I will forever be indebted to her.

I am also grateful to the members of my thesis committee, Dr. Spring M. Berman, Dr. Dimitri P. Bertsekas, and Dr. Konstantinos S. Tsakalis for their continuous guidance and support. Their insights have greatly helped the success of my research. Furthermore, I would like to express my warmest thanks to Dr. Armando A. Rodriguez and Dr. Ying-Cheng Lai for their integral role in inspiring my pursuit of a PhD as undergraduate, and for all of their support in my graduate research.

I would also like to thank Drs. Brett L. Kotschwar and Steven P. Kaliszewski of the at Arizona State University (ASU) School of Mathematical and Statistical Sciences for helping foster my love of mathematics, instrumental in this work.

It has been a pleasure to work alongside my fellow colleagues at ASU – Soham Sarkar, Junmin Zhong, Ruofan Wu, Ling-Wei Kong, Kaustav Mondal, Sai Sravan Manne, Shi Lu, Abdullah Altawaitan, and many others. I am honored to have worked alongside these fellow minds to produce leading research.

Finally, I am grateful to my family and friends for their love and encouragement, especially my parents Anita and John Wallace, my brother Ben Wallace, my lifelong friends Ben Cohen, Michael Coughlin, Kathryn Chamberlin, and many more. I would not be where I am today without them.

This research was supported by the National Science Foundation (NSF) under Grants 1563921, 1808752, 2211740, and NSF Graduate Research Fellowship Grant 026257-001.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Motivation

This work addresses control of continuous-time (CT) affine nonlinear systems. Since the seminal works of Bode [9] and Nyquist [10], historically control systems literature has been developed with a focus on the CT setting. Indeed, excellent successes have been achieved in the disciplines of classical control [11, 12], nonlinear systems [13, 14], optimal control [15, 16], adaptive control [17], and robust control [18, 19] among many others, forming the basis of modern control systems today.

The method of Reinforcement Learning (RL) emerged systematically in the early 1980s [20, 21] and has since shed further insights on solving complex decision and control problems via learning from data and approximation ideas, enabling it to prove a numerically-effective approach to tackling the central "curse of dimensionality" in dynamic programming (DP) [22]. Specifically, great successes have been achieved in discrete-time reinforcement learning (DT-RL) [23–25], both in terms of theoretical results [26–37] and applications successes [38–59].

On the other hand, results in continuous-time reinforcement learning (CT-RL), are by comparison lacking in theoretical and applications successes. There remains an immense need to develop CT-RL algorithms which can overcome these limitations and synthesize real-world designs on the order of the aforementioned results in CT classical control.

Indeed, there is no shortage of motivation to effectively tackle the continuous-time nonlinear optimal control problem. From an applications standpoint, there is a wealth

of well-motivated real-world systems which are inherently continuous-time in nature. In the central fields of robotics and autonomous vehicles, for instance, systems are naturally modeled by the mechanics of Euler-Lagrange, which are coupled second-order ordinary differential equations (ODEs) [60, 61]. In wastewater treatment, influent and effluent flows are modeled as continuous fluid dynamical processes. Bacteria and substrate concentration models are based on the continuous decay regeneration theory [53]. Many chemical processes such as distillation columns [62] are also fundamentally continuous in nature and can be modeled by PDEs or nonlinear ODEs under appropriate assumptions.

Furthermore, the lack of results in CT-RL has, to date, eluded insightful and quantitative explanation by the field at large [63, 64]. Leading CT-RL focused review works; e.g., [63], tend to summarize several CT-RL works only at a high level, they do not substantively focus on underlying quantitative performance issues and important design considerations, and their evaluations do not touch upon why and how the CT-RL methods struggle to synthesize designs. An entirely novel comprehensive performance diagnosis framework and new design approaches are needed.

1.2 State of the Field, Related Work, and Challenges

Background: Dynamic Programming (DP). The origins of modern approaches to optimal control problems are rooted in the 1960s with the inception of dynamic programming (DP) by Bellman [22]. As the first conceptualization of solving challenging nonlinear control problems using recursive methods readily implementable on digital computers, DP has inspired influential works from numerous authors [15, 21, 23, 65, 66]. While researchers recognize the great potential of optimal control, the central "curse of dimensionality" has plagued the field and limited applications. Reinforcement learning (RL) emerged as a systematic method in the

early 1980s [20, 21] with the potential to combat the curse of dimensionality. RL has since become a major breakthrough for addressing key challenges in complex nonlinear control problems. The two original solution approaches to solving Bellman's principle of optimality, namely the policy iteration (PI) and value iteration (VI) algorithms [21, 67] were developed in the RL setting in the context of Markov decision processes (MDPs), and as a result many of the historic and current RL results have been developed for MDP problems. Characteristic of these formulations is to treat optimal decision and control problems stochastically in discrete state/action spaces [68].

An important branch of research work on RL for decision and control is covered under the scope of adaptive/approximate DP (ADP) [23, 65, 69, 70], which focuses on using approximation and learning to solve the optimal control problem. The works of Werbos [70–72] represent some of the earliest and most influential conceptualizations of RL in the controls setting. As has become convention throughout the community, we will henceforth use the terms RL and ADP interchangeably.

DT-RL Theoretical and Application Successes. DT-RL algorithms (cf. [24–26] for review) have demonstrated excellent stability, convergence, and approximation guarantees. They have also substantively addressed a variety of control design requirements, such as stability robustness [73], input saturation [74], and fault tolerance [75]. Additional representative theoretical works include [27–32] that are based on the PI framework, and [33–37] that are based on the VI framework. Collectively, these results address important properties of learning convergence, solution optimality, and system stability for discrete-time nonlinear systems.

Successful applications of DT-RL algorithms have provided further validation of RL as perhaps the most promising and potentially powerful solution to complex control problems. These include discrete-time deep RL and policy gradient methods

[76–80], as well as Deep Q-Networks (DQNs) [39]. Deep RL methods have successfully tackled problems in robotics applications [38], Atari games [39], and the game of Go [40, 41], to name a few. DT-RL methods have also demonstrated great successes in addressing complex continuous state and control problems. These results include energy-efficient data centers [42], aggressive ground robot position control [43, 44], power system stability enhancement [45–47], industrial process control [48, 49], Apache helicopter stabilization, tracking, and reconfiguration control [50–52], waste water treatment [53], and wearable robots to enable continuous and stable walking [54–59].

Early Continuous-Time Optimal Control Numerical Methods 1960s-1990s. Solutions to the optimal control problem can be obtained either via Pontryagin's minimum principle (a necessary condition) or by solving the Hamilton-Jacobi-Bellman (HJB) equation (a sufficient condition) [15]. Development of open-loop numerical solution methods for solving the HJB equation began in the 1960s-1970s [81–83], wherein the state and co-state equations obtained from the Hamiltonian formulation of the optimal control problem were solved numerically as the solution to an associated two-point boundary-value problem. General partial differential equation (PDE) solution methods were widely employed in solving the HJB equation in the early 1990s, when they became computationally feasible. These techniques included the method of characteristics [84], series approximation [85], and finite-difference/finite-element methods [86–88], each of which suffered from poor numerical conditioning and high memory requirements due in large part to Bellman's curse of dimensionality. In addition, the numerical methods did not address the issue of closed-loop stability, particularly for finite-iteration and finite-series truncations [89]. Successive approximation methods were developed to address dimensionality issues, such as the seminal successive Galerkin approximation (SGA) algorithm of Beard and McLain

[89–91]. The SGA method reduced the HJB equation (a nonlinear, first-order PDE) to a sequence of generalized-HJB (GHJB) equations (linear, first-order PDEs), each of which were solved approximately using Galerkin projections. The SGA method is PI-based and requires initialization by a stabilizing policy. It was one of the first to provide comprehensive closed-loop stability results, with region of attraction containing that of the initial policy [89].

ADP CT-RL 2000s-Present. The algorithms developed through to the 1990s were largely model-based offline methods. With increasing computational capacity in the 2000s-2010s came the transition to partially model-free and model-free online ADP methods [24]. It was in this period that Lewis and Abu-Khalaf published their groundbreaking work on the constrained-input nonlinear $\mathcal{H}^2$ problem [92], which began to combine the operator equation approaches of Beard *et al.* [89–91] with learning approximation ideas. These results would later be extended to the $\mathcal{H}^\infty$ problem [93] and culminated in [94]. During this time, Lewis also published a variety of other CT-RL works in areas such as fuzzy logic [95], robot manipulators [96], and power systems [97]. Vrabie and Lewis (2009) then greatly accelerated the transition to online ADP methods with the development of their seminal integral reinforcement learning (IRL) algorithm [1], a PI-based method whose policy evaluation and policy improvement steps are carried out by means of an integral equivalent of the Bellman equation. IRL does not require knowledge of system internal dynamics f, a significant formulation achievement ushering in a new era of partially model-free and model-free ADP algorithms. Shortly thereafter, Vamvoudakis and Lewis (2010) developed synchronous policy iteration (SPI) [2], which by contrast updates the actor/critic weights independently by means of modified Levenberg–Marquardt gradient-descent tuning laws. As would become the case for many subsequent algorithms, SPI requires insertion of probing noise and a persistence of excitation (PE) assumption to ensure proper learn-

ing and weight convergence. Vamvoudakis, Vrabie, and Lewis later combined SPI and IRL in [98], the resulting hybrid SPI-IRL algorithm not requiring internal dynamics f like IRL [1] and allowing for simultaneous dynamic tuning of the critic and actor networks like SPI [2]. Modares, Lewis, and Naghibi-Sistani [99] implemented SPI in conjunction with a system identifier neural network (NN) and experience replay (ER), making SPI model-free.

Towards the 2010s to present, the various works of Z.-P. Jiang *et al.* have also come to the forefront of ADP-based CT-RL developments. Y. Jiang and Z.-P. Jiang (2014) developed Robust ADP (RADP) [3], which accommodates unknown nonlinear matched and unmatched dynamical uncertainties. RADP represented one of the first CT-RL algorithms to systematically address stability robustness alongside the usual convergence and stability results. RADP collects data over a single window and performs its PI on the same data for each iteration in an off-policy fashion. As a trade-off of this clever data reuse, RADP requires injection of probing noise and a PE-like assumption similar to SPI. Jiang and Jiang would later go on to apply RADP to power systems [100, 101], sensorimotor control [102], and interconnected systems [103]. From the mid 2010s to present, Jiang has developed two other novel CT-RL algorithms. Jiang and Jiang (2015) developed global ADP (GADP) [104] for polynomial nonlinear systems. GADP features a relaxed PI algorithm and guarantees global asymptotic stability (GAS) of the closed-loop system. The authors proved that under sufficient conditions, the optimal policy is polynomial and may be solved for successively via a sum of squares (SOS) program which avoids NN approximation. Jiang and Bian (2021) then developed a first-of-its kind VI algorithm [4] for CT systems, which like its DT counterpart does not require an initial stabilizing policy. This formulation also features an impressive suite of global stability results. Unlike most ADP methodologies which are actor-critic network approaches, the developed

VI framework involves a conventional critic network in addition to a unique system Hamiltonian network. The weights for both are tuned according to a temporal-differential projection-based dynamic update.

Multiple authors besides Lewis and Jiang have made significant contributions to ADP CT-RL. Wang, Liu, and Ma (2014) [105] developed an ADP methodology for a class of uncertain CT nonlinear systems. An approximate optimal control law is designed for a nominal system, and then a robustifying term is added to ensure that the closed-loop system is UUB in the presence of dynamic uncertainties and NN approximation error. Yang, Liu, and Wang (2014) [106] addressed unknown nonlinear systems with input constraints. The critic and actor networks are trained simultaneously in real time with gradient-descent-based tuning laws reminiscent to those of SPI [2]. In a similar manner to SPI, the system states and network weights are shown to be UUB. Yang, Wunsch, and Yin (2017) [107] showed that the Hamiltonian can be used for TD-based optimal control of CT nonlinear systems. They developed a Hamiltonian-based PI algorithm and ADP framework. Yaghmaie and Braun (2019) [108] provided solution methods for input-constrained systems when the HJB equation does not admit a $\mathcal{C}^1$ solution using the method of vanishing viscosity. Li, Sun, and Tong (2019) [109] developed a fuzzy fault-tolerant optimal control algorithm for single-input-single-output (SISO) nonlinear systems. Zhao, Na, and Gao (2020) [110] proposed a robust ADP approach for nonlinear systems with unmatched uncertainties similar to RADP [3]. The robust control problem is transformed into an equivalent optimal control problem, for which a critic NN and dynamic tuning algorithm are introduced to find approximate optimal solutions.

ADP CT-RL Applications Challenges. The aforementioned ADP CT-RL methods have proven seminal to the development of the field, and they offer substantial theoretical guarantees. However, in comparison to the comprehensive real-world suc-

cesses in DT-RL discussed above, the application studies of CT-RL remain relatively limited. Survey of the leading CT-RL methods reveals that most address weight convergence, uniform value/policy approximation, and closed-loop stability. However, results in each of these three areas often require sufficient conditions (e.g., results usually require "sufficiently many" basis functions to approximate the value and policy functions, etc.) The studies performed in this work reveal that the qualitative nature of these conditions presents designers challenges in translating theoretical results to practical controller synthesis for their particular application. Another central sufficient condition commonly required by these methods is that the system states be persistently exciting (PE); that is, in response to sufficiently exciting inputs the system states can be used in system identification and thus result in learning parameter convergence. However, the algorithm results do not provide constructive methods for testing or attaining PE, which proves a key challenge when applying these methods to design learning controllers.

For the above reasons, we observe that CT-RL application works on real-world systems generally fall short of implementing the CT-RL algorithms directly. For example, [111] proposes a model-free CT-RL algorithm for attitude control of multiple quadrotors, but the quadrotor model used neglects motor dynamics and assumes an algebraic relationship between the rotor speeds and body torques. Similarly, [112] implements a VI method for wheeled ground robots, but the model has to be reduced to that of a wheel inverted pendulum system and then linearized in order to accommodate the VI algorithm. Finally, [101] and [102] implement RADP [3] on power systems and sensorimotor control, respectively, but the system dynamics are assumed to be partially or fully linear so that the nonlinear RADP algorithm [3] simplifies to solving a sequence of linear equations.

To the great credit of these ADP CT-RL works, it must be noted that the

continuous-time optimal control problem presents formidable structural challenges not present in discrete time. DT-RL techniques focus on approximately solving the Bellman equation [22] (a difference equation), as do many temporal difference (TD) methods [113, 114]. However, these TD methods do not apply to solving the CT-RL Hamilton-Jacobi-Bellman (HJB) equation [15] (a nonlinear partial differential equation (PDE)). Directly solving the HJB is more difficult from a numerics perspective.

Recent Developments in Deep CT-RL 2020s-Present. A handful of recent works in deep CT-RL have begun to synthesize designs, but these results still are limited in comparison to the substantial DT-RL successes discussed above. The concept of applying function approximation to solve the Hamilton-Jacobi-Bellman (HJB) equation traces back to the seminal work of Doya [115]. [116] develops a data-driven Q-learning approach to Kleinman's algorithm [117], but the method is limited to linear systems. The authors of [118] focus on developing PI-based solutions to solve the HJB equation. Their theoretical results may be comprehensive, but they are based on quite stringent assumptions. [119] introduces a semi-discrete version of the HJB equation, which allows for a Q-learning algorithm to use data collected in discrete time without discretizing or approximating the system dynamics. The proposed method was tested on some relatively simple OpenAI GYM benchmarks. The method is promising, but the results still need improvement to be comparable to the state-of-the-art, especially since hyperparameter tuning has a significant impact on the results. [120] introduces a model-based approach which is based on learning time differentials of environment dynamics in order to learn the optimal policy. The results are limited to when there is no environment noise. A robust model-based optimal control formulation based on fitted value iteration (FVI) is developed in [7, 8], where it is shown applicable to cart-pole and real pendulum control problems. However, even as a model-based approach, it still faces state distribution mismatch challenge and the

associated issue when scaling up to high-dimensional problems. Furthermore, the FVI frameworks require data from on the order of $\approx 10^6$ system trajectories to converge on these simple examples, and they assume the system may be instantaneously re-initialized to remain within an *a priori* state fitting grid. These deep RL algorithm properties present issues in real-world control applications (especially in the flight control applications considered in this work) where trajectory data is limited and cannot be initialized arbitrarily. As is common with deep RL, FVIs also lack the key convergence, optimality, and closed-loop stability guarantees required for mission-critical control applications, as the fitting of the FVI value function approximator is not necessarily a contraction [7, 8].

Design Limitations in Existing RL Methods of Control for Hypersonic Vehicles (HSVs). To demonstrate substantive CT-RL results developed in this work, we focus our flagship analyses on a significant unstable, nonminimum phase hypersonic vehicle (HSV) application [5, 6, 121]. Rather than developing RL control methods for general CT dynamical systems and applying them to HSVs, a number of works develop methods specifically for HSV application. However, these existing frameworks exhibit design limitations from the perspective of real-world flight control [122]. For example, [123] develops a composite RL/observer-based attitude control method for HSVs. However, the HSV model used is a simplified approximation of the standard Wang and Stengel model [5, 6, 121] wherein Mach dependencies are ignored in the aerodynamic coefficients, a significant modeling limitation in the high-Mach hypersonic regime. The neural control methods developed in [124, 125] use this same simplified model, and the model considered in the adaptive critic design (ACD) methods [126, 127] also neglects Mach dependencies. Other ADP works such as the backstepping-based neural framework [128], the feedback linearization framework [129], and the sliding mode framework [130] require access to high-order partial

derivative knowledge of the dynamics, which is restrictive in a learning setting and particularly susceptible to model uncertainty. The stability results of [123–127, 131] are relatively limited boundedness results on the approximation and tracking error. The results require that multiple complicated stability inequality conditions hold simultaneously pointwise along the closed-loop trajectories, and no constructive method is provided for ensuring that the inequalities are met. The controller structures resulting from these theoretical assumptions are highly complex, thereby preventing numerical comparability to well-established classically-based methods of flight control [5, 6, 122, 132–135] and application of classical control insights more broadly [11, 136].

Just as significantly, the existing RL-based HSV control works fail to present substantive ablation studies of the effects of system modeling error on closed-loop stability and performance of the learning control solution, either 1) only presenting results on a nominal model, or 2) presenting results on the nominal model and a single selection of error parameters. Such results are not sufficient for proving RL methods on mission-critical flight control applications, especially for high-performance flight systems such as HSVs. Furthermore, none of the leading methods present ablation results on effects of varying system initial conditions, nor do they present studies of underlying algorithm numerical properties (e.g., algorithm conditioning), both particularly vital for RL methods in which data quality concerns are paramount for proper excitation and learning solution quality [137]. In all, significantly increased standards of algorithm numerical validation are needed to make RL methods reliably applicable to flight control, and new RL evaluation frameworks tailored to aerospace systems need to be developed.

Fundamental Questions to be Addressed. Due to the illustrated gaps in knowledge between discrete-time and continuous-time in both theory and application, the first questions essential to the advancement of CT-RL are ones of diagnosis:

1). Why have CT-RL methodologies lagged so far behind their DT-RL counterparts, both in terms of theoretical/algorithm development and potential adoption by real-world applications?

2). Why are substantial CT-RL applications works yet to be demonstrated in spite of longstanding theoretical results?

It is therefore pivotal to have a comprehensive analysis focusing on CT-RL algorithm insights and inherent limitations. Once these limitations have been diagnosed, the field may move forward to questions of control synthesis:

3). Having quantitatively identified the key sources of CT-RL performance limitations, how may we design new CT-RL algorithms which substantively address these limitations in frameworks which supply both theoretical and real-world synthesis guarantees?

Characterization of New Algorithms: Three Novel Constructive RL Design Elements. This work proposes a numerics-driven, designer-centric framework to improve algorithm learning quality – the first such approach in CT-RL. First, for systems which exhibit a physically-motivated partition into distinct dynamical loops, our proposed decentralization framework breaks the optimal control problem into lower-dimensional subproblems in each of the loops, thereby reducing numerical complexity and dimensionality as well as allowing decentralized physics-driven design

and troubleshooting. Second, we develop a multi-injection (MI) solution which realigns the RL excitation framework with classical input/output insights. This creates effective PE while greatly simplifying the process. Third, we introduce a modulation-enhanced excitation (MEE) framework to prescale the learning weight regression via nonsingular transformations of the system state variables, resulting in improved scaling and numerics.

Contributions. Our contributions first address the CT-RL diagnosis issue:

1). We develop a novel learning algorithm performance diagnosis framework to quantitatively identify the key performance limitations facing CT-RL algorithms. This framework's in-depth, designer-focused quantitative analyses reveal gaps between CT-RL theoretical promises and practical synthesis.

Having performed this first-of-its-kind diagnosis, we then substantively address the remaining synthesis problem:

2). We introduce a novel RL design paradigm with three important decentralization, multi-injection, and modulation-enhanced excitation design elements to systematically improve PE/numerics.

3). We develop a novel suite of excitable integral reinforcement learning (EIRL) algorithms combining these elements, and we systematically demonstrate how application of each element successively improves conditioning performance relative to existing ADP methods, making these algorithms capable of realistic and complex controller synthesis.

4). Leveraging classical control insights, we prove rigorous convergence, solution optimality, and closed-loop stability guarantees of the new algorithms.

5). We conduct in-depth numerical evaluations of the proposed EIRL algorithms on real-world application problems, demonstrating significant performance enhancements relative to the state-of-the-art CT-RL methods in both ADP and deep RL.

Broad Applicability of Proposed Frameworks. We keep the MI framework formulation general, illustrating that this idea may be readily applied to most existing ADP-based RL control methods for PE/conditioning improvements. Meanwhile, a variety of compelling real-world applications admit natural decentralizing dynamical partitions. For example, in robotics applications, the Euler-Lagrange equations partition states along the system degrees of freedom [60]. In particular, ground robot dynamics decompose into a translational loop associated with the vehicle speed and a rotational loop associated with the vehicle pose [138, 139]. Helicopter dynamics admit partitions along each of the vehicle's longitudinal, lateral, and vertical degrees of freedom, as well as each of its three rotational degrees of freedom [50–52]. Quadcopter/UAV dynamics partition in an entirely analogous fashion [140]. The longitudinal dynamics of aircraft also separate into a translational/velocity loop and a rotational/flightpath angle loop [122, 141, 142]. This translational/rotational decentralization has demonstrated great real-world success in classical control frameworks, even for high-performance aerospace systems such as hypersonic vehicles (HSVs). [132, 133, 135].

1.4 Organization of Dissertation

The first part of this dissertation develops the CT-RL performance diagnosis framework in Section 2. This includes:

1). A formal definition of the continuous-time optimal control problem studied in

this work in Section 2.1.

2). Description of the neural networks, training, and theoretical results for each of the four central ADP-based CT-RL methods [1–4] are discussed in Sections 2.2, 2.3, and 2.4, respectively.

3). The remaining sections (2.5, 2.6 and 2.7) provide comprehensive and quantitative performance evaluations of the four methods, uncovering key performance limitations and design insights.

4). Finally, we conclude our diagnostic study with a discussion in Section 2.8.

The second part develops the proposed suite of EIRL algorithms in Section 3:

1). We first define the basic problem formulation and dynamical assumptions of the methods in Section Section 3.1.

2). We then develop the EIRL suite of algorithms in Section 3.2.

3). Finally, we proving convergence, solution optimality, and closed-loop stability of the EIRL suite in Section 3.6.

The third part of this work consists of in-depth numerical performance evaluations of the proposed EIRL methods in relation to the CT-RL state of the art:

1). Section 4 contains our initial algorithm development work evaluation studies on a real-world hypersonic vehicle (HSV) model. Section 4.2 first systematically demonstrates how application of decentralization, MI, and MEE successively improves algorithm conditioning in comparison to the original ADP IRL algorithm [1] on which EIRL is based. Section 4.3 focuses on the flagship decentralized EIRL (dEIRL) method and conducts HSV lift-coefficient modeling error ablation studies to evaluate dEIRL's closed-loop optimality recovery capability.

2). Having demonstrated significant performance improvements over existing ADP methods, our evaluations of Section 5 study dEIRL learning and closed-loop performance in comparison to the leading continuous FVI (cFVI) [7] and robust FVI (rFVI) [8] deep CT-RL methods. These studies focus on three different pendulum, jet aircraft, and differential drive mobile robot (DDMR) environments, and they include ablations with respect to modeling error and initial conditions. We evaluate the methods with respect to convergence/solution optimality, average return and policy cost performance, value function approximation performance, closed-loop performance, and numerical/parameter/data complexity.

3). In our final evaluation of Section 6, we substantively combine all three decentralization, MI, and MEE design elements in a comprehensive evaluation on the flagship HSV system. This study evaluates dEIRL in comparison to classical LQR and feedback linearization (FBL) methods with respect to:

- frequency response performance generalization with respect to varying modeling error (Section 6.2);
- closed-loop step response performance generalization with respect to varying modeling error (Section 6.3);
- solution optimality and conditioning generalization with respect to varying initial conditions and modeling error in a single parameter (Section 6.4);
- solution optimality and conditioning generalization with respect to deterministic sweeps of simultaneous modeling errors in two parameters (Section 6.5);
- closed-loop performance generalization with respect to randomly-distributed simultaneous modeling errors in all parameters (Section 6.6).

These evaluations study effects of multiple modeling errors in lift, drag, and pitch moment coefficient. They comprise a comprehensive suite of 37 quantitative learning, stability, frequency-domain, and closed-loop performance metrics evaluated over a total of 12,872 independent learning trial ablations of modeling error and initial conditions.

Chapter 2

CONTINUOUS-TIME REINFORCEMENT LEARNING CONTROL: A REVIEW OF THEORETICAL RESULTS, INSIGHTS ON PERFORMANCE, AND NEEDS FOR NEW DESIGNS

In this section, we develop the novel CT-RL performance diagnosis framework motivated in the Introduction Section 1. We first provide a comprehensive review of the four foundational CT-RL control algorithms [1–4] discussed in Section 1.2, deciphering key theoretical assumptions/results and highlighting their significance to a) the development of RL solutions to optimal control problems, and b) impact on related literatures. We conduct in-depth, designer-focused quantitative analyses revealing gaps between CT-RL theoretical promises and practical synthesis. We outline the needs of future innovative approaches in CT-RL, motivating the algorithm development to take place in Section 3. This CT-RL study was originally conducted in [137].

2.1 Problem Formulation

Notation. Throughout this work, $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{Z}$, and $\mathbb{N}$ denote the sets of reals, non-negative reals, integers, and naturals, respectively. For $n \in \mathbb{N}$, we denote $\mathbb{R}^n$ as the n-dimensional Euclidean space. $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^n$ or operator norm for matrices, unless decorated otherwise. We refer the reader to [13, pp. 117] for the definition of positive (semi)definite functions, and [13, pp. 144] for class $\mathcal{K}$, $\mathcal{K}_\infty$ and $\mathcal{KL}$ functions. In this work, the set $\Omega \subset \mathbb{R}^n$ is assumed to be compact and contain the origin $x = 0$ in its interior.

Problem Formulation: Optimal Control Problem. The background provided

here follows largely from the works [90, 91]. Consider the nonlinear time-invariant affine system,

$$\dot{x} = f(x) + g(x)u, \tag{2.1}$$

where $x \in \mathbb{R}^n$ is the state vector, $u : \mathbb{R}_+ \to \mathbb{R}^m$ is a measurable locally essentially bounded control, $f : \mathbb{R}^n \to \mathbb{R}^n$, and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$. We make the following assumptions on the system (2.1):

Assumption 2.1.1 ([91] Dynamical Assumptions). f and g are Lipschitz on Ω, and $f(0) = 0$.

We denote $x(t)$ as the solution at time $t \geq 0$ to the ODE (2.1) with initial condition $x(0) = x_0 \in \mathbb{R}^n$ evolving under the control $u(t) = \mu(x(t))$ given by the feedback control law $\mu : \mathbb{R}^n \to \mathbb{R}^m$. Define the infinite horizon performance index

$$J(x_0, \mu) = \int_0^\infty r\big(x(\tau), \mu(x(\tau))\big)d\tau, \tag{2.2}$$

where $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_+,\ r(x, u) = Q(x) + u^T Ru$ is the running cost and is assumed to satisfy the following:

Assumption 2.1.2 ([91] Cost Structure Assumptions). $Q : \mathbb{R}^n \to \mathbb{R}_+$ is a positive definite, monotonically-increasing function, and the system (2.1) is zero-state observable through Q. $R \in \mathbb{R}^{m\times m}$ satisfies $R = R^T > 0$.

Definition 2.1.1 ([91] Admissible Policies). A control policy $\mu : \mathbb{R}^n \to \mathbb{R}^m$ is admissible with respect to the cost (2.2) on Ω, denoted $\mu \in \mathcal{A}(\Omega)$, if μ is continuous on Ω, $\mu(0) = 0$, μ stabilizes the system (2.1) on Ω (cf. [90, Definition 3.1.2]), and $J(x_0, \mu)$ in (2.2) is finite for all $x_0 \in \Omega$.

The optimal regulation problem is to find the optimal control $\mu^* \in \mathcal{A}(\Omega)$ and its associated optimal cost function V^* (if they exist) such that

$$\begin{aligned} V^*(x_0) &= \min_{\mu \in \mathcal{A}(\Omega)} J(x_0, \mu) \\ \mu^*(x_0) &= \arg\min_{\mu \in \mathcal{A}(\Omega)} J(x_0, \mu) \quad , \quad \forall\, x_0 \in \Omega, \\ &= -\tfrac{1}{2} R^{-1} g^T(x_0) \nabla V^*(x_0) \end{aligned} \tag{2.3}$$

subject to the dynamics (2.1). Define the Hamiltonian function $H : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^{1\times n}$ as

$$H(x, u, p) = p\,[f(x) + g(x)u] + r(x, u). \tag{2.4}$$

Definition 2.1.2 ([91] Generalized HJB (GHJB) Equation). For an admissible control $\mu \in \mathcal{A}(\Omega)$, the function $V \in \mathcal{C}^1(\Omega)$ satisfies the GHJB equation, written $\mathrm{GHJB}(V, \mu) = 0$, if

$$H\left(x, \mu(x), (\nabla V(x))^T\right) = 0, \quad \forall x \in \Omega, \quad V(0) = 0. \tag{2.5}$$

In the affine case, (2.5) is given by

$$\nabla V^T\,[f + g\mu] + Q + \mu^T R \mu = 0, \quad V(0) = 0. \tag{2.6}$$

We next list the key properties of the GHJB equation.

Lemma 2.1.1 ([91, Lemma 8] GHJB Equation Properties). Suppose that the system (2.1) satisfies Assumption 2.1.1, and that the cost structure (2.2) satisfies Assumption 2.1.2. Then for each admissible policy $\mu \in \mathcal{A}(\Omega)$, there exists a unique $\mathcal{C}^1$ solution V to the equation $\mathrm{GHJB}(V, \mu) = 0$ (2.5). V is a Lyapunov function on

Ω for the closed-loop system comprised of (2.1) and $u = \mu(x)$ (in particular, V is positive definite). Furthermore, $\text{GHJB}(V, \mu) = 0$ if and only if $V(x) = J(x, \mu)$, where J is the performance index given in (2.2).

We next discuss the HJB equation and its fundamental importance to the optimal control problem.

Definition 2.1.3 ([91] HJB Equation). The function $V^* \in \mathcal{C}^1(\Omega)$ satisfies the HJB equation, written $\text{HJB}(V^*) = 0$, if

$$\text{HJB}(V^*) = \text{GHJB}\big(V^*, -\frac{1}{2}R^{-1}g^T\nabla V^*\big) = 0, \;\; V^*(0) = 0. \tag{2.7}$$

In the affine case, (2.7) is given by

$$(\nabla V^*)^T f - \frac{1}{4}(\nabla V^*)^T g R^{-1} g^T \nabla V^* + Q = 0, \;\; V^*(0) = 0. \tag{2.8}$$

The following theorem establishes sufficient conditions for the existence and uniqueness of solutions to the HJB equation (2.8). It also ties the solutions of the HJB equation to the optimal control problem.

Theorem 2.1.1 ([90] HJB Equation Properties). Suppose that Assumptions 2.1.1 and 2.1.2 hold. Then there exists a unique positive definite $\mathcal{C}^1$ solution V^* to the HJB equation (2.8), and V^* is the optimal value function in (2.3); i.e., the associated control μ^* given by (2.3) is admissible and uniquely minimizes the performance index (2.2) over the admissible controls $\mathcal{A}(\Omega)$.

Through the classical PI Algorithm 1, the GHJB equation (a first-order, linear PDE) may be solved successively to search for the solution of the HJB equation (a first-order, nonlinear PDE). Solving the GHJB PDE, although simpler than the HJB

PDE, is still a challenging problem. Subsequent sections outline numerically-tractable methods by means of neural network approximation and ADP.

Algorithm 1 PI Algorithm.

1: **Hyperparameters:** Initial admissible policy $\mu_0 \in \mathcal{A}(\Omega)$.
2: **for** $i = 0, 1, \dots$ **do**
3: *Policy Evaluation:* Evaluate the performance index (2.2) for the policy μ_i by solving $\text{GHJB}(V_i, \mu_i) = 0$ (2.5).
4: *Policy Improvement:* Update the control by

$$\begin{aligned}\mu_{i+1}(x) &= \arg\min_{v \in \mathcal{A}(\Omega)} \left\{ H\left(x, v(x), (\nabla V_i(x))^T\right)\right\} \\ &= -\frac{1}{2} R^{-1} g^T(x) \nabla V_i(x). \end{aligned} \tag{2.9}$$

5: **end for**

Theorem 2.1.2 ([90] PI Algorithm Properties). Let all hypotheses be as in Theorem 2.1.1. Suppose that $\mu_0 \in \mathcal{A}(\Omega)$ is admissible, and consider the sequences $\{\mu_i\}_{i=0}^{\infty}$ and $\{V_i\}_{i=0}^{\infty}$ generated by the PI Algorithm 1. Then the following hold:

(i) $\mu_i \in \mathcal{A}(\Omega)$ for all $i \geq 0$;

(ii) $V_{i+1}(x) \leq V_i(x)$ for all $x \in \Omega$, $i \geq 0$;

(iii) $V_i \to V^*$ and $\mu_i \to \mu^*$ uniformly on Ω.

(iv) If Υ_i denotes the basin of attraction of policy μ_i $(i = 0, 1, \dots)$, and if Υ^* denotes the basin of attraction of μ^*, then for each $i \geq 0$ we have $\Omega \subset \Upsilon_i \subset \Upsilon_{i+1} \subset \Upsilon^*$.

2.2 Neural Network (NN) Structures Considered

In what follows, let $\{\phi_j\}_{j=1}^{\infty}$, $\{\psi_j\}_{j=1}^{\infty}$, $\{\theta_j\}_{j=1}^{\infty}$, and $\{\tilde{\psi}_j\}_{j=1}^{\infty}$ be sequences of linearly independent $\mathcal{C}^1$ basis functions which vanish at the origin, with ϕ_j, ψ_j, $\theta_j : \mathbb{R}^n \to \mathbb{R}$ and $\tilde{\psi}_j : \mathbb{R}^n \to \mathbb{R}^{1\times m}$ for $j \in \mathbb{N}$.

Critic Network. Consider the critic network

$$V^*(x) = \hat{V}(x, c) + \epsilon_1(x), \qquad \hat{V}(x, c) = c^T \Phi(x), \tag{2.10}$$

where $N_1 \in \mathbb{N}$, $\Phi : \mathbb{R}^n \to \mathbb{R}^{N_1}$, $\Phi(x) = \begin{bmatrix} \phi_1(x) & \cdots & \phi_{N_1}(x) \end{bmatrix}^T$, $c \in \mathbb{R}^{N_1}$ is the critic weight vector, and $\epsilon_1 : \Omega \to \mathbb{R}$ is the critic NN approximation error function. For the sake of brevity, we shall whenever possible denote the critic (2.10) by $\hat{V}(x)$.

Actor Network. Consider the actor network

$$\mu^*(x) = \hat{\mu}(x) + \epsilon_2(x), \tag{2.11}$$

where $\epsilon_2 : \Omega \to \mathbb{R}^m$ is the actor NN approximation error function. The considered methodologies (IRL [1], SPI [2], RADP [3], and CT-VI [4]) use three distinct actor networks $\hat{\mu}$ (2.11). The first structure, used by RADP [3], is given by

$$\hat{\mu}(x, W) = W^T \Psi(x), \tag{2.12}$$

where $N_2 \in \mathbb{N}$, $\Psi : \mathbb{R}^n \to \mathbb{R}^{N_2}$, $\Psi(x) = \begin{bmatrix} \psi_1(x) & \cdots & \psi_{N_2}(x) \end{bmatrix}^T$, and $W \in \mathbb{R}^{N_2 \times m}$ is a weight matrix. The second structure, used by IRL [1] and SPI [2], is motivated by the structural form of μ^* assumed in (2.3)

$$\hat{\mu}(x, w) = -\frac{1}{2} R^{-1} g^T(x) \nabla \Phi^T(x) w, \tag{2.13}$$

where $w \in \mathbb{R}^{N_2}$ is the actor weight vector. We note for this structure that $N_2 \leftarrow N_1$ is imposed, and knowledge of the input dynamics g is required. This is in contrast to the structure (2.12), which is model-free. The third structure, used by CT-VI [4], is discussed next.

Hamiltonian Network. Consider the Hamiltonian network

$$\begin{aligned} H^*(x,u) &\triangleq H\left(x,u,(\nabla V^*(x))^T\right) \\ &= \hat{H}(x,u,\overline{v}) + \epsilon_3(x,u), \end{aligned} \tag{2.14}$$

where the Hamiltonian function H is as defined in (2.4), and $\epsilon_3 : \Omega \times \mathbb{R}^m \to \mathbb{R}$ is the Hamiltonian NN approximation error function. We consider the following approximation structure for the Hamiltonian NN

$$\hat{H}(x,u,\overline{v}) = \overline{v}^T \Sigma(x,u) + (Q(x) + u^T R u), \tag{2.15}$$

where $N_3 \in \mathbb{N}$, $v \in \mathbb{R}^{N_3}$, $w \in \mathbb{R}^{N_2}$, $\overline{v} \triangleq \begin{bmatrix} w^T & v^T \end{bmatrix}^T$, $\tilde{\Psi} : \Omega \to \mathbb{R}^{N_2 \times m}$, $\tilde{\Psi}(x) = \begin{bmatrix} \tilde{\psi}_1^T(x) & \cdots & \tilde{\psi}_{N_2}^T(x) \end{bmatrix}^T$, and $\Sigma : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{N_2+N_3}$ is defined as

$$\Sigma(x,u) = \begin{bmatrix} \tilde{\Psi}(x)u \\ \Theta(x) \end{bmatrix}. \tag{2.16}$$

Here $\Theta : \mathbb{R}^n \to \mathbb{R}^{N_3}$, $\Theta(x) = \begin{bmatrix} \theta_1(x) & \cdots & \theta_{N_3}(x) \end{bmatrix}^T$. We thus choose the following basis for the Hamiltonian NN

$$\{\sigma_j(x,u)\}_{j=1}^{N_2+N_3+1} = \left\{\tilde{\psi}_j(x)\, u\right\}_{j=1}^{N_2} \bigcup \{\theta_j(x)\}_{j=1}^{N_3} \bigcup \{r(x,u)\}. \tag{2.17}$$

The selection of the basis functions (2.17), together with the definition of $H^*(x,u)$ (2.14), that $V^*(x)$ satisfies the HJB equation (2.8), and that given $x \in \mathbb{R}^n$ the approximation (2.14) must hold for any $u \in \mathbb{R}^m$, implies that the following approximations

are to be made:

$$w^T\tilde{\Psi}(x) \approx (\nabla V^*(x))^T g(x) \in \mathbb{R}^{1\times m}, \tag{2.18}$$

$$\begin{aligned} v^T\Theta(x) &\approx (\nabla V^*(x))^T f(x) \\ &= \frac{1}{4}(\nabla V^*(x))^T g(x) R^{-1} g^T(x) \nabla V^*(x) - Q(x). \end{aligned} \tag{2.19}$$

We are now ready to define the actor network adopted by CT-VI [4], which is given by

$$\hat{\mu}(x, w) = -\frac{1}{2} R^{-1} \left(w^T \tilde{\Psi}(x) \right)^T. \tag{2.20}$$

Remark 2.2.1 (CT-VI Basis Selection). In [4, Section IV-B], the authors include the control penalty function $u^T R u$ in the Hamiltonian network basis $\{\sigma_j(x,u)\}_{j=1}^{N_2+N_3+1}$ (2.17) instead of the full running cost $r(x,u)$. This basis selection, in turn, makes the term $-Q(x)$ in (2.19) disappear, which in principle reduces the complexity of the required approximation. In spite of this intuition, we find the selection of basis (2.17) to be more numerically reliable in practice. Therefore, we employ (2.17) in our evaluations of Sections 2.6-2.7.

Remark 2.2.2 (Notation). In the single-input ($m = 1$) case, the RADP actor weight matrix $W \in \mathbb{R}^{N_2\times m}$ (2.12) becomes a weight vector which we naturally denote $w \in \mathbb{R}^{N_2}$, as it is for the other three methods. Comparison of the RADP and CT-VI actor implementations (2.12) and (2.20), respectively, reveals these two structures to be identical modulo multiplication by the scalar term $-1/2R$. Thus, for CT-VI we use the RADP activation functions $\{\psi_j\}_{j=1}^{N_2}$ in place of the activation functions $\{\tilde{\psi}_j\}_{j=1}^{N_2}$. We adopt these conventions in the evaluations of Sections 2.6-2.7, which focus on single-input systems.

2.3 Algorithms and Training

We begin by defining some common notation. Each of these algorithms requires a CT-RL learning time $t_f > 0$ and associated learning window $t \in [0, t_f]$ over which to collect state-action data. For the two PI-based algorithms (IRL and RADP), we denote i as the iteration index and $i^* \in \mathbb{N}$ as the final iteration. The PI-based algorithms require collection of $l \in \mathbb{N}$ data samples per iteration. RADP reuses the same data for each iteration, so we denote its sample times as $\{t_k\}_{k=0}^{l}$ (i.e., $0 = t_0 < t_1 < \cdots < t_l = t_f$). IRL requires new data at each iteration $0 \leq i \leq i^*$, so we denote its sample instants as $\{\{t_k^i\}_{k=0}^{l}\}_{i=0}^{i^*}$ (i.e., $0 = t_0^0 < t_1^0 < \cdots < t_l^0 = t_0^1 < \cdots < t_l^{i^*} = t_f$). The algorithms have various termination criteria, so to unify notation we use the subscript "f" to denote an algorithm's final output value of the respective parameter (e.g., final critic weights c_f, final critic network output $\hat{V}_f$).

2.3.1 Integral Reinforcement Learning (IRL) [1]

Given a state trajectory $\{x(t)\}_{t \in \mathbb{R}_+}$ of the system (2.1), define $\Delta\phi_j : \mathbb{R}_+^2 \to \mathbb{R}$ $(j = 1, \ldots, N_1)$ as $\Delta\phi_j(t_0, t_1) = \phi_j(x(t_1)) - \phi_j(x(t_0))$. Next, define $\Delta\Phi : \mathbb{R}_+^2 \to \mathbb{R}^{N_1}$ by,

$$\Delta\Phi(t_0, t_1) = \begin{bmatrix} \Delta\phi_1(t_0, t_1) \\ \vdots \\ \Delta\phi_{N_1}(t_0, t_1) \end{bmatrix}. \tag{2.21}$$

For a strictly increasing sequence $\{t_k\}_{k=0}^{l}$, define $A_\phi : \mathbb{R}_+^{l+1} \to \mathbb{R}^{l\times N_1}$ by

$$A_\phi(t_0,\dots,t_l) = \begin{bmatrix} \Delta\Phi^T(t_0,t_1) \\ \vdots \\ \Delta\Phi^T(t_{l-1},t_l) \end{bmatrix}. \tag{2.22}$$

Next, for an admissible policy $\mu \in \mathcal{A}(\Omega)$, define the integral reinforcement function $\xi : \mathbb{R}_+^2 \times \mathcal{A}(\Omega) \to \mathbb{R}_+$ by

$$\xi(t_0,t_1,\mu) = \int_{t_0}^{t_1} \big(Q(x) + \mu^T(x)R\mu(x)\big)\, d\tau. \tag{2.23}$$

Similarly, define the function $\Xi : \mathbb{R}_+^{l+1} \times \mathcal{A}(\Omega) \to \mathbb{R}_+^{l}$ by

$$\Xi\big(t_0,\dots,t_l,\mu\big) = \begin{bmatrix} \xi(t_0,t_1,\mu) \\ \vdots \\ \xi(t_{l-1},t_l,\mu) \end{bmatrix}. \tag{2.24}$$

At iteration i $(i = 0, 1, \dots)$, IRL collects state trajectory data $\{x(t_k^i)\}_{k=0}^{l}$ at the time instants $\{t_k^i\}_{k=0}^{l}$ under the control $u = \hat{\mu}_i(x)$. At $t = t_l^i$, it updates its weights by solving the least-squares solution $c_i \in \mathbb{R}^{N_1}$ to the system of equations

$$\begin{aligned} \mathbf{A}_{IRL}^i\, c_i &= -\,\Xi\big(t_0^i,\dots,t_l^i,\hat{\mu}_i\big), \\ \mathbf{A}_{IRL}^i &\triangleq A_\phi(t_0^i,\dots,t_l^i) \in \mathbb{R}^{l\times N_1}, \end{aligned} \tag{2.25}$$

where A_ϕ, Ξ are as defined in (2.22) and (2.24), respectively.

Algorithm 2 IRL Algorithm [1].

Hyperparameters: t_f, i^*, l, sample times $\{\{t_k^i\}_{k=0}^l\}_{i=0}^{i^*}$, $\mu_0 \in \mathcal{A}(\Omega)$, and IC $x_0 \in \Omega$.
Initialization: Let $\hat{\mu}_0 \leftarrow \mu_0$.
for $i = 0 : i^*$ **do**
    Apply control $u = \hat{\mu}_i(x)$ to system (2.1) , collecting data $\{x(t_k^i)\}_{k=0}^l$ and $\{\xi(t_k^i, t_{k+1}^i, \hat{\mu}_i)\}_{k=0}^{l-1}$ (2.23).
    Perform weight update c_i (2.25) and policy update $\hat{\mu}_{i+1}(x) \leftarrow \hat{\mu}(x, c_i)$ (2.13).
end for
Apply final policy $\hat{\mu}_f = \hat{\mu}_{i^*+1}$.

2.3.2 Synchronous Policy Iteration (SPI) [2]

SPI updates its critic weights $\{c(t)\}_{t\in[0,t_f]}$ over the learning window $[0, t_f]$ dynamically via the tuning law

$$\dot{c} = -\alpha_1 \frac{\sigma_2}{m_s^2} \left[\sigma_2^T c + r(x, \hat{\mu}(x, w))\right], \tag{2.26}$$

where $\alpha_1 > 0$ is a tuning gain, $\hat{\mu}(x, w)$ is given by the actor network (2.13), $\sigma_2(x) = \nabla\Phi(x)\left[f(x) + g(x)\hat{\mu}(x, w)\right]$, and $m_s(x) = (\sigma_2^T(x)\sigma_2(x) + 1)$ is a normalization term.

Remark 2.3.1 (SPI Actor Tuning). The authors [2] prescribe the following actor tuning law

$$\dot{w} = -\alpha_2 \left[\left(F_2 w - F_1 \frac{\sigma_2^T}{m_s} c\right) - \frac{1}{4} D(x) w \frac{\sigma_2^T}{m_s^2} c\right], \tag{2.27}$$

where $\alpha_2 > 0$, $F_1 \in \mathbb{R}^{N_1} > 0$, $F_2 \in \mathbb{R}^{N_1 \times N_1}$, $F_2 = F_2^T > 0$ are tuning parameters, and $D(x) = \nabla\Phi(x) g(x) R^{-1} g^T(x) \nabla\Phi^T(x)$. After extensive exploration, we were unable to find parameter values α_2, F_1, F_2 which yield stable state trajectory and weight responses for the examples studied in this work. In the code (found at [143]) for a previous rendition [144] of the SPI algorithm [2], the authors implement the following modified tuning law, which we observe to function properly and hence use throughout

this work instead of (2.27)

$$\dot{w} = -\alpha_2 \left[\left(F_2 w - F_2 c \right) - \frac{1}{4} D(x) w \frac{\sigma_2^T}{m_s^2} c \right]. \tag{2.28}$$

We note that after adopting the modified tuning law (2.28), the closed-loop stability and convergence results for SPI [2] (cf. Theorems 2.4.3 and 2.4.4, respectively) are no longer guaranteed. Examining the update (2.28) qualitatively, we note that the rightmost terms in (2.28) vanish as $\|x\| \to 0$ and as $\|x\| \to \infty$. Thus, in these regimes (2.28) can be approximated by $\dot{w} \approx -\alpha_2 F_2 (w - c)$, which resembles a linear tracking control law whereby the actor weights $w(t)$ track the critic weights $c(t)$.

Algorithm 3 SPI Algorithm [2].

Hyperparameters: t_f, tuning gains $\alpha_1, \alpha_2 > 0$, $F_1 > 0$, $F_2 = F_2^T > 0$ (2.26), e, $\mu_0 \in \mathcal{A}(\Omega)$ (cf. Assumption 2.4.1), IC $x_0 \in \Omega$, $c_0 \in \mathbb{R}^{N_1}$ (cf. Remark 2.5.1), and $w_0 \in \mathbb{R}^{N_1}$ such that $\hat{\mu}(x, w_0) \in \mathcal{A}(\Omega)$ (2.13).
Initialization: Let $c(0) \leftarrow c_0$, $w(0) \leftarrow w_0$.
for $t \in [0, t_f]$ **do**
    Apply control $u(t) = \hat{\mu}(x(t), w(t)) + e(t)$ (2.13) to system (2.1), tuning critic weights $c(t)$ via (2.26) and actor weights $w(t)$ via (2.27) (or (2.28), cf. Remark 2.3.1).
end for
Terminate e. Apply final policy $\hat{\mu}_f(x) = \hat{\mu}(x, w(t_f))$ (2.13).

2.3.3 Robust Adaptive Dynamic Programming (RADP) [3]

In what follows, suppose that the system (2.1) evolves under the control $u = \mu_0(x) + e$ (cf. Assumption 2.4.2), generating the trajectory $\{x(t)\}_{t \in \mathbb{R}_+}$. For an admissible policy $\mu \in \mathcal{A}(\Omega)$, define the function $\Delta_\psi : \mathbb{R}_+^2 \times \mathcal{A}(\Omega) \to \mathbb{R}^{mN_2}$ by

$$\Delta_\psi(t_0, t_1, \mu) = \int_{t_0}^{t_1} \left[R \left(u - \mu(x) \right) \right] \otimes \Psi(x) \, d\tau, \tag{2.29}$$

where $\otimes$ denotes the Kronecker tensor product. Given a strictly increasing sequence $\{t_k\}_{k=0}^{l}$ and a policy $\mu \in \mathcal{A}(\Omega)$, define $A_\psi : \mathbb{R}_+^{l+1} \times \mathcal{A}(\Omega) \to \mathbb{R}^{l \times mN_2}$ by

$$A_\psi\big(t_0, \dots, t_l, \mu\big) = 2 \begin{bmatrix} \Delta_\psi^T(t_0, t_1, \mu) \\ \vdots \\ \Delta_\psi^T(t_{l-1}, t_l, \mu) \end{bmatrix}. \tag{2.30}$$

At iteration i ($i = 0, 1, \dots$) of the RADP algorithm, the weights $c_i \in \mathbb{R}^{N_1}$, $W_i \in \mathbb{R}^{N_2 \times m}$ are solved for as the least-squares solution to the system of equations

$$\begin{aligned} \mathbf{A}_{RADP}^i \begin{bmatrix} c_i \\ \operatorname{vec}(W_i) \end{bmatrix} &= -\,\Xi\big(t_0, \dots, t_l, \hat{\mu}_i\big), \\ \mathbf{A}_{RADP}^i &\triangleq \begin{bmatrix} A_\phi(t_0, \dots, t_l) & A_\psi\big(t_0, \dots, t_l, \hat{\mu}_i\big) \end{bmatrix} \\ &\in \mathbb{R}^{l \times (N_1 + mN_2)}, \end{aligned} \tag{2.31}$$

where $\operatorname{vec}(W) \in \mathbb{R}^{N_2 m}$ denotes the vectorization of the matrix $W \in \mathbb{R}^{N_2 \times m}$, and the functions A_ϕ, Ξ, A_ψ are as defined in (2.22), (2.24), and (2.30), respectively.

Algorithm 4 RADP Algorithm [3].

1: **Hyperparameters:** t_f, i^*, l, sample times $\{t_k\}_{k=0}^{l}$, e, $\mu_0 \in \mathcal{A}(\Omega)$ (cf. Assumption 2.4.2), IC $x_0 \in \Omega$, and $W_0 = 0$.
   Initialization: Let $\hat{\mu}_0(x) \leftarrow \hat{\mu}(x, W_0)$ (2.12).
2: Apply control $u = \mu_0(x) + e$ to system (2.1), collecting state-action data $\{(x(t), u(t))\}_{t \in [0, t_f]}$.
3: **for** $i = 0 : i^*$ **do**
4: Calculate for policy $\hat{\mu}_i$ the data $\{\xi(t_k, t_{k+1}, \hat{\mu}_i)\}_{k=0}^{l-1}$ (2.23) and $\{\Delta_\psi(t_k, t_{k+1}, \hat{\mu}_i)\}_{k=0}^{l-1}$ (2.29).
5: Perform weight update c_i, W_i (2.31) and policy update $\hat{\mu}_{i+1}(x) \leftarrow \hat{\mu}(x, W_i)$ (2.12).
6: **end for**
7: Terminate e. Apply final policy $\hat{\mu}_f = \hat{\mu}_{i^*+1}$.

Remark 2.3.2 (RADP Robustness Results). The RADP algorithm as presented in [3, Algorithm 1] adds a robustifying term to the final policy $\hat{\mu}_f = \hat{\mu}_{i^*+1}$ produced by Algorithm 4 for its stability robustness results (cf. [3, Section III-B]). Yet, our initial attempts to implement this robustness term were thwarted by closed-loop stability issues we observe from Algorithm 4 in practice (cf. Sections 2.6-2.7). As noted by the authors in [3, Remark 3.2], in the absence of dynamic uncertainties the RADP algorithm may be run entirely without the developed robustifying term, which is the procedure followed in this paper.

2.3.4 Continuous-Time Value Iteration (CT-VI) [4]

For CT-VI, a measurable essentially bounded input u (cf. Assumption 2.4.4) is applied to the system (2.1) over the window $[0, t_f]$, generating the trajectory $\{x(t)\}_{t\in[0,t_f]}$. After data has been collected, CT-VI then tunes its weights dynamically over a learning time scale $s \in [0, s_f]$, which is independent of the system time scale $t \in [0, t_f]$. CT-VI updates its critic weights $\{c(s)\}_{s\in[0,s_f]}$ via the tuning law

$$\frac{d}{ds}c(s) = K_\phi^{-1}(t_f)\int_0^{t_f} \Phi(x)\hat{H}\big(x,\, \hat{\mu}(x, w(s)),\, \overline{v}(s)\big)d\tau, \tag{2.32}$$

where

$$K_\phi(t_f) = \int_0^{t_f} \Phi(x)\Phi^T(x)d\tau \in \mathbb{R}^{N_1\times N_1}. \tag{2.33}$$

CT-VI updates its actor weights $\{w(s)\}_{s\in[0,s_f]}$ and Hamiltonian weights $\{v(s)\}_{s\in[0,s_f]}$ via the tuning law

$$\overline{v}(s) = K_\sigma^{-1}(t_f)\int_0^{t_f} \Sigma(x,u)\left(\frac{d}{d\tau}\hat{V}(x, c(s)) + r(x,u)\right)d\tau, \tag{2.34}$$

where $\overline{v} = \begin{bmatrix} w^T & v^T \end{bmatrix}^T$ (cf. Section 2.2), and

$$K_\sigma(t_f) = \int_0^{t_f} \Sigma(x)\Sigma^T(x)d\tau \in \mathbb{R}^{(N_2+N_3)\times(N_2+N_3)}. \tag{2.35}$$

Algorithm 5 CT-VI Algorithm [4].

1: **Hyperparameters:** t_f, s_f, control u (cf. Assumption 2.4.4), IC $x_0 \in \Omega$, $c_0 \in \mathbb{R}^{N_1}$ such that $\hat{V}(x, c_0)$ (2.10) is positive definite and radially unbounded, $w_0 = 0$, $v_0 = 0$.
Initialization: Let $c(0) \leftarrow c_0$, $w(0) \leftarrow w_0$, $v(0) \leftarrow v_0$.
2: Apply control u to system (2.1), collecting state-action data $\{(x(t), u(t))\}_{t\in[0,t_f]}$.
3: **for** $s \in [0, s_f]$ **do**
4: Tune critic weights $c(s)$ via (2.32) and the actor, Hamiltonian weights $w(s)$, $v(s)$ via (2.34).
5: **end for**
6: Apply final policy $\hat{\mu}_f(x) = \hat{\mu}(x, w(s_f))$ (2.20).

2.4 Theoretical Results

This section discusses the key assumptions and properties of the four algorithms studied in this work. Throughout this section, we assume that the baseline hypotheses of Section 2.1 hold, which ensure that the optimal control problem is well-posed. Table 2.1 lists the terms needed to understand subsequent analysis and provides specific references to their definitions.

2.4.1 IRL

For IRL, the main convergence and stability results rely upon the following technical lemma, which is a restatement of [1, Lemma 3]:

Lemma 2.4.1. Suppose for admissible $\mu \in \mathcal{A}(\Omega)$ that the system (2.1) is simulated under the control $u = \mu(x)$, generating the state trajectory $\{x(t)\}_{t\in\mathbb{R}_+}$. Given that the

Table 2.1: Relevant Terms and Definitions

Term	Reference
Persistence of Excitation (PE)	[17, Def. 4.3.1, pp. 177]
Lyapunov Stability, Asymptotic Stability (AS)	[13, Def. 4.1, pp. 112]
Basin of Attraction, Global AS (GAS)	[13, pp. 122]
Local AS (LAS), Regional AS (RAS), Semiglobal Stabilization	[13, pp. 473]
Practical Stabilization (PS)	[145, Def. 1.2.1, pp. 9]
Uniformly Ultimately Bounded (UUB)	[13, Def. 4.6, pp. 169]
Input-to-State Stability (ISS)	[13, Def. 4.7, pp. 175]

set $\{\phi_j\}_{j=1}^{N_1}$ is linearly independent, for each $t_0 \geq 0$ and $l \geq N_1$ there exists a strictly increasing sequence $\{t_k\}_{k=1}^{l}$ such that the matrix $A_\phi(t_0, \ldots, t_l) \in \mathbb{R}^{l \times N_1}$ (2.22) has full column rank N_1.

We next move on to the key stability/convergence results.

Theorem 2.4.1 (IRL – Admissibility of Policies $\hat{\mu}_i$). Suppose $\mu_0 \in \mathcal{A}(\Omega)$ is admissible. There exists $N_{1,0} \in \mathbb{N}$ such that whenever $N_1 \geq N_{1,0}$, the policies $\{\hat{\mu}_i\}_{i=1}^{\infty}$ generated by Algorithm 2 are each admissible.

Theorem 2.4.2 (IRL – Uniform Approximation). For each $\epsilon > 0$, there exist $N_{1,0}, i_0 \in \mathbb{N}$ such that whenever $N_1 \geq N_{1,0}$, and $i^* \geq i_0$, we have

$$\left\| \hat{V}_f - V^* \right\|_\infty < \epsilon, \quad \|\hat{\mu}_f - \mu^*\|_\infty < \epsilon, \tag{2.36}$$

where $\hat{V}_f = \hat{V}_{i^*} = \hat{V}(x, c_{i^*})$ (2.10) and $\hat{\mu}_f = \hat{\mu}_{i^*+1} = \hat{\mu}(x, c_{i^*})$ (2.13) are as generated by Algorithm 2. Here $\|\cdot\|_\infty$ denotes the uniform norm on $\mathcal{C}(\Omega)$.

2.4.2 SPI

As with many ADP algorithms, SPI [2] has a persistence of excitation (PE) requirement.

Assumption 2.4.1 (SPI – PE Assumption). The signal $\overline{\sigma}_1 = \sigma_1/(\sigma_1^T\sigma_1 + 1)$, $\sigma_1 = \nabla\Phi(x)\left[f(x) + g(x)\mu^*(x)\right]$ is PE.

We also require the use of the following lemma.

Lemma 2.4.2 ([2, Lemma 1]). The solution $\hat{c}^*$ to the least squares minimization (2.37) exists and is unique, where

$$\begin{aligned}\hat{c}^* &= \min_{c\in\mathbb{R}^{N_1}} \left\| H\left(x, \mu^*(x), \nabla\hat{V}^T(x,c)\right)\right\|_{L^2(\Omega)} \\ &= \min_{c\in\mathbb{R}^{N_1}} \left\| c^T\nabla\Phi\left[f + g\mu^*\right] + r(x,\mu^*)\right\|_{L^2(\Omega)}. \end{aligned} \tag{2.37}$$

Before presenting the key stability and approximation results for SPI [2], we make the note that they require application of original actor tuning law (2.27), which we had to modify to (2.28) (cf. Remark 2.3.1 for discussion).

Theorem 2.4.3 (SPI – UUB Stability). Let tuning for the critic network (2.10) be provided by (2.26) and tuning for the actor network (2.13) be provided by (2.27). Suppose that tuning parameters are selected according to [2, Appendix]. Finally, consider the system (2.1) simulated under the control $u(t) = \hat{\mu}(x(t), w(t)) + e(t)$ (2.13), and assume the PE Assumption 2.4.1 is satisfied. Then there exists $N_{1,0} \in \mathbb{N}$ such that whenever $N_1 \geq N_{1,0}$, the closed-loop system state $x(t)$, critic error $\tilde{c} = \hat{c}^* - c$ (2.37), and actor error $\tilde{w} = \hat{c}^* - w$ are UUB.

Theorem 2.4.4 (SPI – Uniform Approximation). Let all hypotheses be as in Theorem 2.4.3. Then for each $\epsilon > 0$, there exists $N_{1,0} \in \mathbb{N}$ such that whenever $N_1 \geq N_{1,0}$, there exists $t_{f,0} = t_{f,0}(N_1)$ such that $t_f \geq t_{f,0}$ implies the uniform approximation result (2.36) holds for $\hat{V}_f(x) = \hat{V}(x, c(t_f))$ (2.10) and $\hat{\mu}_f(x) = \hat{\mu}(x, w(t_f))$ (2.13).

2.4.3 RADP

For RADP [3], we require that the initial policy $\mu_0 \in \mathcal{A}(\Omega)$ be admissible and satisfy the following assumption.

Assumption 2.4.2. The policy $\mu_0 \in \mathcal{A}(\Omega)$ is admissible and is such that for the exploration noise e there exists a compact set $\Omega_0 \subset \Omega$ containing the origin in its interior for which given any initial condition $x_0 \in \Omega_0$, Ω is an invariant set for the trajectory $x(t)$ generated by the closed-loop system composed of (2.1) and $u = \mu_0(x) + e$.

Assumption 2.4.3 (RADP – PE-Like Assumption). There exist $l_0 \in \mathbb{N}$ and $\delta > 0$ such that for all $l \geq l_0$, we have

$$\delta I_{N_1+mN_2} \leq \frac{1}{l} \sum_{k=0}^{l-1} \zeta_{i,k} \zeta_{i,k}^T, \qquad \forall\, i \geq 0, \tag{2.38}$$

where for $k = 0, \ldots, l-1$,

$$\zeta_{i,k} = \begin{bmatrix} \Delta\Phi(t_k, t_{k+1}) \\ 2\Delta_\psi(t_k, t_{k+1}, \hat{\mu}_i) \end{bmatrix} \in \mathbb{R}^{N_1+mN_2}, \tag{2.39}$$

and the functions $\Delta\Phi, \Delta_\psi$ are as defined in (2.22) and (2.30), respectively.

We are now ready to present the key stability and approximation results for RADP [3].

Theorem 2.4.5 (RADP – Admissibility of Policies $\hat{\mu}_i$). Suppose that Assumptions 2.4.2 and 2.4.3 hold. There exist $N_{1,0}$, $N_{2,0} \in \mathbb{N}$ such that whenever $N_1 \geq N_{1,0}$ and $N_2 \geq N_{2,0}$, the policies $\{\hat{\mu}_i\}_{i=1}^{\infty}$ generated by Algorithm 4 are each admissible.

Theorem 2.4.6 (RADP – Uniform Approximation). Suppose that Assumptions 2.4.2 and 2.4.3 hold. For each $\epsilon > 0$, there exist $N_{1,0}$, $N_{2,0}$, $i_0 \in \mathbb{N}$ such that whenever $N_1 \geq N_{1,0}$, $N_2 \geq N_{2,0}$, and $i^* \geq i_0$, the uniform approximation result (2.36) holds for $\hat{V}_f = \hat{V}_{i^*} = \hat{V}(x, c_{i^*})$ (2.10) and $\hat{\mu}_f = \hat{\mu}_{i^*+1} = \hat{\mu}(x, W_{i^*})$ (2.12) as generated by Algorithm 4.

2.4.4 CT-VI

CT-VI [4] has a PE-like assumption which we outline here.

Assumption 2.4.4 (CT-VI – PE-Like Assumption). The measurable essentially bounded input u is such that there exist $\delta > 0$ and $t_0 > 0$ such that for all $t_f \geq t_0$, the trajectory $\{x(t)\}_{t\in[0,t_f]}$ under u remains in Ω, and

$$\delta I_{N_1} < \frac{1}{t_f} K_\phi(t_f), \qquad \delta I_{N_2+N_3} < \frac{1}{t_f} K_\sigma(t_f), \tag{2.40}$$

where the matrices $K_\phi(t_f)$, $K_\sigma(t_f)$ are as defined in (2.33) and (2.35), respectively.

We are now ready to move on to the key results.

Theorem 2.4.7 (CT-VI – Regional Practical Stabilization (RPS)). Suppose that Assumption 2.4.4 holds. For each $\epsilon > 0$ such that $B_\epsilon(0) \subset \Omega$, there exist t_f, $s_f > 0$, N_1, N_2, $N_3 \in \mathbb{N}$ such that

$$\nabla V^*(x)^T \left[f(x) + g(x)\hat{\mu}(x, c(s_f)) \right] < 0, \quad \forall\, x \in \Omega \backslash B_\epsilon(0), \tag{2.41}$$

the weights $c(s)$, $w(s)$, $v(s)$ being tuned by Algorithm 5.

Remark 2.4.1. The results of Theorem 2.4.7 provided in the original CT-VI work (cf. [4, Theroem 3]) actually guarantee semiglobal PS; i.e., the compact set $\Omega \subset \mathbb{R}^n$ in

Theorem 2.4.7 may be made arbitrarily large. However, the definition of admissibility in [4] requires policies to be GAS. In our context, admissible policies $\mu \in \mathcal{A}(\Omega)$ only guarantee RAS on a fixed compact set $\Omega \subset \mathbb{R}^n$, so the associated stability results for CT-VI are only regional when applied here. This subtlety is addressed by the authors in [4, Remark 6].

Theorem 2.4.8 (CT-VI – Uniform Approximation). Suppose that Assumption 2.4.4 holds, and that Q is continuous. For each $\epsilon > 0$, there exist $N_{1,0}$, $N_{2,0}$, $N_{3,0} \in \mathbb{N}$, $t_f, s_f > 0$, and compact $\Omega_u \subset \mathbb{R}^m$ containing $u = 0$ in its interior sufficiently large such that whenever $N_1 \geq N_{1,0}$, $N_2 \geq N_{2,0}$, and $N_3 \geq N_{3,0}$, we have that the uniform approximation result (2.36) holds for $\hat{V}_f(x) = \hat{V}(x, c(s_f))$ (2.10) and $\hat{\mu}_f = \hat{\mu}(x, w(s_f))$ (2.20) as generated by Algorithm 5, and

$$\left\| \hat{H}_f - H^* \right\|_\infty < \epsilon, \tag{2.42}$$

where $\hat{H}_f(x, u) = \hat{H}(x, u, \overline{v}(s_f))$ (2.15) as generated by Algorithm 5, and $\|\cdot\|_\infty$ in (2.42) is the uniform norm on $\mathcal{C}(\Omega \times \Omega_u)$.

2.4.5 Summary and Discussion of Methodologies

Table 2.2 provides an overview of the essential features of the four methodologies considered.

Remark 2.4.2 (Initial Admissible Policy). IRL, SPI, and RADP require an initial admissible policy $\mu_0 \in \mathcal{A}(\Omega)$. In contrast to IRL and RADP, which (modulo Assumption 2.4.2 for RADP) are structurally unconstrained in their selection of $\mu_0 \in \mathcal{A}(\Omega)$, SPI requires that the initial policy μ_0 be implementable in the actor network (2.13) as $\mu_0(x) = \hat{\mu}(x, w_0)$ for some $w_0 \in \mathbb{R}^{N_1}$. This is comparatively quite restrictive and depends on the input dynamics g and critic basis functions $\{\phi_j\}_{j=1}^{N_1}$ available.

Table 2.2: Summary of CT–RL Methodologies

Algorithm	Dynamics Required	PE?	Data Reuse?	Convergence Results	Stability Results
IRL	g	No [a]	No	Uniform	RAS
SPI	f, g	Yes	No	Uniform	UUB [b]
RADP	None	Yes	Yes	Uniform	RAS
CT-VI	None	Yes	Yes	Uniform	RPS

[a] Cf. Remark 2.4.4.

[b] Cf. Remark 2.3.1.

Meanwhile, as is the case in the DT setting, strictly speaking CT-VI does not require an initial stabilizing policy [4]. The authors [4] suggest re-initializing the trajectory $x(t)$ whenever it leaves Ω in the learning interval $[0, t_f]$ (cf. [4, Remark 3]). However, this is not a luxury afforded in a real-world online learning scenario, so realistically speaking a designer will likely require an initial policy $\mu_0 \in \mathcal{A}(\Omega)$ to run CT-VI.

Remark 2.4.3 (Pseudoinversion, Conditioning). Three of the four algorithms studied here (IRL, RADP, and CT-VI) require use of the Moore-Penrose pseudoinverse, for which condition number plays a fundamental role in solution accuracy and sensitivity [146]. In the case of IRL and RADP, at iteration i the matrices $\mathbf{A}^i_{IRL}$ (2.25) and $\mathbf{A}^i_{RADP}$ (2.31) are pseudoinverted to yield the least-squares solutions for their respective weight updates. For CT-VI, in this work we implement the matrix inverses $K_\phi^{-1}(t_f)$ (2.32) and $K_\sigma^{-1}(t_f)$ (2.34) via pseudoinversion for improved computation speed and accuracy.

Remark 2.4.4 (PE Assumptions). SPI relies on the PE Assumption 2.4.1, while RADP and CT-VI rely on the PE-like Assumptions 2.4.3 and 2.4.4, respectively. As has long been understood, for nonlinear systems there do not exist systematic frameworks for verifying PE or for selecting a probing noise to ensure PE [2]. As we

will illustrate in Sections 2.6-2.7, in practice it is a challenge to excite the system to yield quality state trajectory data, even for simple academic examples.

IRL, strictly speaking, does not have a PE requirement. However, performance of this algorithm is still deeply tied to the quality of state trajectory data, as full column rank of $\mathbf{A}_{IRL}^{i}\mathbb{R}^{l\times N_1}$ is required at each iteration i for the least-squares weight update (2.25). Lemma 2.4.1 furnishes the existence of sample instances to meet the rank condition, but as will be seen in Sections 2.6-2.7, lack of ability to insert a probing noise e makes it difficult to systematically ensure good conditioning.

2.5 Performance Evaluation Setup

In this section, we offer four sets of fundamental evaluations of the studied CT-RL algorithms. Throughout we keep our focus from the perspective of a designer, working from a ground-up assessment which illustrates performance effectiveness, efficiency, limitations, and insights. The first three evaluations examine a second-order academic system (2.43) to establish performance baselines and insights. The fourth evaluation examines a cart inverted pendulum system (2.58) to assess the potential of real-world implementability.

These studies were performed in MATLAB R2021a, on an NVIDIA RTX 2060, Intel i7 (9th Gen) processor. All numerical integrations in this work are performed in MATLAB's adaptive `ode45` solver to ensure solution accuracy. The complete MATLAB software suite used to produce the data in this paper is available at [147].

2.5.1 Setup – 2nd Order System

The first three evaluations consider the following second-order academic system from [1, Section 6.2]

$$f(x) = \begin{bmatrix} -x_1 + x_2 + 2x_2^3 \\ -\frac{1}{2}(x_1 + x_2) + \frac{1}{2}x_2(1 + 2x_2^2)\sin^2(x_1) \end{bmatrix}, \qquad g(x) = \begin{bmatrix} 0 \\ \sin(x_1) \end{bmatrix}. \quad (2.43)$$

Linearization reveals that the origin of this system is an unstable equilibrium point. We run each algorithm over the IC sweep $x(0) = [-1 : 0.25 : 1]^2 \setminus \{(0, 0)\}$. In the first three evaluations, a "trial" corresponds to each of the ICs. We define the running cost as $Q(x) = x_1^2 + x_2^2 + 2x_2^4$, $R = 1$. This example was constructed such that the optimal value V^* and policy μ^* are available in closed-form and are given by

$$V^*(x) = \frac{1}{2}x_1^2 + x_2^2 + x_2^4, \quad (2.44)$$

$$\mu^*(x) = -\sin(x_1)(x_2 + 2x_2^3). \quad (2.45)$$

Basis Functions. Examining the optimal value V^* (2.44) and policy μ^* (2.45), for the first evaluation we select the following minimum-dimension critic basis (2.10)

$$\Phi(x) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_2^4 \end{bmatrix}, \quad \nabla\Phi(x) = \begin{bmatrix} 2x_1 & 0 \\ 0 & 2x_2 \\ 0 & 4x_2^3 \end{bmatrix}, \quad (2.46)$$

By inspection of (2.44) and (2.46), the optimal critic weights are $c^* = \left[\frac{1}{2}, 1, 1\right]^T$. For the second evaluation, we increment the problem dimension by adding the non-

essential basis function $\phi_4(x) = x_1 x_2$; i.e.,

$$\Phi(x) = \begin{bmatrix} x_1^2 & x_2^2 & x_2^4 & x_1 x_2 \end{bmatrix}^T, \tag{2.47}$$

which by inspection has optimal weights $c^* = \left[\frac{1}{2},\, 1,\, 1,\, 0\right]^T$. In the third evaluation, for the sake of comparison we choose the critic basis as identical to that of the example it was originally studied in [1, Section 6.2], with the essential basis function x_2^4 removed; i.e.,

$$\Phi(x) = \begin{bmatrix} x_1^2 & x_2^2 & x_1 x_2 & x_1^4 & x_1^3 x_2 & x_1^2 x_2^2 & x_1 x_2^3 \end{bmatrix}^T. \tag{2.48}$$

With the removal of this term, the optimal value function V^* (2.44) and policy μ^* (2.45) can no longer be approximated exactly. This is a more realistic scenario than the previous set of exact basis functions. Since it is well-known that in general there is no closed-form solution to HJB equation, a designer would naturally bias their selection towards such lower-order terms.

Recall that since the system (2.43) is single-input (i.e., $m = 1$), the same basis functions may be used for the RADP actor basis (2.12) and the CT-VI actor basis (2.20) (cf. Remark 2.2.2). Throughout the first three evaluations, we use the minimum-dimension basis

$$\Psi(x) = \sin(x_1) \begin{bmatrix} x_2 \\ x_2^3 \end{bmatrix}. \tag{2.49}$$

The optimal actor weights w^* are given by $w^* = [-1,\, -2]^T$ in the network (2.12), $w^* = c^*$ in the network (2.13), and $w^* = [2,\, 4]^T$ in the network (2.20). After working

out the algebra, it can be checked that

$$\frac{1}{4}(\nabla V^*)^T g R^{-1} g^T \nabla V^* = \sin^2(x_1)\left[x_2^2 + 4x_2^4 + 4x_2^6\right]. \tag{2.50}$$

Examination of (2.50) and (2.19) motivates the minimal choice of Hamiltonian basis

$$\Theta(x) = \begin{bmatrix} \sin^2(x_1)x_2^2 \\ \sin^2(x_1)x_2^4 \\ \sin^2(x_1)x_2^6 \\ x_1^2 \\ x_2^2 \\ x_2^4 \end{bmatrix}, \quad v^* = \begin{bmatrix} 1 \\ 4 \\ 4 \\ -1 \\ -1 \\ -2 \end{bmatrix}. \tag{2.51}$$

The first three terms in $\Theta(x)$ compose (2.50), while the last three terms compose $-Q(x)$ in (2.19) (cf. Remark 2.2.1 for discussion).

Initial Stabilizing Policy μ_0. In [1], the authors use the initial stabilizing policy

$$\mu_0(x) = -\frac{1}{2}\sin(x_1)(3x_2 - 0.2x_1^2 x_2 + 12x_2^3). \tag{2.52}$$

However, examining the minimal critic basis $\Phi(x)$ and its Jacobian $\nabla\Phi(x)$ (2.46), we see that we do not have access to the $x_1^2 x_2$ cross term for implementation of (2.52) in the actor network (2.13). Thus, in the spirit of continuity and maintaining a consistent comparison across the methodologies, we choose the similar stabilizing policy

$$\mu_0(x) = -\frac{1}{2}\sin(x_1)(3x_2 + 12x_2^3) \tag{2.53}$$

for the first two evaluations. For similar reasons, the critic basis (2.48) in the third

evaluation necessitates that modify the policy (2.53). We choose for the third evaluation

$$\mu_0(x) = -5\sin(x_1)x_2. \tag{2.54}$$

Exploration Noise e**.** We consider the following three default low-, medium-, and high-excitation noises

$$e_1(t) = 5\cos(t), \quad e_2(t) = 10\cos(t), \quad e_3(t) = 20\cos(t). \tag{2.55}$$

In the first evaluation we further perturb e_3 as

$$e_3(t) = 20\cos(t) + \cos(0.1t) \tag{2.56}$$

because CT-VI exhibits convergence issues without the addition of the small low-frequency term. Similarly, in the second evaluation we further perturb e_1 as

$$e_1(t) = 5\cos(t) + 0.25\cos(0.1t) \tag{2.57}$$

because CT-VI fails to converge for a few of the initial conditions in the sweep with the default exploration noise e_1 (2.55). Furthermore, by our search no small-amplitude perturbation of the exploration noise e_3 (2.55) is able to make CT-VI converge for all initial conditions in the second evaluation. As a result, in the second evaluation we choose the default exploration noise e_3 (2.55), and CT-VI is not run for this noise. These selections are summarized in Table 2.3.

Hyperparameter and Weight Initialization. Hyperparameter selections are listed in Table 2.4. We use a default learning time $t_f = 10$s. IRL's lack of probing

Table 2.3: Exploration Noises for the First Three Evaluations

Eval.	$e_1(t)$	$e_2(t)$	$e_3(t)$
1	(2.55)	(2.55)	(2.56)
2	(2.57)	(2.55)	(2.55)
3 [a]	(2.55)	N/A	N/A

[a]Noises customized for each method to maximize chances of convergence. See Section 2.6.3 for details.

noise necessitates a shorter learning time (cf. Section 2.6.1), whereas SPI's dynamic tuning laws require a longer learning time. For IRL, we collect $l = 10$ samples per iteration, while for RADP we collect $l = 50$ samples (all at equally-spaced time instants). We collect more points for RADP because its associated least-squares minimization is higher-dimensional ($N_1 + N_2$, cf. (2.31)) than that of IRL (N_1, cf. (2.25)).

Table 2.4: Hyperparameters for the First Three Evaluations

Eval.	Alg.	t_f (s)	s_f (s)	i^*	l
1, 2	IRL	5	N/A	5	10
	SPI	500	N/A	N/A	N/A
	RADP	10	N/A	5	50
	CT-VI	10	125	N/A	N/A
3	IRL	5	N/A	15	10
	SPI	500	N/A	N/A	N/A
	RADP	10	N/A	15	50
	CT-VI	10	125	N/A	N/A

Additional hyperparameters and initial weights are as follows. For SPI, we use $\alpha_1 = 10$ in the critic tuning law (2.26) and $\alpha_2 = 10$, $F_2 = 5I_{N_1}$ in the actor tuning law (2.28). For IRL, we initialize the critic weights c_0 to implement the policy μ_0 (2.53) in the actor structure (2.13) for the first two evaluations and the policy μ_0 (2.54) in the third evaluation. The actor weights w_0 for SPI are set to identical values as the IRL critic weights c_0 for all evaluations. This initializes IRL and SPI with the same policy μ_0 in their shared actor network structure (2.13).

Remark 2.5.1 (SPI – Critic Weight Initialization). Technically speaking, the initial critic weights c_0 for SPI may be selected independently of the initial actor weights w_0. However, as noted in Remark 2.3.1, the modified actor tuning law (2.28) implemented resembles a tracking control law which makes the actor weights track the critic weights. Thus, if the initial critic weights c_0 do not correspond to a stabilizing policy in the actor network (2.13), we observe that closed-loop instability results as the actor weights $w(t)$ converge to the destabilizing critic weights $c(t)$. Thus, in this work we initialize critic and actor weights to the same values.

For RADP, we initialize the actor weights $w_0 = 0$ as per Algorithm 4. Finally for CT-VI as per Algorithm 5, we initialize the actor weights $w_0 = 0$, Hamiltonian weights $v_0 = 0$, and critic weights c_0 such that the initial critic network is given by $\hat{V}(x, c_0) = x_1^2 + x_2^2$.

2.5.2 Setup – Cart Inverted Pendulum System

The fourth and final evaluation considers the cart inverted pendulum system [11]

$$\begin{aligned} \ddot{\mathrm{x}} &= \frac{m_p l \dot{\theta}^2 \sin\theta - m_p g \sin\theta\cos\theta + u}{m_c + m_p \sin^2\theta}, \\ \ddot{\theta} &= \frac{-\ddot{\mathrm{x}}}{l}\cos\theta + \frac{g}{l}\sin\theta, \end{aligned} \tag{2.58}$$

where x is the cart position (measured in meters), θ is the pendulum angular displacement (measured in radians relative to the upright position, clockwise positive), u is the horizontal force applied to the cart, m_c is the mass of the cart, m_p is the mass of the pendulum, l is the length of the pendulum, and g is the gravitational field constant. We use the standard values $m_c = 1\text{kg}$, $m_p = 0.1\text{kg}$, $l = 0.5\text{m}$, and $g = 9.81\text{m/s}^2$. This simplified model assumes no cart or pendulum friction, and that the mass of the pendulum is concentrated in a point at its end. With state vari-

ables $\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T = \begin{bmatrix} \mathrm{x} & \dot{\mathrm{x}} & \theta & \dot{\theta} \end{bmatrix}^T$, the dynamical equations (2.58) may be expressed in state-space form (2.1) as

$$f(x) = \begin{bmatrix} x_2 \\ \frac{m_p \sin x_3 \left(l x_4^2 - g \cos x_3\right)}{m_c + m_p \sin^2 x_3} \\ x_4 \\ \frac{\sin x_3 \left(\frac{g}{l}(m_c + m_p) - m_p x_4^2 \cos x_3\right)}{m_c + m_p \sin^2 x_3} \end{bmatrix}, \qquad g(x) = \frac{1}{m_c + m_p \sin^2 x_3} \begin{bmatrix} 0 \\ 1 \\ 0 \\ -\frac{\cos x_3}{l} \end{bmatrix}. \tag{2.59}$$

We use the standard Q-R cost structure $r(x, u) = x^T Q x + u^T R u$ and the natural choice $Q = I_4$, $R = 1$.

Basis Functions. We note for this example that the running cost $r(x, u)$ is an even function of (x, u) and that the state dynamics $f(x) + g(x)u$ is an odd function of (x, u). It can be checked that this implies the optimal value function V^* is even and the optimal policy μ^* is odd. With this insight, we select as our critic basis $\{\phi_j\}_{j=1}^{N_1}$ the even monomials of total degree two (i.e., $N_1 = 10$). We select for the actor basis $\{\psi_j\}_{j=1}^{N_2}$ the odd monomials of total degree less than or equal to three (i.e., $N_2 = 24$). Lastly, we select the Hamiltonian basis functions $\{\theta_j\}_{j=1}^{N_3}$ also as the even monomials of total degree two (i.e., $N_3 = N_1 = 10$). We believe these selections to be the natural first-choice for a designer beginning their analysis of this system.

Initial Stabilizing Policy μ_0. For the initial stabilizing policy μ_0, we examine the linearization (A, B) about the origin and design for it the linear quadratic regulator (LQR) full-state feedback control law $u(x) = Kx$, where $K = R^{-1} B^T P \in \mathbb{R}^{m \times n}$, and $P \in \mathbb{R}^{n \times n}$, $P = P^T > 0$ is the unique positive definite solution of the Riccati equation

$$0 = A^T P + P A - P B R^{-1} B^T P + Q. \tag{2.60}$$

We use the locally-stabilizing nonlinear control law

$$\mu_0(x) = -R^{-1}g^T(x)Px. \tag{2.61}$$

Exploration Noise e**.** We default to the exploration noise $e(t) = e_1(t) = 5\cos(t)$ (2.55). With the policy μ_0 (2.61) and initial condition $x_0 = [1,\ 0,\ 15°,\ 0]^T$, this exploration noise yields stable cart position oscillations on the order of 3m and pendulum oscillations on the order of 20°, so qualitatively the noise allows the four algorithms to collect rich trajectory data under the initial stabilizing policy without exciting the pendulum instability. We observe the initial policy to achieve stability for exploration noise amplitudes of up to ~ 7.5.

Hyperparameter and Weight Initialization. For IRL, we choose a learning time $t_f = 5$s. Since we have increased the number of critic basis functions to $N_1 = 10$, we increase the number of data points per iteration from $l = 10$ to $l = 15$. Number of iterations i^* for IRL is changed experimentally, so we leave discussion to Section 2.7. For SPI, we choose a shorter collection window of $t_f = 100$s for reasons explained in Section 2.7. We choose all SPI tuning gains/matrices identical to those of Section 2.5.1 (modulo dimension increases). For RADP, we use $i^* = 20$ iterations; otherwise, all hyperparameters for RADP and CT-VI are chosen identical to those of the first three evaluations. Finally, all weight initializations are performed identically to Section 2.5.1, now corresponding to the policy μ_0 (2.61).

2.6 Performance Evaluation and Analysis – 2nd Order System

2.6.1 Evaluation 1 – Exact Minimal Bases

Results. IRL behaves well in regards to approximation and weight convergence. To illustrate this point, Table 2.5 displays the mean, max, and standard deviation critic

weight errors $\|c^* - c_f\|$ observed across the initial condition sweep for the exploration noise e_3 (2.56) in the first column, and the last three columns correspond to the three respective weights in the basis (2.46). IRL exhibits a final critic weight error $\|c^* - c_f\|$ of less than 10^{-9} for all trials. It also has a short average run time of around 0.15s per trial, as seen in Table 2.6 which shows the average run time of each algorithm over the IC sweep for the first evaluation.

Table 2.5: Eval. 1: Critic Weight Error for Noise e_3 (2.56)

Algorithm	Data	$\|c^* - c_f\|$	$\|c_1^* - c_{f,1}\|$	$\|c_2^* - c_{f,2}\|$	$\|c_3^* - c_{f,3}\|$
IRL	mean	8.9e-11	5.15e-15	1.74e-14	8.9e-11
	max	8.04e-10	1.68e-14	1.23e-13	8.04e-10
	std	1.78e-10	3.37e-15	2.97e-14	1.78e-10
SPI	mean	1.07e-04	3.12e-06	7.35e-05	7.78e-05
	max	1.33e-04	4.00e-06	9.15e-05	9.64e-05
	std	6.09e-06	4.74e-07	4.21e-06	4.45e-06
RADP	mean	6.77	0.925	6.18	2.62
	max	7.56	1.03	6.92	2.89
	std	0.348	0.0628	0.333	0.127
CT-VI	mean	0.0169	0.00382	0.0155	0.00539
	max	0.471	0.105	0.434	0.15
	std	0.0744	0.0166	0.0686	0.0236

Next, we discuss conditioning. Table 2.7 shows the IC sweep average condition number of the matrices pseudoinverted for IRL, RADP, and CT-VI for the first three evaluations. In the case of IRL and RADP, the respective matrices $\mathbf{A}_{IRL}^i$ (2.25) and $\mathbf{A}_{RADP}^i$ (2.31) change numerically with iteration count i, so we have taken the IC sweep average over the final-iteration matrices $\mathbf{A}_{IRL}^{i^*}$ and $\mathbf{A}_{RADP}^{i^*}$. IRL struggles with significant conditioning issues, having an average final-iteration condition number on the order of 10^5.

SPI also exhibits good convergence properties (Table 2.5). We note that conditioning analyses do not apply to SPI, which is an adaptive/gradient-descent-based method and does not require regression in its weight updates. However, these dynamic weight

Table 2.6: Eval. 1: Average Run Time (s)

Algorithm	e_1	e_2	e_3
IRL	0.14	0.15	0.15
SPI	2.96	2.99	3.17
RADP	0.20	0.21	0.23
CT-VI	9.15	17.21	23.00

updates require a significantly longer collection window $t_f = 500$s to converge, and thus SPI takes a much longer 3s on average to run (Table 2.6).

RADP achieves good approximation performance for the two smaller noises e_1, e_2 (2.55) (comparable numerically with that of IRL and SPI), but as seen in Table 2.5 this degrades for the large exploration noise e_3 (2.56). Its mean critic weight error $\|c^* - c_f\|$ is 6.77 with a standard deviation of only 0.348, suggesting that the error is large across the sweep. RADP fares quite well with conditioning (Table 2.7), which remains on the order of 10 for all exploration noises. It also has a short average run time of around 0.2s per trial (Table 2.6).

CT-VI converges consistently overall for the smaller exploration noises e_1, e_2 (2.55), with max final critic weight error $\|c^* - c_f\|$ of less than 0.01 and max actor weight error $\|w^* - w_f\|$ of 0.238. The maximum Hamiltonian weight error $\|v^* - v_f\|$ is higher at 2.08 (observed for e_2), but CT-VI exhibits a mean error $\|v^* - v_f\|$ of only 0.0672 for this exploration noise, so the peak of 2.08 is an outlier. For the large excitation e_3 (2.56) CT-VI performs well overall, with a mean critic weight error $\|c^* - c_f\|$ of only 0.0169 (Table 2.5), but there are outliers in which the error gets as large as 0.471 at maximum.

Like IRL, CT-VI struggles with conditioning. In Table 2.7, conditioning of the matrix $\kappa(K_\phi(t_f))$ averages on the order of 10^2 for all exploration noises and $\kappa(K_\sigma(t_f))$ averages on the order of 10^6 for e_1 and on the order of 10^4 for e_2, e_3. Since $K_\phi(t_f) \in \mathbb{R}^{N_1 \times N_1}$ with $N_1 = 3$ and $K_\sigma(t_f) \in \mathbb{R}^{(N_2+N_3)\times(N_2+N_3)}$ with $N_2 + N_3 = 8$, it is expected

Table 2.7: Evals. 1-3: Mean Condition Number

Algorithm	Matrix	Eval. 1			Eval. 2			Eval. 3 [a]
		e_1	e_2	e_3	e_1	e_2	e_3	e_1
IRL [b]	$\mathbf{A}_{IRL}^{i^*}$	1.48e+05	"	"	5.62e+05	"	"	5.75e+11
RADP	$\mathbf{A}_{RADP}^{i^*}$	34.94	22.21	17.19	19.68	23.10	25.61	155.45
CT-VI	$K_\phi(t_f)$	207.66	631.69	413.30	366.14	866.12	N/A [c]	6.17e+04
	$K_\sigma(t_f)$	6.96e+06	5.49e+04	4.87e+04	6.88e+06	5.49e+04	N/A[c]	1.06e+07

[a]Exploration noises e_2, e_3 not executed for Evaluation 3 (cf. Section 2.6.3 for details).

[b]No exploration noise injected for IRL, so we put its data under the e_1 column and leave its other entries blank.

[c]CT-VI not executed for exploration noise e_3 in Evaluation 2 (cf. Section 2.6.2 for details).

that in general the conditioning of $K_\sigma(t_f)$ will fare worse than that of $K_\phi(t_f)$. Evidence of the demanding computational requirements of CT-VI is further witnessed in Table 2.6. CT-VI requires by far the longest run time at 10-20s per trial. We note in addition that run time for CT-VI increases substantially with increasing exploration noise amplitude, more than doubling on average over the exploration noises tested.

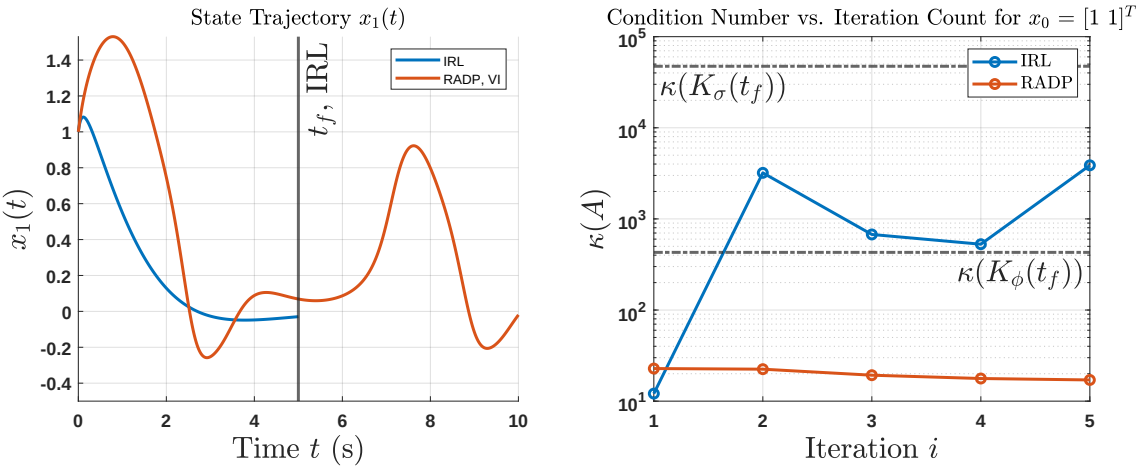
Insights. **1) Algorithms Perform Well for Lower Excitations in Baseline Example.** For the lower excitations e_1, e_2 (2.55) all algorithms successfully converge to the optimal weights regardless of the initial condition selected (with the exception of a few outliers). Since this example is low-order and we have chosen exact bases, the alignment between theoretical guarantees and observed synthesis is to be expected.

2) Algorithm Structure Significantly Impacts Conditioning and Numerical Performance. We point the reader to Figure 2.1a, which displays the state trajectory $x_1(t)$ corresponding to exploration noise e_1 (2.55) and IC $x_0 = [1,\ 1]^T$. RADP and CT-VI use the same trajectory data to perform their learning, yet CT-VI's conditioning is three orders of magnitude worse than RADP's (cf. Table 2.7). This simple example illustrates the stark impact of algorithm structure on inherent conditioning and numerical properties. For this reason, conditioning should be considered in the design process using similar approaches presented in this study, not just in post-hoc

analysis.

3) Avoid Large Excitation for RADP. We offer designers this general observation of RADP: Across all evaluations conducted, RADP exhibits convergence issues for large exploration noises. This is of practical concern to real-world designers, who are in general concerned with achieving *sufficient* excitation to meet PE requirements. We caution that over-excitation is a significant phenomenon which occurs even for low-order academic examples.

Limitations. 1) CT-VI Numerical Complexity Issues Associated with Tuning Structure. In this simple example, we have already experienced divergence issues with CT-VI for the large exploration noise e_3, which necessitates the addition of the low-frequency perturbation $\cos(0.1t)$ in (2.56). We believe these issues can be explained by examining the tuning procedure of CT-VI in (2.32) and (2.34). The Hamiltonian weights $\overline{v}(s)$ yielded by the pseudoinversion of $K_\sigma(t_f)$ in (2.34) are nested in the integral (2.32) involved in the pseudoinversion of $K_\phi(t_f)$. The pseudoinversion of $K_\phi(t_f)$ yields the derivative $\frac{d}{ds}c(s)$ in the critic weight tuning law (2.32), which itself must be integrated with respect to the weight tuning time s. In sum, the CT-VI algorithm comprises an alternating chain of two pseudoinversions sandwiched between three nested (vector-valued) integrations. When combined with the high condition numbers seen in Table 2.7, we conclude that the weight convergence issues stem from these numerical considerations. The run time of CT-VI exceeds that of IRL and RADP by a factor of 100 (Table 2.6), a further empirical indication of numerical complexity issues.



(a) (b)

Figure 2.1: Eval. 1: Data for Exploration Noise e_1 (2.55), IC $x_0 = [1, 1]^T$. (a) Learning-Phase State Trajectory $x_1(t)$. (b) Condition Number Versus Iteration Count.

2) IRL's Lack of Probing Noise Causes Data Quality Degradation as State is Regulated to Origin. The two PI-based algorithms (IRL and RADP) exhibit vastly different conditioning properties (cf. Table 2.7). Returning to Figure 2.1a offers clear explanation. Recall that IRL does not allow for insertion of a probing noise. As the system is simulated under the initial stabilizing policy μ_0 (2.53) and successive stabilizing policies $\hat{\mu}_1, \ldots, \hat{\mu}_{i^*}$, the state is regulated to the origin. Meanwhile, examining the form of the i-th iteration weight update matrix $\mathbf{A}_{IRL}^i \in \mathbb{R}^{l \times N_1}$ (2.25) $(i = 0, \ldots, i^*)$, we see that continuity of the state trajectory $x(t)$ and continuity of the critic basis functions $\Phi(x)$ implies that $\mathbf{A}_{IRL}^i$ vanishes as variations in the state trajectory samples $\{x(t_k^i)\}_{k=0}^l$ vanish. This explains the steep upward trend in IRL's iteration-wise condition number plotted in Figure 2.1b, beginning at around 10 for $i = 1$ and increasing to almost 10^4 for $i = 5$. It is for this reason that a shorter collection window $t_f = 5$s is necessary for IRL. If the default $t_f = 10$s used for the other methods is chosen, the condition number $\kappa\left(\mathbf{A}_{IRL}^{i^*}\right)$ regularly exceeds 10^8 for this example. We hence observe lack of probing noise insertion as a fundamental lim-

itation of the IRL methodology, even though strictly speaking it does not have a PE requirement (cf. Table 2.2 and Remark 2.4.4). As a result of probing noise insertion, RADP has access to richer trajectory data and its conditioning remains low.

For the same reasons, we make the note that the conditioning of IRL fares significantly worse for ICs chosen nearer the origin. Thus, Figure 2.1b with the IC $x_0 = [1,\, 1]^T$ is a best-case across the sweep. In order to combat this conditioning issue, for each iteration i the authors [1, Section 6.2] collect data from multiple trajectories with randomized initial conditions. While this is a legitimate learning procedure, strictly speaking it is not permissible in an online learning scenario.

Remark 2.6.1. Concluding Remarks for Evaluation 1: The Curse of Conditioning in CT-RL. After considering Table 2.7, Figure 2.1b, and the subsequent analysis, we wish to characterize whether the observed conditioning issues are *emergent* phenomena or if they are *inherent* to the CT-RL methodologies themselves.

Certainly, this is not an issue of *dimensionality*. The system (2.43) is second-order and single-input, and the basis dimensions $N_1 = 3$, $N_2 = 2$, $N_3 = 6$ are chosen to be minimal for this problem. Real-world applications will inevitably be higher-dimensional than this one.

Neither is this an issue of *approximation*. Indeed, the example was constructed such that the optimal value V^* and policy μ^* are available in closed-form, and the bases chosen can achieve exact approximation.

Having ruled out the usual culprits of problem dimension and approximation error, we consider that the fundamental conditioning issues illustrated here are intrinsic to the algorithms themselves. For in many respects, the problem structure of this evaluation represents the *best-case* performance that could be hoped for from these algorithms. Unfortunately, we shall soon see that the underlying numerics compound subsequent issues of dimensionality and approximation, altogether severely limiting

the applicability of these CT-RL methods to real-world design problems.

2.6.2 Evaluation 2 – Critic Basis with $N_1 = 4$ Terms

Results. **IRL** achieves good approximation performance comparable with that of the first evaluation for all probing noises. We present critic weight error data for the exploration noise e_3 (2.55) in Table 2.8. IRL achieves a critic weight error $\|c^* - c_f\|$ of less than 10^{-9} at max. Meanwhile, examination of Table 2.7 shows that the addition of one critic basis function has increased IRL's average conditioning by a factor of five to 5.6×10^5. For the sake of comparison, we have again plotted condition number versus iteration count for the exploration noise e_1 (2.57) and IC $x_0 = [1,\ 1]^T$ in Figure 2.2. Comparison with Figure 2.1b corroborates the trend in Table 2.7 that conditioning has degraded across the board. In particular, the conditioning of IRL now approaches the 10^5 mark for this trial, almost a tenfold increase from the previous evaluation. This illustrates that IRL suffers from scalability issues.

Finally, addition of one basis function has not increased the run time of any of the algorithms significantly, so we have omitted run time data here for the sake of brevity. Note, however, that we wished to carry out more thorough analyses of run time performance, but for all algorithms weight convergence issues halted these evaluations before we could introduce sufficient dimensional scaling. Given that these algorithms repeatedly use pseudoinversion which scales as $\sim \mathcal{O}(l^3)$, in principle run time scaling will likely present significant challenges in the future.

SPI achieves a max critic weight error $\|c^* - c_f\|$ of 0.0465 for exploration noises e_1 (2.57) and e_2 (2.55), and 0.0176 for e_3 (Table 2.8). For all exploration noises, its actor weight error $\|w^* - w_f\|$ remains less than 0.001 at max.

RADP We again observe the pattern that RADP performs comparably to IRL and SPI for small excitations but struggles for the large excitation e_3 (2.55), with an

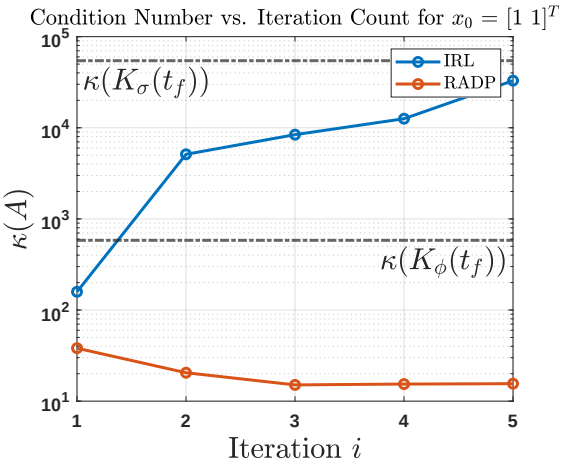


Figure 2.2: Eval. 2: Condition Number Versus Iteration Count for Exploration Noise e_1 (2.57), IC $x_0 = [1,\ 1]^T$.

Table 2.8: Eval. 2: Critic Weight Error for Noise e_3 (2.55)

Alg.	Data	$\|c^* - c_f\|$	$\|c_1^* - c_{f,1}\|$	$\|c_2^* - c_{f,2}\|$	$\|c_3^* - c_{f,3}\|$	$\|c_4^* - c_{f,4}\|$
IRL	mean	1.54e-10	3.7e-14	7.8e-14	1.54e-10	8.91e-14
	max	1.24e-09	2.3e-13	3.94e-13	1.24e-09	4.95e-13
	std	2.56e-10	5.69e-14	8.66e-14	2.56e-10	1.16e-13
SPI	mean	0.0168	0.000803	0.00849	0.0143	0.00248
	max	0.0176	0.000926	0.00883	0.0149	0.00268
	std	0.00022	3.59e-05	0.000132	0.000199	7.46e-05
RADP	mean	2.87	0.468	2.62	1.09	0.0224
	max	6.73	1.06	6.14	2.55	0.0997
	std	2.76	0.45	2.51	1.04	0.0217

average critic weight error $\|c^* - c_f\|$ of 2.87, max of 6.73, and standard deviation of 2.76 (Table 2.8). Comparison of these numbers with those of Table 2.5 shows that the max error is comparable for the two evaluations. Meanwhile, the mean error for this evaluation (2.87) is smaller than that of the previous evaluation (6.77). However, the present evaluation standard deviation (2.76) is a factor of ten higher than previous (0.348). This anecdotally suggests that the addition of a basis function to the critic (2.47) has made the weight convergence of RADP more volatile for large exploration noises. In spite of these issues, RADP does zero the redundant basis

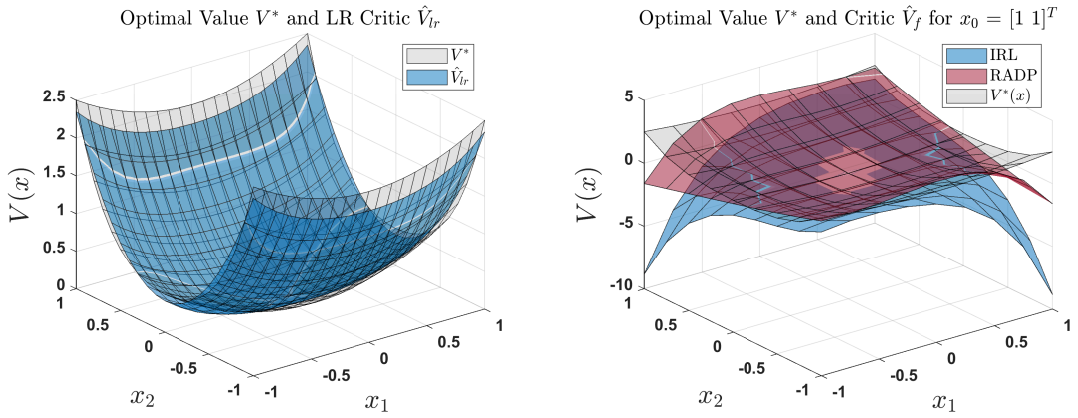
function $\phi_4(x) = x_1 x_2$ (2.47) for e_3 quite well overall (Table 2.8). Finally RADP's conditioning has remained low at approximately 20 (Table 2.7).

CT-VI is not run for the exploration noise e_3 (2.55) due to convergence issues (cf. Section 2.5.1 for discussion), so its data is absent in Table 2.8. It does achieve good convergence properties for the smaller two excitations, for which its critic, actor, and Hamiltonian weight errors remain less than 0.01 at max. Examining Table 2.7, the conditioning of $K_\sigma(t_f)$ is nearly identical to the previous evaluation, which is expected since the bases $\Psi(x)$ (2.49) and $\Theta(x)$ (2.51) composing this matrix have remained unchanged. On the other hand, the condition number $\kappa\left(K_\phi(t_f)\right)$ has increased from 208 in the previous evaluation to 366 for the exploration noise e_1 (+76%) and from 632 to 866 for e_2 (+37%).

Limitations. 1) Convergence and Conditioning Degradation with Addition of One Critic Basis Function: Dimensional Scalability Concerns. IRL still performs well in terms of convergence, but its conditioning has degraded significantly as a result of the addition. SPI, the gradient-descent algorithm for which conditioning issues do not apply, fares the best overall; its convergence properties remain largely unchanged. RADP displays a slight degradation in conditioning and substantial increases in weight volatility. Finally, CT-VI exhibits weight divergence on both the low- and high-amplitude ends of the exploration noise spectrum, due in large part to condition number increase of the matrix $K_\phi(t_f)$ (2.33) structurally affected by the additional critic basis function. Now, all that remains is a central band of amplitudes around $e_2(t) = 10\cos(t)$ for which this algorithm can run properly.

2.6.3 Evaluation 3 – Realistic Choice of Critic Basis

Now that the optimal value function V^* (2.44) cannot be approximated exactly by our choice of critic basis (2.48), we first establish that good approximation is still



(a) (b)

Figure 2.3: Eval. 3: (a) Optimal Value V^* and LR Critic $\hat{V}_{lr}$. (b) Optimal Value V^* and Final Critic $\hat{V}_f$ for Exploration Noise e_1 (2.55), IC $x_0 = [1,\ 1]^T$.

achievable. We perform linear regression (LR) of the optimal value function V^* (2.44) in the basis functions (2.48) over the box $[-1, 1]^2$, yielding the $L^2([-1, 1]^2)$-optimal weights,

$$c_{lr} = \begin{bmatrix} 0.2140 & 1.7436 & 0 & 0.2 & 0 & 0.1905 & 0 \end{bmatrix}^T. \tag{2.62}$$

The associated LR critic $\hat{V}_{lr}(x) = \hat{V}(x, c_{lr})$ is plotted alongside the optimal value function V^* (2.44) in Figure 2.3a. The approximation achieved is quite accurate by visual inspection.

Results. IRL, RADP We begin our study with the two PI-based algorithms. Comparison of the conditioning data in Table 2.7 to the previous evaluation shows that the conditioning for IRL has increased six orders of magnitude from 5.6×10^5 to 5.8×10^{11}. RADP conditioning has fared better, but it still has increased by a factor of eight from 20 to 155. To analyze convergence properties, we plot the critic weights c_i versus iteration count i for IRL and RADP in Figures 2.4c and 2.4d, respectively. These responses correspond to the exploration noise e_1 (2.55) and IC $x_0 = [1,\ 1]^T$,

which we observe as qualitatively representative.

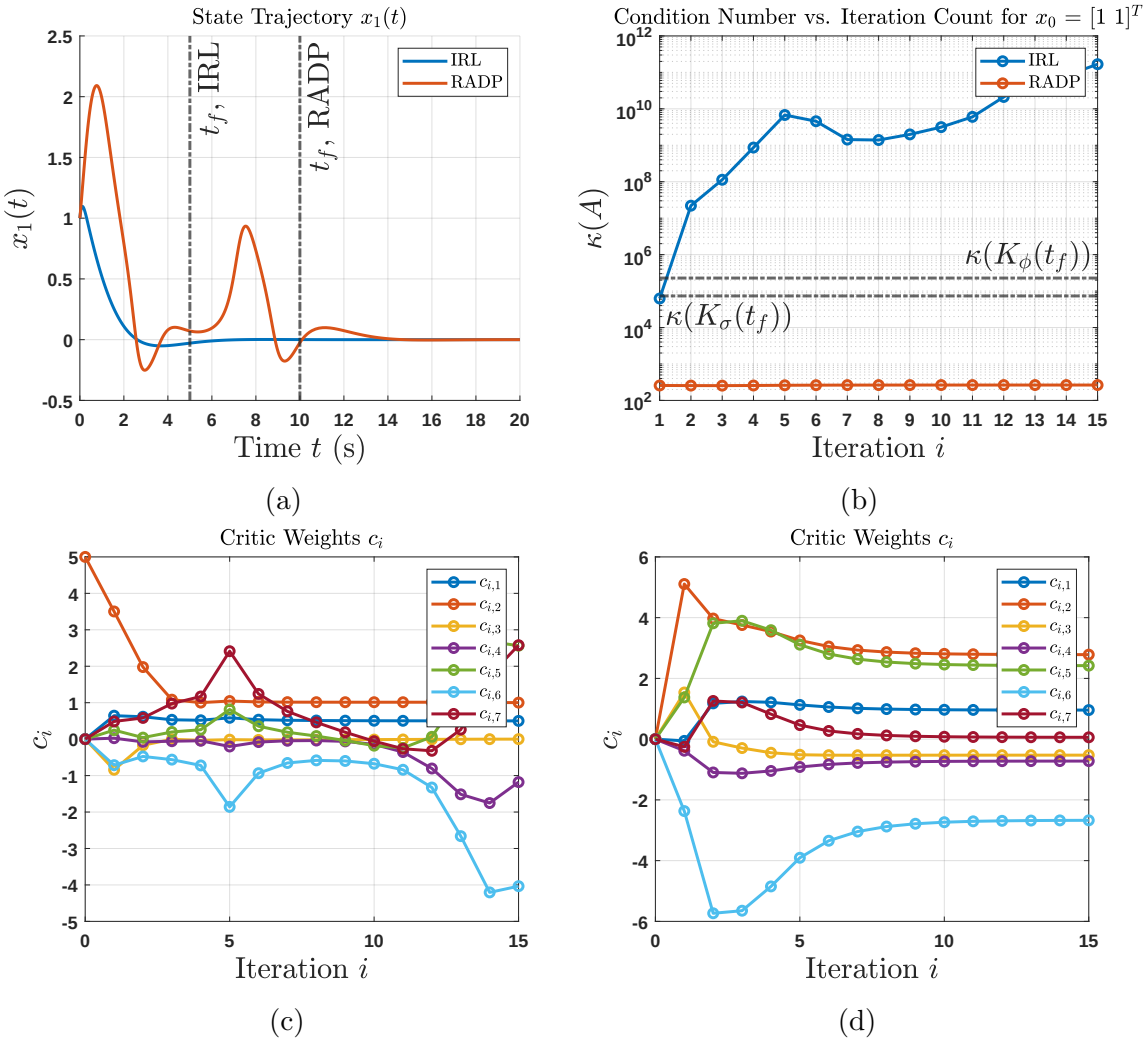


Figure 2.4: Eval. 3: Data for Exploration Noise e_1 (2.55), IC $x_0 = [1, 1]^T$. Top: (a) State Trajectory $x_1(t)$. (b) Condition Number Versus Iteration Count. Bottom: Critic Weights c_i Versus Iteration Count for (c) IRL, (d) RADP.

One observes two distinct regimes of behavior for IRL in Figure 2.4c: An initial phase from $i = 1$ to $i = 10$ iterations where the weights oscillate and drift, and a second phase from $i = 10$ to $i = 15$ iterations where the weights begin to diverge. The reason for the latter weight divergence is clear upon viewing the corresponding $x_0 = [1, 1]^T$ state trajectory data in Figure 2.4a. For IRL's learning on $t = [0, 5]$, the

latter third of the trajectory is near zero. We have thus encountered the same scenario as in Section 2.6.1, where IRL struggles with conditioning in latter iterations as the state trajectory approaches the origin. Now that the basis order has increased and the basis functions can no longer achieve exact approximation, IRL cannot perform its learning quickly enough to outpace data quality degradation.

The observed weight oscillations/drifting in the first $i = 10$ iterations of Figure 2.4c are readily explained by the iteration-wise condition number of $\mathbf{A}^i_{IRL}$ plotted in Figure 2.4b. Here we observe that even for early iterations (when the quality of trajectory data is relatively high), the conditioning of the IRL problem exceeds 10^4 across the board. By comparison to the previous evaluation (for which conditioning begins on the order of 10^2) this is a 100-fold increase.

Table 2.9 displays the mean and standard deviation of each of the final critic weights $c_{f,j}$, $j = 1, \ldots, 7$. As a result of poor conditioning, IRL exhibits large standard deviations for the latter four weights. However, we do note that the weights $c_{f,1}$, $c_{f,2}$, and $c_{f,3}$ exhibit mean values of 0.5, 1, and ~ 0, respectively, with near-zero standard deviation. These mean values agree with the respective values of the optimal weights c^* for the exact basis (2.47). The critic weights for RADP, by contrast, have converged in Figure 2.4d. Unfortunately, the values to which the weights converge are not consistent across the IC sweep (Table 2.9).

Table 2.9: Eval. 3: Critic Weight Mean, Standard Deviation

Algorithm	Data	$c_{f,1}$	$c_{f,2}$	$c_{f,3}$	$c_{f,4}$	$c_{f,5}$	$c_{f,6}$	$c_{f,7}$
IRL	mean	0.50	1.00	-3.1e-04	-0.58	0.87	-1.89	1.04
	std	1.6e-04	9.0e-04	4.8e-04	0.57	1.03	1.33	1.07
RADP	mean	0.70	1.63	-0.24	-0.32	0.54	-1.11	1.03
	std	0.19	0.39	0.25	0.45	0.91	0.94	0.82

Having rounded off the weight convergence analysis of IRL and RADP, it now remains to examine the approximation errors of the final critics $\hat{V}_f(x) = \hat{V}(x, c_f)$

(2.10) they produce in relation to the optimal value function V^* (2.44). We display these functions in Figure 2.3b for exploration noise e_1 (2.55) and IC $x_0 = [1,\, 1]^T$. Unfortunately, both algorithms exhibit wide variation. Indeed, neither critic $\hat{V}_f$ is even positive definite.

SPI, CT-VI Unfortunately, the tuning of both of these algorithms breaks down for this example. We run SPI for the IC $x_0 = [1,\, 1]^T$. After $t = 35$ seconds of tuning, the weights are $c(35) \approx w(35) = [0.771,\, 1.972,\, 0.345,\, -0.234,\, -0.154,\, -0.0826,\, -0.179]^T$. If the simulation is continued beyond $t = 35$s, the state trajectory diverges. A similar phenomenon prevails regardless of the IC x_0 in the sweep, the eventual divergence occurring within the first couple dozen seconds of simulation. We test the policy $\hat{\mu}(x, w(35))$ (2.13) corresponding to the actor weights $w(35)$ without any exploration noise to find that it is indeed stabilizing on $[-1, 1]^2$. Thus, although the SPI tuning has kept the actor weights $w(t)$ stabilizing from $t = 0$ to $t = 35$, the actor network shortly after $\hat{\mu}(x, w(35))$ is unable to reject the exploration noise e_1 (2.55) amidst the unstable dynamics of the system (2.43). This is discouraging, given that the exploration noise e_1 (2.55) chosen has the smallest amplitude of any tested in this work. Indeed, SPI performs quite well for this exploration noise in previous evaluations.

For CT-VI, regardless of our choice of hyperparameters (i.e., probing noise e, learning time t_f) the weight tuning laws (2.32) and (2.34) diverge. We present CT-VI's average condition number data for the standard choices $e = e_1$ (2.55) and $t_f = 10$s in Table 2.7. Compared to the previous evaluation, the average condition number of $K_\sigma(t_f)$ for the exploration noise e_1 has nearly doubled from 6.9×10^6 to 10.6×10^6. Due to the increase in critic basis dimension from $N_1 = 4$ to $N_1 = 7$, the condition number of $K_\phi(t_f)$ has increased two orders of magnitude from 366 to 6.2×10^4. Unfortunately, we conclude that these sheer condition numbers render CT-VI unusable for this example.

Limitations. **1) IRL Lack of Probing Noise Results in Hyperparameter Deadlock.** A designer assessing the performance of IRL in Figure 2.4c might be tempted to reduce the learning time t_f in hopes of restricting to higher-quality trajectory data and thereby improving weight convergence. Alas, these efforts are not fruitful. Recall in Section 2.6.1 we deduced that the regression matrix $\mathbf{A}_{IRL}^i$ (2.25) vanishes as variations in the state trajectory samples $\{x(t_k^i)\}_{k=0}^l$ vanish. This condition occurs as the differences in sample time instants $\{t_k^i\}_{k=0}^l$ converge to zero; i.e., as learning time $t_f \to 0$, final iteration $i^* \to \infty$, or number of samples $l \to \infty$. On one hand, the increase in final iteration count from $i^* = 5$ previously to $i^* = 15$ here is unavoidable: Generally speaking, higher-order problems require more iterations to converge, and we observe this example as no exception. On the other hand, there is little room to reduce the number of samples from $l = 10$, since $l \geq N_1 = 7$ is needed for full column rank in the weight update (2.25). Thus, both of these hyperparameters have been virtually minimized to ensure best conditioning possible. It is perhaps of no surprise then that we observe reducing the learning time t_f (or increasing sample count l and/or final iteration i^*, all of which we tried) only exacerbates the poor early-iteration conditioning seen in Figure 2.4b, which in turn magnifies the early-iteration weight oscillations in Figure 2.4c. With all options exhausted, we unfortunately conclude that the designer is deadlocked in an effort to balance IRL hyperparameter selection with the underlying numerics.

2) Conditioning Ceiling Causes Discrepancy Between Theoretical Approximation Results and Observed Approximation Performance. Both IRL and RADP guarantee uniform approximation of the optimal value function V^* on compact subsets (cf. Theorems 2.4.2 and 2.4.6 respectively). We note the subtlety that these approximation results are sufficient conditions which are not constructive; i.e., they don't furnish estimates of the number of basis functions N_1 required to achieve

a desired approximation $\epsilon > 0$. The critic approximation issues observed in Figure 2.3b reveal that when increasing the critic basis dimension N_1, a practical conditioning ceiling incapacitates approximation performance long before the theoretical threshold can be attained.

3) SPI Excitation Requirements Exceed Closed-Loop Stability Thresholds. Unfortunately, the SPI instability issue cannot be remedied by decreasing the exploration noise amplitude. For amplitudes ~ 3.5 and above, the state eventually diverges. For amplitudes below this threshold, the weight tuning freezes due to insufficient excitation. We further tried modulating the exploration noise by a decaying exponential term; i.e., $e(t) = 5\cos(t)e^{-at}$ for decay rate $a > 0$. Unfortunately, these efforts yield much the same qualitative behavior as varying the exploration noise amplitude. Thus, for SPI we are unable to find a balance between stability and sufficient excitation. We believe these issues to be, in part, due to the modified actor tuning law (2.28) which we had to adopt in this work (cf. Remark 2.3.1). Due to the actor tuning modifications, the original stability guarantees provided in [2] no longer apply.

Remark 2.6.2. Concluding Remarks for Evaluations 1-3, Performance Limitations. For this third evaluation, the two PI-based algorithms, IRL and RADP, can be successfully run across the IC sweep. However, underlying conditioning issues either render weight convergence inconsistent (RADP), or prevent convergence entirely (IRL). The two dynamic tuning algorithms, SPI and CT-VI, cannot execute to termination due to either closed-loop instability (SPI) or weight divergence (CT-VI). In the case of SPI, our probing noise selection has to meet the conflicting demands of disturbance rejection and sufficient excitation, for which a suitable balance cannot be sought. In the case of CT-VI, conditioning degradation associated with increased problem complexity has rendered its tuning laws numerically intractable. The new phenomena observed for SPI aside, all of the key issues witnessed here are direct out-

growths of the novel diagnosis we performed in our first evaluation, now compounded by the long-understood curses of dimensionality and approximation.

2.7 Performance Evaluation and Analysis – Cart Inverted Pendulum System

Having thoroughly analyzed the second-order academic example (2.43), we now apply the fundamental design insights gained to the cart inverted pendulum system (2.58), a benchmark control problem which has direct implications in real-world applications. We at first ran tests over the initial cart positions $\mathrm{x}_0 = [-1 : 0.1 : 1]$ m, pendulum angles $\theta_0 = [-30 : 1 : 30]$ deg, and zero initial translational/rotational velocities $\dot{\mathrm{x}}_0 = 0$ m/s, $\dot{\theta}_0 = 0$ deg/s. However, in our analyses we encountered algorithm performance issues of similar nature regardless of the IC chosen. Thus for the purposes of illustration, here we examine the initial condition $x_0 = [1\mathrm{m},\ 0,\ 15^\circ,\ 0]^T$.

Results. **IRL** We recall from the analysis presented in Section 2.6.3 that, in general, IRL conditioning degrades with increasing number of iterations i^*. Running IRL for this example at $i^* = 2$ is not problematic (although, of course, $i^* = 2$ is insufficiently many iterations for convergence). Running IRL for $i^* \geq 3$, on the other hand, yields a new issue which has not been observed previously. We examine the state trajectory data in Figure 2.5. Eventually, after $i = 2$ iterations, the critic weights are no longer stabilizing, and state trajectory diverges. Due to poor conditioning (observed on the order of 10^6 for this example), the critic weights c_i oscillate drastically, which in turn makes the policies $\hat{\mu}_i$ update abruptly and excite natural inverted pendulum instability. Corroborated by our insights gained in Section 2.6.3, decreasing the collection time t_f or increasing the number of samples l or number of iterations i^* only worsens the conditioning for this example.

SPI exhibits new behavior in this example as well. The weight tuning is observed to freeze (i.e., the weights $c(t)$, $w(t)$ remain virtually constant over $[0, t_f]$). After

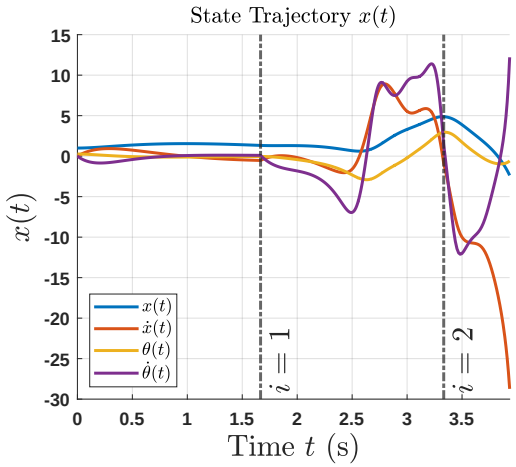


Figure 2.5: Eval. 4: IRL State Trajectory $x(t)$ for IC $x_0 = [1,\ 0,\ 15°,\ 0]^T$.

diagnosis, we find the culprit to be the term $\sigma_2/m_s^2 = \sigma_2/(\sigma_2^T\sigma_2 + 1)^2 \in \mathbb{R}^{N_1}$ in the critic tuning law (2.26) vanishing across the state trajectory $x(t)$. The critic tuning law (2.26) is a Levenberg-Marquartdt algorithm modified by the authors [2] where $(\sigma_2^T\sigma_2 + 1)^2$ is used for normalization instead of the usual $(\sigma_2^T\sigma_2 + 1)$ (cf. [2, below Equation (23)]). Unfortunately, it is the squaring of this normalization term (alongside $\sigma_2 \in \mathbb{R}^{N_1}$ having large norm across the trajectory) which has caused the vanishing.

Naturally, a designer will increase the amplitude of the probing noise e or the critic tuning gain $\alpha_1 > 0$ in an attempt to unfreeze the critic weight learning. Unfortunately, increasing the amplitude of e from 5 to 7.5 does not fix the issue, and as we have noted in Section 2.5.2, increasing the amplitude beyond this point makes the closed-loop system unstable. Meanwhile, the critic weights still freeze when increasing the tuning gain by a factor of 100 from $\alpha_1 = 10$ to $\alpha_1 = 1000$ (and for any intermediate selections we tried). This analysis suggests that the modifications made to the Levenberg-Marquartdt algorithm by [2] in the tuning law (2.26) have frozen the critic weight learning for this higher-order example.

RADP Regardless of the IC or hyperparameters chosen, we were unable to yield a stabilizing controller $\hat{\mu}_f = \hat{\mu}_{i^*+1}$ from RADP for this system. Similarly to the third evaluation (cf. Table 2.9), RADP's final weight values are observed to be highly sensitive to IC x_0. Furthermore, for some ICs (e.g., $x_0 = [1,\ 0,\ 30^\circ,\ 0]^T$), the weights oscillate indefinitely and fail to converge. These convergence and stability issues are likely a result of conditioning. For $x_0 = [1,\ 0,\ 15^\circ,\ 0]^T$, we observed the condition number of $\mathbf{A}^i_{RADP} \in \mathbb{R}^{l\times(N_1+N_2)}$ (2.31) to exceed 10^8 for each iteration i (an increase of six orders of magnitude from the previous evaluation). Given that the increment in critic basis dimension is relatively minor ($N_1 = 7$ in Section 2.6.3 to $N_1 = 10$ here), we attribute this drastic conditioning degradation to the increase in actor basis dimension (from the minimal $N_2 = 2$ actor basis in previous evaluations to the realistic $N_2 = 24$ here).

CT-VI Previously-identified numerical issues persist as we transition toward a real-world design problem. Regardless of hyperparameter selections, the weight responses diverge. For the natural choices listed in Section 2.5.2, running the algorithm for the initial condition $x_0 = [1,\ 0,\ 15^\circ,\ 0]^T$ yields condition numbers of 7.8×10^5 for $K_\phi(t_f)$ and 5.2×10^{13} for $K_\sigma(t_f)$. Unfortunately, conditioning has claimed its last victim, concluding our analysis of this example.

2.8 Discussion

This work provides an extensive review of four seminal CT-RL control methods (IRL [1], SPI [2], RADP [3], and CT-VI [4]), discussing the key theoretical assumptions and results. Our review shows these methods to be well-principled in approach, each offering an impressive suite of theoretical guarantees. All algorithms guarantee uniform convergence to the optimal value and policy, which extends beyond the baseline weight convergence results seen in the RL literature. Furthermore, each en-

sures closed-loop stability in one of its various notions. RADP even provides stability robustness results.

These theoretical successes aside, our first-of-its-kind analytical framework illustrates through comprehensive evaluation studies a fundamental divergence between CT-RL theoretical guarantees and controller synthesis performance. Our in-depth evaluations lead us to posit that it is ultimately this analysis/synthesis discrepancy which underpins the fundamental CT/DT gap in the RL control literature. As we experienced difficulties in achieving realistic control performance goals when implementing these algorithms on small-scale systems, we further analyzed step by step what hinders their performance and why. Our observations are summarized below:

Challenges Facing the CT-RL Optimal Control Problem. To the credit of the existing CT-RL algorithms, the CT-RL optimal control problem is considerably more difficult than its DT-RL counterpart. Altogether, 1) combating the multitude of structural challenges in continuous state, action, and time, alongside 2) the usual dimensionality, approximation, and PE issues, while 3) rigorously proving convergence, closed-loop stability, and other controls-centric performance guarantees proves to be a three-pronged challenge perhaps unparalleled by any other problem in control systems.

Design Challenges and Performance Limitations Facing Current CT-RL Algorithms. 1) Systematically Achieving PE Proves Difficult. As noted in Remark 2.4.4, there does not exist a systematic way to ensure the PE condition for nonlinear systems. The evaluations conducted here reaffirm the severity of this challenge. Indeed, as manifested empirically by the conditioning data seen in Table 2.7, collecting quality state trajectory data proves difficult, even for low-order systems and bases. The challenge becomes especially acute for open-loop unstable systems, where the designer must balance excitation within the confines of the disturbance

rejection capabilities of the controller. Here we note that SPI, which tunes its weights via gradient descent and hence does not face conditioning concerns, still exhibits significant PE issues, its weights either freezing due to insufficient excitation or failing to stabilize when the excitation is increased.

2) Underlying Complexity of Existing Algorithms Causes Performance Limitations. This work showcases the promising theoretical guarantees offered by existing CT-RL algorithms. However, significant algorithm complexity is required in order to prove these guarantees, resulting in numerical problems (e.g., CT-VI's nested pseudoinversion/integration tuning structure). Indeed, we pose conditioning issues as central – and intrinsic – to these algorithms and their performance shortcomings. In reality, the overly-complex nature of these algorithms makes them intractable.

3) Large Number of Hyperparameters Hinders Systematic Design. Another side effect of the observed algorithm complexity is the large number of hyperparameters required by each. For example, SPI requires the designer to choose the learning time t_f, probing noise e, tuning parameters $\alpha_1, \alpha_2 > 0$, $F_1 > 0$, $F_2 = F_2^T > 0$, as well as initial weights c_0, w_0 – finding a selection which yields convergent weights is a challenge in and of itself. We attempt to systematically select hyperparameters in Section 2.5 and justify our rationale (e.g., selecting smaller learning time t_f for IRL to avoid data quality degradation, choosing a larger learning time t_f for SPI to allow its gradient-descent tuning laws to converge, etc.), but as we encounter performance issues these efforts inevitably give way to haphazard algorithm-specific troubleshooting. Ultimately, not being able to systematically select hyperparameters to achieve good performance for these design algorithms defeats the purpose of their theoretical guarantees.

4) Dimensional Scalability Issues Limit Real-World Applicability. Bellman's curse of dimensionality has long explained scalability issues, but these algorithms

exhibit severe numerical breakdowns to even small increments in problem dimension (e.g., addition of one basis function to the critic). Each eventually experiences weight convergence issues and resultantly large approximation errors. Solutions are found to be highly sensitive to initial conditions, signaling difficulty for generalizability.

Directions of Future Research. The limitations of CT-RL algorithms illustrated by this work motivate several potentially fruitful and compelling directions of future research:

1) Leveraging Established Classical Results. CT-RL is at the very early stages of development. Practically-useful RL design methods validated by systematic performance evaluations are needed. To this end, in the near future RL algorithm development may benefit from adapting/incorporating classical and model-based architectural features. Such innovations will allow RL algorithm designers to draw from well-established and practically-tested classical theory to provide much-needed insights on RL controller synthesis and to shed light on performance guarantees/limitations. Conducting transparent "apples-to-apples" performance comparisons with classical techniques is necessary to formalize CT-RL as a control method.

2) Taking Advantage of Modeling or Models. By virtue of capturing the interacting dynamics between the agent and the environment, a well-developed model may allow the learning controller to more efficiently explore the state space and thereby improve value function approximation. Such models can be obtained from an effective and efficient system identification process, or from a first-principles physical model of the environment such as a kinematic model in the case of mechanical/robotic systems.

Modeling may bring several additional benefits: 1) Modeling may reduce the learning controller's demand for training data and/or reduce the stringency of PE requirements, thereby mitigating data deficiencies which commonly arise and hinder

learning control performance. 2) Incorporating a well-defined model structure directly in the algorithm design may alleviate the numerical complexity issues illustrated in this work to some extent. 3) An offline controller may be designed first to be used before the online learning controller adapts to the specific task needs, and 4) The learning controller may take advantage of system dynamics by rolling-out or planning ahead in solving certain sequential decision and control problems.

Yet, it is important to note that a poorly-constructed model may defeat these purposes and may even introduce additional adverse effects to the resulting controller. Unfortunately, few systematic evaluations have been conducted on the effect of modeling errors due to approximation by a neural network or other modeling methods. Instead, often the existing neural-network dynamical approximation works simply assume that the approximation error is within an ϵ bound, after which control results are obtained by illustrating on small, handcrafted examples. Oftentimes these works use simple neural networks with radial basis functions, which have rarely been associated with the most recent successes of neural network applications. Systematic approaches and evaluations are called for to examine the trade-offs between the advantages of incorporating a model versus the adversities induced by inevitable model inaccuracy. These issues are out of the scope of the current work, as little results have been developed to directly account for them in a realistic problem-solving context.

3) Exploring Nonlinear Network Structures for Improved Approximation /Scaling Performance. Each of the four methods studied here requires a linearly-independent basis, which is by comparison a strong requirement, since almost none of the successful demonstrations of RL control rely on linearly-independent bases. Furthermore, CT-RL methods for control almost universally employ single-layer, linear NN approximation structures (such as polynomial basis functions), which again is not representative in comparison to the approximation capabilities of deep networks.

Future works which relax such assumptions and take advantage of deep networks could perhaps improve the dimensional scaling and approximation issues exhibited by current CT-RL algorithms. Along this vein, a fruitful future area of study may try to explain why computation-based methods such as (deep) neural networks are effective, at least in case studies and benchmark problems.

4) Performing Systematic Comparative Studies of CT-RL and DT-RL Algorithms. As illustrated in Section 1, DT-RL methods have achieved great successes in a variety of controls applications. Having now examined their CT counterparts, a future study which delves into their inherent differences could perhaps shed light on the CT/DT gap and thereby uncover new insights for future CT-RL algorithm development. There is a need to investigate at the fundamental level what causes a loss of learning efficiency in CT-RL methods, and how to flexibly collect data, reuse data, and remove various theoretical constraints posed as assumptions in developing the major control results.

Conclusion. These four CT-RL works are instrumental and have inspired an ever-increasing number of follow-up results. This novel study has illustrated some key areas in which to improve the seminal ideas developed by these works.

Chapter 3

EXCITABLE INTEGRAL REINFORCEMENT LEARNING (EIRL)

Having now conducted the first-of-its-kind CT-RL numerical diagnosis in Section 2, we arrive at the fundamental performance limitations of ADP-based CT-RL algorithms enumerated in the Introduction Section 1. In sum, these algorithms face significant design challenges due to their complexity, numerical conditioning, and dimensional scaling issues. Despite advanced theoretical results, existing ADP CT-RL synthesis methods are inadequate in solving even small, academic problems.

The goal of this section is thus to introduce a suite of new (decentralized) excitable integral reinforcement learning (EIRL) CT-RL algorithms for control of affine nonlinear systems. Our design approach relies on three important factors. First, our methods are applicable to physical systems that can be partitioned into smaller subproblems. This constructive consideration results in reduced dimensionality and greatly improved intuitiveness of design. Second, we introduce a new excitation framework to improve persistence of excitation (PE) and numerical conditioning performance via classical input/output insights. Third, we develop a modulation-enhanced excitation (MEE) to further improve scaling and numerics through design-motivated nonsingular state transformations. We leverage Kleinman's well-tested algorithm [117] to rigorously prove convergence/stability guarantees of the developed methods. These algorithms output an optimal LQR controller K^* associated with the linearization of the nonlinear dynamics under consideration. The EIRL algorithm was originally developed in [148].

3.1 Problem Formulation

System. We consider the continuous-time nonlinear time-invariant affine system

$$\dot{x} = f(x) + g(x)u, \tag{3.1}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is the control vector, $f : \mathbb{R}^n \to \mathbb{R}^n$, and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$. We assume that f and g are Lipschitz on a compact set $\Omega \subset \mathbb{R}^n$ containing the origin $x = 0$ in its interior, and that $f(0) = 0$. We consider the quadratic cost

$$J(x_0) = \int_0^\infty (x^T Q x + u^T R u)\, d\tau, \tag{3.2}$$

where $Q \in \mathbb{R}^{n\times n}$, $Q = Q^T \geq 0$ and $R \in \mathbb{R}^{m\times m}$, $R = R^T > 0$ are the state and control penalty matrices, respectively.

LQR Problem (Background). The LQR problem considers the continuous-time linear time-invariant system

$$\dot{x} = Ax + Bu, \tag{3.3}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ are the state and control vectors, respectively, $A \in \mathbb{R}^{n\times n}$, and $B \in \mathbb{R}^{n\times m}$. We assume that (A, B) is stabilizable and $(Q^{1/2}, A)$ is detectable [136]. Under the above dynamical assumptions, the LQR optimal control u^* associated with the quadruple (A, B, Q, R) exists, is unique, and assumes the form of a full-state feedback control law [136]

$$u^* = -K^* x, \tag{3.4}$$

where $K^* = R^{-1}B^T P^*$, and $P^* \in \mathbb{R}^{n\times n}$, $P^* = P^{*T} > 0$ is the unique positive definite solution to the control algebraic Riccati equation (CARE)

$$A^T P^* + P^* A - P^* B R^{-1} B^T P^* + Q = 0. \tag{3.5}$$

Kleinman's Algorithm for Linear Systems [117]. We utilize some successive approximation concepts from Kleinman's algorithm together with state-action data (x, u) from the nonlinear system (3.1) to achieve efficient nonlinear EIRL learning. Kleinman's algorithm iteratively solves for the optimal LQR control K^* (3.4). Suppose that $K_0 \in \mathbb{R}^{m\times n}$ is such that $A - BK_0$ is Hurwitz. For iteration i ($i = 0, 1, \dots$), let $P_i \in \mathbb{R}^{n\times n}$, $P_i = P_i^T > 0$ be the symmetric positive definite solution of the algebraic Lyapunov equation (ALE)

$$(A - BK_i)^T P_i + P_i(A - BK_i) + K_i^T R K_i + Q = 0. \tag{3.6}$$

Having solved the ALE P_i (3.6), the controller $K_{i+1} \in \mathbb{R}^{m\times n}$ is updated recursively as

$$K_{i+1} = R^{-1}B^T P_i. \tag{3.7}$$

Relevant Operators for Learning.

Definition 3.1.1. For $n \in \mathbb{N}$, let

$$\underline{n} \triangleq \tfrac{n(n+1)}{2}. \tag{3.8}$$

For $l \in \mathbb{N}$ and a strictly increasing sequence $\{t_k\}_{k=0}^{l}$, whenever $x, y : [t_0, t_l] \to \mathbb{R}^n$,

define the matrix $\delta_{x,y} \in \mathbb{R}^{l \times \underline{n}}$ as

$$\delta_{x,y} = \begin{bmatrix} \big(x(t_1) + y(t_0)\big)^T \underline{\otimes} \big(x(t_1) - y(t_0)\big)^T \\ \big(x(t_2) + y(t_1)\big)^T \underline{\otimes} \big(x(t_2) - y(t_1)\big)^T \\ \vdots \\ \big(x(t_l) + y(t_{l-1})\big)^T \underline{\otimes} \big(x(t_l) - y(t_{l-1})\big)^T \end{bmatrix}, \tag{3.9}$$

where $\underline{\otimes}$ denotes the symmetric Kronecker product (cf. Appendix A). Whenever x, y are square-integrable, define $I_{x,y} \in \mathbb{R}^{l \times \underline{n}}$ as

$$I_{x,y} = \begin{bmatrix} \int_{t_0}^{t_1} x^T \underline{\otimes} y^T \, d\tau \\ \int_{t_1}^{t_2} x^T \underline{\otimes} y^T \, d\tau \\ \vdots \\ \int_{t_{l-1}}^{t_l} x^T \underline{\otimes} y^T \, d\tau \end{bmatrix}. \tag{3.10}$$

3.2 Algorithms and Training

In the classical LQR problem, Kleinman's algorithm [117] reduces the CARE, a quadratic matrix equation, to an iterative sequence of algebraic Lyapunov equations (ALEs), linear matrix equations. Inspired by Kleinman's approach, EIRL and dEIRL use state-action data generated by the *nonlinear* system (3.1) to iteratively solve the CARE associated with its linearization via a sequence of simpler linear regression problems. dEIRL reduces the dimension of the regressions by taking advantage of the decentralized dynamical structure (3.21).

Single-Injection (SI) and Multi-Injection (MI). We first note that EIRL and dEIRL can each be implemented with single-injection (SI) and multi-injection (MI) modes. As such, we propose a suite of four CT-RL algorithms in this work. Figure 3.1 shows a standard negative feedback structure, consisting of a controller K and plant

P (each of which may be linear or nonlinear). For SI, a probing noise d is injected at the plant input (cf. Figure 3.1). This is how probing noise is generally applied for CT-RL algorithms [137]. In the case of MI, a reference command r, selected by the designer to improve PE, is also injected. We will discuss these two modes further in Sections 3.4.1 and 3.4.2.

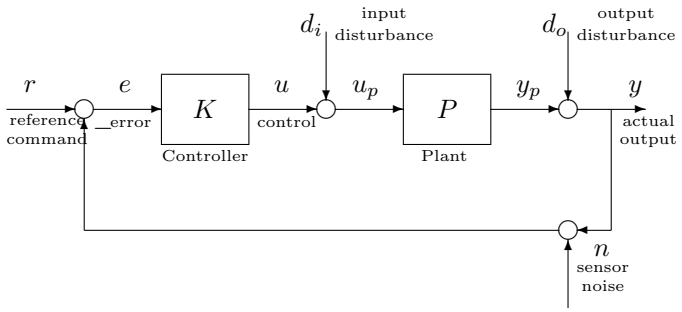


Figure 3.1: Standard Negative Feedback Structure.

Critic Network Structure. The critic network is given by

$$V(x) = (x \underline{\otimes} x)^T \operatorname{svec}(P_i) \tag{3.11}$$

where svec is the symmetric vectorization operator (cf. Appendix A), $\operatorname{svec}(P_i) \in \mathbb{R}^{\underline{n}}$ is the weight vector yielded from the EIRL learning regression (3.18) (discussed shortly), and the basis (of dimension $\underline{n}$) consists of the monomials of degree two $x \underline{\otimes} x \in \mathbb{R}^{\underline{n}}$. Applying standard symmetric Kronecker product identities (Appendix A) yields yields $V(x) = (x \underline{\otimes} x)^T v(P_i) = x^T P_i x$ – the same quadratic approximation form of Kleinman's algorithm.

Policy Structure. Once having solved for the value function approximator $V(x)$ (3.11), we construct a corresponding sequence of learning policies of the form $u(x) = -K_i x$ (see Figure 3.1). We generate these policies K_i from the critic network weights

svec(P_i) (3.18) via the nonlinear EIRL learning procedure described below.

3.2.1 Single-Injection Excitable Integral Reinforcement Learning (SI-EIRL)

Given an iteration $i \geq 0$, we use the method of integral reinforcement [1, 98, 149] to construct a learning update for the next iteration policy $u(x) = -K_{i+1}x$. Let $t_0 < t_1$ be given. The critic network approximates the integral cost J (3.2), which implies that along environment trajectories, we have

$$V(x(t_1)) - V(x(t_0)) \approx J(x(t_1)) - J(x(t_0)) = \int_{t_0}^{t_1} x^T Q x + u^T R u \, d\tau \tag{3.12}$$

The right-hand-side of (3.12), called the integral reinforcement signal, requires only state-action data (x, u) from the nonlinear system (3.1). (3.12) is exact only when $V = J$, and the learning objective is to minimize the residual network approximation error in (3.12). In order to recast (3.12) in a form amenable to regression, we first rearrange the terms in (3.1) as

$$\begin{aligned} \dot{x} &= w(x) + g(x)u + A_i x + B K_i x, \\ w(x) &\triangleq f(x) - Ax, \qquad A_i \triangleq A - BK_i. \end{aligned} \tag{3.13}$$

Here, the drift term $w(x) \triangleq f(x) - Ax \in \mathbb{R}^n$ fully captures 1) the system nonlinearities, 2) dynamical coupling, and 3) possible model uncertainties, while A, B are known nominal linearization terms of f, g in (3.1). We emphasize that the equation (3.13) is still exact to the original nonlinear dynamics (3.1). Since (3.13) contains the current-iterate policy K_i, it may be used to solve for the next-iterate policy K_{i+1} when combined with the integral reinforcement equation (3.12) as follows. We differentiate

the value function V along system trajectories, yielding

$$V(x(t_1)) - V(x(t_0)) = \int_{t_0}^{t_1} \frac{d}{d\tau} \{V(x)\}\, d\tau. \tag{3.14}$$

Along the solutions of the nonlinear system (3.1), this is

$$\begin{aligned} V(x(t_1)) - V(x(t_0)) &= x^T(t_1) P_i x(t_1) - x^T(t_0) P_i x(t_0) \\ &= \int_{t_0}^{t_1} \left[2\big(w(x) + g(x)u + BK_i x\big)^T P_i x + x^T \big(A_i^T P_i + P_i A_i\big) x \right] d\tau. \end{aligned} \tag{3.15}$$

Applying standard symmetric Kronecker product algebraic identities (cf. Appendix A) and rearranging terms, (3.15) becomes

$$\begin{aligned} &\left[-2 \int_{t_0}^{t_1} \big(w(x) + g(x)u + BK_i x\big) \mathbin{\underline{\otimes}} x \, d\tau \right. \\ &\qquad \left. + \big(x(t_1) + x(t_0)\big) \mathbin{\underline{\otimes}} \big(x(t_1) - x(t_0)\big) \right]^T \operatorname{svec}(P_i) \\ &\quad = \left[\int_{t_0}^{t_1} x \mathbin{\underline{\otimes}} x \, d\tau \right]^T \operatorname{svec}\big(A_i^T P_i + P_i A_i\big) \qquad (3.16) \\ &\quad = -\left[\int_{t_0}^{t_1} x \mathbin{\underline{\otimes}} x \, d\tau \right]^T \operatorname{svec}\big(Q + K_i^T R K_i\big), \qquad (3.17) \end{aligned}$$

where the last equality (3.17) follows from the fact that $P_i = P_i^T > 0$ satisfies the ALE (3.6). The integral reinforcement equation (3.17) is now of the required form for learning regression: The terms in brackets $\big[-2 \int_{t_0}^{t_1} \dots \big]^T \operatorname{svec}(P_i)$ contain the system trajectory integral and difference data and will form a single row of the learning matrix $\mathbf{A}_i$ (3.19), multiplied on the right by the critic weight vector $\operatorname{svec}(P_i) \in \mathbb{R}^{\underline{n}}$. Meanwhile, the term in $\operatorname{svec}\big(Q + K_i^T R K_i\big)$ requires only integral state data x and will form a single element of the learning vector $\mathbf{b}_i$ (3.20).

We now use the integral reinforcement (3.17) (which comprises a single trajectory sample) to construct the learning matrices $\mathbf{A}_i \in \mathbb{R}^{l\times \underline{n}}$, $\mathbf{b}_i \in \mathbb{R}^l$ (3.18) from l such samples. Specifically, For $l \in \mathbb{N}$ and a strictly increasing sequence $\{t_k\}_{k=0}^l$, applying (3.17) at the sample instants $\{t_k\}_{k=0}^l$ and manipulating further, we arrive at the least-squares regression

$$\mathbf{A}_i \operatorname{svec}(P_i) = \mathbf{b}_i, \tag{3.18}$$

where $\mathbf{A}_i \in \mathbb{R}^{l\times \underline{n}}$, $\mathbf{b}_i \in \mathbb{R}^l$ are given by

$$\mathbf{A}_i = -2\big[I_{x,w+gu} + I_{x,x}\big(I_n \underline{\otimes} BK_i\big)^T\big] + \delta_{x,x}, \tag{3.19}$$

$$\mathbf{b}_i = -I_{x,x} \operatorname{svec}(Q_i), \qquad Q_i \triangleq Q + K_i^T R K_i. \tag{3.20}$$

Here, $I_{x,\bullet} \in \mathbb{R}^{l\times \underline{n}}$ (3.10) and $\delta_{x,x} \in \mathbb{R}^{l\times \underline{n}}$ (3.9) are simply "storage" matrices containing integral data $I_{x,\bullet} \leftarrow \int_{t_{k-1}}^{t_k} x\, d\tau$ and difference data $\delta_{x,x} \leftarrow x(t_k) - x(t_{k-1})$ between trajectory samples as they appear in the integral reinforcement equation (3.17).

Having solved for the critic weights $\operatorname{svec}(P_i)$ (3.18), we update the controller K_{i+1} with (3.7), after which we return to (3.27) to yield the next-iteration weights $\operatorname{svec}(P_{i+1})$, and so on. This nonlinear EIRL learning procedure is summarized in Figure 3.2.

Remark 3.2.1 (EIRL Algorithm vs. Original IRL Formulation [1]: Probing Noise, Data Reuse). EIRL accommodates learning in controller design via appropriate probing noise injection, which the original IRL algorithm [1] does not include in its formulation. Lack of probing noise proves a practical design hindrance, as it renders proper system excitation nearly impossible [137]. Furthermore, the symmetric Kronecker product algebra derived for the term $I_{x,x}\big(I_n \underline{\otimes} BK_i\big)^T = I_{x,BK_ix}$ (3.19),

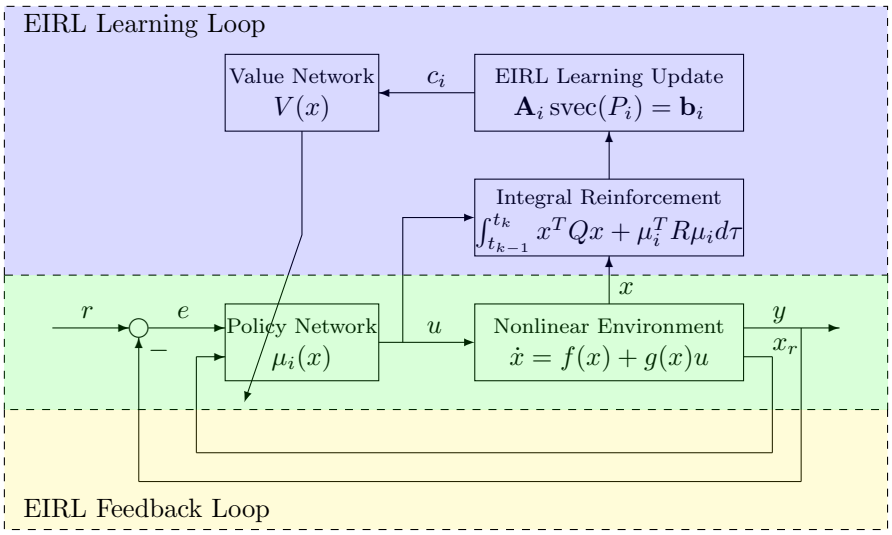


Figure 3.2: EIRL Nonlinear Learning and Feedback Structure.

enabled by Kleinman's structure [117], allows the EIRL algorithm to reuse the same state trajectory data collected under the initial stabilizing policy K_0 for generation of the sequence $\{K_i\}_{i=1}^{\infty}$. This is in contrast to the original IRL formulation [1], which for iteration i requires state-action data to be simulated under the stabilizing policy K_i before updating to K_{i+1}.

Remark 3.2.2 (EIRL Algorithm vs. Subsequent IRL Formulation [149]: Accommodating Nonlinear Systems, Dimensionality Reductions). The EIRL formulation is inspired by [149], but EIRL offers various practical improvements. Firstly, EIRL accommodates nonlinear systems, while [149] applies to linear systems only. More importantly, comparing the learning regression (3.18) with [149, Equation (11)], we see that (3.18) is lower-dimensional ($\underline{n}$ for EIRL versus $\underline{n} + mn$ in [149]). This is because the controller $K_{i+1} \in \mathbb{R}^{m \times n}$ is no longer a part of the regression vector in (3.18). As a result, knowledge of the system input dynamics g (and hence B) is required in (3.18). Our tradeoff for reduced dimensionality in exchange for system knowledge results in control solutions that were not achievable by previous methods, which struggle for low-order academic examples ($n = 2$, $m = 1$) [137].

3.3 Decentralization

In this section, we illustrate how EIRL may be readily generalized to a decentralized system which affords a physically-motivated dynamical partition with $N \geq 1$ loops.

3.3.1 Setup

Consider a decentralized nonlinear system (f, g) of the following form. We present $N = 2$ loops for illustration, but all results generalize to $N > 2$ loops:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} + \begin{bmatrix} g_{11}(x) & g_{12}(x) \\ g_{21}(x) & g_{22}(x) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{3.21}$$

No assumptions are made on dynamic coupling between the loops; i.e., the loops may be fully coupled. Here, $x_j \in \mathbb{R}^{n_j}$, $u_j \in \mathbb{R}^{m_j}$ $(j = 1, \ldots, N)$ with $\sum_{j=1}^{N} n_j = n$ and $\sum_{j=1}^{N} m_j = m$. For convenience, we define $g_j : \mathbb{R}^n \to \mathbb{R}^{n_j \times m}$, $g_j(x) = \begin{bmatrix} g_{j1}(x) & \cdots & g_{jN}(x) \end{bmatrix}$. We consider the block-diagonal Q-R cost structure

$$Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}, \quad R = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}, \tag{3.22}$$

where $Q_j \in \mathbb{R}^{n_j \times n_j}$, $Q_j = Q_j^T \geq 0$, and $R_j \in \mathbb{R}^{m_j \times m_j}$, $R_j = R_j^T > 0$ $(j = 1, \ldots, N)$. Kleinman's algorithm (Section 3.1) may be applied to a decentralized linear system (A, B):

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{3.23}$$

where we analogously define $B_j = \begin{bmatrix} B_{j1} & \cdots & B_{jN} \end{bmatrix} \in \mathbb{R}^{n_j \times m}$. This yields sequences $\{P_{i,j}\}_{i=0}^{\infty}$ in $\mathbb{R}^{n_j \times n_j}$ and $\{K_{i,j}\}_{i=1}^{\infty}$ in $\mathbb{R}^{m_j \times n_j}$ from the ALE

$$(A_{jj} - B_{jj}K_{i,j})^T P_{i,j} + P_{i,j}(A_{jj} - B_{jj}K_{i,j}) + K_{i,j}^T R_j K_{i,j} + Q_j = 0. \tag{3.24}$$

Thus, analogously to (3.11) the critic network for dEIRL is given by

$$V(x) = \sum_{j=1}^{N} V_j(x_j), \qquad V_j(x_j) = (x_j \underline{\otimes} x_j)^T \operatorname{svec}(P_{i,j}) \tag{3.25}$$

where now $\operatorname{svec}(P_{i,j}) \in \mathbb{R}^{\underline{n}_j}$ is yielded from dEIRL learning (3.27), described subsequently.

3.3.2 Single-Injection Decentralized Excitable Integral Reinforcement Learning (SI-dEIRL)

Let any loop $1 \leq j \leq N$ be given. Similarly to (3.13), rearranging terms in the decentralized nonlinear dynamics (3.21) yields

$$\begin{aligned} \dot{x}_j &= w_j(x) + g_j(x)u + A_{i,j}x_j + B_{jj}K_{i,j}x_j, \\ w_j(x) &\triangleq f_j(x) - A_{jj}x_j, \qquad A_{i,j} \triangleq A_{jj} - B_{jj}K_{i,j}. \end{aligned} \tag{3.26}$$

Given designer-selected sample count $l_j \in \mathbb{N}$ and sample instants $\{t_{k,j}\}_{k=0}^{l_j}$, following a derivation entirely analogous to that presented in Section 3.2.1, we arrive at the decentralized least-squares regression

$$\mathbf{A}_{i,j} \operatorname{svec}(P_{i,j}) = \mathbf{b}_{i,j}, \tag{3.27}$$

where the learning matrices $\mathbf{A}_{i,j} \in \mathbb{R}^{l_j \times \underline{n}_j}$, $\mathbf{b}_{i,j} \in \mathbb{R}^{l_j}$ are

$$\mathbf{A}_{i,j} = -2\big[I_{\mathcal{B}(x_j, w_j + g_j u)} + I_{\mathcal{B}(x_j,x_j)} W_{i,j}^T\big] + \delta_{x_j x_j}, \tag{3.28}$$

$$\mathbf{b}_{i,j} = -I_{\mathcal{B}(x_j,x_j)} v(Q_{i,j}), \qquad Q_{i,j} \triangleq Q_j + K_{i,j}^T R_j K_{i,j} \tag{3.29}$$

and where $I_{x_j,\bullet} \in \mathbb{R}^{l \times \underline{n}_j}$ (3.10) and $\delta_{x_j,x_j} \in \mathbb{R}^{l \times \underline{n}_j}$ (3.9) are defined analogously to the respective matrices in the EIRL update (3.27). Having solved for the critic weights $\mathrm{svec}(P_{i,j})$ (3.27), we update the policy analogously to (3.7):

$$K_{i+1,j} = R_j^{-1} B_{jj}^T P_{i,j}. \tag{3.30}$$

Table 3.1: Data and Dynamical Information Required

System Type	EIRL		dEIRL Loop j	
	Data	Dyn [a]	Data	Dynamics [b]
Nonlin, Coupled	(x, u)	f, g	(x, u)	f_j, g_j
Lin, Coupled	(x, u)	B	(x, u)	A_{jk} $(k \neq j)$, B_j
Nonlin, Decoup	(x, u)	f, g	(x_j, u_j)	f_j, g_{jj}
Lin, Decoup	(x, u)	B	(x_j, u_j)	B_{jj}

[a] For definitions: f, g (3.1), B (3.3).

[b] For definitions: f_j, g_j, g_{jj} (3.21), A_{jk}, B_j, B_{jj} (3.23).

Remark 3.3.1 (Dynamical Information of Physical Process Required). Table 3.1 summarizes the state-action data and dynamical information required to run EIRL and decentralized EIRL (dEIRL) in loop $1 \leq j \leq N$. These physics-based algorithms require a nominal model (f, g), which in turn leads to a nominal linearization (A, B). However, they utilize state-action data (x, u) from the actual physical process to learn the optimal policy for the true nonlinear dynamics. In the case the system is linear, EIRL does not require knowledge of the drift dynamics A, a common feature of IRL-based algorithms [1, 98, 149]. Meanwhile for dEIRL, in the case

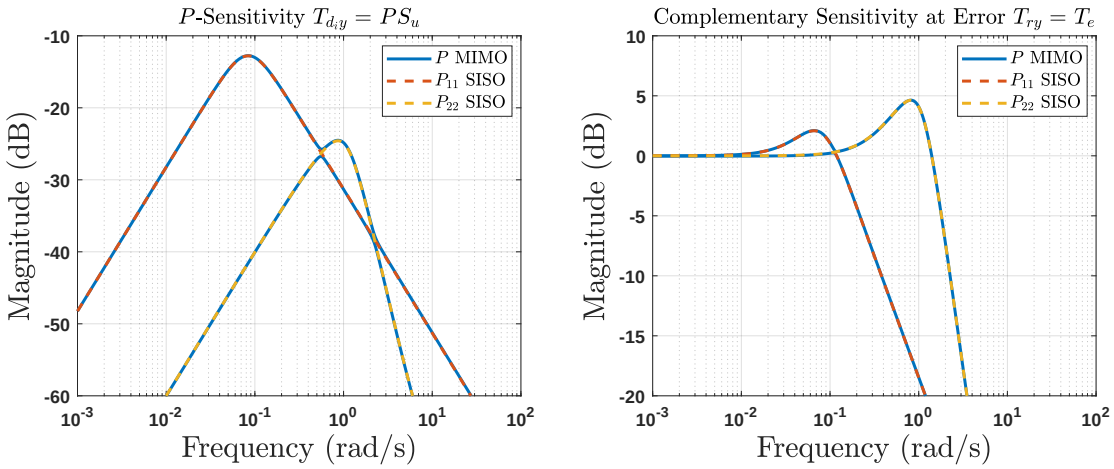
that the system is linear of the form (3.23), then $w_j(x) = \sum_{k \neq j} A_{jk} x_k$. Knowledge of the dominant diagonal drift dynamics term A_{jj} is no longer required; rather, the designer only requires knowledge of the off-diagonal dynamics A_{jk}. In the case that the system (3.21) is decoupled, f_j is a function of x_j only. Furthermore, in this case $g_j(x)u = g_{jj}(x_j)u_j$. Altogether, this says that only collection of state-action data (x_j, u_j) is required to run dEIRL in loop j, rather than the entire dataset (x, u). Thus, dynamical decoupling implies algebraic decoupling in the dEIRL algorithm.

3.4 Multi-Injection (MI)

3.4.1 Probing Noise Injection: Insights on a Fundamental Conflict between RL and Classical Control Principles

PE requirements are often invoked in proofs of CT-RL algorithm convergence in ADP [1–4]. To achieve PE, in ADP it has long been standard practice to apply a probing noise d to the system (3.1). Meanwhile in machine learning, extensive exploration in the environment resulting in a vast number of data samples during learning practically achieves PE [7, 8]. Unfortunately, the empirical analysis in [137] illustrates that a lack of constructive design to meet the requirement of PE introduces myriad practical design complications in ADP, even for simple second-order examples. To gain insights on the issue, we first define a few relevant closed-loop maps. Returning to Figure 3.1 in the case of a linear plant P and controller K, we define the P-sensitivity $PS_u = T_{dy}$ as the closed-loop map from plant input disturbance d to plant output y, and we define the complementary sensitivity at the error $T_e = T_{ry}$ as the closed-loop map from reference command r to plant output y [136]. To illustrate typical input/output behavior, Figure 3.3 shows these two closed-loop frequency responses for the nominal HSV design in Section 4 – both the exact MIMO frequency

response (blue solid curve) and SISO approximations in each of the loops (dashed curves).



(a) (b)

Figure 3.3: Closed-Loop Frequency Responses: The Probing Noise Injection Issue, Visualized. (a): P-Sensitivity $PS_u = T_{d_iy}$. (b): Complementary Sensitivity at the Error $T_e = T_{ry}$.

Let us examine the frequency response in loop $j = 2$ of the HSV (associated with flightpath angle $y_2 = \gamma$, yellow dashed curve), which will turn out to be the most numerically-troublesome in Section 4. Since probing noise is inserted at the *plant input*, the effective closed-loop map from probing noise d to output y is the P-sensitivity T_{dy}. With this in mind, one glance at the yellow dashed SISO T_{dy} response in Figure 3.3a immediately reveals issues: Any probing noise will be attenuated by at least -25 dB (a factor of about 20) as a best-case, and only so near frequencies of $\omega \approx 1$ rad/s. Even more troubling, any probing noise frequency content below $\omega \approx 10^{-1}$ rad/s and above $\omega \approx 2.5$ rad/s will be attenuated by more than -40 dB – a factor of 100. In light of this simple linear classical analysis, it is no wonder that achieving sufficient system excitation via probing noise injection proves to be a significant issue for the nonlinear flightpath angle learning in loop $j = 2$.

This real-world example illustrates a broader and more fundamental divergence

between RL and classical control. From a classical perspective, a control designer is *pleased* by the P-sensitivity in Figure 3.3a; indeed, good input disturbance rejection is a pillar of sound control system design [11, 136]. On the other hand, from an RL perspective, a designer is *troubled* by such a P-sensitivity response, for a best-case attenuation of -25 dB means that sufficient system excitation is likely impossible to achieve within reasonable bounds of control effort. Hence, we arrive at the unfortunate conclusion that the now-standard RL practice of probing noise injection has pitted learning requirements *diametrically* at odds with classical control principles. This motivates MI designs.

3.4.2 Multi-Injection (MI) EIRL and dEIRL

In light of the fundamental conflict illustrated in Section 3.4.1, the question naturally arises: How may RL control algorithms be modified to accommodate an excitation framework which is in alignment with classical control principles? The remedy lies again in application of simple input-output intuitions. Just as input-disturbance rejection is a pillar of good control system design, so too is (low-frequency) reference command following [11, 136]. Graphically representative of this principle is the complementary sensitivity T_{ry} plotted in Figure 3.3b, which remains near 0 dB (i.e., unity $y \approx r$) at low frequencies. We hence pose the following *multi-injection* solution: Instead of inserting only a probing noise d at the plant input, we apply the control $u = -Ke + d$, where the excitation $Ke = K(r - y)$ (Figure 3.1) results from insertion of a designed reference command r.

MI offers multiple benefits: First and foremost, reference command injection allows the designer to modulate system excitation via the complementary sensitivity map T_{ry}, which is much favorable in comparison to the P-sensitivity T_{dy} from an input-output perspective (Figure 3.3). Second, the designer now has multiple excita-

tion "knobs" (namely, the usual probing noise d in addition to the reference command r) to tweak in order to improve data quality. These excitations d, r may be deterministic (as we choose in our evaluations below, for continuity with the leading CT-RL works [1–4] and to exploit the input-output insights discussed), or they may be chosen as random noises if the designer sees fit for their application. In light of observed CT-RL PE issues [137], this increased excitation flexibility is of great practical use.

Lastly, by virtue of lumping the additional reference command term r in the control u, MI doesn't explicitly alter RL algorithm theoretical formulations and can be readily implemented on nonlinear compensators in the general RL context. Specifically, suppose a CT-RL algorithm has been formulated under the standard probing noise injection excitation framework; i.e., requires application of a control of the form $u = \mu(x) + d$ for some stabilizing policy μ. Since the optimal control problem already requires full-state information, it is not restrictive to designate a subset of the state x as measurement variables y for reference injection. Suppose that $p \leq n$ variables of the state x have been chosen so, and assume after possibly re-indexing that x has the form $x = \begin{bmatrix} y^T & x_r^T \end{bmatrix}^T$, where $x_r \in \mathbb{R}^{n-p}$ denotes the rest of the state. If we apply a reference excitation r to the closed-loop system, note that the control

$$u = \mu(e, x_r) + d = \mu(y, x_r) + \tilde{d}, \tag{3.31}$$

$$\tilde{d} \triangleq d + (\mu(e, x_r) - \mu(y, x_r)) \tag{3.32}$$

is also of the form $u = \mu(x) + \tilde{d}$ required for execution. Since the probing noise d is an excitation signal in RL formulations, the choice $d = \tilde{d}$ is permissible. Thus, MI is a viable candidate for improving PE properties of existing RL methods involving the usual probing noise formulation. In exchange for this enhanced MI excitation tool, computational burden has increased slightly if the compensator K is dynamic

(which is the case, e.g., when integral augmentation is performed [11, 136]). In this case, the excitation Ke needs to be simulated dynamically online, in addition to the usual response Ky. However, we make the important notes that in spite of this slight computational cost, MI does not affect dimensionality of the underlying learning problem, nor does it impose additional requirements on system dynamics knowledge.

To conclude this section, we summarize the EIRL and dEIRL execution procedure in Algorithm 6, both in their SI and MI modes.

Algorithm 6 EIRL/dEIRL Algorithm.

Hyperparameters: (Selection methods in Section 4.1).

- $N \in \mathbb{N}$ loops ($N = 1$: EIRL, $N > 1$: dEIRL).

In each loop $j = 1, \dots, N$:

- Cost Q_j, R_j (3.2), stabilizing controller $K_{0,j}$ (3.30).
- Learning Params: Sample period $T_{s,j}$, number of samples l_j, number of iterations i_j^* (cf. Table 4.1).
- Excitations: Probing noise d_j, plus reference command r_j for MI.

Data Collection: In each loop $j = 1, \dots, N$, apply

if SI **then**
    d_j
else if MI **then**
    d_j, r_j
end if

to system (3.1), starting from initial condition $x_0 \in \mathbb{R}^n$ simulating under initial stabilizing policy $K_{0,j}$, collecting environment state-action data $\left\{\left(x_j(kT_{s,j}), u_j(kT_{s,j})\right)\right\}_{k=0}^{l_j}$.

Learning:

for each loop $j = 1, \dots, N$ **do**
    for each algorithm iteration $i = 0, \dots, i_j^*$ **do**
        Perform regression $\operatorname{svec}(P_{i,j})$ (3.27).
        Update critic NN $V_j(x_j) \leftarrow (x_j \operatorname{svec} x_j)^T \operatorname{svec}(P_{i,j})$ (3.25).
        Perform inversion $\operatorname{svec}(P_{i,j}) \mapsto P_{i,j}$ (Prop. **??**).
        Update policy $K_{i,j} \leftarrow K_{i+1,j}$ (3.30).
    end for
end for

Termination: Final policies $K_{i_j^*,j}$ ($j = 1, \dots, N$).

3.5 Modulation-Enhanced Excitation (MEE)

In this section, we develop our novel modulation-enhanced excitation (MEE) framework, which uses invertible transformations of the system state variables in order to modulate the (d)EIRL regression (3.27) for improved algorithm conditioning. This framework makes extensive use of the new symmetric Kronecker product algebra developed in Appendix A.

3.5.1 A Motivating Example

First, we motivate the need for the developed MEE framework via an illustrative example. Consider the system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -0.1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{3.33}$$

The system (3.33) is a diagonal linear system consisting of a high-bandwidth loop $j = 1$ (associated with x_1, u_1) and a low-bandwidth loop $j = 2$ (associated with x_2, u_2). Such weakly-coupled two-loop systems with a decade separation in bandwidth are quite common in real-world applications (e.g., the HSV example of Section 4).

Oftentimes, control/actuator saturation is a major concern for designing in hardware, especially for RL excitation purposes in lower-bandwidth loops. For example, in robotics applications, designers must consider the load specifications of the motors/servomechanisms, which have significant impact on achievable closed-loop performance [43, 44]. In aerospace applications, control surface deflections have strict bounds to avoid aerodynamic stall [122]. To account for control saturation concerns in this example, suppose the designer has constraints of $u_1(t) \in [-1, 1]$ in the high-bandwidth loop $j = 1$ and $u_2(t) \in [-0.1, 0.1]$ in the low-bandwidth loop $j = 2$. We

consider natural designer first-choices for the cost structure: $Q = I_2$, $R = \texttt{diag}(1, 10)$, where the larger control penalty on u_2 reflects the designer's control saturation concerns in this channel. Similarly, for EIRL's hyperparameters, we select a sample period $T_s = 0.1$ (i.e., a sample frequency approximately a decade above the highest-bandwidth mode in the plant), $i^* = 5$ iterations, and $l = 5$ data points. For probing noises, we choose $d_1(t) = \cos(t)$, $d_2(t) = 0.1\cos(0.1t)$, the amplitudes reflecting the designer's consideration of control saturation in each channel, and the frequencies placed at the bandwidths of the respective loop modes.

Suppose that the designer opts for a single-loop design $N = 1$ (i.e., not using decentralization and hence executing EIRL (Section 3.2.1) rather than dEIRL (Section 3.3.2). Noting that the open-loop system (3.33) is stable, Theorem 3.6.2 (developed in Section 3.6 shortly) guarantees that the stabilizing controller $K_0 = 0 \in \mathbb{R}^{2\times 2}$ will result in convergence of the (d)EIRL algorithm to the optimal controller K^*. Running EIRL, the final controller K_{i^*} converges to within 1.62×10^{-9} of the optimal K^*. However, the learning regression has a large peak condition number of 138.47 (cf. conditioning data in Table 3.2). Intuitively, the culprit stems from the max control effort requirements placed in the low-bandwidth loop $j = 2$. As a result, the designer can excite the low-bandwidth loop $j = 2$ at only a tenth the control effort of the high-bandwidth loop $j = 1$. Since the probing noise frequencies were placed at the respective loop bandwidths, the state response $x_1(t)$ in the high-bandwidth loop exhibits approximately ten times the amplitude of the response $x_2(t)$ in the low-bandwidth loop, resulting in scaling and thus conditioning issues in the regression matrix $\mathbf{A}_{i,j}$ (3.28).

The designer's insight to fix the issue is clear: The state response $x_2(t)$ in the low-bandwidth loop needs to be scaled up by a factor of ten to improve scaling. This raises the central questions: How may we address this significant scaling issue in a

systematic design framework which leverages physical insights (in this case, our saturation constraints) while achieving excitation and thus good numerical conditioning? Crucially, how can we ensure that such a framework preserves dEIRL's key theoretical convergence and closed-loop stability guarantees (to be developed in Section 3.6)?

In a real-world analogue to this scenario, the designer oftentimes has no physical means of recourse to address these conditioning issues: The excitation level in the high-bandwidth loop $j = 1$ cannot be reduced without degrading PE and hence learning performance in this loop. Furthermore, oftentimes unit scaling between unlike physical measurements renders the equilibration of responses physically intractable (e.g., in the HSV example studied in Section 4, velocity oscillations on the order of 100 ft/s are needed to achieve good PE in the translational loop, yet flightpath angle oscillations on the order of 100 deg in the rotational loop are nonsensical). This simple example illustrates that the problem runs deeper: Even when the system has been excited to the greatest possible extent, physical constraints and/or unit intermingling may still leave the learning regression poorly conditioned. These fundamental design concerns make the symmetric Kronecker product results of the next section all the more vital.

Table 3.2: MEE Motivating Example: Max/Min Conditioning

Algorithm	Loop j	$\max_i(\kappa(\mathbf{A}_{i,j}))$	$\min_i(\kappa(\mathbf{A}_{i,j}))$
EIRL	1	138.47	36.04
EIRL w/ MEE	1	14.05	7.14
dEIRL	1	1.00	1.00
	2	1.00	1.00

Before developing the results, we here demonstrate how they intuitively solve the problem: Choosing the natural modulation matrix $S = \texttt{diag}(1, 10) \in \mathrm{GL}(2)$ drastically improves EIRL conditioning, reducing it by a factor of ten from 138.47

before MEE to 14.05 after MEE (Table 3.2). Thus, using little beyond common-sense principles, MEE can offer conditioning reductions of an order of magnitude to designers using the EIRL/dEIRL algorithm.

We conclude this section by employing a decentralized design (i.e., dEIRL) in each of the $N = 2$ loops. Using identical hyperparameters, the resulting final controllers $K_{i^*,1}$, $K_{i^*,2}$ converge to within 1.38×10^{-11} and 1.49×10^{-9} of the optimal controllers K_1^*, K_2^* in each loop, respectively. Furthermore, dEIRL has unity conditioning in each loop (since the dimension of each is $\underline{n}_1 = \underline{n}_2 = 1$), illustrating the general principle that dEIRL's use of physically-motivated dynamical insights enables further learning performance improvements.

3.5.2 Kleinman's Algorithm & Modulation

Let a decentralized loop $1 \leq j \leq N$ be given, and suppose that $K_{0,j} \in \mathbb{R}^{m_j \times n_j}$ is such that $A_{jj} - B_{jj}K_{0,j}$ is Hurwitz in loop j. We may then apply Kleinman's algorithm (Section 3.1), yielding sequences $\{P_{i,j}\}_{i=0}^{\infty}$ in $\mathbb{R}^{n_j \times n_j}$ and $\{K_{i,j}\}_{i=0}^{\infty}$ in $\mathbb{R}^{m_j \times n_j}$ from the ALE

$$A_{i,j}^T P_{i,j} + P_{i,j} A_{i,j} + Q_{i,j} = 0. \tag{3.34}$$

where the matrices $A_{i,j}$ and $Q_{i,j}$ are given by (3.26) and (3.29), respectively. We have seen, vis. (A.24) of Appendix A, that the ALE (3.34) is equivalent to the following vectorized ALE regression

$$(A_{i,j} \mathbin{\underline{\oplus}} A_{i,j})^T \operatorname{svec}(P_{i,j}) = -\operatorname{svec}(Q_{i,j}). \tag{3.35}$$

Now, suppose $S = \texttt{diag}(S_1, \ldots, S_N) \in \mathrm{GL}(n)$, $S_j \in \mathrm{GL}(n_j)$ $(j = 1, \ldots, N)$, is any nonsingular coordinate transformation $\tilde{x} = Sx$, partitioned in the decentralized form

$$\tilde{x}_j = S_j x_j. \tag{3.36}$$

This induces the following transformed LQR problem in loop j, associated with the quadruple $(\tilde{A}_{jj}, \tilde{B}_{jj}, \tilde{Q}_j, R_j)$, where

$$\tilde{A}_{jj} = S_j A_{jj} S_j^{-1}, \qquad \tilde{B}_{jj} = S_j B_{jj}, \qquad \tilde{Q}_j = S_j^{-T} Q_j S_j^{-1}. \tag{3.37}$$

By similarity, the controller $\tilde{K}_{i,j} = K_{i,j} S_j^{-1}$ is such that $\tilde{A}_{i,j} = \tilde{A}_{jj} - \tilde{B}_{jj}\tilde{K}_{i,j}$ is Hurwitz. This motivates the following modulated ALE

$$\tilde{A}_{i,j}^T \tilde{P}_{i,j} + \tilde{P}_{i,j} \tilde{A}_{i,j} + \tilde{Q}_{i,j} = 0. \tag{3.38}$$

Modulation by nonsingular coordinate transformations is common practice in the study of matrix equations, oftentimes offering significant theoretical/numerical advantages for purposes of solving [150]. Nonsingular state transformations are also commonly used in classical control of multivariable control systems to improve directionality properties [136].

3.5.3 (d)EIRL & Modulation: MEE Framework

Now, consider the analogue in the (d)EIRL algorithm. Associate with the nonsingular coordinate transformation $S_j \in \mathrm{GL}(n_j)$ the transformed problem $(\tilde{f}_j, \tilde{g}_j, \tilde{Q}_j, R_j)$ in loop j, where

$$\tilde{f}_j = S_j \circ f_j \circ S^{-1}, \qquad \tilde{g}_j = S_j \circ g_j \circ S^{-1}. \tag{3.39}$$

This induces the following modulated dEIRL least-squares regression, analogous to (3.27), which we term the MEE regression for brevity:

$$\tilde{\mathbf{A}}_{i,j}\,\mathrm{svec}(\tilde{P}_{i,j}) = \tilde{\mathbf{b}}_{i,j}. \tag{3.40}$$

We will develop the properties of the MEE regression in our subsequent theoretical results of Section 3.6.

3.6 Theoretical Results

In this section, we prove key convergence, optimality, and closed-loop stability guarantees of the presented methodologies. We also prove the key symmetric Kronecker product results which enable the use of MEE (Section 3.5) to further improve algorithm numerical properties while preserving the original control solution before modulation. Throughout this section, we assume that the baseline dynamical assumptions outlined in Section 3.1 hold for well-posedness of the underlying optimal control problem.

3.6.1 Convergence, Optimality, and Closed-Loop Stability of (d)EIRL

In order to develop the convergence, optimality, and closed-loop stability guarantees of the (d)EIRL algorithm, we leverage the theoretical properties of Kleinman's algorithm:

Theorem 3.6.1 (Convergence, Optimality, and Closed-Loop Stability of Kleinman's Algorithm [117]). Let the assumptions of Section 3.1 hold. Then we have the following:

(i) $A - BK_i$ is Hurwitz for all $i \geq 0$.

(ii) $P^* \leq P_{i+1} \leq P_i$ for all $i \geq 0$.

(iii) $\lim_{i\to\infty} K_i = K^*$, $\lim_{i\to\infty} P_i = P^*$.

We now move on to EIRL. Before proceeding to the main theoretical results, we require the following two lemmas:

Lemma 3.6.1. Suppose that the controller $K_i \in \mathbb{R}^{m\times n}$ is stabilizing, and that the matrix $\mathbf{A}_i \in \mathbb{R}^{l\times \underline{n}}$ (3.19) has full column rank. Then $P_i \in \mathbb{R}^{n\times n}$, $P_i = P_i^T > 0$ is the unique positive definite solution to the ALE (3.6) if and only if P_i satisfies the least-squares regression (3.18) at equality. In particular, the least-squares solution of the EIRL algorithm (3.18) yields the solution of the associated ALE (3.6).

Proof: The forward direction was proved in (3.15)–(3.17). Conversely, suppose $\operatorname{svec}(P) \in \mathbb{R}^{\underline{n}}$ minimizes the least-squares regression (3.18). Since $\mathbf{A}_i$ has full column rank, the solution $\operatorname{svec}(P) \in \mathbb{R}^{\underline{n}}$ is unique. Moreover, letting $P_i = P_i^T > 0$ be the unique positive definite solution to the ALE (3.6), we have seen that $\operatorname{svec}(P_i) \in \mathbb{R}^{\underline{n}}$ satisfies (3.18) at equality. Thus, $\operatorname{svec}(P) = \operatorname{svec}(P_i)$. Since svec is a bijection (Proposition A.3.1), this implies $P = P_i$ is the solution to the ALE (3.6). ■

Lemma 3.6.2. Suppose that $l \in \mathbb{N}$ and the sample instants $\{t_k\}_{k=0}^{l}$ are chosen such that the matrix $I_{x,x}$ (3.10) has full column rank $\underline{n}$. If K_i is stabilizing, then the matrix $\mathbf{A}_i \in \mathbb{R}^{l\times \underline{n}}$ (3.18) has full column rank $\underline{n}$.

Proof: Follows similarly from the proof of [149, Lemma 6]. Suppose $\operatorname{svec}(P) \in \mathbb{R}^{\underline{n}}$ is such that $\mathbf{A}_i \operatorname{svec}(P) = 0$. Then (3.16) (which holds for *any* symmetric matrix) implies that $\mathbf{A}_i \operatorname{svec}(P) = I_{x,x} \operatorname{svec}(M)$, where $M \in \mathbb{R}^{n\times n}$, $M = M^T$ is given by

$$M = A_i^T P + P A_i. \tag{3.41}$$

(3.41) is an ALE, which since $M = M^T$ and since $A_i = A - BK_i$ is Hurwitz has the unique solution $P = \int_0^\infty e^{A_i^T \tau}(-M)e^{A_i \tau}\, d\tau$ [136]. Meanwhile, by full column rank of $I_{x,x}$, that $I_{x,x}\,\mathrm{svec}(M) = 0$ implies $\mathrm{svec}(M) = 0$, or $M = 0$. Since $M = 0$, we see $P = 0$, whence $\mathrm{svec}(P) = 0$. Altogether, we have shown that $\mathbf{A}_i$ has a trivial right null space, so it has full column rank. ■

Theorem 3.6.2 (Convergence, Optimality, and Closed-Loop Stability of EIRL Algorithm). Suppose that $l \in \mathbb{N}$ and the sample instants $\{t_k\}_{k=0}^{l}$ are chosen such that the matrix $I_{x,x}$ (3.10) has full column rank $\underline{n}$. If K_0 is such that $A - BK_0$ is Hurwitz, then the EIRL algorithm and Kleinman's algorithm are equivalent in that the sequences $\{P_i\}_{i=0}^{\infty}$ and $\{K_i\}_{i=1}^{\infty}$ produced by both are identical. Thus, the convergence, optimality, and stability conclusions of Kleinman's algorithm (Theorem 3.6.1) hold for the EIRL algorithm as well.

Proof: Follows by induction on i, after application of Lemmas 3.6.2 and 3.6.1. ■

Entirely analogous versions of Lemmas 3.6.1 and 3.6.2 hold for the dEIRL algorithm developed in Section 3.3.2, and hence the results of Theorem 3.6.2 extend to the dEIRL algorithm:

Theorem 3.6.3 (Convergence, Optimality, and Closed-Loop Stability of dEIRL Algorithm). Suppose for $1 \leq j \leq N$ that $l_j \in \mathbb{N}$ and the sample instants $\{t_{k,j}\}_{k=0}^{l_j}$ are chosen such that the matrix I_{x_j,x_j} (3.10) has full column rank $\underline{n}_j$. If $K_{0,j}$ is such that $A_{jj} - B_{jj}K_{0,j}$ is Hurwitz, then the dEIRL algorithm and Kleinman's algorithm are equivalent in that the sequences $\{P_{i,j}\}_{i=0}^{\infty}$ and $\{K_{i,j}\}_{i=1}^{\infty}$ produced by both are identical. Thus, the convergence, optimality, and closed-loop stability conclusions of Kleinman's algorithm (Theorem 3.6.1) hold for the dEIRL algorithm as well.

Remark 3.6.1 (dEIRL Algorithm: Decentralized Learning, with or without Dynamic Coupling). The dEIRL algorithm (via Theorem 3.6.3) guarantees convergence to the solution of the *linear* quadratic regulator problem associated with loop j: $(A_{jj}, B_{jj}, Q_j, R_j)$ from state trajectory data generated by the *nonlinear* system (f, g) (3.21), *regardless* of if (f, g) is dynamically coupled between loops $j = 1, \ldots, N$. Note that Theorem 3.6.3 involves only a fixed single loop $1 \leq j \leq N$, both in terms of assumptions and results. We in particular call attention to the crux of the hypotheses required in Theorem 3.6.3: full-column rank of the matrix $I_{x_j,x_j} \in \mathbb{R}^{l_j \times \underline{n}_j}$ (3.10). This matrix places requirements on the quality of state trajectory data x_j in loop j only. Thus, the dEIRL algorithm is truly decentralized: The loops $j = 1, \ldots, N$ may be updated entirely independently, and the designer may focus on data quality in the individual loops rather than for the aggregate system. As a result, dEIRL offers significant real-world benefits to designers in terms of dimensionality, allowing a single higher-dimensional problem to be partitioned into lower-dimensional subproblems.

3.6.2 Modulation Invariance of (d)EIRL

In this section, develop the fundamental symmetric Kronecker product results which form the basis of our proposed MEE framework (Section 3.5). The results apply in their entirety to both the EIRL and dEIRL algorithms, but we will focus on the dEIRL case here for simplicity. At this point in the development, two questions are natural: 1) How do the original sequences $\{P_{i,j}\}_{i=0}^{\infty}$, $\{K_{i,j}\}_{i=0}^{\infty}$ (3.34) output by Kleinman's algorithm relate to the modulated sequences $\{\tilde{P}_{i,j}\}_{i=0}^{\infty}$, $\{\tilde{K}_{i,j}\}_{i=0}^{\infty}$ (3.38)? Noting by Theorem 3.6.3 that dEIRL and Kleinman's algorithm are equivalent, this first question also addresses the relations between the respective sequences produced by dEIRL. And, 2) How does prescaling interact with the symmetric Kronecker product algebra developed in Appendix A? That is, how does prescaling affect the terms

in the ALE regression (3.35) and the dEIRL regression (3.27), and what structural parallels exist between the two?

Theorem 3.6.4 (Kleinman's Algorithm: Modulation Invariance). $P_{i,j} \in \mathbb{S}^{n_j}$, $P_{i,j} > 0$ satisfies the ALE (3.34) if and only if $\tilde{P}_{i,j} = S_j^{-T} P_{i,j} S_j^{-1}$ satisfies the modulated ALE (3.38).

Proof: We have seen, vis. (A.24) of Appendix A, that the modulated ALE (3.38) is equivalent to

$$(\tilde{A}_{i,j} \underline{\oplus} \tilde{A}_{i,j})^T \operatorname{svec}(\tilde{P}_{i,j}) = -\operatorname{svec}(\tilde{Q}_{i,j}). \tag{3.42}$$

Applying the symmetric Kronecker product algebra of Proposition A.3.3, we may expand (3.42) as

$$\begin{aligned}(S_j \underline{\otimes} S_j)^{-T} (A_{i,j} \underline{\oplus} A_{i,j})^T (S_j \underline{\otimes} S_j)^T \operatorname{svec}(\tilde{P}_{i,j}) \\ = -(S_j \underline{\otimes} S_j)^{-T} \operatorname{svec}(Q_{i,j}).\end{aligned} \tag{3.43}$$

By Proposition A.3.3 5S)S), we may multiply both sides by $(S_j \underline{\otimes} S_j)^T \in \mathrm{GL}(\underline{n}_j)$, yielding the equivalent regression

$$(A_{i,j} \underline{\oplus} A_{i,j})^T (S_j \underline{\otimes} S_j)^T \operatorname{svec}(\tilde{P}_{i,j}) = -\operatorname{svec}(Q_{i,j}). \tag{3.44}$$

However, from comparison of (3.44) and the symmetric vectorization of the original ALE (3.35), we conclude that $(S_j \underline{\otimes} S_j)^T \operatorname{svec}(\tilde{P}_{i,j}) = \operatorname{svec}(P_{i,j})$. Applying Proposition

A.3.3 again,

$$(S_j \underline{\otimes} S_j)^T \operatorname{svec}(\tilde{P}_{i,j}) = \operatorname{svec}(\pi(S_j^T \tilde{P}_{i,j} S_j)) \\ = \operatorname{svec}(S_j^T \tilde{P}_{i,j} S_j). \tag{3.45}$$

In all, $S_j^T \tilde{P}_{i,j} S_j = P_{i,j}$, implying the desired result. The reverse direction follows by a symmetric argument. ■

We now have a powerful answer to question 1) posed above: Kleinman's algorithm (and hence the dEIRL algorithm) is invariant with respect to nonsingular state modulation in the sense that if the sequences $\{\tilde{P}_{i,j}\}_{i=0}^{\infty}$, $\{\tilde{K}_{i,j}\}_{i=0}^{\infty}$ are generated under the modulated problem with potentially-improved numerics, then the solution sequences $\{P_{i,j}\}_{i=0}^{\infty}$, $\{K_{i,j}\}_{i=0}^{\infty}$ of the original problem may be backed out by

$$P_{i,j} = S_j^T \tilde{P}_{i,j} S_j, \qquad K_{i,j} = \tilde{K}_{i,j} S_j. \tag{3.46}$$

Furthermore, the above proof also answers question 2) in the case of Kleinman's algorithm: The modulated ALE regression (3.42) is equivalent to (3.44), in which we observe the that the original ALE regression matrix $(A_{i,j} \underline{\oplus} A_{i,j})^T \in \mathrm{GL}(\underline{n}_j)$ (3.35) is multiplied on the right by the modulation matrix $(S_j \underline{\otimes} S_j)^T \in \mathrm{GL}(\underline{n}_j)$. The regression target vector $-\operatorname{svec}(Q_{i,j}) \in \mathbb{R}^{\underline{n}_j}$ is unchanged between the original regression (3.35) and equivalent modulated regression (3.44).

The symmetric Kronecker product algebraic results developed in Appendix A are essential to the derivation of the MEE regression (3.40). In particular:

Proposition 3.6.1. The operations $\delta_{x,y}$ (3.9) and $I_{x,y}$ (3.10) satisfy the following:

1) $\delta_{Ax,Ay} = \delta_{x,y}(A \underline{\otimes} A)^T$, $A \in \mathbb{R}^{m \times n}$.

2) $I_{Ax,Ay} = I_{x,y}(A \underline{\otimes} A)^T$, $A \in \mathbb{R}^{m \times n}$.

3) $I_{Ax,Bx} = I_{x,x}(A \underline{\otimes} B)^T$, $A, B \in \mathbb{R}^{m \times n}$.

Proof: Follows from Proposition A.3.3 6S). ■

These key algebraic properties enable the basis of our proposed MEE framework:

Theorem 3.6.5 (MEE Framework and the dEIRL Algorithm: Modulation Invariance). $P_{i,j} \in \mathbb{S}^{n_j}$, $P_{i,j} > 0$ satisfies the dEIRL regression (3.27) if and only if $\tilde{P}_{i,j} = S_j^{-T} P_{i,j} S_j^{-1}$ satisfies the MEE regression (3.40). Furthermore, the original regression (3.27) and MEE regression (3.40) are related by

$$\tilde{\mathbf{A}}_{i,j} = \mathbf{A}_{i,j}(S_j \underline{\otimes} S_j)^T, \qquad \tilde{\mathbf{b}}_{i,j} = \mathbf{b}_{i,j}. \tag{3.47}$$

Proof: The first assertion follows immediately from Theorems 3.6.4 and 3.6.3. The relation (3.47) follows from application of the symmetric Kronecker product algebra developed in Propositions A.3.3 and 3.6.1. ■

Theorem 3.6.5 definitively concludes our answer to question 2) for the dEIRL algorithm and our proposed MEE framework, revealing substantial parallels to the classical Kleinman's algorithm. Crucially, the dEIRL regression matrix $\mathbf{A}_{i,j} \in \mathbb{R}^{l_j \times \underline{n}_j}$ (3.28) is mutliplied on the right by the *same* modulation matrix $(S_j \underline{\otimes} S_j)^T \in \mathrm{GL}(\underline{n}_j)$ to form the MEE regression matrix $\tilde{\mathbf{A}}_{i,j}$ (3.40). As is the case with Kleinman's algorithm, the regression target vector $\mathbf{b}_{i,j} \in \mathbb{R}^{l_j}$ (3.29) remains unchanged under MEE. Furthermore, this vector is given by $\mathbf{b}_{i,j} = -I_{x_j,x_j} \operatorname{svec}(Q_{i,j})$, which is simply the product of the integral matrix I_{x_j,x_j} (3.10) with the ALE regression vector $-\operatorname{svec}(Q_{i,j})$ (3.35). The parallelisms under which these two algorithms interact with the symmetric Kronecker product algebra developed in this work presents a significant practical advantage to real-world control designers: The same physics-based prescaling insights which readily apply to solving classical control problems may be ported directly to dEIRL's MEE framework. We summarize these key algebraic properties in Table 3.3.

Table 3.3: Kleinman's Algorithm and dEIRL: Symmetric Kronecker Product Algebraic Structure under Modulation

Term	Kleinman Loop j		dEIRL Loop j	
	Original	w/ Modulation	Original	w/ Modulation (MEE)
Dynamics	A_{jj}, B_{jj} (3.23)	$S_j A_{jj} S_j^{-1}$, $S_j B_{jj}$ (3.37)	f_j, g_j (3.21)	$S_j \circ f_j \circ S^{-1}$, $S_j \circ g_j \circ S^{-1}$ (3.39)
Cost Structure	Q_j, R_j (3.22)	$S_j^{-T} Q_j S_j^{-1}$, R_j (3.37)	Q_j, R_j (3.22)	$S_j^{-T} Q_j S_j^{-1}$, R_j (3.37)
ALE Solution	$P_{i,j}$ (3.34)	$S_j^{-T} P_{i,j} S_j^{-1}$ (3.46)	$P_{i,j}$ (3.34)	$S_j^{-T} P_{i,j} S_j^{-1}$ (3.46)
Controller	$K_{i,j}$ (3.30)	$K_{i,j} S_j^{-1}$ (3.46)	$K_{i,j}$ (3.30)	$K_{i,j} S_j^{-1}$ (3.46)
Regression Matrix	$(A_{i,j} \underline{\oplus} A_{i,j})^T$ (3.35)	$(A_{i,j} \underline{\oplus} A_{i,j})^T (S_j \underline{\otimes} S_j)^T$ (3.44)	$\mathbf{A}_{i,j}$ (3.28)	$\mathbf{A}_{i,j} (S_j \underline{\otimes} S_j)^T$ (3.47)
Regression Target Vector	$-\operatorname{svec}(Q_{i,j})$ (3.35)	$-\operatorname{svec}(Q_{i,j})$ (3.44)	$-I_{x_j,x_j} \operatorname{svec}(Q_{i,j})$ (3.29)	$-I_{x_j,x_j} \operatorname{svec}(Q_{i,j})$ (3.47)

Chapter 4

EVALUATION STUDY: EIRL SUITE VERSUS ADP ON HYPERSONIC VEHICLE (HSV) SYSTEM

The evaluations of this section are designed to show how application of our three novel design elements decentralization (Section 3.3), multi-injection (Section 3.4), and modulation-enhanced excitation (Section 3.5) successively improve algorithm numerical performance. We demonstrate these design elements on a significant unstable, nonminimum phase HSV application, comparing numerical performance to the original ADP-based IRL algorithm [1] and to the ADP numerical performance evaluations of Section 2. For an in-depth analysis of the HSV model considered in this evaluation, please see Appendix B.

4.1 Setup and Hyperparameter Selection

HSV Longitudinal Model. Consider the following HSV longitudinal model [5, 6, 121]

$$\begin{aligned}
\dot{V} &= \frac{T\cos\alpha - D}{m} - \frac{\mu\sin\gamma}{r^2}, \\
\dot{\gamma} &= \frac{L + T\sin\alpha}{mV} - \frac{(\mu - V^2 r)\cos\gamma}{Vr^2}, \\
\dot{\theta} &= q, \\
\dot{q} &= \frac{\mathcal{M}}{I_{yy}}, \\
\dot{h} &= V\sin\gamma,
\end{aligned} \tag{4.1}$$

where V is the vehicle airspeed, γ is the flightpath angle (FPA), α is the angle of attack (AOA), $\theta \triangleq \alpha + \gamma$ is the pitch attitude, q is the pitch rate, and h is the vehicle

altitude. Here $r(h) = h + R_E$ is the total distance from the earth's center to the vehicle, $R_E = 20,903,500$ ft is the radius of the earth, and $\mu = Gm_E = 1.39 \times 10^{16}$ ft^3/s^2, where G is Newton's gravitational constant and m_E is the mass of the earth. $L, D, T, \mathcal{M}$ are the lift, drag, thrust, and pitching moment, respectively. Expressions for these terms can be found in Appendix B, where we conduct an in-depth analysis of this HSV model.

These studies consider a single modeling error parameter $\nu_L \in \mathbb{R}$ (B.4) which is unknown (nominally 1) representing modeling error in the basic lift increment coefficient $C_{L,\alpha}$. The system (B.1) is fifth-order, with states $x = [V, \gamma, \theta, q, h]^T$. The controls are $u = [\delta_T, \delta_E]^T$, and we examine the outputs $y = [V, \gamma]^T$. As in [5], we study a steady level flight cruise condition $q_e = 0, \gamma_e = 0°$, at $M_e = 15$, $h_e = 110,000$ ft, which corresponds to an equilibrium airspeed $V_e = 15,060$ ft/s. At this flight condition, the vehicle is trimmed at $\alpha_e = 1.7704°$ by the controls $\delta_{T,e} = 0.1756$ ($T_e = 4.4966 \times 10^4$ lb), $\delta_{E,e} = -0.3947°$.

Hyperparameter Selection. For sake of comparison, we hold all hyperparameter selections constant across Evaluations 1 and 2.

Cost Structure. We select cost by applying classically-based optimal control principles [11, 136]. In the velocity loop $j = 1$, we choose the state penalty $Q_1 = I_2$ and control penalty $R_1 = 15$, while in the FPA loop $j = 2$ we choose the state penalty $Q_2 = \texttt{diag}(1, 1, 0, 0)$ and control penalty $R_2 = 0.01$. These cost structure parameters were selected to yield optimal designs K_1^* (4.5), K_2^* (4.6), which achieve closed-loop step response specifications comparable to previous works [132–134]: A 90% rise time in velocity $t_{r,V,90\%} = 31.99$ s and FPA $t_{r,\gamma,90\%} = 4.56$ s, a 1% settling time in velocity $t_{s,V,1\%} = 78.18$ s and FPA $t_{s,\gamma,1\%} = 8.643$ s, percent overshoot in velocity $M_{p,V} = 4.24\%$ and FPA $M_{p,\gamma} = 3.988\%$. Next, using the decentralized control method described in [132, 134] (which performs decentralized LQR designs on

simplified versions of the diagonal plant terms P_{jj}), we arrive at the following initial stabilizing controllers

$$K_{0,1} = \begin{bmatrix} 0.2582 & 4.3570 \end{bmatrix}, \tag{4.2}$$

$$K_{0,2} = \begin{bmatrix} 10.0000 & 26.3299 & 1.6501 & 1.0124 \end{bmatrix}, \tag{4.3}$$

$$K_0 = \begin{bmatrix} K_{0,1} & 0 \\ 0 & K_{0,2} \end{bmatrix}. \tag{4.4}$$

Excitation Signals. For the exploration noise d (used by all methods except the original IRL formulation [1]) we choose $d_1(t) = 0.1\sin\left(\frac{2\pi}{25}t\right) + 0.1\sin\left(\frac{2\pi}{250}t\right) + 0.2$ and $d_2(t) = 10\sin\left(\frac{2\pi}{6}t\right) + 5\cos\left(\frac{2\pi}{50}t\right) + 2.5\sin\left(\frac{2\pi}{25}t\right)$. It is no coincidence for the noises d_1, d_2 that we have placed the dominant terms at the frequencies $\omega = \frac{2\pi}{25} \approx 0.25$ rad/s and $\omega = \frac{2\pi}{6} \approx 1$ rad/s, respectively – the peak P-sensitivity T_{dy} frequencies in the respective channels (cf. Figure 3.3a). This choice maximizes the excitation efficiency. For the reference command r (used in the MI mode only), we choose $r_1(t) = 10\cos\left(\frac{2\pi}{10}t\right) + 10\sin\left(\frac{2\pi}{25}t\right) + 50\sin\left(\frac{2\pi}{200}t\right)$ and $r_2(t) = 0.02\cos\left(\frac{2\pi}{3}t\right) + 0.1\sin\left(\frac{2\pi}{6}t\right) + 0.25\sin\left(\frac{2\pi}{15}t\right)$. Similarly to the probing noise excitations d_1, d_2, we have chosen the dominant terms in r_1, r_2 corresponding to the input/output behavior of the complementary sensitivity map T_{ry} in Figure 3.3b.

Learning Hyperparameters. We systematically select these parameters based on insights of the system physics and natural dynamical behavior. These include sample period $T_s = t_k - t_{k-1}$, number of samples collected l, and number of iterations i^*. Selections for all methods are summarized in Table 4.1. The parameters for the original IRL algorithm [1] were selected as the ones with the best condition number from those over a range of candidates. As has been systematically demonstrated in the CT-RL comparative study [137], a relatively short sample period $T_s = 0.15$ s is

favorable for the original IRL algorithm [1] due to the lack of ability to insert probing noise d (see Section 4.2 for further discussion). Also, $l = 25$ data samples yields the best numerical conditioning results consistent with prior analyses performed in [137], and $i^* = 5$ iterations is run as with the other algorithms. Lastly, for IRL we use the critic basis functions $x \underline{\otimes} x$ in order to minimize its critic network dimensionality, the same choice we make for EIRL (cf. Section 3.2 for discussion).

Table 4.1: Learning Hyperparameters for Section 4

Algorithm	Loop j	$T_{s,j}$ (s)	l_j	i_j^*
IRL (old) [1]	1	0.15	25	5
EIRL	1	5	25	5
dEIRL	1	6	15	5
	2	2	25	5

Meanwhile, for dEIRL the designer can select the sample period $T_{s,j}$ in accordance with the natural bandwidth of the dynamics in the respective loop j – a crucial numerical advantage of decentralization. In this example, the complementary sensitivity response T_{ry} in Figure 3.3b shows that the velocity $j = 1$ and FPA $j = 2$ loops are separated by a decade in bandwidth. As such, using a single sample period to capture the dynamical features of both of these loops is naturally troublesome. Reaffirming these intuitions experimentally, a longer sample period $T_{s,1} = 6$ s is observed numerically-favorable in the lower-bandwidth velocity loop, while a shorter sample period $T_{s,2} = 2$ s is favorable in the higher-bandwidth FPA loop. In the case of EIRL, the designer is afforded the luxury of probing noise injection (and/or reference command injection in the MI case), but is still required to choose a single sample period T_s. It is then intuitive that a sample period $T_s = 5$ s (i.e., between the favorable dEIRL selections $T_{s,2} = 2$ s and $T_{s,1} = 6$ s) is observed to yield the best conditioning for EIRL.

For selection of the number of data points l, we note that this system has regression

dimensions $\underline{n} = 21$, $\underline{n}_1 = 3$, $\underline{n}_2 = 10$, which serve as lower bounds for the number of samples collected in the respective regression. The lower dimensionality of the velocity loop $j = 1$ allows for fewer data points $l_1 = 15$ to be collected for dEIRL than for the other methods, which indeed proved numerically favorable.

System Initial Conditions (ICs). Since the original IRL algorithm [1] does not allow probing noise injection, the only means of excitation available to the designer is initialization of the system away from trim x_e. To this end, for the original IRL algorithm [1], we initialize the HSV at the airspeed $V_0 = V_e + 1000$ ft/s and FPA $\gamma_0 = \gamma_e + 2^\circ = 2^\circ$, and otherwise all remaining ICs are set to trim. For the algorithms developed in this work, we need not bother with this practice and hence initialize the system at trim $x_0 = x_e$.

Hardware, Software. These evaluations were performed in MATLAB R2021a, on an NVIDIA RTX 2060, Intel i7 (9th Gen) processor. All numerical integrations in this work are performed in MATLAB's adaptive `ode45` solver to ensure solution accuracy. All code developed for this work can be found at [151].

4.2 Evaluation 1: Conditioning and Convergence Study on Nominal Model

In this section, we demonstrate conditioning and convergence effectiveness [137] of EIRL and dEIRL. Specifically, we first show the significant conditioning reduction moving from the original IRL formulation [1] to any of the methods developed here. Second, we illustrate the successive conditioning improvements associated with multi-injection (SI $\rightarrow$ MI) and decentralization (EIRL $\rightarrow$ dEIRL). As a convention, we include the prefix SI- (e.g., SI-EIRL) to make explicit when an algorithm is in SI mode; otherwise, without a prefix the algorithm is assumed to be in MI mode (e.g., EIRL). For this comparative study, we consider the nominal HSV model described in Section 4.1 (i.e., zero lift-coefficient modeling error $\nu = 1$ (B.4)).

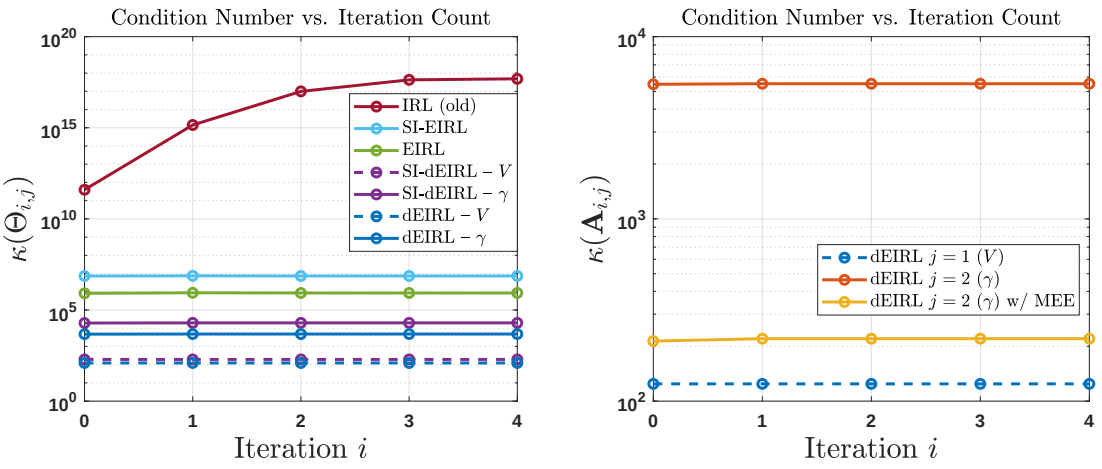


Figure 4.1: Eval. 1: Condition Number Versus Iteration Count i. (a): Conditioning of the Original IRL Algorithm [1], SI-EIRL, EIRL, SI-dEIRL, and dEIRL. (b): Conditioning of dEIRL Before and After MEE (Re-Scaled from (a) for Legibility Purposes).

Table 4.2: Eval. 1: Max/Min Conditioning

Algorithm	Loop j	$\max_i(\kappa(\mathbf{A}_{i,j}))$	$\min_i(\kappa(\mathbf{A}_{i,j}))$	$i_{\max\kappa}$	$i_{\min\kappa}$
IRL (old)	1	5.00e+17	3.97e+11	4	0
SI-EIRL	1	7.52e+06	7.34e+06	1	0
EIRL	1	8.79e+05	8.27e+05	1	0
SI-dEIRL	1	193.19	193.16	0	1
	2	1.97e+04	1.93e+04	1	0
dEIRL	1	123.26	123.25	0	1
	2	4.81e+03	4.74e+03	1	0
dEIRL w/ MEE	1	123.26	123.25	0	1
	2	220.13	213.53	1	0

Before we begin, we motivate why conditioning is integral to the study of CT-RL algorithm learning performance [137]. Existing ADP-based CT-RL algorithms which use linear regressions in the form of Equations (3.18) and (3.27) to yield their NN weights exhibit condition numbers on the order of $\kappa(\Theta) \approx 10^5 - 10^{11}$ for simple second-order academic examples [137, Table VII]. The original IRL algorithm [1] has conditioning of $\kappa(\mathbf{A}_i) = 5 \times 10^{17}$ in this HSV study (cf. Table 4.2). It is then to

be expected that IRL's NN weights exhibit large oscillations on the order of 8,000 and fail to converge (cf. Figure 4.2a). The culprit stifling the learning performance of IRL (and of the existing ADP-based CT-RL suite, more broadly [137]) is poor conditioning. It is thus vital that future works show rigorous numerical conditioning studies when proving new CT-RL algorithms.

Conditioning Analysis. The entirety of this analysis is summarized in Figure 4.1a and Table 4.2. Figure 4.1a displays the iteration-wise condition number of each of the methods studied (i.e., of the matrix $\mathbf{A}_i$ (3.19) for IRL [1] and EIRL, and $\mathbf{A}_{i,j}$, $j = 1, 2$ (3.28) for dEIRL). Table 4.2 displays the maximum and minimum of the conditioning data presented in Figure 4.1a (i.e., taken over iteration $0 \leq i \leq i^* - 1$), alongside the iteration of max conditioning $i_{\max \kappa}$ and min conditioning $i_{\min \kappa}$. Figure 4.1a reveals that the original IRL algorithm [1] suffers from the worst conditioning of the methods presented, featuring a monotonic increase from 4×10^{11} at $i = 0$ to 5×10^{17} at $i = 4$ (Table 4.2). Conditioning on this order in a monotonic degradation pattern has been demonstrated by IRL previously and is associated with PE issues as the system approaches the origin under the stabilizing policy K_0 without probing noise excitation (cf. [137, Section IX-A] for in-depth analysis). By contrast, SI-EIRL (which naturally possesses the worst conditioning properties of the methods developed here, by virtue of not taking advantage of MI or decentralization) features steady conditioning on the order of 7.5×10^6 (Table 4.2). This represents an eleven order of magnitude improvement in worst-case conditioning over prior ADP methods.

Remark 4.2.1 (Multi-Injection: An Order of Magnitude Reduction in Conditioning). Moving down Table 4.2, the effect of MI is generally an order of magnitude conditioning reduction. Conditioning of SI-EIRL is on the order of 7.5×10^6, and EIRL 8.5×10^5. SI-dEIRL in loop $j = 2$ has conditioning on the order of 2×10^4, and dEIRL 4.81×10^3 in this loop. The one case where this trend does not hold is in dEIRL

loop $j = 1$, wherein SI-dEIRL conditioning is 193 and dEIRL 123. We attribute the less dramatic reduction to diminishing returns associated with the already-favorable conditioning in this loop, although a reduction of 36% is still significant.

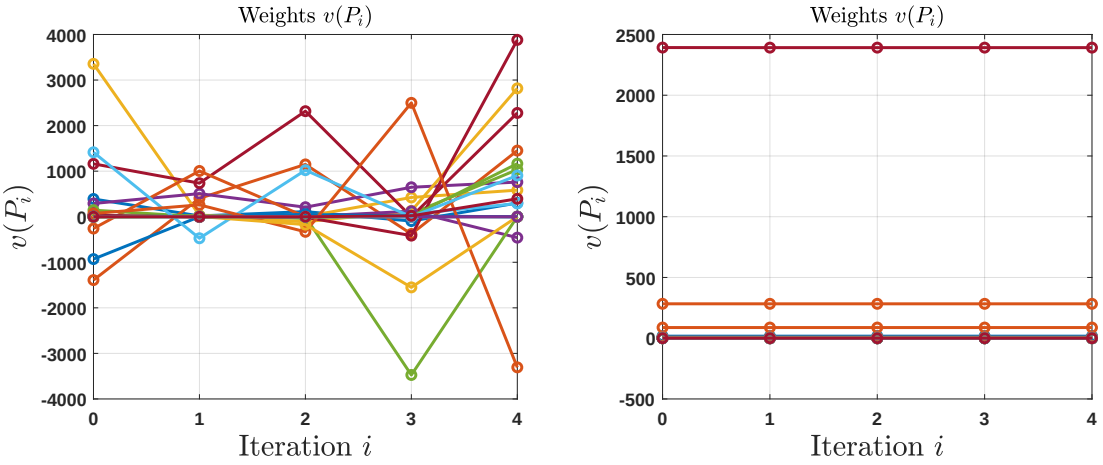
Remark 4.2.2 (Decentralization: A Two-to-Four Order of Magnitude Reduction in Conditioning). Examining Table 4.2 again, we see that decentralization achieves numerical improvements even more impressive than those of MI. Moving from SI-EIRL to its decentralized SI-dEIRL counterpart, conditioning is reduced from 7.5×10^6 to 193 in loop $j = 1$ and 2×10^4 in loop $j = 2$ – a reduction of four and two orders of magnitude, respectively. The trend is similar from EIRL (8.5×10^5) to dEIRL (123 in loop $j = 1$ and 4.81×10^3 in loop $j = 2$), a respective reduction of three and two orders of magnitude.

Remark 4.2.3 (Modulation-Enhanced Excitation: An Order of Magnitude Reduction in Conditioning). Examination of Table 4.2 shows that worst-case conditioning is already acceptable in the velocity loop $j = 1$ at 124.38. Thus, no modulation $S_1 = I_2$ in loop $j = 1$ is necessary. However, conditioning in the higher-dimensional, unstable, nonminimum phase FPA loop $j = 2$ is worse at 4.81×10^3. Furthermore, just as in the motivating example studied in Section 3.5.1, a few minutes of investigation yields a physically-intuitive explanation of the cause of the conditioning issue. Within the FPA loop $j = 2$ is the FPA subsystem γ itself (stable, nonminimum phase), alongside the attitude subsystem θ, q (unstable, minimum phase). The FPA dynamics have a bandwidth roughly a decade below that of the attitude dynamics. As a result, the pitch θ generally exhibits larger responses than the FPA, and the pitch rate q by virtue of differentiation magnifies this response amplitude discrepancy.

As can be seen, this simple radians/degrees conversion reduces worst-case conditioning by factor of 25 from 4.81×10^3 without MEE to 220.13 with MEE, a condi-

tioning reduction observed iteration-wise across the board in Figure 4.1b. In light of the higher dimension and dynamical challenges associated with the FPA loop $j = 2$, a near equalization of the conditioning in this loop with that of the velocity loop $j = 1$ is a substantial real-world numerical result.

Remark 4.2.4 (Decentralization, Multi-Injection, and Modulation-Enhanced Excitation: The Curses of Conditioning and Dimensionality Mollified). In all, the cumulative numerical improvements from the original IRL formulation [1] to dEIRL with MEE represent momentous reductions of fifteen orders of magnitude in worst-case conditioning. Moving from SI-EIRL to successive application of MI and decentralization reduces conditioning by four orders of magnitude in loop $j = 1$ and three orders of magnitude in loop $j = 2$, and MEE reduces the conditioning by an additional order of magnitude in loop $j = 2$.



(a) (b)

Figure 4.2: Eval. 1: Weight Responses svec(P_i) (3.18). (a): Original IRL Algorithm [1]. (b): SI-EIRL (Section 3.2.1).

Convergence/Solution Optimality Analysis. This model has the following op-

timal LQ controllers

$$K_1^* = \begin{bmatrix} 0.2582 & 4.3577 \end{bmatrix}, \tag{4.5}$$

$$K_2^* = \begin{bmatrix} 10.0000 & 26.3393 & 1.6514 & 0.9921 \end{bmatrix}, \tag{4.6}$$

$$K^* = \left[\begin{array}{cccccc} 0.2581 & 4.3622 & 0.0074 & 0.0814 & 0.0000 & 0.0001 \\ \hline -0.2865 & -1.1120 & 9.9959 & 26.3120 & 1.6512 & 0.9921 \end{array}\right]. \tag{4.7}$$

We have plotted the weight responses for the original IRL algorithm [1] in Figure 4.2a. Due to poor conditioning (Figure 4.1a, Table 4.2), the weights update erratically and fail to converge. This same qualitative weight behavior was illustrated systematically in [137, Figure 4c] on a simple second-order academic example (for which conditioning was on the order of 10^{11}), so it is not surprising that we encounter it for the HSV here. For comparison, we have plotted the corresponding weight responses for SI-EIRL in Figure 4.2b. These two methods use the same basis functions, yet the SI-EIRL weights converge nicely in the $i^* = 5$ iterations. Since SI-EIRL naturally has the worst conditioning properties of the methods developed in this work, Figure 4.2b represents the "worst-case" weight response of the proposed methods.

Remark 4.2.5 (All Proposed Methods Deliver Real-World Synthesis Guarantees). Each of the proposed methods successfully converges to their respective optimal controller K^*. Indeed, the largest final controller error $\|K_{i^*} - K^*\|$ exhibited by any of the methods is only 4.63×10^{-3} (by SI-EIRL). For comparison, dEIRL exhibits final controller errors $\|K_{i^*,1} - K_1^*\| = 1.11 \times 10^{-6}$ and $\|K_{i^*,2} - K_2^*\| = 2.70 \times 10^{-5}$. Thus, this evaluation study affirms that the proposed methods achieve real-world convergence performance in exact accordance with their theoretical guarantees (Section 3.6) – perhaps a first in ADP-based CT-RL [137].

Sample/Training Time Efficiency. Due to leveraging Kleinman's structure (cf. Section 3.1) in combination with nonlinear state-action data (x, u), these EIRL algorithms have high data/time efficiency. All of the proposed methods require at most $l = 25$ trajectory data samples (cf. Table 4.1). As a result of this data efficiency, the developed methods all converge in 2.74 seconds of training time at most (longest: dEIRL).

4.3 Evaluation 2: dEIRL Optimality Recovery Generalization with Respect to Modeling Error

Having now demonstrated a systematic framework for improving learning numerics through an in-depth conditioning analysis of the developed algorithms, for this study we hone our focus to the flagship method: dEIRL. Having demonstrated dEIRL's convergence properties on the nominal HSV model $\nu_L = 1$ (B.4), we now analyze its convergence when the model is perturbed from the nominal model to $\nu_L = 0.9$ (a 10% modeling error) and $\nu_L = 0.75$ (a 25% modeling error), resulting in a more challenging control problem (cf. Appendix B.2 for discussion).

Conditioning Analysis. Before delving into the main convergence results, we first provide a brief conditioning analysis. For $\nu_L = 0.9$, dEIRL has max conditioning $\max_i(\kappa(\mathbf{A}_{i,1})) = 111.13$ and $\max_i(\kappa(\mathbf{A}_{i,2})) = 259.64$, while for $\nu_L = 0.75$, dEIRL has max conditioning $\max_i(\kappa(\mathbf{A}_{i,1})) = 90.89$ and $\max_i(\kappa(\mathbf{A}_{i,2})) = 268.01$. Overall, conditioning has remained largely unchanged in the velocity loop $j = 1$ (cf. Table 4.2). Even in the higher-dimensional, unstable, nonminimum-phase FPA loop $j = 2$ directly affected by the lift-coefficient modeling error ν_L (B.4), conditioning performance has only degraded slightly. In sum, dEIRL possesses inherently-favorable conditioning properties which are robust with respect to (even severe) modeling errors – a vital real-world performance validation.

Convergence/Solution Optimality Analysis. Running dEIRL for $i^* = 5$ iterations and $\nu_L = 0.9$, we have

$$K_{i^*,1} = \begin{bmatrix} 0.2582 & 4.3579 \end{bmatrix}, \tag{4.8}$$

$$K_1^* = \begin{bmatrix} 0.2582 & 4.3580 \end{bmatrix}, \tag{4.9}$$

$$K_{i^*,2} = \begin{bmatrix} 10.0171 & 27.1052 & 1.5828 & 0.9741 \end{bmatrix}, \tag{4.10}$$

$$K_2^* = \begin{bmatrix} 10.0000 & 27.0327 & 1.5685 & 0.9671 \end{bmatrix}. \tag{4.11}$$

For $\nu_L = 0.75$, we have

$$K_{i^*,1} = \begin{bmatrix} 0.2582 & 4.3585 \end{bmatrix}, \tag{4.12}$$

$$K_1^* = \begin{bmatrix} 0.2582 & 4.3586 \end{bmatrix}, \tag{4.13}$$

$$K_{i^*,2} = \begin{bmatrix} 10.0397 & 28.5706 & 1.4712 & 0.9539 \end{bmatrix}, \tag{4.14}$$

$$K_2^* = \begin{bmatrix} 10.0000 & 28.2496 & 1.4303 & 0.9238 \end{bmatrix}. \tag{4.15}$$

In Table 4.3, we show the controller error reduction $\left\|K_{i,j} - K_j^*\right\|$ from the initial controllers $K_{0,1}$ (4.2), $K_{0,2}$ (4.3) (i.e., the nominal decentralized LQR controllers [132, 134]) to the final controllers $K_{i^*,1}$ (4.8), (4.12) and $K_{i^*,2}$ (4.10), (4.14), respectively.

Table 4.3: Eval. 2: dEIRL Optimality Recovery

ν_L (B.4)	Loop j	$\left\|K_{i,j} - K_j^*\right\|$		% Reduction
		$i = 0$ (Nom. LQ)	$i = i^*$	$i = 0 \to i^*$
0.9	1	9.10e-04	2.30e-05	97.5
	2	0.709	0.0761	89.3
0.75	1	0.00151	7.30e-05	95.2
	2	1.934	0.327	83.1

Remark 4.3.1 (dEIRL Solution Optimality Recovery). Examining Table 4.3,

for the 10% modeling error $\nu_L = 0.9$, dEIRL reduces controller optimality error by at least an order of magnitude in each loop. For the 25% modeling error $\nu_L = 0.75$, dEIRL reduces controller optimality error by at least 80% in each loop. It is at this point intuitive that dEIRL converges closer to the optimal controllers for the smaller 10% modeling error than for the severe 25% modeling error, and that reductions are more significant in the velocity loop $j = 1$ than in the FPA loop $j = 2$.

This capability is one of great practical utility. Previously, if a designer synthesized an initial LQR controller $K_{0,j}$ (i.e., optimal with respect to the nominal linear drift dynamics A_{jj}), they had to content themselves with this design and had no real-world means of improvement. Here we demonstrate that, via a nominal model ($\nu_L = 1$), dEIRL outputs a controller $K_{i^*,j}$ significantly closer to the optimal K_j^* than the original estimate $K_{0,j}$.

Closed-Loop Performance Analysis. Having illustrated that dEIRL numerically recovers the optimal controllers K_j^* in each loop, we conclude this section with an analysis of how it recovers optimal closed-loop performance. To this end, we issue a 100 ft/s step-velocity command and a 1° step-FPA command to the (nonlinear, coupled) perturbed HSV models, simulating under the nominal LQ controllers $K_{0,j}$ ($\nu_L = 1$), dEIRL controllers $K_{i^*,j}$, and optimal LQ controllers K_j^* ($\nu_L \neq 1$). The resulting closed-loop step response characteristics in each loop j are listed in Table 4.4, including the 90% rise time $t_{r,y_j,90\%}$, 1% settling time $t_{s,y_j,1\%}$, and percent overshoot M_{p,y_j} ($j = 1, 2$).

Table 4.4: Eval. 2: Closed-Loop Step Response Characteristics

ν_L (B.4)	Loop j	Algorithm	$t_{r,y_j,90\%}$ (s)	$t_{s,y_j,1\%}$ (s)	M_{p,y_j} (%)
0.9	1	Nom. LQ	32.33	78.34	4.25
		dEIRL	31.99	78.26	4.24
		Opt. LQ	31.94	78.27	4.24
	2	Nom. LQ	4.75	9.97	7.11
		dEIRL	4.80	9.33	5.26
		Opt. LQ	4.52	9.16	5.12
0.75	1	Nom. LQ	32.33	78.76	4.27
		dEIRL	31.62	78.11	4.25
		Opt. LQ	32.17	78.41	4.25
	2	Nom. LQ	5.13	16.75	12.41
		dEIRL	4.85	10.28	7.98
		Opt. LQ	4.65	10.01	7.59

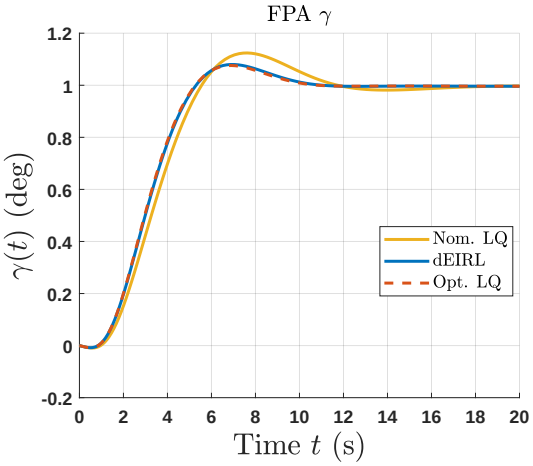


Figure 4.3: Closed-Loop 1° FPA Step Response for 25% Lift-Coefficient Modeling Error $\nu_L = 0.75$ (B.4).

As can be seen in Table 4.4, dEIRL successfully recovers the closed-loop step response characteristics of the optimal LQ controllers. The performance recovery is most apparent in the FPA loop $j = 2$, where for the severe modeling error $\nu_L = 0.75$ the performance of the nominal LQ controller is significantly degraded compared to that of dEIRL and the optimal. In particular, the 1% FPA settling time $t_{s,\gamma,1\%}$ of the nominal LQ controller is almost 17 s, and only 10 s for dEIRL and the optimal LQ. Similarly, percent overshoot in FPA $M_{p,\gamma}$ is over 12% for the nominal LQ controller, and only 8% for dEIRL and the optimal LQ. The corresponding FPA step response for the 25% lift-coefficient modeling error $\nu_L = 0.75$ is plotted in Figure 4.3. Corroborated numerically by Table 4.4, Figure 4.3 shows that dEIRL has qualitatively recovered the LQ-optimal closed-loop step response performance in spite of the severe 25% modeling error. As a brief aside, the first $t = 1$ s of the FPA response in Figure 4.3 exhibits the characteristic inverse nonminimum phase behavior attributable to the parasitic downward lift induced by pitch-up elevator deflections δ_E.

4.4 Discussion

This evaluation presents a suite of innovative CT-RL algorithms to address the performance limitations facing ADP-based CT-RL algorithms discovered in Section 2. We develop these new algorithms with accompanying results on theoretical convergence and stability guarantees. Our in-depth quantitative performance evaluations of the four algorithms show that decentralization, MI, and MEE (i.e., the dEIRL algorithm) achieve conditioning reductions of multiple orders of magnitude. Furthermore, evaluations show convergence and stability results corroborating theoretical analysis, and that the algorithms successfully recover the optimal controller and closed-loop performance in the presence of severe modeling errors. Altogether, our numerical studies demonstrate that dEIRL is a highly intuitive and effective design tool. This

is a significant step forward from results of previous CT-RL algorithms, which ultimately fail to synthesize stabilizing designs even for simple academic systems (cf. Section 2).

We would like to point out that decentralization allows the designer to select learning parameters optimized to the inherent physics of each loop, rather than being forced to select a single set of "middle-ground" parameters which fails to adequately address individual-loop learning needs (cf. Section 4.1). Furthermore, weak dynamical coupling means that the excitation selections d_j, r_j in one loop have little effect on data quality in other loops, and the rest of the hyperparameters have no inter-loop effects. This allows the designer to troubleshoot at the individual loop level, greatly simplifying the design process. With all the demonstrated benefits of decentralization, it is noted that this decentralization is physics-driven, with many important applications in areas such as robotics and aerial vehicles.

Chapter 5

EVALUATION STUDY: DEIRL VERSUS DEEP RL FITTED VALUE ITERATION (FVI) ON THREE DIFFERENT ENVIRONMENTS

5.1 Implementation and Training Procedures

In this section, we first demonstrate how dEIRL learning generalizes with respect to systematic and simultaneous sweeps of 1) system initial conditions (ICs) and 2) system modeling error on the three environments studied. In particular, we focus analysis on how dEIRL delivers its convergence, solution optimality, and closed-loop stability guarantees of Section 3.6 in practical learning control problems.

We then provide a comprehensive evaluation of our proposed dEIRL method in comparison to the SOTA deep CT-RL works in FVI [7, 8], including performance generalization with respect to system ICs and modeling error.

Baselines. In reporting evaluation results below, we use the following short-form descriptions for the baseline methods tested:

- "Continuous FVI (cFVI)": SOTA deep CT-RL method developed in [7].
- "Robust FVI (rFVI)": Robust variant of SOTA FVI method developed in [8].
- "Nominal LQ": The classical LQR design performed on the nominal linearization (A, B) of the nonlinear system (f, g) (3.1). Unlike dEIRL, the nominal LQ fails to take into account 1) system nonlinearity, and 2) model uncertainty.
- "Optimal LQ": The classical LQR design performed on the linearization of the *actual* nonlinear process (f, g) (3.1) (i.e., the linearization includes model-

ing error). Unlike dEIRL, the optimal LQR fails to take into account system nonlinearity.

Performance Measures. In these studies, we provide comprehensive evaluations of standard performance measures in: average reward, convergence speed, learning success rate, generalization with respect to system ICs and modeling error, and time/data/free parameter complexity. In addition, analyze performance with respect to the following learning control measures:

- Cost Performance: The infinite-horizon cost $J(x_0)$ (3.2) delivered by the policy in the nonlinear optimal control task. As a note, in controls conventions the cost $J(x_0) > 0$ is a positive number to be minimized (lower = better).
- Estimation Error: The difference $J(x) - V(x)$ between the cost $J(x)$ at state $x \in \mathbb{R}^n$ and the value function approximation $V(x)$ at $x \in \mathbb{R}^n$. It is desired that the estimation error $J - V$ be as small in magnitude as possible (so the critic is accurate), and negative whenever it deviates from zero (so the critic underestimates actual policy performance).
- Policy Optimality Error: The difference $\|\mu - K^*\|$ in operator norm between a given policy μ and the optimal control K^*.

In addition, we measure the following closed-loop time-domain performance measures which are central to the continuous-time nonlinear dynamical control task studied:

- 90% Rise Time $t_{r,y,90\%}$: The time taken for the closed-loop response $y(t)$ to rise to 90% of its commanded value.
- 1% Settling Time $t_{s,y,1\%}$: The time taken for the closed-loop response $y(t)$ to settle within $\pm 1\%$ of its commanded value.

- Percent Overshoot $M_{p,y}$: The maximum by which the closed-loop response $y(t)$ exceeds the step reference command $r(t) \equiv r$. Expressed in percent as a ratio of the size of the command r as follows

$$M_{p,y} = \max_{t \geq 0} \left\{ \frac{y(t) - r}{r} \right\} \times 100\%. \tag{5.1}$$

All Code/Data Available. All dEIRL code and datasets for this study are available in Supplemental and at [152]. All FVI results [7, 8] are generated by the open-source code developed by the authors available at [153].

Hardware. These studies were performed in MATLAB R2022b, on an NVIDIA RTX 2060, Intel i7 (9th Gen) processor. All numerical integrations in this work are performed in MATLAB's adaptive `ode45` solver to ensure solution accuracy.

Software. All dEIRL code and datasets developed for this work is available in Supplemental and at [152]. All FVI results [7, 8] were generated from the open-source repository developed by the authors [153].

5.1.1 Selection of Three Environments

Extensive evaluations of dEIRL are performed on three environments: a pendulum, jet aircraft, and differential drive mobile robot (DDMR). In-depth analyses of these three systems can be found in Appendices C, D, and E, respectively. The

dynamics of the environments are given by

$$\begin{array}{rcl|rcl} \dot{\theta} & = & \omega & \dot{V} & = & \frac{m_c d}{\hat{m}}\omega^2 - \frac{2\overline{\beta}}{\hat{m}r^2}V + \frac{k_t}{\hat{m}k_g r}i_{a_r} + \frac{k_t}{\hat{m}k_g r}i_{a_l} \\ \dot{\omega} & = & \frac{mgL}{2I}\sin\theta + \frac{\tau}{I} & \dot{\omega} & = & \frac{-m_c d}{\hat{I}}\omega V - \frac{\overline{\beta}d_w^2}{2\hat{I}r^2}\omega + \frac{d_w k_t}{2\hat{I}k_g r}i_{a_r} - \frac{d_w k_t}{2\hat{I}k_g r}i_{a_l} \\ & & & \dot{i}_{a_r} & = & \frac{-k_g k_b}{l_a r}V - \frac{k_g k_b d_w}{2l_a r}\omega - \frac{r_a}{l_a}i_{a_r} + \frac{1}{2l_a}\overline{e}_a + \frac{1}{2l_a}\Delta e_a \\ & & & \dot{i}_{a_l} & = & \frac{-k_g k_b}{l_a r}V + \frac{k_g k_b d_w}{2l_a r}\omega - \frac{r_a}{l_a}i_{a_l} + \frac{1}{2l_a}\overline{e}_a - \frac{1}{2l_a}\Delta e_a \end{array} \tag{5.2}$$

$$\begin{bmatrix} \dot{V} \\ \dot{\gamma} \\ \dot{q} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} -D_V & -g\cos\alpha_e & 0 & 0 \\ \frac{L_V}{V_e} & 0 & 0 & \frac{L_\alpha}{V_e} \\ 0 & 0 & M_q & M_\alpha \\ \frac{-L_V}{V_e} & 0 & 1 & \frac{-L_\alpha}{V_e} \end{bmatrix} \begin{bmatrix} V \\ \gamma \\ q \\ \alpha \end{bmatrix} + \begin{bmatrix} T_{\delta_T} & 0 \\ 0 & 0 \\ 0 & M_{\delta_E} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_T \\ \delta_E \end{bmatrix}. \tag{5.3}$$

Pendulum. This system (5.2, left) has states $x = (\theta, \omega)$, where θ is the pendulum angle (measured zero pointing upward, positive counterclockwise), ω is the pendulum angular velocity, and the single-input control $u = \tau$ is the torque applied to the pendulum base.

Jet Aircraft. This system (5.3) has states $x = (V, \gamma, q, \alpha)$, where V is the airspeed, γ is the flightpath angle (FPA), q is the pitch rate, and α is the angle of attack (AOA). It has controls $u = (\delta_T, \delta_E)$, where δ_T is the throttle setting (associated with the airspeed V in the translational loop $j = 1$), and δ_E is the elevator deflection (associated with the FPA γ and attitude q, α in the rotational loop $j = 2$).

Differential drive mobile robot (DDMR). This system (5.2, right) also lends itself to decentralization, with states $x = (V, \omega, i_{a_r}, i_{a_l})$, where V is the velocity, ω is the angular velocity, and i_{a_r}, i_{a_l} are the right and left DC motor armature currents, respectively. The controls are $u = (\overline{e}_a, \Delta e_a)$, where $\overline{e}_a$ is the average of the armature

voltages applied to the right and left DC motors (associated with the speed V in the translational loop $j = 1$), and Δe_a is the difference of the right/left voltages (associated with the rotational velocity ω in the rotational loop $j = 2$).

5.1.2 Training Procedures

An episode is initialized by resetting the environment at a given initial condition x_0, and the environment is simulated out to the training horizon T required for the respective algorithm in order to collect the resulting state-action data (x, u).

Learning Trial: dEIRL. Due to dEIRL's data efficiency (requiring only on the order of $l = 20$ total trajectory samples), state-action data (x, u) from a single episode (i.e., initialized at single IC x_0) provides sufficient data to execute learning for a single seed to algorithm termination after i^* iterations. This low data complexity allows dEIRL to use online data collected from the actual physical process (i.e., with modeling error $\nu \neq 1$).

Learning Trial: FVIs. Deep RL FVIs, on the other hand, have higher data complexity to train their deep network approximators and require training data from over 5 million epsiodes to execute to learning completion (cf. Table 5.11), for details see [7, 8]. As a result, the only practical means of training FVI is in simulation. Since the modeling error ν for a given system is not known *a priori*, this means that FVI must train on the nominal model (modulo adversary perturbations in the rFVI case [8]). Thus, the results presented in this work for FVI are attained through training on the nominal model $\nu = 1$, a training procedure identical to that presented by the original authors in [7, 8].

Initial Condition (IC) Generation. System ICs for training are generated via the following uniform distrubutions $\mathcal{U}$ for the pendulum, jet aircraft, and DDMR,

respectively:

$$x_0 \sim \mathcal{U}(\pm\pi \text{ rad}, \pm 8 \text{ rad/s}), \tag{5.4}$$

$$x_0 \sim \mathcal{U}(\pm 15 \text{ ft/s}, \pm 3 \text{ deg}, \pm 30 \text{ deg/s}, \pm 10 \text{ deg}), \tag{5.5}$$

$$x_0 \sim \mathcal{U}(\pm 3 \text{ m/s}, \pm 90 \text{ deg/s}, \pm 0 \text{ A}, \pm 0 \text{ A}). \tag{5.6}$$

As a note: For the pendulum system, we have chosen the identical uniform distribution $\mathcal{U}$ (5.4) for state initialization as is chosen in the original FVI benchmark studies [7, 8].

The only exception to the above IC generation procedure is the systematic grid sweeps conducted in the dEIRL initial condition and modeling error generalization studies of Section 5.3. Here, the initial conditions x_0 are swept over a grid of values $x_0 \in G_{x_0}$. The IC grids G_{x_0} for the pendulum, jet aircraft, and DDMR are given respectively as:

$$G_{x_0} = [-\tfrac{\pi}{3} : \tfrac{\pi}{6} : \tfrac{\pi}{3}] \text{ rad } \times [-\tfrac{\pi}{3} : \tfrac{\pi}{6} : \tfrac{\pi}{3}] \text{ rad/s}, \tag{5.7}$$

$$G_{x_0} = [90 : 2 : 110] \text{ ft/s } \times [-2 : 0.5 : 2] \text{ deg}, \tag{5.8}$$

$$G_{x_0} = [1.5 : 0.125 : 2.5] \text{ m/s } \times [-30 : 5 : 30] \text{ rad/s}, \tag{5.9}$$

where all other ICs for the higher-order jet aircraft and DDMR environments are set to zero. Note that for all systems in this work, the IC grid G_{x_0} is centered about the respective equilibrium point x_e (cf. Appendices C–E for discussion of equilibria of each system).

Modeling Error Generation. In the modeling error generalization studies of Sections 5.3 and 5.4, the environment modeling error ν is swept over a grid of values $\nu \in G_\nu$. The modeling erro grids G_ν for the pendulum, jet aircraft, and DDMR are

given respectively as:

$$G_\nu = [1 : 0.01 : 1.25], \tag{5.10}$$

$$G_\nu = [1 : -0.025 : 0.75], \tag{5.11}$$

$$G_\nu = [1 : 0.025 : 1.25]. \tag{5.12}$$

The direction of the perturbation (i.e., $\nu > 1$ or $\nu < 1$) is chosen to maximize the difficulty of the respective learning problem (cf. Appendices C-E for in-depth discussion).

5.1.3 Evaluation Procedures

dEIRL Combined Initial Condition/Modeling Error Generalization Study (Section 5.3). Here, dEIRL is run over the combined trial space $(x_0, \nu) \in G_{x_0} \times G_\nu$, where the respective IC grids G_{x_0} for each environment are given by (5.7)–(5.9) and the respective modeling error grids G_ν for each environment are given by (5.10)–(5.12). This corresponds to 1,620 learning trials for the pendulum (81 ICs x_0 × 20 modeling errors ν), 1,089 trials for the jet (99 x_0 × 11 ν), and 1,287 trials for the DDMR (117 x_0 × 11 ν).

dEIRL and FVIs Return Initial Condition Generalization Study (Section 5.4.1). dEIRL, cFVI, and rFVI are trained over 20 seeds with data generated by the nominal model $\nu = 1$ and system ICs initialized over the uniform training distributions $\mathcal{U}$ given by (5.4)–(5.6). At each algorithm iteration, the return of the trained policies is evaluated over 100 episodes of the environment. For evaluation, system ICs for training are generated via the following uniform distrubutions $\mathcal{U}$ for the pendulum,

jet aircraft, and DDMR, respectively:

$$x_0 \sim \mathcal{U}(\pm\pi \text{ rad}, \pm 0.01 \text{ rad/s}), \tag{5.13}$$

$$x_0 \sim \mathcal{U}(\pm 10 \text{ ft/s}, \pm 2 \text{ deg}, \pm 0.01 \text{ deg/s}, \pm 0.01 \text{ deg}), \tag{5.14}$$

$$x_0 \sim \mathcal{U}(\pm 0.5 \text{ m/s}, \pm 30 \text{ deg/s}, \pm 0 \text{ A}, \pm 0 \text{ A}). \tag{5.15}$$

As a note: For the pendulum system, we have chosen the identical uniform distribution $\mathcal{U}$ (5.13) for state initialization as is chosen in the original FVI benchmark studies [7, 8].

dEIRL and FVIs Cost, Estimation Error, and Closed-Loop Performance Generalization Study (Sections 5.4.2–5.4.4). dEIRL, cFVI, and rFVI are trained according to the training procedure described in Section 5.1. For display purposes of generating the surface plots in Figures 5.4–5.9, we then select a single seed for each algorithm for evaluation and present data for 0%, 10%, and 25% modeling errors. For FVIs, we choose random number seed 42, the same as was chosen for evaluation in original works [7, 8]. For dEIRL, we initialize its single-episode learning trajectory at the system equilibrium $x_0 = x_e$. The resulting final policies are then evaluated with respect to their relative costs $J_{xFVI}(x) - J_{dEIRL}(x)$ (3.2) and value function approximation errors $J(x) - V(x)$ for all states x in the following evaluation grids G_x for the pendulum, jet aircraft, and DDMR, respectively:

$$G_x = [-60 : 0.5 : 60] \text{ deg } \times [-60 : 0.5 : 60] \text{ deg/s}, \tag{5.16}$$

$$G_x = [90 : 2 : 110] \text{ ft/s } \times [-2 : 0.1 : 2] \text{ deg}, \tag{5.17}$$

$$G_x = [1.5 : 0.125 : 2.5] \text{ m/s } \times [-30 : 5 : 30] \text{ rad/s}, \tag{5.18}$$

where all other ICs for the higher-order jet aircraft and DDMR environments are set

to zero. Note that for all systems in this work, as is the case with the IC grids G_{x_0} (5.7)–(5.7) the evaluation grids G_x (5.16)–(5.16) are centered about the respective equilibrium point x_e (cf. Appendices C–E for discussion of equilibria of each system). It is these grids which are used to generate the surface plots of Figures 5.4–5.9, and the corresponding tabular data in Tables 5.7–5.10.

Finally, for the closed-loop response evaluations, we issue step reference commands these trained policies in each input/output channel of the three environments at 0%, 10%, and 25% modeling error. For the pendulum, we study swing-up performance by initializing at full displacement $\theta_0 = \pi$, $\dot{\theta}_0 = 0$ and issuing a reference command $r(t) = 0$ rad (upright position). For the jet aircraft and DDMR, we initialize ICs to equilibrium $x_0 = x_e$. For the jet, we issue a reference command in velocity $y_1 = V$ of $r_1(t) = 110$ ft/s (i.e., 10 ft/s faster than equilibrium $V_e = 100$ ft/s), and we issue a reference command in flightpath angle $y_2 = \gamma$ of $r_2(t) = 1$ deg. For the DDMR, we issue a reference command in velocity $y_1 = V$ of $r_1(t) = 3$ m/s (i.e., 1 m/s faster than equilibrium $V_e = 2$ m/s), and we issue a reference command in angular velocity $y_2 = \omega$ of $r_2(t) = 30$ deg/s. It is these issued reference commands which generate the closed-loop responses in Figures 5.10–5.12 and corresponding tabular data in Table 5.10.

5.2 Hyperparameter Selections

5.2.1 Hyperparameter Selections: Shared Hyperparameters

State, Control Penalty Gains. For the pendulum, we use identical penalty selections to those in the original cFVI studies [7, 8]; namely,

$$Q_1 = \texttt{diag}(1, 0.1), \qquad R_1 = 0.5. \tag{5.19}$$

For the jet aircraft, consider the decentralized design framework described in Section D.2. We choose the following cost structure

$$\begin{aligned} Q_1 &= \texttt{diag}(0.005, 0.05), & R_1 &= 5, \\ Q_2 &= \texttt{diag}(0.5, 1, 0, 0), & R_2 &= 1. \end{aligned} \tag{5.20}$$

These state/control penalties were chosen to yield optimal LQ controllers K_1^* (D.5), K_2^* (D.6) achieving nominal closed-loop step response specifications comparable to existing benchmarks [122]: A 90% rise time in speed $t_{r,V,90\%} = 9.297$ s and FPA $t_{r,\gamma,90\%} = 4.52$ s, a 1% settling time in speed $t_{s,V,1\%} = 14.47$ s and FPA $t_{s,\gamma,1\%} = 7.20$ s, percent overshoot in speed $M_{p,V} = 0.09\%$ and FPA $M_{p,\gamma} = 0.25\%$.

For the DDMR, consider the decentralized design framework described in Section E.2. We choose the following cost structure

$$\begin{aligned} Q_1 &= 10I_2, & R_1 &= 0.75, \\ Q_2 &= \texttt{diag}(25, 7.5), & R_2 &= 1. \end{aligned} \tag{5.21}$$

These state/control penalties were chosen to yield optimal LQ controllers K_1^* (E.9), K_2^* (E.10) achieving nominal closed-loop step response specifications comparable to existing benchmarks [43, 44]: A 90% rise time in speed $t_{r,V,90\%} = 3.778$ s and angular velocity $t_{r,\omega,90\%} = 1.27$ s, a 1% settling time in speed $t_{s,V,1\%} = 5.556$ s and angular velocity $t_{s,\omega,1\%} = 6.73$ s, percent overshoot in speed $M_{p,V} = 0\%$ and angular velocity $M_{p,\omega} = 16.9\%$.

5.2.2 Hyperparameter Selections: dEIRL

Initial Stabilizing Controller. For the pendulum, we use the initial stabilizing controller

$$K_{0,1} = \begin{bmatrix} 13.5108 & 5.8316 \end{bmatrix}, \tag{5.22}$$

which we obtained from cost structure selections $Q_1 = \texttt{diag}(0.5, 0.25)$, and $R_1 = 0.01$. For the jet aircraft in loop j $(j = 1, 2)$, we use the initial stabilizing controllers

$$K_{0,1} = \begin{bmatrix} 0.0316 & 0.1168 \end{bmatrix}, \tag{5.23}$$

$$K_{0,2} = \begin{bmatrix} -1.7321 & -3.4191 & -0.3427 & -0.9709 \end{bmatrix}, \tag{5.24}$$

which we obtained from a decentralized design with cost structure selections $Q_1 = 0.01I_2$, $R_1 = 10$, $Q_2 = \texttt{diag}(1.5, 2.5, 0, 0)$, and $R_2 = 0.5$. For the DDMR in loop j $(j = 1, 2)$, we use the initial stabilizing controllers

$$K_{0,1} = \begin{bmatrix} 2.2361 & 3.4966 \end{bmatrix}, \tag{5.25}$$

$$K_{0,2} = \begin{bmatrix} 8.6603 & 12.4403 \end{bmatrix}, \tag{5.26}$$

which we obtained from a decentralized design with cost structure selections $Q_1 = 5I_2$, $R_1 = 1$, $Q_2 = \texttt{diag}(7.5, 2.5)$, and $R_2 = 0.1$.

The remainder of the dEIRL hyperparameter selections can be found in Table 5.1. Examination of Table 5.1 shows that these hyperparameter selections comprise little more than "round-number" designer first-choices, requiring only insights of the system dynamics and a few minutes of trial-and-error to obtain.

Table 5.1: dEIRL Hyperparameter Selections

Hyperparameter	Pendulum	Jet Aircraft		DDMR	
	Loop $j=1$	Loop $j=1$	Loop $j=2$	Loop $j=1$	Loop $j=2$
Sample Period $T_{s,j}$ (s)	1	2	0.5	4	1
Number of Samples l_j	15	15	30	20	15
Final Iteration i_j^*	5	5	5	5	5
Ref Cmd r_j	$10\sin(\frac{2\pi}{10}t)$	$5\sin(\frac{2\pi}{50}t)$	$0.1\sin(\frac{2\pi}{2.5}t)$	$2\sin(\frac{2\pi}{10}t)$	$5\sin(\frac{2\pi}{50}t)$
(deg | m/s, deg | m/s, deg/s)	$+5\sin(\frac{2\pi}{5}t)$	$+10\sin(\frac{2\pi}{25}t)$	$+0.1\sin(\frac{2\pi}{1.5}t)$	$+\sin(\frac{2\pi}{5}t)$	$+5\sin(\frac{2\pi}{5}t)$ $+5\sin(\frac{2\pi}{2.5}t)$
Initial Controller $K_{0,j}$	(5.22)	(5.23)	(5.24)	(5.25)	(5.26)

5.2.3 Hyperparameter Selections: cFVI, rFVI

As with our selections of the pendulum model structure and parameters (cf. Section C), for our pendulum studies we have selected hyperparameters identical to those of the original cFVI/rFVI evaluations [7, 8], with two exceptions. In [7, 8], the authors use a logcos control penalty function scaled so that its curvature at the origin $u = 0$ is $2R$; i.e., so that its curvature agrees with that of a quadratic penalty $u^T Ru$. In order to make comparisons consistent across the methods studied, and in order to produce a more widely-applicable performance benchmark for real-world designers, we have decided to apply the standard quadratic control penalty $u^T Ru$ for all methods. Likewise, the authors in [7, 8] wrap the penalty function of the pendulum angle state to be periodic in $[0, 2\pi)$, a practice which we have dropped for consistency of comparison and generalizability of benchmarking. Finally, due to these changes we observed that more iterations were necessary for rFVI to converge in training the pendulum system (cf. Figure 5.3), so we increased its iteration count from 100 previously [7, 8] to 150 here (cf. Table 5.2).

For the jet and DDMR examples that are new to FVIs, we have chosen hyperparameters in light of the successes achieved by the selections in [7, 8], tailored to maximize learning performance for these specific systems.

Hyperparameter selections for cFVI and rFVI can be found in Table 5.2. These

parameter selections are overall quite standard and have indeed demonstrated great learning performance successes on second-order, unstable systems in previous studies [7, 8].

Table 5.2: cFVI, rFVI Hyperparameter Selections

Hyperparameter	Pendulum		Jet Aircraft		DDMR	
	cFVI	rFVI	cFVI	rFVI	cFVI	rFVI
Time Step (s)	0.008	0.008	0.008	0.008	0.008	0.008
Time Horizon (s)	5	5	20	20	5	5
Discounting γ	0.99	0.99	0.99	0.99	0.99	0.99
Network Dimension	$[3 \times 96]$	$[3 \times 96]$	$[3 \times 96]$	$[3 \times 96]$	$[3 \times 96]$	$[3 \times 96]$
# Ensemble	4	4	4	4	4	4
Activation	Tanh	Tanh	Tanh	Tanh	Tanh	Tanh
Learning Rate	1e-5	1e-5	3e-5	3e-5	3e-5	3e-5
Weight Decay	1e-6	1e-6	1e-6	1e-6	1e-6	1e-6
Hidden Layer Gain	1.41	1.41	1.41	1.41	1.41	1.41
Output Layer Gain	1.00	1.00	1.00	1.00	1.00	1.00
Output Layer Bias	-0.1	-0.1	-0.1	-0.1	-0.1	-0.1
Diagonal Softplus Gain β_L	1.0	1.0	7.5	7.5	1.0	1.0
Batch Size	256	128	256	256	256	256
# Batches	200	200	200	200	200	200
Eligibility Trace	0.85	0.85	0.85	0.85	0.85	0.85
n-step Trace Weight	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4
# Iterations	100	150	100	100	100	100
# Epochs/Iteration	20	20	20	20	20	20
State Adversary $\|\xi_x\|_{\max}$	0.0	0.025	0.0	0.025	0.0	0.025
Action Adversary $\|\xi_u\|_{\max}$	0.0	0.1	0.0	0.1	0.0	0.1
Model Adversary $\|\xi_\theta\|_{\max}$	0.0	0.15	0.0	0.1	0.0	0.009
Obs Adversary $\|\xi_o\|_{\max}$	0.0	0.025	0.0	0.025	0.0	0.025

5.3 dEIRL Modeling Error and Initial Condition Ablation Study

When benchmarking the characteristics of a new RL control framework, studies must address the central question: Does the RL algorithm deliver better performance than existing, well-established classical methods? In this evaluation, we provide substantive quantitative analysis demonstrating that dEIRL offers significant performance improvements over classical LQR. In short, in the face of severe modeling errors, for all systems dEIRL reliably delivers a 90% reduction in operator-norm error with respect to the optimal controller over a nominal LQR design. In the process of substantiating this claim, we also establish the key convergence and conditioning properties of dEIRL with respect to significant variations in 1) system initial conditions $x_0 \in G_{x_0}$ (5.9), and 2) modeling error $\nu \in G_\nu$ (5.12). For a detailed discussion of the dynamical models, see Appendices C–E. All hyperparameter selections and definitions of the performance metrics examined in these studies can be found in Section 5.1.

Solution Optimality Generalization. A complete evaluation of dEIRL over a systematic sweep of ICs and modeling errors for all three environments can be found in Tables 5.3–5.5. Running dEIRL over the IC sweep $x_0 \in G_{x_0}$ for varying modeling errors $\nu \in G_\nu$, we plot dEIRL's controller optimality error $\left\|K_{i^*,j} - K_j^*\right\|$ over the sweep in Figure 5.1. Tables 5.3, 5.4, and 5.5 provide worst-case, mean, and standard deviation data of the metrics presented in Figures 5.1 for the pendulum, jet, and DDMR, respectively.

dEIRL Training on Nominal Model. Examining dEIRL's nominal-model learning $\nu = 1$ in Figure 5.1 (blue curve) and Tables 5.3–5.5, we see that dEIRL successfully converges to the optimal controller for the nominal model $\nu = 1$ regardless of the IC chosen for all systems. Indeed, dEIRL exhibits a worst-case controller error

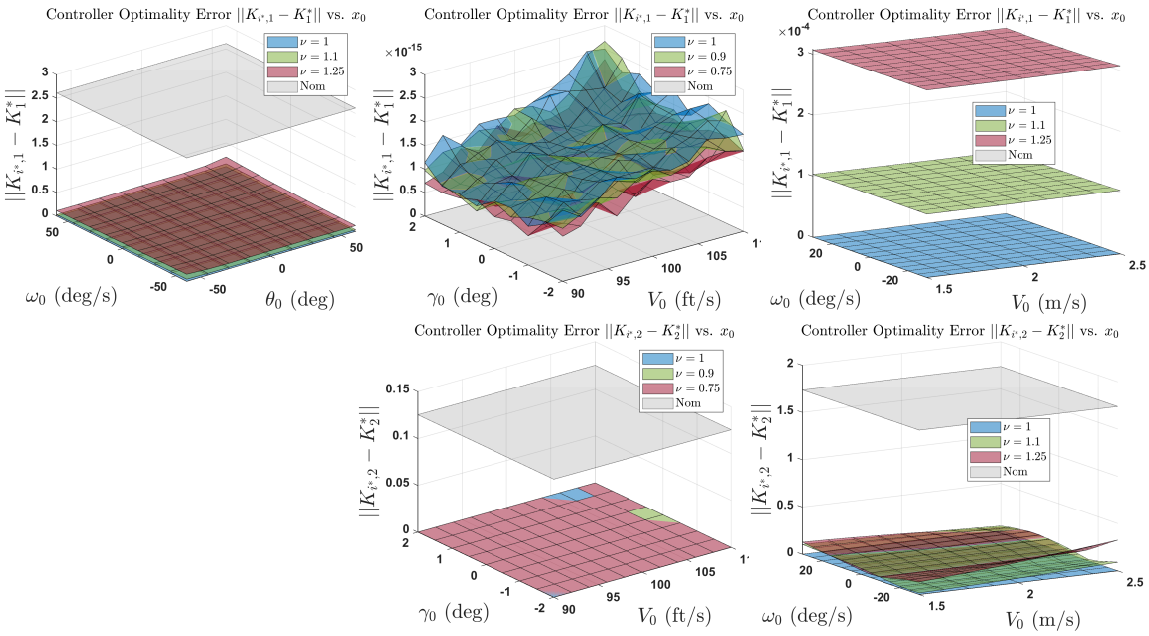


Figure 5.1: dEIRL Controller Optimality Error $\left\|K_{i^*,j} - K_j^*\right\|$ Versus Modeling Error ν and IC $x_0 \in G_{x_0}$ Ablation Study Results. First Row: Loop $j = 1$, Second Row: Loop $j = 2$. First Column: Pendulum (Note: Single-Loop $j = 1$), Second Column: Jet Aircraft, Third Column: DDMR. Gray: Controller Optimality Error of Nominal LQ Design at 25% Modeling Error.

$\|K_{i^*,1} - K_1^*\| = 3.71 \times 10^{-5}$ for the pendulum system. For the jet aircraft, dEIRL has worst-case controller errors of $\|K_{i^*,1} - K_1^*\| = 2.55 \times 10^{-15}$ in the translational loop $j = 1$ and $\|K_{i^*,2} - K_2^*\| = 2.42 \times 10^{-8}$ in the rotational loop $j = 2$. For the DDMR, dEIRL has worst-case controller errors of $\|K_{i^*,1} - K_1^*\| = 4.78 \times 10^{-9}$ in the translational loop $j = 1$ and $\|K_{i^*,2} - K_2^*\| = 5.17 \times 10^{-5}$ in the rotational loop $j = 2$. Thus, dEIRL matches the optimal performance of classical LQR when training on a nominal model and achieves real-world convergence performance in accordance with its theoretical guarantees (cf. Section 3.6). As a comparison, the FVIs also converge nicely, but they still exhibit appreciable variance between seeds (cf. Figure 5.3).

We next examine how dEIRL's solution optimality generalizes with respect to initial conditions x_0 when modeling error is introduced. As a worst case observed over the IC grid $x_0 \in G_{x_0}$ and in the presence of the most severe 25% modeling error

Table 5.3: Initial Condition x_0 Ablation Optimality Error and Conditioning Data – Pendulum

ν	Data Over G_{x_0}	$\|K_{i,1} - K_1^*\|$ Nom LQ	$\|K_{i,1} - K_1^*\|$ dEIRL $i = i^*$	$\|K_{i,1} - K_1^*\|$ % Reduc Nom→dEIRL	$\max_i \kappa(\mathbf{A}_{i,1})$ dEIRL
0%	worst	0	3.71e-05	N/A [a]	42.25
	avg		1.75e-05	N/A	23.17
	std		1.19e-05	N/A	11.82
10%	worst	1.04	0.06	94.12	39.12
	avg		0.03	96.75	22.82
	std		0.01	0.53	10.79
25%	worst	2.61	0.20	92.18	39.64
	avg		0.13	95.21	25.56
	std		0.01	0.54	10.34

[a] Not applicable for the nominal model $\nu = 1$.

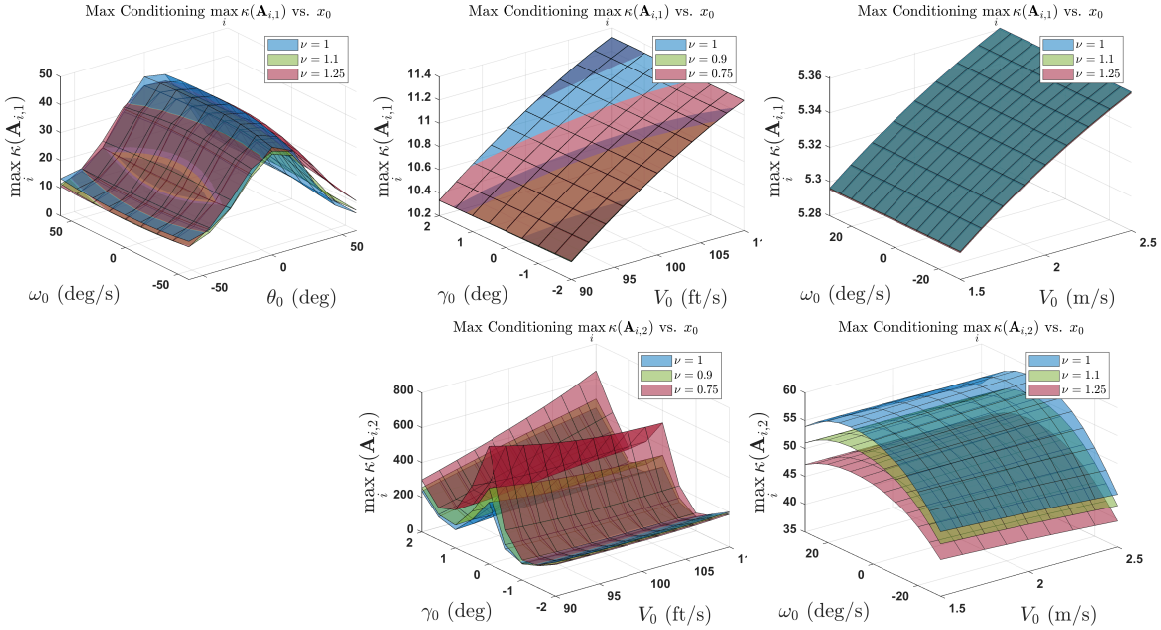


Figure 5.2: dEIRL Iteration-Wise Max Condition Number $\max_i \kappa(\mathbf{A}_{i,j})$ Versus Modeling Error ν and IC $x_0 \in G_{x_0}$ Ablation Study Results. First Row: Loop $j = 1$, Second Row: Loop $j = 2$. First Column: Pendulum (Note: Single-Loop $j = 1$), Second Column: Jet Aircraft, Third Column: DDMR.

tested, dEIRL converges to within 0.20, 1.78×10^{-8}, and 0.32 of the optimal policy K^* for the pendulum, jet, and DDMR, respectively. By comparison, the optimality errors of the nominal LQ controllers K_0 are 2.61, 0.13, and 1.74, respectively. Thus, for the pendulum we see that dEIRL's policy error $\|K_{i^*} - K^*\|$ is *at most* 0.20/2.61 (8%) the policy error of the nominal LQ design $\|K_0 - K^*\|$; i.e., a reduction of *at least* 92% for a given initial condition $x_0 \in G_{x_0}$. Similarly, dEIRL offers a reduction of at least 99.99% and 81% from the nominal LQ design for the jet and DDMR, respectively, thus demonstrating generalizability of learning with respect to environment errors.

Table 5.4: Initial Condition x_0 Ablation Optimality Error and Conditioning Data – Jet Aircraft

ν	Data	$\|K_{i,1} - K_1^*\|$			$\|K_{i,2} - K_2^*\|$			$\max_i \kappa(\mathbf{A}_{i,1})$	$\max_i \kappa(\mathbf{A}_{i,2})$
	Over G_{x_0}	Nom LQ	dEIRL $i = i^*$	% Reduc Nom→dEIRL	Nom LQ	dEIRL $i = i^*$	% Reduc Nom→dEIRL	dEIRL	dEIRL
0%	worst	0	2.55e-15	N/A [a]	0	2.42e-08	N/A [a]	11.32	432.07
	avg		1.61e-15	N/A		1.93e-08	N/A	10.87	195.13
	std		4.09e-16	N/A		2.54e-09	N/A	0.31	86.28
10%	worst	0	2.66e-15	N/A [b]	0.05	2.15e-08	99.99	11.32	483.68
	avg		1.49e-15	N/A		1.71e-08	99.99	10.87	217.39
	std		3.76e-16	N/A		2.29e-09	7.06e-06	0.31	104.41
25%	worst	0	2.27e-15	N/A [b]	0.13	1.79e-08	99.99	11.33	676.39
	avg		1.26e-15	N/A		1.38e-08	99.99	10.88	290.91
	std		3.29e-16	N/A		1.94e-09	1.55e-06	0.31	159.55

[a] Not applicable for the nominal model $\nu = 1$.

[b] Modeling error ν does not affect optimal controller K_1^* in loop $j = 1$ for the jet aircraft (cf. Appendix D.2).

Conditioning Generalization. Running dEIRL over the IC sweep $x_0 \in G_{x_0}$ for varying modeling errors $\nu \in G_\nu$, we plot dEIRL's iteration-wise max condition number $\max_i \kappa(\mathbf{A}_{i,j})$ for each learning trial over the sweep in Figure 5.2. Tables 5.3, 5.4, and 5.5 provide worst-case, mean, and standard deviation data of the metrics presented in Figure 5.2 for the pendulum, jet, and DDMR, respectively.

Examining Figure 5.2, we see that conditioning remains low for the pendulum at less than 42.25 regardless of the IC or modeling error (Table 5.3). For the jet aircraft, conditioning remains low in the lower-dimensional translational loop $j = 1$

at less than 11.33 regardless of IC or modeling error. Meanwhile, conditioning is higher in the higher-dimensional rotational/FPA loop $j = 2$, where it is seen that the worst-case IC sweep conditioning increases from 432.07 to 6776.39 from 0% to 25% modeling error (Table 5.4). Conditioning on this order is to be expected for the FPA loop on an aircraft longitudinal model, as we have seen in our previous evaluation study on the HSV (cf. Table 4.2). Finally, conditioning remains low in both loops of the DDMR at a max of 5.36 in the translational loop $j = 1$ and a max of 58.46 in the rotational loop $j = 2$ (Table 5.5). Overall, dEIRL conditioning generalizes very well with respect to a combination of varying ICs $x_0 \in G_{x_0}$ and modeling errors $\nu \in G_\nu$ for each of these three CT-RL environments.

Table 5.5: Initial Condition x_0 Ablation Optimality Error and Conditioning Data – DDMR

ν	Data	$\|K_{i,1} - K_1^*\|$			$\|K_{i,2} - K_2^*\|$			$\max_i \kappa(\mathbf{A}_{i,1})$	$\max_i \kappa(\mathbf{A}_{i,2})$
	Over G_{x_0}	Nom LQ	dEIRL $i = i^*$	% Reduc Nom→dEIRL	Nom LQ	dEIRL $i = i^*$	% Reduc Nom→dEIRL	dEIRL	dEIRL
0%	worst	0	4.78e-09	N/A [a]	0	5.17e-05	N/A[a]	5.36	58.46
	avg		4.67e-09	N/A		1.19e-05	N/A	5.33	54.52
	std		5.56e-11	N/A		9.06e-06	N/A	0.02	3.79
10%	worst	0	1.03e-04	N/A [b]	0.68	0.10	84.85	5.36	55.02
	avg		1.03e-04	N/A		0.06	91.06	5.33	51.46
	std		3.31e-07	N/A		0.03	4.07	0.02	3.38
25%	worst	0	3.08e-04	N/A[b]	1.74	0.32	81.35	5.36	50.57
	avg		3.05e-04	N/A		0.10	94.25	5.33	47.52
	std		9.56e-07	N/A		0.06	3.23	0.02	2.87

[a] Not applicable for the nominal model $\nu = 1$.

[b] Modeling error ν does not affect optimal controller K_1^* in loop $j = 1$ for the DDMR (cf. Appendix E.2).

5.4 Quantitative Comparisons between dEIRL and Deep RL FVIs

This evaluation focuses on comparing our current dEIRL to the leading deep RL FVIs [7, 8]. Training and evaluation procedures for all algorithms are described in detail in Section 5.1.

5.4.1 *Average Return Generalization to System ICs*

dEIRL, cFVI, and rFVI are trained with trajectory data generated by the same training uniform IC distribution for each of the respective environments, and the average return is then evaluated over the respective evaluation uniform IC distribution. These distributions are given in Section 5.1. The learning curves are plotted in Figure 5.3 over 20 training seeds, and the average return, variance, and success rate is tabulated in Table 5.6. As a note, for this optimal regulation dynamic control task we define "success" as a policy achieving closed-loop stability with respect to the environment dynamics (f, g) (3.1).

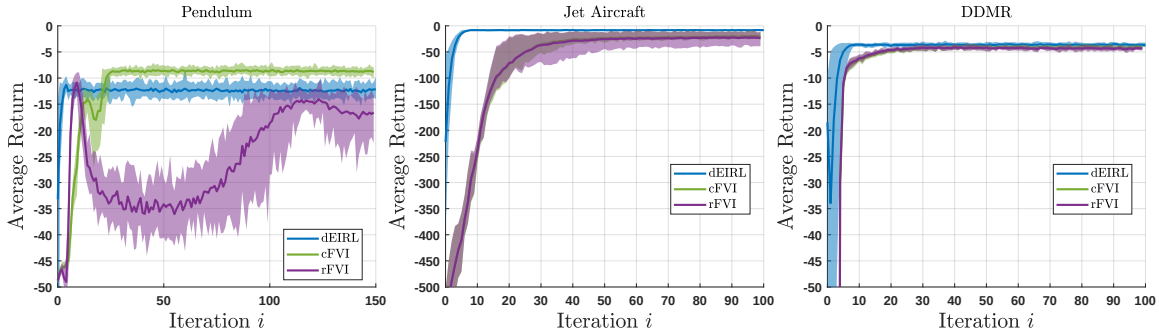


Figure 5.3: Learning Curves of cFVI and rFVI Obtained Over 20 Seeds for the Pendulum (Left), Jet Aircraft (Middle), and DDMR (Right). The Shaded Area Displays the Min/Max Range Between Seeds, as is Presented in the Original Works [7, 8].

Table 5.6: Average Return and Success Rate on Three Environments

Algorithm	Pendulum		Jet aircraft		DDMR	
	Success [%]	Return $[\mu \pm 2\sigma]$	Success [%]	Return $[\mu \pm 2\sigma]$	Success [%]	Return $[\mu \pm 2\sigma]$
dEIRL	100	-12.17 ± 2.30	100	-8.26 ± 1.21	100	-3.69 ± 0.51
cFVI	100	-8.85 ± 1.12	100	-20.50 ± 7.26	100	-4.12 ± 0.44
rFVI	100	-16.58 ± 5.36	100	-23.37 ± 16.04	100	-4.32 ± 0.46

As can be seen from Table 5.3, all methods successfully stabilize the closed-loop

system for the three environments. The FVI algorithms exhibit overall consistent learning behavior as shown in the original works [7, 8], a result confirmed independently here on SOTA environments. Indeed, cFVI edges out dEIRL's IC generalization on the pendulum, delivering a higher average return and lower variance. However, for all environments dEIRL converges much more quickly (within 5-10 iterations) than the FVIs (usually 50-100 iterations). On the two higher-dimensional, multi-input jet aircraft and DDMR models studied, dEIRL exhibits the best average return and the lowest variance. For instance, Table 5.6 shows that dEIRL exhibits a return of -8.26 ± 1.21, followed by cFVI's -20.50 ± 7.26 and then by rFVI's -23.37 ± 16.04. Overall, rFVI tends to exhibit the lowest return and highest variance with respect to varying ICs.

5.4.2 Cost Performance Generalization to Modeling Error

Figure 5.4 shows the cost difference $J_{cFVI} - J_{dEIRL}$ between cFVI and dEIRL (first row), and the difference $J_{rFVI} - J_{dEIRL}$ between rFVI and dEIRL (second row) for the nominal pendulum model $\nu = 1$ (left column), a 10% modeling error $\nu = 1.1$ (middle column), and a 25% modeling error $\nu = 1.25$ (right column). Note that wherever this difference is positive, dEIRL delivers *better* performance than the respective FVI algorithm. Table 5.7 presents the corresponding min, max, average, and standard deviation data. Figure 5.5 and Table 5.8 are laid out analogously for the jet aircraft, and Figure 5.6 and Table 5.9 are laid out analogously for the DDMR.

Several key trends emerge from Tables 5.7–5.9: 1) dEIRL achieves the lowest cost for all three systems as modeling error ν is increased, and the best modeling error generalization overall. 2) For both multi-loop systems (i.e., the jet and DDMR), dEIRL achieves lowest cost pointwise, regardless of modeling error. 3) The FVIs perform quite well. Indeed, the top left plot of Figure 5.4 shows that cFVI performance edges

out that of dEIRL for the nominal pendulum far from the origin $x = 0$. However, when modeling error is introduced (top middle 10%, top right 25%), cFVI performance degrades significantly, a trend we observe for all three systems (see individual analyses below). By contrast, rFVI degradation is less pronounced, but at the cost of inferior overall performance.

Cost Performance – Pendulum. In short, dEIRL and cFVI perform comparably overall, cFVI edging out dEIRL near the nominal model but degrading with increasing modeling error. By contrast, rFVI's performance is significantly worse than dEIRL's or cFVI's. Examining the first row of Figure 5.4, the dEIRL and cFVI policies deliver highly comparable cost performance J in a large region around the origin for the nominal model $\nu = 1$ and 10% modeling error $\nu = 1.1$. Toward the fringes of the test domain, the performance of cFVI edges out that of dEIRL slightly, by as much as -0.243 (Table 5.7). Specifically, cFVI performs better near the corners $x = (-60, -60)$ and $x = (60, 60)$; i.e., for large initial pendulum displacements and velocities in the direction of the displacement. This is perhaps intuitive, since these trajectories depart furthest from the origin. Here, the pendulum nonlinearities are strongest, and cFVI's deep network will have an approximation advantage over dEIRL's quadratic cost approximator. However, we note that is precisely in these two regions that the performance of cFVI degrades the heaviest in relation to dEIRL when the modeling error is increased. For a 25% modeling error $\nu = 1.25$, the top right plot of Figure 5.4 shows that in these corners the dEIRL policy is far superior, by as much as 0.499 at max (Table 5.7).

Meanwhile, the second row of Figure 5.4 shows that rFVI performs comparably to dEIRL and cFVI near the origin, but its policy performance degrades significantly on the same fringes. Indeed, Table 5.7 shows that dEIRL exhibits lower cost pointwise relative to rFVI, and rFVI's worst-case cost increases from 6.50 at nominal to 8.59 at

the 25% modeling error relative to dEIRL. Overall, the cost performance of dEIRL and cFVI are much more comparable, and rFVI exhibits the worst performance across the board.

Cost Performance – Jet Aircraft. For the jet aircraft, dEIRL perhaps enjoys the largest performance advantage over the FVIs of any of the systems tested. Examination of the jet cost data in Table 5.8 shows that dEIRL delivers the lowest cost pointwise regardless of modeling error. Furthermore, the cost discrepancy between the FVIs and dEIRL is the largest of the three systems tested, averaging at least 3.08 for cFVI and 3.72 for rFVI. rFVI's degradation is less severe than cFVI's, but rFVI delivers inferior cost performance overall, as much as 10.34 higher than dEIRL's at max. Meanwhile, examining Figure 5.5 shows that cFVI and rFVI exhibit similar cost performance behavior for this system. Both compare well with dEIRL for lower initial airspeeds V, but a large performance discrepancy develops at higher airspeeds. The discrepancy is seen to be slightly worse in the rFVI case.

Cost Performance – DDMR. Figure 5.6 shows the cost difference data for the DDMR. Note that dEIRL delivers lower cost J than cFVI and rFVI *pointwise*, regardless of modeling error – an important result for training on this more complicated real-world DDMR system. The performance of cFVI in relation to dEIRL degrades with increasing modeling error, the maximum performance discrepancy $J_{cFVI} - J_{dEIRL}$ increasing by 50% from 0.273 on the nominal model $\nu = 1$ to 0.414 at $\nu = 1.25$ (cf. Table 5.9). On the other hand, rFVI's performance improves with modeling error, its max performance discrepancy decreasing from 2.27 at $\nu = 1$ to 1.60 at $\nu = 1.25$. Nevertheless, as with the pendulum system, rFVI exhibits the worst overall cost performance of the three methods.

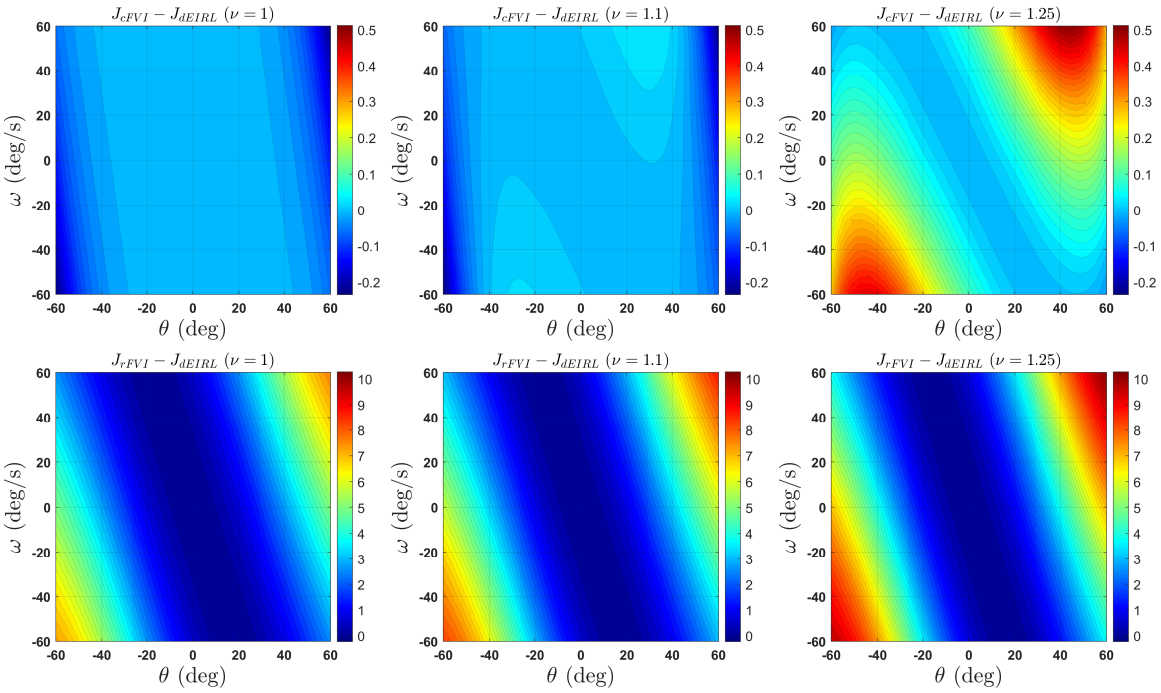


Figure 5.4: Cost Performance Results of Pendulum Model. First Row: Cost Difference $J_{cFVI} - J_{dEIRL}$ (3.2). Second Row: Cost Difference $J_{rFVI} - J_{dEIRL}$ (3.2). Left: Nominal Model $\nu = 1$ (E.8). Middle: 10% Modeling Error $\nu = 1.1$. Right: 25% Modeling Error $\nu = 1.25$.

Table 5.7: Pendulum Training Cost/Approximation Data

Function	Data	ν (C.4)		
		1	1.1	1.25
$J_{cFVI} - J_{dEIRL}$	min	-0.230	-0.24	-0.02
	max	4.63e-04	0.04	0.51
	avg	-0.02	-0.01	0.12
	std	0.04	0.04	0.12
$J_{rFVI} - J_{dEIRL}$	min	-1.33e-06	2.43e-06	-3.81e-04
	max	7.72	8.80	10.27
	avg	2.16	2.50	2.99
	std	1.92	2.20	2.61
$J(x) - V(x)$ dEIRL	min	-1.76	-2.28	-3.24
	max	1.93e-04	0.01	0.043
	avg	-0.26	-0.325	-0.44
	std	0.35	0.45	0.63
$J(x) - V(x)$ cFVI	min	-0.02	-0.01	-0.01
	max	0.03	2.75	8.29
	avg	-4.07e-03	0.63	1.88
	std	0.01	0.61	1.83
$J(x) - V(x)$ rFVI	min	-11.80	-7.98	-1.54
	max	-0.04e-04	-1.42e-03	-1.42e-03
	avg	-3.11	-2.16	-0.54
	std	2.81	1.93	0.44

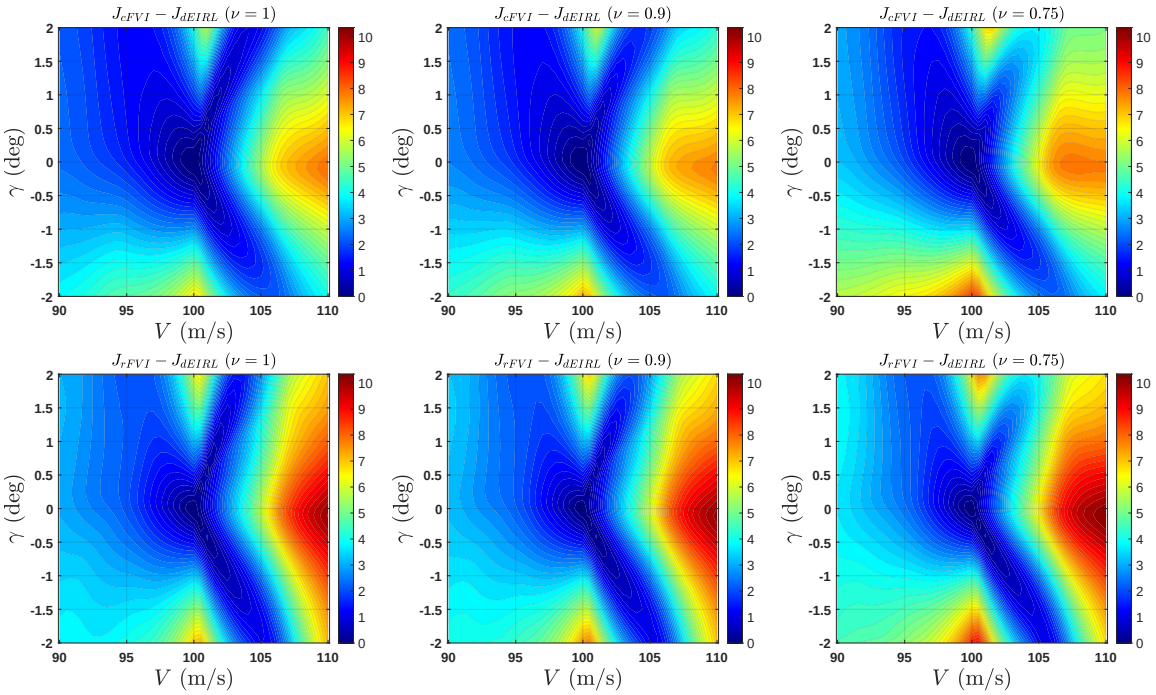


Figure 5.5: Cost Performance Results of Jet Aircraft Model. First Row: Cost Difference $J_{cFVI} - J_{dEIRL}$ (3.2). Second Row: Cost Difference $J_{rFVI} - J_{dEIRL}$ (3.2). Left: Nominal Model $\nu = 1$ (C.4). Middle: 10% Modeling Error $\nu = 0.9$. Right: 25% Modeling Error $\nu = 0.75$.

Table 5.8: Jet Aircraft Training Cost/Approximation Data

Function		Data	ν (D.2)		
			1	0.9	0.75
$J_{cFVI} - J_{dEIRL}$		min	0.00	0.00	0.00
		max	8.04	7.90	8.57
		avg	3.08	3.25	3.74
		std	1.80	1.82	1.98
$J_{rFVI} - J_{dEIRL}$		min	0.00	0.00	0.00
		max	10.15	10.23	10.34
		avg	3.72	3.86	4.22
		std	2.23	2.24	2.29
$J(x) - V(x)$	dEIRL	min	0.00	0.00	0.00
		max	11.77	13.01	15.66
		avg	3.15	3.43	4.01
		std	2.75	3.00	3.52
$J(x) - V(x)$	cFVI	min	-0.46	-0.14	-5.90e-04
		max	31.29	32.33	35.82
		avg	8.28	8.72	9.81
		std	6.57	6.69	7.14
$J(x) - V(x)$	rFVI	min	-3.44	-3.12	-2.42
		max	32.75	33.73	36.30
		avg	7.93	8.35	9.30
		std	7.16	7.26	7.56

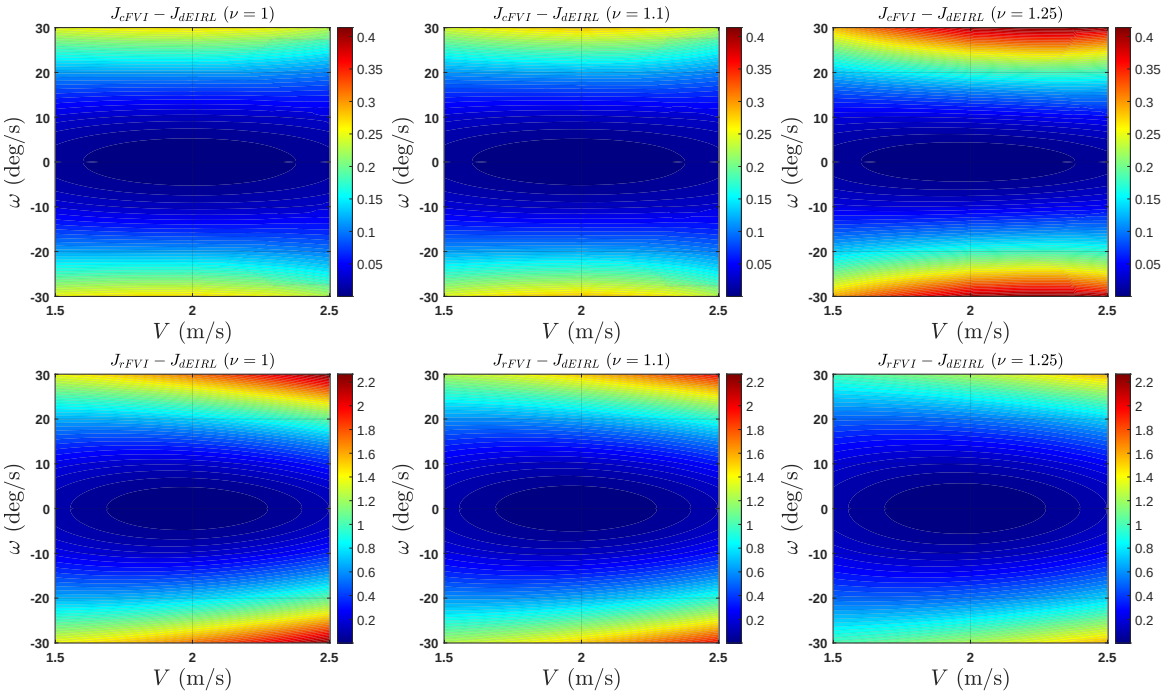


Figure 5.6: Cost Performance Results of DDMR Model. First Row: Cost Difference $J_{cFVI} - J_{dEIRL}$ (3.2). Second Row: Cost Difference $J_{rFVI} - J_{dEIRL}$ (3.2). Left: Nominal Model $\nu = 1$ (E.8). Middle: 10% Modeling Error $\nu = 1.1$. Right: 25% Modeling Error $\nu = 1.25$.

Table 5.9: DDMR Training Cost/Approximation Data

Function	Data	ν (E.8)		
		1	1.1	1.25
$J_{cFVI} - J_{dEIRL}$	min	1.11e-05	1.07e-05	1.08e-05
	max	0.27	0.28	0.41
	avg	0.09	0.09	0.13
	std	0.08	0.08	0.11
$J_{rFVI} - J_{dEIRL}$	min	1.77e-03	1.77e-03	1.77e-03
	max	2.27	1.99	1.60
	avg	0.63	0.56	0.46
	std	0.54	0.47	0.38
$J(x) - V(x)$ dEIRL	min	-0.70	-0.76	-0.93
	max	0.82	0.99	1.15
	avg	0.01	0.02	0.01
	std	0.20	0.23	0.27
$J(x) - V(x)$ cFVI	min	-0.01	-1.05e-03	-1.05e-03
	max	0.33	0.806	1.80
	avg	0.10	0.24	0.50
	std	0.09	0.16	0.38
$J(x) - V(x)$ rFVI	min	-6.08	-5.80	-5.36
	max	3.15e-04	3.31e-04	3.57e-04
	avg	-1.79	-1.72	-1.60
	std	1.30	1.23	1.13

5.4.3 Critic Network Approximation Performance Generalization to Modeling Error

Figures 5.7, 5.8, and 5.9 show the the critic network error $J - V$ for dEIRL (first row), cFVI (second row), and rFVI (third row) for the pendulum, jet, and DDMR systems, respectively. In general, it is desirable for the difference $J - V$ to be as small in magnitude as possible (so the critic is accurate) and to be negative if it does deviate from zero (so the critic underestimates the policy performance).

Generally speaking, as is the case with cost performance, dEIRL exhbits the smallest critic network error when modeling error is introduced and the best generalization overall. cFVI does an excellent job of approximating its policy cost for the nominal model but experiences significant degradation. For example, Table 5.9 for the DDMR shows that cFVI's worst-case critic error increases by 454% from $\nu = 1$ to $\nu = 1.25$, as compared to dEIRL's 39%. Meanwhile, rFVI struggles with cost approximation to a larger degree than dEIRL or cFVI; however, rFVI's worst-case cost approximation improves from 6.08 at nominal to 5.36 at 25% modeling error (cf. Table 5.9), demonstrating favorable generalization.

Approximation Performance – Pendulum. Examining Figure 5.9, the overall picture is clear: dEIRL exhibits the most consistent approximation performance, while the performance of the two FVI methods is much more sensitive to modeling error.

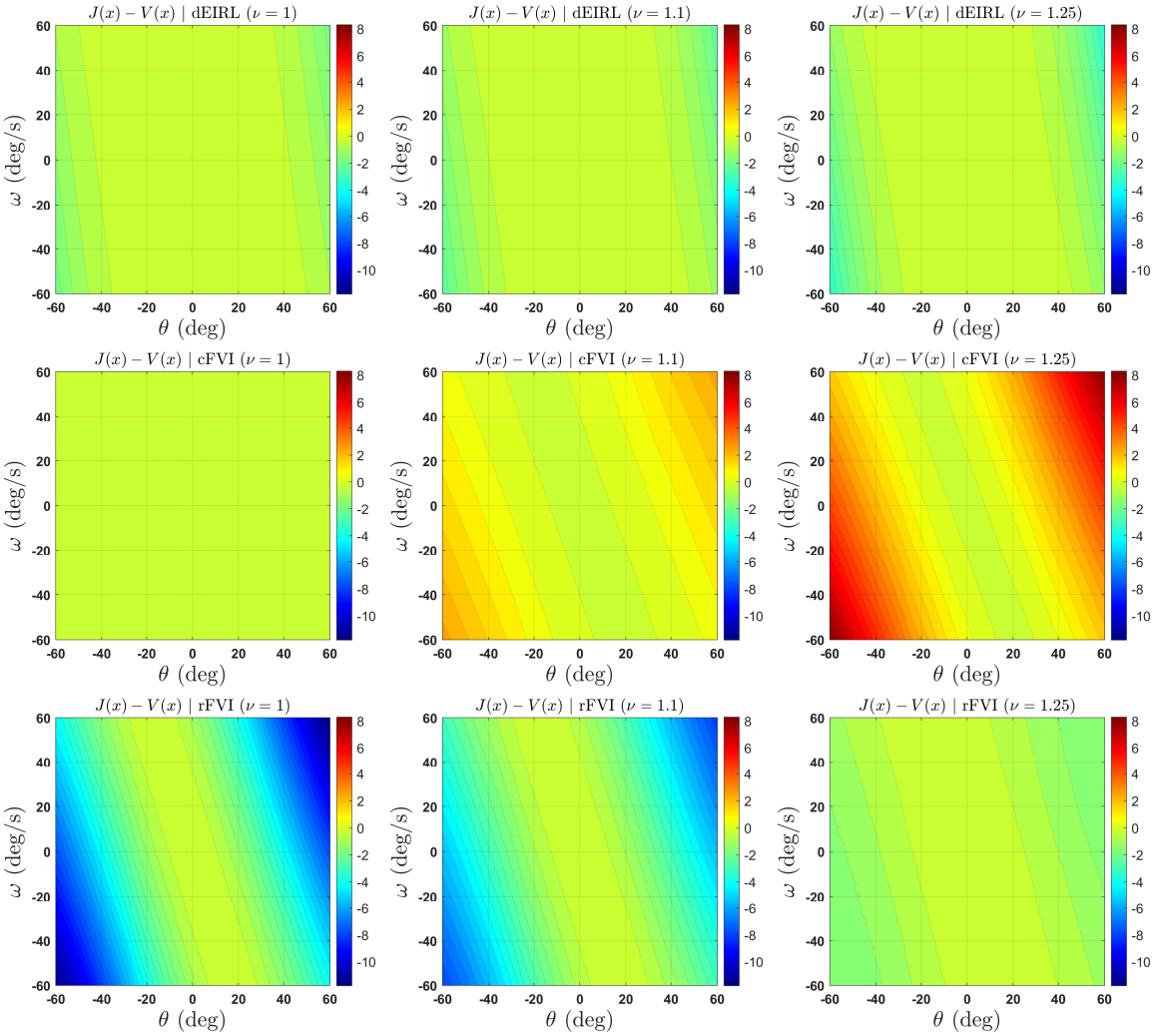


Figure 5.7: Critic NN Approximation Error $J(x) - V(x)$ of Pendulum Model. Left: Nominal Model $\nu = 1$ (C.4). Middle: 10% Modeling Error $\nu = 1.1$. Right: 25% Modeling Error $\nu = 1.25$. First Row: dEIRL. Second Row: cFVI [7]. Third Row: rFVI [8].

dEIRL's network is highly accurate in a large region around the origin and slightly underestimates the policy performance toward the fringes. This underestimation in-

creases in magnitude monotonically with the modeling error. However, the degradation is gradual, beginning at a worst-case approximation error of -1.758 for the nominal model $\nu = 1$ and decreasing to only -3.24 for a 25% modeling error $\nu = 1.25$ (Table 5.7).

cFVI's critic does an excellent job of approximating the policy cost for the nominal model $\nu = 1$, the two functions falling within 0.0338 of each other at max (Table 5.7). Given cFVI's deep network critic structure, such approximation performance is intuitive. However, with the introduction of a 10% modeling error $\nu = 1.1$, the critic approximation quality degrades significantly. Examining Table 5.7, cFVI overestimates its policy's performance by 0.627 on average, 2.75 at max for $\nu = 1.1$, compared to dEIRL's underestimation by only -0.325 on average, -2.28 at min for the same modeling error. Thus, for even mild modeling errors, dEIRL exhibits an underestimation behavior preferable to the overestimation behavior of cFVI, and in magnitude dEIRL's critic error is approximately half that of cFVI on average. We note that cFVI also tends to overestimate its policy performance for the DDMR system as well (see below), suggesting this is a common performance characteristic of the method. The discrepancies grow more pronounced as we move to the severe 25% modeling error $\nu = 1.25$. Here, dEIRL underestimates its policy performance by -3.24 at minimum, -0.444 on average. In comparison, cFVI overestimates its policy performance by 8.29 at max, 1.88 on average.

On the other hand, rFVI's approximation performance is poor for the nominal model and actually improves with increasing modeling error. rFVI underestimates its policy performance by -10.37 at worst for the nominal model, improving to -6.82 and -1.34 for 10% and 25% modeling errors, respectively. Thus, we conclude that rFVI's value function has successfully adapted to its modeling error adversary ξ_θ, at the cost of inferior performance for models closer to the nominal.

Approximation Performance – Jet Aircraft. dEIRL definitively surpasses the FVIs in approximation performance. Examination of Table 5.8 shows that the worst-case critic network error for dEIRL on the nominal model is only 11.77, compared to cFVI's 31.29 and rFVI's 32.75. Similar results hold when modeling error is introduced. The approximation performance seen visually in Figure 5.8 shows that dEIRL achieves low approximation error across the state domain. Meanwhile, cFVI and rFVI exhibit similar behavior and both struggle at higher airspeeds V.

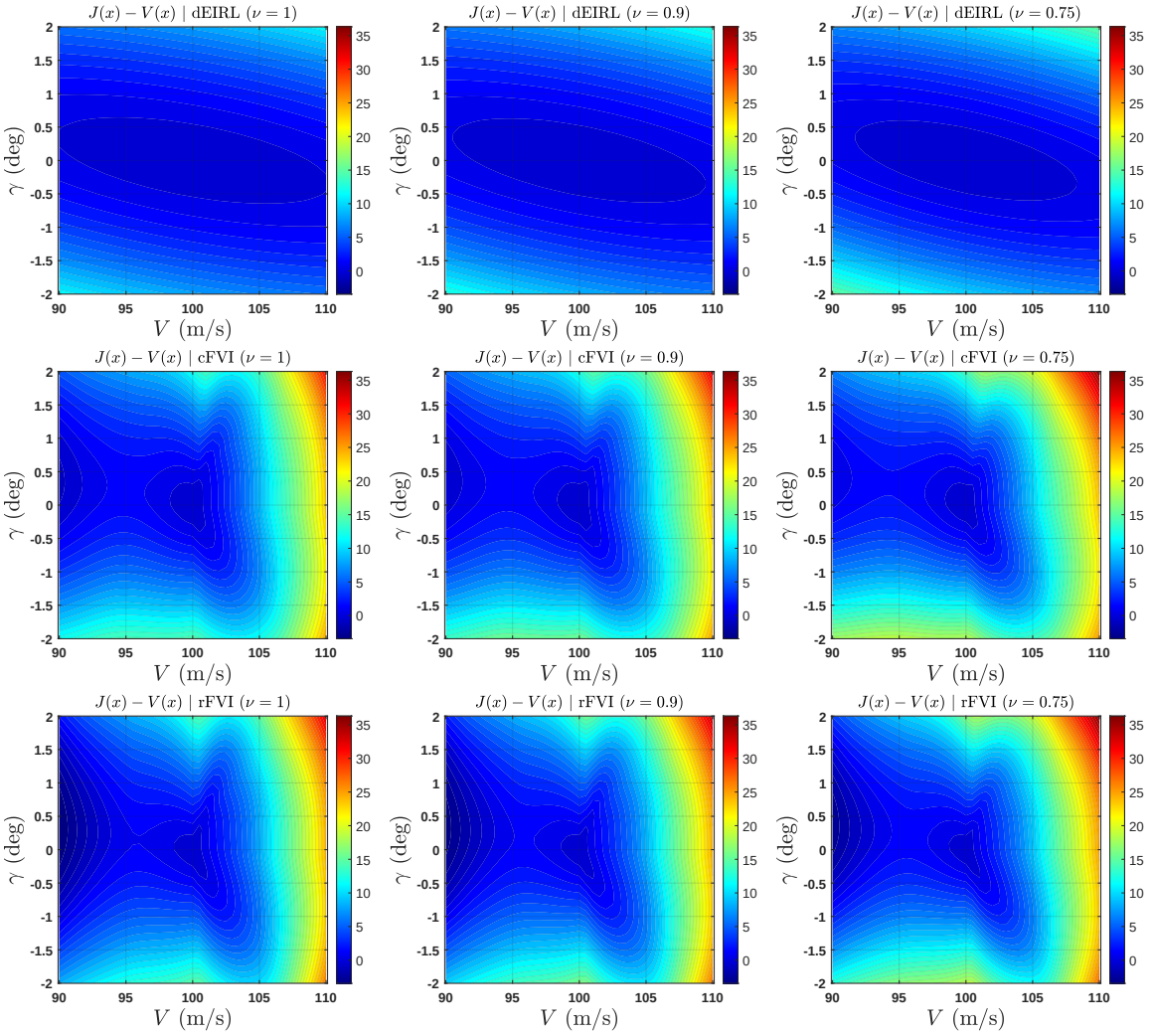


Figure 5.8: Critic NN Approximation Error $J(x) - V(x)$ of Jet Aircraft Model. Left: Nominal Model $\nu = 1$ (D.2). Middle: 10% Error $\nu = 0.9$. Right: 25% Error $\nu = 0.75$. First Row: dEIRL. Second Row: cFVI [7]. Third Row: rFVI [8].

Approximation Performance – DDMR. We display the critic approximation error $J - V$ of dEIRL, cFVI, and rFVI in Figure 5.9, laid out analogously to Figure 5.7. Overall, the trends are similar to those of the pendulum: dEIRL exhbits the most consistent approximation performance, while the performance of the FVI methods is more sensitive to modeling error.

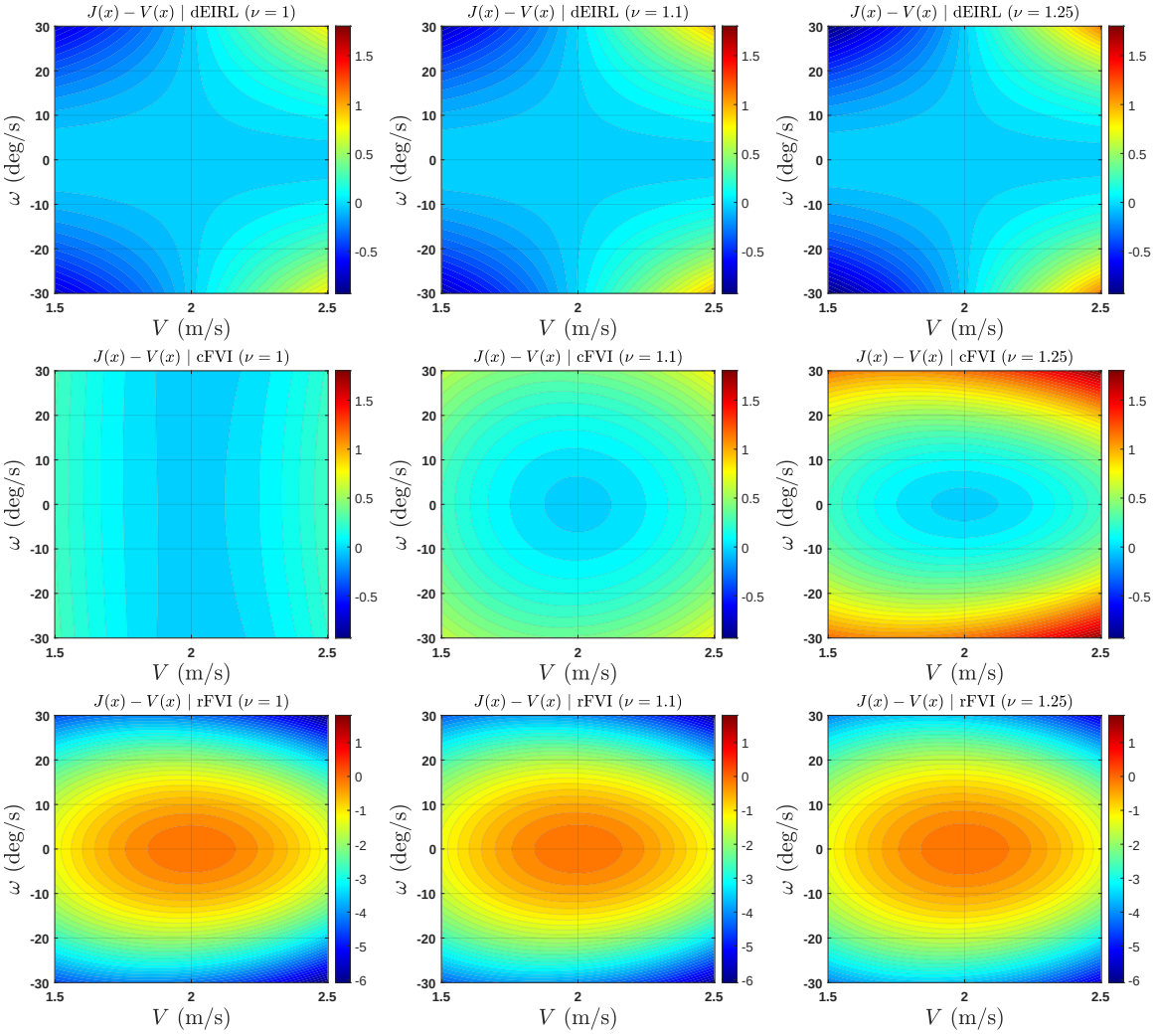


Figure 5.9: Critic NN Approximation Error $J(x) - V(x)$ of DDMR Model. Left: Nominal Model $\nu = 1$ (E.8). Middle: 10% Modeling Error $\nu = 1.1$. Right: 25% Modeling Error $\nu = 1.25$. First Row: dEIRL. Second Row: cFVI [7]. Third Row: rFVI [8]. Note: rFVI Color Normalized Independently for Legibility Purposes.

cFVI does an excellent job of cost approximation in the case of the nominal model $\nu = 1$, edging out the approximation performance of dEIRL on the corners of the test grid. However, dEIRL exhibits much more consistent cost approximation performance in the face of modeling error. Indeed, as with the pendulum system, the approximation advantage of cFVI is lost when modeling error is introduced, eventually overestimating by up to 1.80 at max for the 25% modeling error to dEIRL's 1.15. Meanwhile, rFVI struggles the most with cost approximation, universally underestimating its policy performance by on the order of -6 at minimum.

5.4.4 Closed-Loop Performance Generalization to Modeling Error

Figures 5.10, 5.11, and 5.12 plot the closed-loop step responses of the tested methods on the pendulum, jet, and DDMR (respectively) at 0%, 10%, and 25% modeling errors. The step response metrics corresponding to these figures can be found in Table 5.10. Overall, the FVI responses are either sluggish and/or exhibit large overshoot when compared to dEIRL. As corroborated by Section 5.3, dEIRL recovers the closed-loop performance of the optimal policy for all systems and outperforms the nominal classical LQ design.

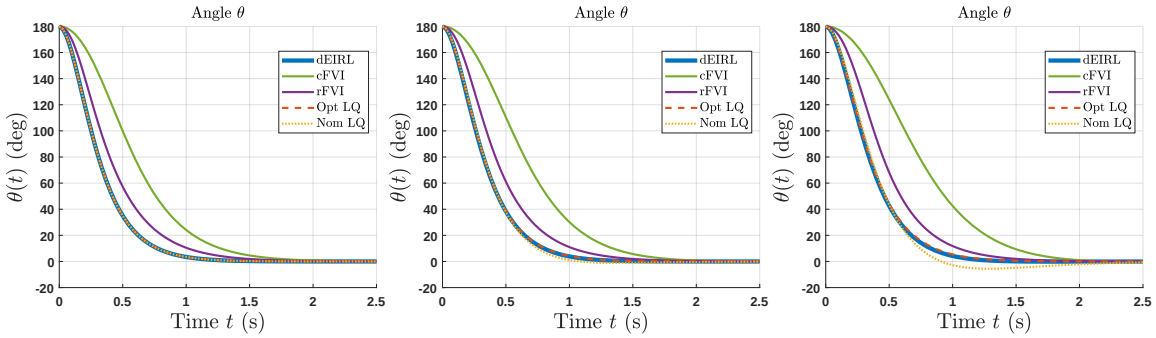


Figure 5.10: Swing-up Closed-Loop Response of Pendulum Model. Left: Nominal Model $\nu = 1$ (C.4). Middle: 10% Modeling Error $\nu = 1.1$. Right: 25% Modeling Error $\nu = 1.25$.

Table 5.10: Closed-Loop Performance Measures Generalization to Modeling Error ν

System	Alg	$t_{s,y,1\%}$ (s)			$t_{r,y,90\%}$ (s)			$M_{p,y}$ (%)		
	ν	0%	10%	25%	0%	10%	25%	0%	10%	25%
Pendulum $y = \theta$	dEIRL	1.16	1.16	1.18	0.67	0.71	0.74	0.00	0.01	0.07
	cFVI	1.80	1.80	1.95	1.10	1.19	1.36	0.00	0.02	0.19
	rFVI	1.52	1.54	1.53	0.85	0.88	0.90	0.00	0.03	0.03
	Opt LQ	1.16	1.22	1.28	0.67	0.70	0.74	0.00	0.02	0.01
	Nom LQ	1.16	0.99	2.12	0.67	0.65	0.71	0.00	0.65	3.09
Jet Aircraft $y_1 = V$	dEIRL	14.42	14.62	14.41	9.61	9.41	9.31	0.09	0.10	0.11
	cFVI	18.58	18.33	17.97	9.92	9.98	10.07	0.00	0.00	0.00
	rFVI	19.20	18.96	18.83	10.28	10.35	10.59	0.00	0.00	0.00
	Opt LQ	14.42	14.62	14.41	9.61	9.41	9.31	0.09	0.10	0.11
	Nom LQ	14.42	14.36	14.27	9.61	9.47	9.51	0.09	0.10	0.12
$y_2 = \gamma$	dEIRL	7.17	6.99	6.44	4.43	4.49	4.44	0.25	0.36	0.69
	cFVI	14.42	14.67	15.20	3.91	3.99	4.16	14.68	16.53	19.96
	rFVI	17.88	18.18	18.58	4.00	4.10	4.23	15.63	17.41	20.60
	Opt LQ	7.17	6.99	6.44	4.43	4.49	4.44	0.25	0.36	0.69
	Nom LQ	7.17	6.67	8.81	4.43	4.39	4.42	0.25	0.37	1.11
DDMR $y_1 = V$	dEIRL	5.33	5.33	5.33	3.83	3.83	3.83	0.43	0.43	0.43
	cFVI	5.45	5.46	5.52	3.94	3.96	3.77	0.55	0.55	0.55
	rFVI	5.77	5.78	5.88	3.75	3.67	3.77	0.10	0.10	0.10
	Opt LQ	5.33	5.33	5.33	3.83	3.83	3.83	0.43	0.43	0.43
	Nom LQ	5.33	5.33	5.33	3.83	3.83	3.83	0.43	0.43	0.43
$y_2 = \omega$	dEIRL	6.68	6.87	7.44	1.24	1.19	1.19	16.93	18.76	21.06
	cFVI	12.81	12.12	11.69	1.28	1.22	1.16	26.02	30.74	38.93
	rFVI	17.51	17.51	17.67	1.53	1.51	1.38	12.06	13.81	16.72
	Opt LQ	6.68	7.26	7.43	1.24	1.19	1.18	16.93	18.65	21.11
	Nom LQ	6.68	6.66	6.15	1.24	1.18	1.13	16.93	20.45	26.64

Closed-Loop Performance – Pendulum. We now study the swing-up performance of dEIRL, cFVI, the optimal LQ controller, and the nominal LQ controller (i.e., optimal for the nominal model $\nu = 1$). Figure 5.10 shows these responses for the nominal model $\nu = 1$ (left), for a 10% modeling error $\nu = 1.1$ (middle), and for a 25% modeling error $\nu = 1.25$ (right). As can be seen, the cFVI exhibits the most sluggish response, followed by rFVI. dEIRL and the LQ controllers are the most responsive. cFVI/rFVI exhibit relatively slow closed-loop responses for the DDMR system as well (see below), suggesting that FVI tends to train to the control penalty more heavily. Regardless of the modeling error, dEIRL successfully recovers the closed-loop performance if the optimal LQ controller. As seen by the increased overshoot of

the nominal LQ controller in the middle and rightmost plots of Figure 5.10, dEIRL successfully outperforms the classical LQR design in the face of modeling error.

Closed-Loop Performance – Jet Aircraft. Figure 5.11 plots the closed-loop responses to a 1 deg step FPA command. As can be seen, dEIRL exhibits a nice, monotonic response with no overshoot. By contrast, the FVIs exhibit a large overshoot transient and comparatively high settling time. The overshoot increases for both FVI algorithms as the modeling error increases.

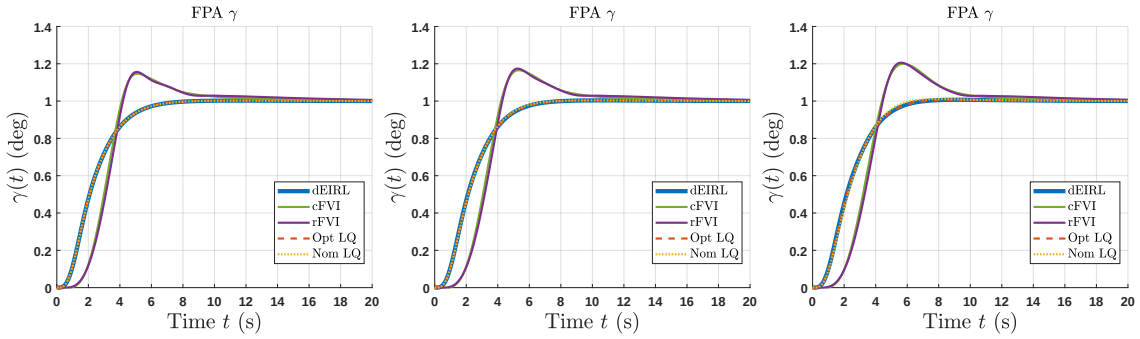


Figure 5.11: Closed-Loop Response of Jet Aircraft Model to 1 deg Step FPA Command. Left: Nominal Model $\nu = 1$ (D.2). Middle: 10% Modeling Error $\nu = 0.9$. Right: 25% Modeling Error $\nu = 0.75$.

Closed-Loop Performance – DDMR. We study the closed-loop responses of dEIRL, cFVI, rFVI, the optimal LQ controller, and the nominal LQ controller to a 30 deg/s step angular velocity command in Figure 5.12. As can be seen, regardless of the modeling error ν, the closed-loop response of the cFVI controller has similar rise time to dEIRL, but with significant overshoot and slow settling time. The cFVI overshoot increases with increasing modeling error. Meanwhile, the rFVI response has similar overshoot to dEIRL, but with relatively long rise time and very sluggish settling time. Regardless of the modeling error, dEIRL successfully recovers the closed-loop performance of the optimal LQ controller. As seen by the increased overshoot of the nominal LQ controller in the middle and rightmost plots of Figure

5.12, dEIRL successfully outperforms the classical LQR design.

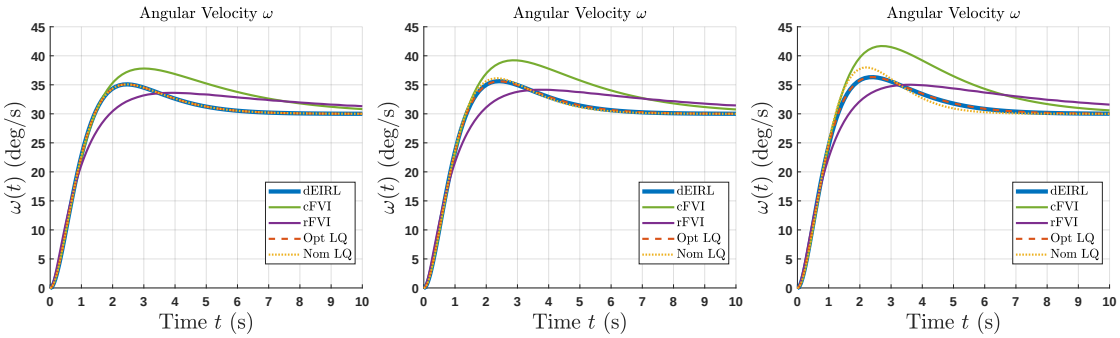


Figure 5.12: Closed-Loop Response of DDMR Model to 30 deg/s Step Angular Velocity Command. Left: Nominal Model $\nu = 1$ (E.8). Middle: 10% Modeling Error $\nu = 1.1$. Right: 25% Modeling Error $\nu = 1.25$.

Algorithm Time/Data/Parameter Complexity. Table 5.11 lists key algorithm complexity parameters for dEIRL and FVIs. On the DDMR, for example, the ratio of dEIRL/FVI for simulations required: 1/5,000,000, data samples: 1/6,000,000, network weights: 1/12,000, training epochs: 1/400, number of hyperparameters: 1/4, and training time: 1/3,000. As a result, we are able to conduct 160 times the number of learning trials for dEIRL in these studies than for FVI.

Table 5.11: dEIRL Versus FVI: Algorithm Time/Data Complexity

Parameter	Pendulum		Jet Aircraft		DDMR	
	dEIRL	cFVI/rFVI	dEIRL	cFVI/rFVI	dEIRL	cFVI/rFVI
# Simulations Required	1	1.05e+07 / 3.84e+06	1	5.12e+06	1	5.12e+06
# Data Samples Required	15	3.45e+08 / 1.73e+08	45	2.30e+08	35	2.30e+08
# NN Weights	3	79,104	13	79,104	6	79,104
# Epochs Required	5	2,000 / 3,000	5	2,000	5	2,000
Avg Training Time (s) [a]	0.17	6.88e+03 / 8.98e+03	4.25	8.18e+03 / 7.98e+03	2.58	6.30e+03 / 6.04e+03
# Trials/Seeds Tested	1,640 [b]	20	1,109	20	1,307	20

[a] Averaged over 20 seeds (cf. Section 5.4.1).

[b] Includes 20 seeds in IC ablation (Section 5.4.1), plus sweeps over IC grids $x_0 \in G_{x_0}$ and modeling error grids $\nu \in G_\nu$ (Section 5.3).

5.5 Discussion

In the context of current ADP and deep RL CT-RL methods, we formulate a model-based dEIRL algorithm which leverages nonlinear learning alongside input/output insights of the environment and Kleinman control structures for data efficiency. dEIRL leads to new CT-RL results on SOTA environments (jet aircraft and DDMR ground robot new to CT-RL). All three CT-RL classes represent different approaches to the learning control problem. dEIRL presents theoretical guarantees, and its learning performance at least matches, and often outperforms, the SOTA deep RL FVIs in terms of 1) policy cost performance, 2) critic network approximation performance, 3) closed-loop time-domain performance, 4) algorithm data/time efficiency, and 5) generalization to modeling error. Yet, dEIRL's efficiency requires knowledge of the environment, and dEIRL only considers Q-R cost structures (3.2). Meanwhile, ADP presents strong analytical results and may not require knowledge of the environment (f, g), but these methods have not been proven for meaningful applications, as only evaluations of simple systems with known optimal solutions are available. Furthermore, ADPs generally restrict to Q-R cost as well. Finally, deep RL FVIs are learning-driven methods with significant empirical promise and generalizability, as independently verified by the new SOTA evaluations we conduct on these algorithms in Section 5.4. These methods also consider flexible cost structures including dense/sparse costs. However, FVIs require the most dynamic knowledge of the three classes, and theoretical results are yet to be developed.

We provide a much-needed first-of-its-kind quantitative performance evaluation of classical-based dEIRL and deep FVI CT-RL methods, revealing the following key insights: 1) FVI's developed training procedure [7] in computer simulation on a nominal model makes cFVI particularly sensitive to modeling error performance degradation.

rFVI attempts to robustify this vulnerability by training under adversarial input [8], which ultimately mitigates degradation at the cost of inferior overall performance. By contrast, dEIRL enables more focused policy training on a single simulation trial collected from the actual system. 2) FVIs exhibit significant time/data complexity in relation to dEIRL, requiring three orders of magnitude more time and six orders of magnitude more simulations to train. 3) dEIRL exhibits a clear cost, approximation, and time-domain performance advantage for the higher-order, multi-loop systems studied in this work, empirically validating our physics-based decentralization paradigm developed for these systems.

Chapter 6

EVALUATION STUDY: COMPREHENSIVE DEIRL ABLATIONS WITH RESPECT TO MODELING ERRORS AND INITIAL CONDITIONS ON HYPERSONIC VEHICLE MODEL

6.1 Setup and Hyperparameter Selection

These evaluations were performed in MATLAB R2022b, on an NVIDIA RTX 2060, Intel i7 (9th Gen) processor. All numerical integrations in this work are performed in MATLAB's adaptive `ode45` solver to ensure solution accuracy. All code and datasets developed for this work can be found at [154].

6.1.1 Hyperparameter Selection

Cost Structure. We select penalty matrices $Q_1 = \texttt{diag}(1.5, 5)$, $R_1 = 7.5$ in the velocity loop $j = 1$ and $Q_2 = \texttt{diag}(100, 150, 0.5, 0)$, $R_2 = 1$ in the FPA loop $j = 2$. These penalties were chosen such that the resulting optimal LQR controllers achieve the closed-loop design Specifications B.3.1 on the nominal nonlinear HSV model (frequency response and step response characteristics given in Tables 6.2 and 6.3, respectively).

Excitation Signals. We choose the exploration noise d, and reference command r based on the preliminary studies conducted on this HSV model [148], which generally place the dominant frequency content near the peak of the respective closed-loop map (i.e., the P-sensitivity $T_{d_i \to y}$ and complementary sensitivity $T_{r \to y}$, respectively) in order to maximize excitation efficiency. For the exploration noise d, we choose $d_1(t) = 0.01 \cos\left(\frac{2\pi}{250} t\right)$ and $d_2(t) = \sin\left(\frac{2\pi}{6} t\right) + 1.5 \cos\left(\frac{2\pi}{25} t\right) + \cos\left(\frac{2\pi}{100} t\right)$. For the

reference command r, we choose $r_1(t) = 5\cos\left(\frac{2\pi}{10}t\right) + 5\sin\left(\frac{2\pi}{25}t\right) + 50\sin\left(\frac{2\pi}{100}t\right)$ and $r_2(t) = 0.03\sin\left(\frac{2\pi}{6}t\right) + 0.015\sin\left(\frac{2\pi}{15}t\right)$.

Learning Hyperparameters. We systematically select these parameters based on insights of the system physics and natural dynamical behavior. These include sample period $T_s = t_k - t_{k-1}$, number of samples collected l, number of iterations i^*, and initial stabilizing controller K_0. For sample period, we choose $T_{s,1} = 6$ s in the velocity loop $j = 1$, and a shorter $T_{s,2} = 2$ s in the FPA loop $j = 2$ to capture the trajectory features of the higher-bandwidth dynamics. For number of samples, we choose $l_1 = 15$, $l_2 = 25$ – higher in the FPA loop $j = 2$ because this loop is higher-dimensional ($n_1 = 2$, $n_2 = 4$). For the number of iterations, $i_1^* = i_2^* = 10$ was observed sufficient for learning convergence. Lastly, we choose initial stabilizing controllers $K_{0,1}$, $K_{0,2}$. In general, these controllers may be chosen arbitrary, as long as they are stabilizing (Theorem 3.6.3). However, for sake of comparison with a nominal classical LQR design, we initialize these as LQR controllers with associated penalties $Q_{1_0} = I_2$, $R_{1_0} = 12.5$, $Q_{2_0} = \texttt{diag}(1, 1, 0, 0)$, $R_{2_0} = 0.025$. These penalties were chosen such that the nominal LQR design $K_0 = \texttt{diag}(K_{0,1}, K_{0,2})$ satisfies the same closed-loop design Specification B.3.1 which we require of the optimal LQR design. As an aside, one could just as well make the choice $Q_{1_0} = Q_1$, $R_{1_0} = R_1$, $Q_{2_0} = Q_2$, $R_{2_0} = R_2$; i.e., select the initial controller as the LQR design optimal for the nominal model. This is the practice followed in the initial evaluations conducted in Section 4, wherein dEIRL exhibits controller optimality reductions $\left\|K_{0,j} - K_j^*\right\| \to \left\|K_{i^*,j} - K_j^*\right\|$ on the order of 90% as modeling error is introduced (cf. Table 4.3). Since this capability is already well-documented, here we choose to present the algorithm a more "challenging" learning problem from the perspective of convergence by initializing the parameters to a controller in specification but further in norm from the optimal.

6.1.2 Implementation and Training Procedures

Modeling Errors ν Tested. The trials of Sections 6.2, 6.4, and 6.3 study the effects of perturbing a single modeling error parameter in lift coefficient ν_L (B.4), drag coefficient ν_D (B.6), and pitch moment coefficient $\nu_{\mathcal{M}}$ (B.8). For these trials, the modeling errors are tested over the following grids of values

$$G_{\nu_L} = [1 : -0.025 : 0.75], \qquad G_{\nu_D} = [1 : 0.025 : 1.25], \qquad G_{\nu_{\mathcal{M}}} = [1 : 0.025 : 1.25], \tag{6.1}$$

i.e., 0%–25% modeling error with a step size of 2.5%. The direction of the respective perturbation ($\nu > 1$ or $\nu < 1$) is chosen as the direction which decreases the HSV's RHPZ/RHPP ratio, thereby presenting the algorithm with the greatest possible learning challenge. The modeling error ablation of Section 6.5 studies modeling error in two parameters simultaneously, over sweep grids in lift/drag of $G_{\nu_L} \times G_{\nu_D}$, lift/pitch moment $G_{\nu_L} \times G_{\nu_{\mathcal{M}}}$, and drag/pitch moment $G_{\nu_D} \times G_{\nu_{\mathcal{M}}}$. Finally, the random modeling error ablation of Section 6.6 studies 10,000 trials of modeling error, wherein all three parameters are simultaneously perturbed, each in a uniform distribution $\mathcal{U}(0.9, 1.1)$ (10% bidirectional disturbance). We choose this uniform distribution to keep our results comparable to the leading CT-RL numerical studies in deep RL [7, 8], which favor uniform distributions in modeling error in order to increase weight on the edge cases of the distribution.

System Initial Conditions (ICs) x_0 Tested. Section 6.4 performs dEIRL learning ablations with respect to varying ICs x_0 over the following grid of values

$$G_{x_0} = [-100 : 25 : 100] \text{ ft/s} \times [-1 : 0.25 : 1] \text{ deg}, \tag{6.2}$$

Table 6.1: Closed-Loop Performance Metrics

Metric Number	Indicator Function	Design Requirement
1	I_S	Closed-loop stability
2 (3)	$I_{V,t_{s,10\%}25}$ ($I_{V,t_{s,10\%}50}$)	10% settling time $\leq$ 25 s (50 s)
4 (5)	$I_{V,t_{s,1\%}75}$ ($I_{V,t_{s,1\%}100}$)	1% settling time $\leq$ 75 s (100 s)
6 (7)	$I_{V,t_{r,90\%}25}$ ($I_{V,t_{r,90\%}30}$)	90% rise time $\leq$ 25 s (30 s)
8 (9)	I_{V,M_p5} (I_{V,M_p10})	Percent overshoot $\leq$ 5% (10%)
10 (11)	I_{V,δ_T20} (I_{V,δ_T25})	Maximum change in throttle setting $\delta_T \leq$ 20% (25%)
12 (13)	$I_{V,\delta_E0.25}$ ($I_{V,\delta_E0.5}$)	Maximum change in elevator $\delta_E \leq$ 0.25 deg (0.5 deg)
14 (15)	$I_{V,\Delta\gamma0.01}$ ($I_{V,\Delta\gamma0.05}$)	Maximum change in FPA $\gamma \leq$ 0.01 deg (0.05 deg)
16 (17)	$I_{\gamma,t_{s,10\%}7.5}$ ($I_{\gamma,t_{s,10\%}10}$)	10% settling time $\leq$ 7.5 s (10 s)
18 (19)	$I_{\gamma,t_{s,1\%}10}$ ($I_{\gamma,t_{s,1\%}15}$)	1% settling time $\leq$ 10 s (15 s)
20 (21)	$I_{\gamma,t_{r,90\%}5}$ ($I_{\gamma,t_{r,90\%}7.5}$)	90% rise time $\leq$ 5 s (7.5 s)
22 (23)	I_{γ,M_p5} (I_{γ,M_p10})	Percent overshoot $\leq$ 5% (10%)
24 (25)	I_{γ,δ_T25} (I_{γ,δ_T50})	Maximum change in throttle setting $\delta_T \leq$ 25% (50%)
26 (27)	I_{γ,δ_E5} ($I_{\gamma,\delta_E7.5}$)	Maximum change in elevator $\delta_E \leq$ 5 deg (7.5 deg)
28 (29)	$I_{\gamma,\Delta V0.15}$ ($I_{\gamma,\Delta V0.25}$)	Maximum change in velocity $V \leq$ 0.15% (0.25%)

meanwhile, the remainder of the state variables are initialized to their trim values x_e. We choose these grid bounds because the closed-loop performance metrics studied in Section 6.6 (cf. Table 6.1) study specifications on velocity commands of 100 ft/s and FPA commands of 1 deg. For the studies of Sections 6.2, 6.5, 6.3, and 6.6 which focus on modeling error effects, ICs are set to trim $x_0 = x_e$.

Algorithm Conditioning. This work presents in-depth studies of dEIRL algorithm conditioning, which has proven a central numerical design limitation for existing CT-RL algorithms [137]. Given a particular learning trial (i.e., associated with a fixed set of modeling error parameters ν and initial conditions x_0), we examine the iteration-wise max conditioning defined as $\max_{0\leq i\leq i^*-1} \kappa(\mathbf{A}_{i,j})$ $(j = 1, 2)$, where $\mathbf{A}_{i,j}$ is the dEIRL learning regression matrix (3.27). This yields the worst-case conditioning exhibited by the algorithm over all iterations of execution for a given learning trial.

6.1.3 Feedback Linearization (FBL) Benchmark Tested [5, 6].

In order to present comparative analyses of dEIRL performance against leading classical flight control methods, in this work we study the same robust FBL

control framework for which this HSV model was originally developed as demonstration [5, 6]. For this method, we select the following LQ parameters for its feedback-linearized performance: $Q_1 = \texttt{diag}(8.54\times10^{-6}, 0.34, 0.86, 47.93)$, $R_1 = 0.89$, $Q_2 = \texttt{diag}(0.5, 0.3, 1, 0.5)$, $R_2 = 0.35$. The parameters Q_1, R_1 for the velocity loop $j = 1$ are chosen identical to the robust control framework [5], whose optimization is configured to minimize failure percentage of closed-loop performance metrics based upon 100 ft/s step velocity commands (cf. Table 6.1). Thus, to prevent biasing results against FBL, our learning IC ablation results in Section 6.4 and our closed-loop response performance results in Sections 6.3 and 6.6 also study 100 ft/s velocity responses. Meanwhile, [5, 6] study outputs $y = [V, h]^T$, so for the parameters Q_2, R_2 in the FPA loop $j = 2$, we make selections to satisfy the closed-loop performance Specifications B.3.1 and hence make FBL numerically-comparable to the other methods.

6.2 Frequency Response Performance Generalization to Modeling Error

In this study, we examine the frequency response performance of the nominal LQ, dEIRL, and optimal LQ with respect to the sensitivity S_e, S_u and complementary sensitivity T_e, T_u at the error e and controls u, respectively (cf. Figure B.3). We examine these maps at 0%, 10%, and 25% modeling errors in lift coefficient ν_L (B.4), drag coefficient ν_D (B.6), and pitch moment coefficient $\nu_{\mathcal{M}}$ (B.8). The peak closed-loop map data is summarized in Table 6.2, and we have plotted the frequency responses of the sensitivity and complementary sensitivity S_e, T_e, S_u, T_u at the error e and controls u with respect to varying modeling error ν in Figures 6.1–6.4.

Examining Table 6.2, we note that regardless of the modeling error tested (in lift ν_L, drag ν_D, or pitching moment $\nu_{\mathcal{M}}$), and regardless of the severity of the modeling error (0%–25%), dEIRL successfully recovers the closed-loop frequency response properties of the optimal controller. Indeed, regardless of the modeling error type or

Table 6.2: Peak Closed-Loop Maps Versus Modeling Error ν

ν		$\|S_e\|_{\mathcal{H}^\infty}$			$\|T_e\|_{\mathcal{H}^\infty}$			$\|S_u\|_{\mathcal{H}^\infty}$			$\|T_u\|_{\mathcal{H}^\infty}$		
		Nom	dEIRL	Opt	Nom	dEIRL	Opt	Nom	dEIRL	Opt	Nom	dEIRL	Opt
0%		6.05	4.76	4.76	4.50	3.29	3.29	0.10	0.07	0.07	5.14	4.17	4.17
10%	L	5.56	4.69	4.70	4.33	3.42	3.30	0.06	0.03	0.08	4.68	3.88	4.12
	D	6.05	4.76	4.76	4.50	3.29	3.29	0.10	0.07	0.07	5.14	4.18	4.17
	$\mathcal{M}$	7.29	4.86	4.59	5.71	3.88	3.22	1.95	0.01	0.07	6.83	3.77	4.40
25%	L	4.91	4.62	4.60	4.33	3.68	3.31	0.06	0.02	0.11	4.01	3.33	4.03
	D	6.06	4.77	4.76	4.50	3.28	3.29	0.10	0.07	0.07	5.15	4.19	4.17
	$\mathcal{M}$	10.32	5.07	4.31	9.17	4.05	3.09	7.74	0.03	0.06	10.68	4.18	4.70

value, dEIRL recovers the $\mathcal{H}^\infty$ norm of the optimal controller for all maps to within 0.96 dB at max (worst: T_e for 25% pitch moment modeling error). In the absence of modeling error, the nominal LQR controller achieves closed-loop peaking comparable to dEIRL and the optimal at the controls u (to be expected, since these methods all inherit LQR performance guarantees at the controls [136]), and its peaking is only slightly higher at the error e. The nominal design's peaking in the sensitivity at the controls is $\|S_u\|_{\mathcal{H}^\infty} \approx 0$ dB, similar to dEIRL and the optimal controller. As a note, LQR guarantees $\|S_u\|_{\mathcal{H}^\infty} = 0$ dB [136], the slight numerical deviations observed from the theoretical guarantee arising from the decentralized controller structure (cf. Appendix B.3). The nominal's peak in the complementary sensitivity at the controls is $\|T_u\|_{\mathcal{H}^\infty}$ is also comparable (5.14 dB) to dEIRL and the optimal (4.17 dB). As a note, LQR guarantees $\|T_u\|_{\mathcal{H}^\infty} \leq 6$ dB [136]. Meanwhile, the nominal's peaking at the error e is comparable to that of dEIRL and the optimal, generally within 1 dB.

Furthermore, by virtue of recovery of the optimal closed-loop performance for a given modeling error, dEIRL's peaking degrades little with respect to increasing modeling error. The worst-case increase in $\mathcal{H}^\infty$ norm for any map and modeling error type is for the complementary sensitivity at the error T_e with respect to pitch moment modeling error $\nu_{\mathcal{M}}$, for which dEIRL's peak $\|T_e\|_{\mathcal{H}^\infty}$ increases by 0.76 dB from 3.29 dB at 0% modeling error to 4.05 dB at 25% modeling error.

By contrast, the nominal LQR controller's closed-loop frequency response perfor-

mance degrades significantly with respect to modeling error. The degradation is most pronounced with respect to modeling error in pitching moment coefficient $\nu_{\mathcal{M}}$, as pictured at the error e in Figures 6.1 and 6.2. As can be seen, the peaking for dEIRL and the optimal remains relatively constant, dEIRL successfully capturing the optimal closed-loop response. Meanwhile, the nominal's peaking increases significantly from 0% to 25% modeling error: 6.05 dB to 10.32 dB (S_e), and 4.33 dB to 9.17 dB (T_e). Similar increases are observed at the controls u (cf. Table 6.2). At the controls u, the nominal's peaking increases from 0% to 25% modeling error: 0.10 dB to 7.74 dB (S_u), and 5.14 dB to 10.68 dB (T_u).

6.2.1 Plots: Sensitivity at Error S_e

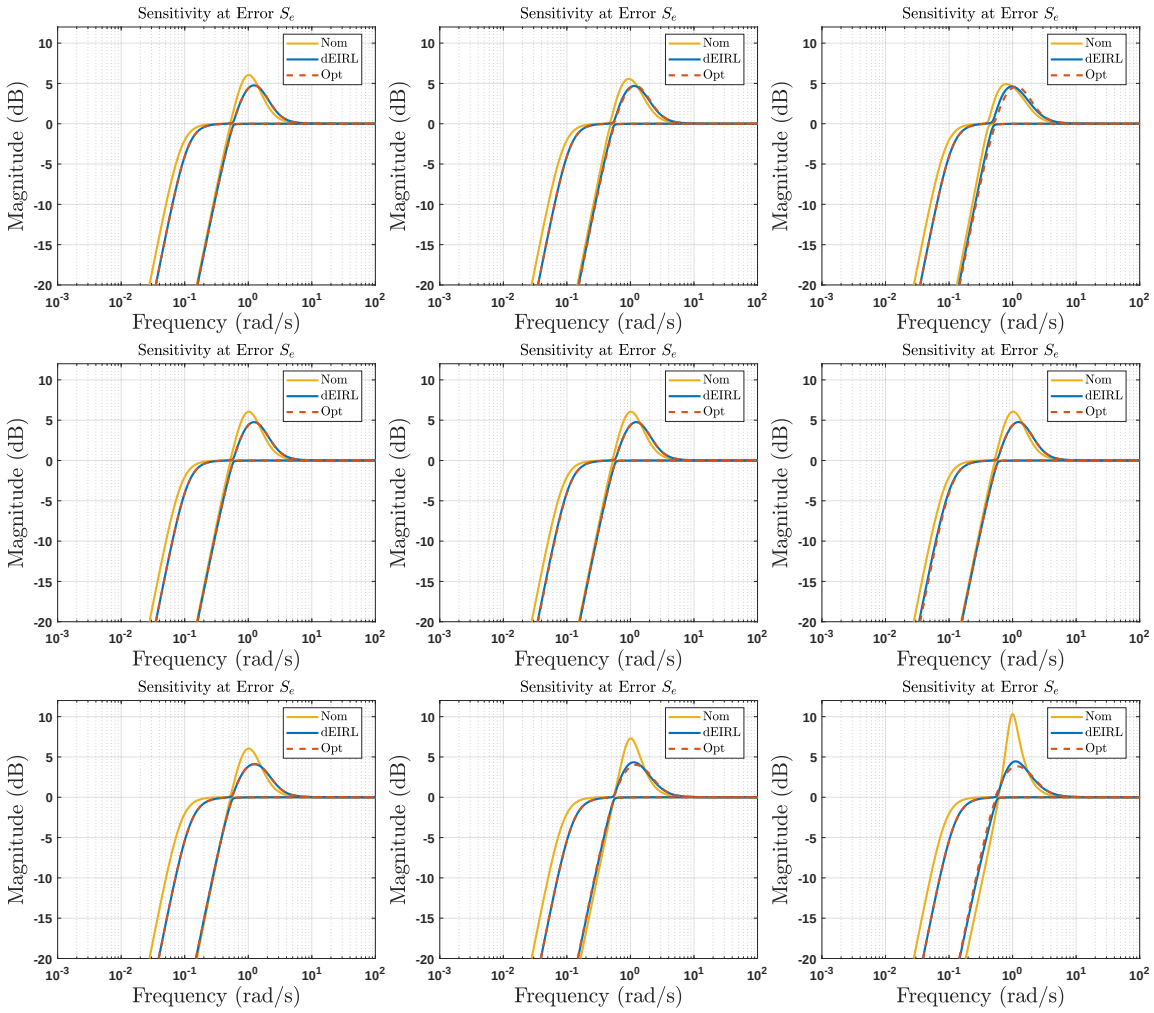


Figure 6.1: Sensitivity at Error S_e Versus Modeling Error ν. First Row: Lift Coefficient Sweep ν_L. Second Row: Drag Coefficient Sweep ν_D. Third Row: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$. First Column: 0% Modeling Error. Second Column: 10% Modeling Error. Third Column: 25% Modeling Error.

6.2.2 Plots: Complementary Sensitivity at Error T_e

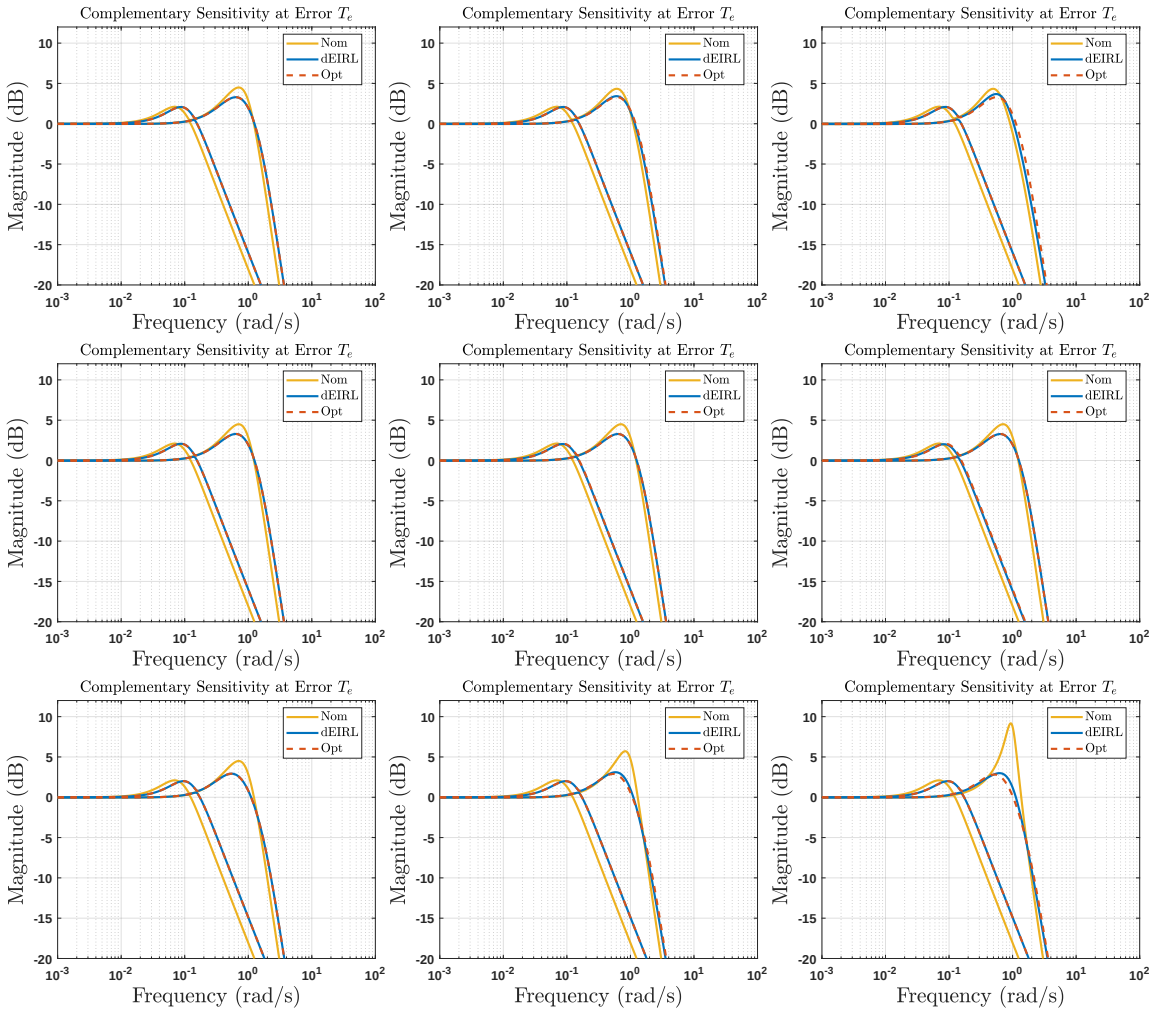


Figure 6.2: Complementary Sensitivity at Error T_e Versus Modeling Error ν. First Row: Lift Coefficient Sweep ν_L. Second Row: Drag Coefficient Sweep ν_D. Third Row: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$. First Column: 0% Modeling Error. Second Column: 10% Modeling Error. Third Column: 25% Modeling Error.

6.2.3 Plots: Sensitivity at Controls S_u

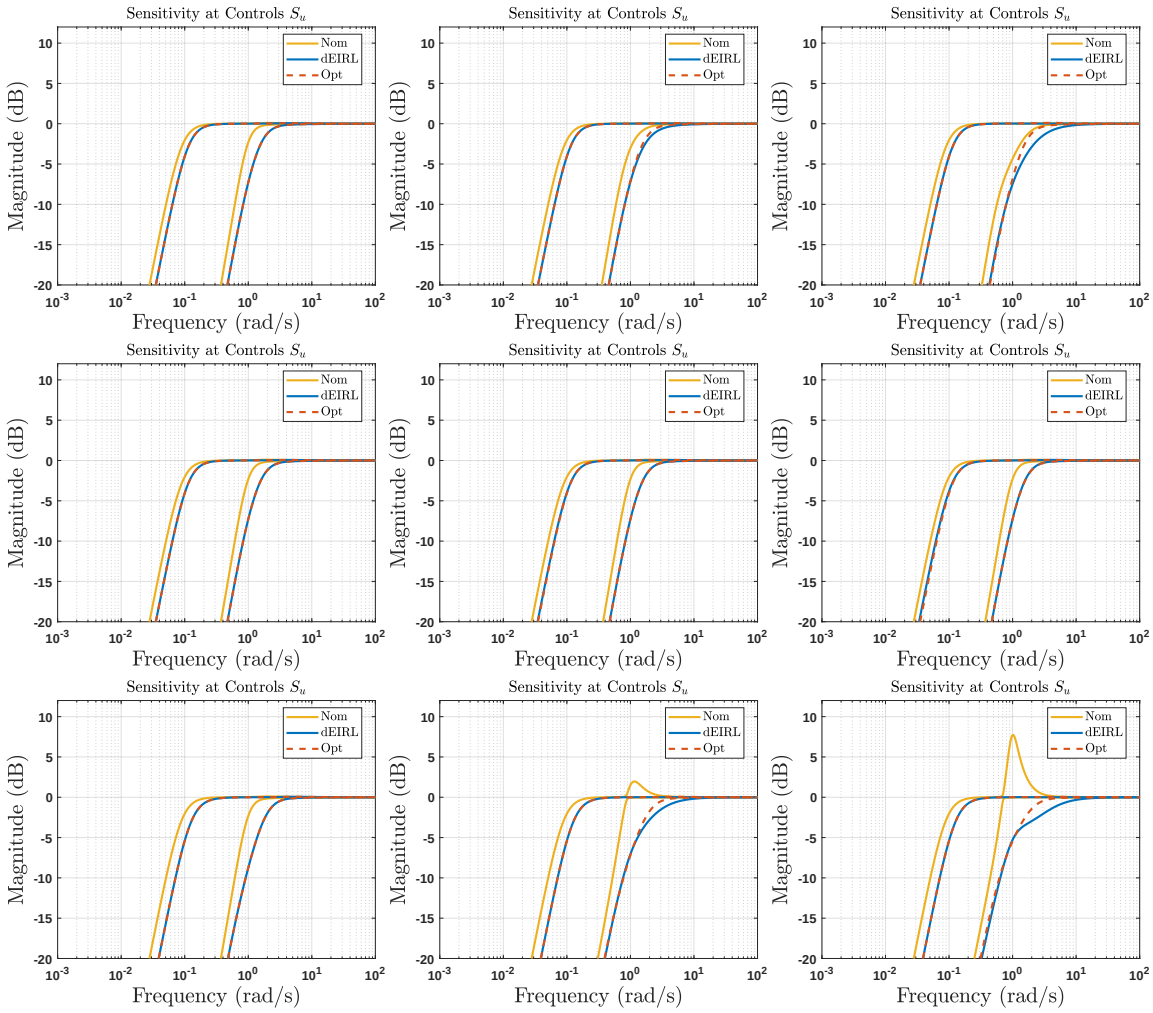


Figure 6.3: Sensitivity at Controls S_u Versus Modeling Error ν. First Row: Lift Coefficient Sweep ν_L. Second Row: Drag Coefficient Sweep ν_D. Third Row: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$. First Column: 0% Modeling Error. Second Column: 10% Modeling Error. Third Column: 25% Modeling Error.

6.2.4 Plots: Complementary Sensitivity at Controls T_u

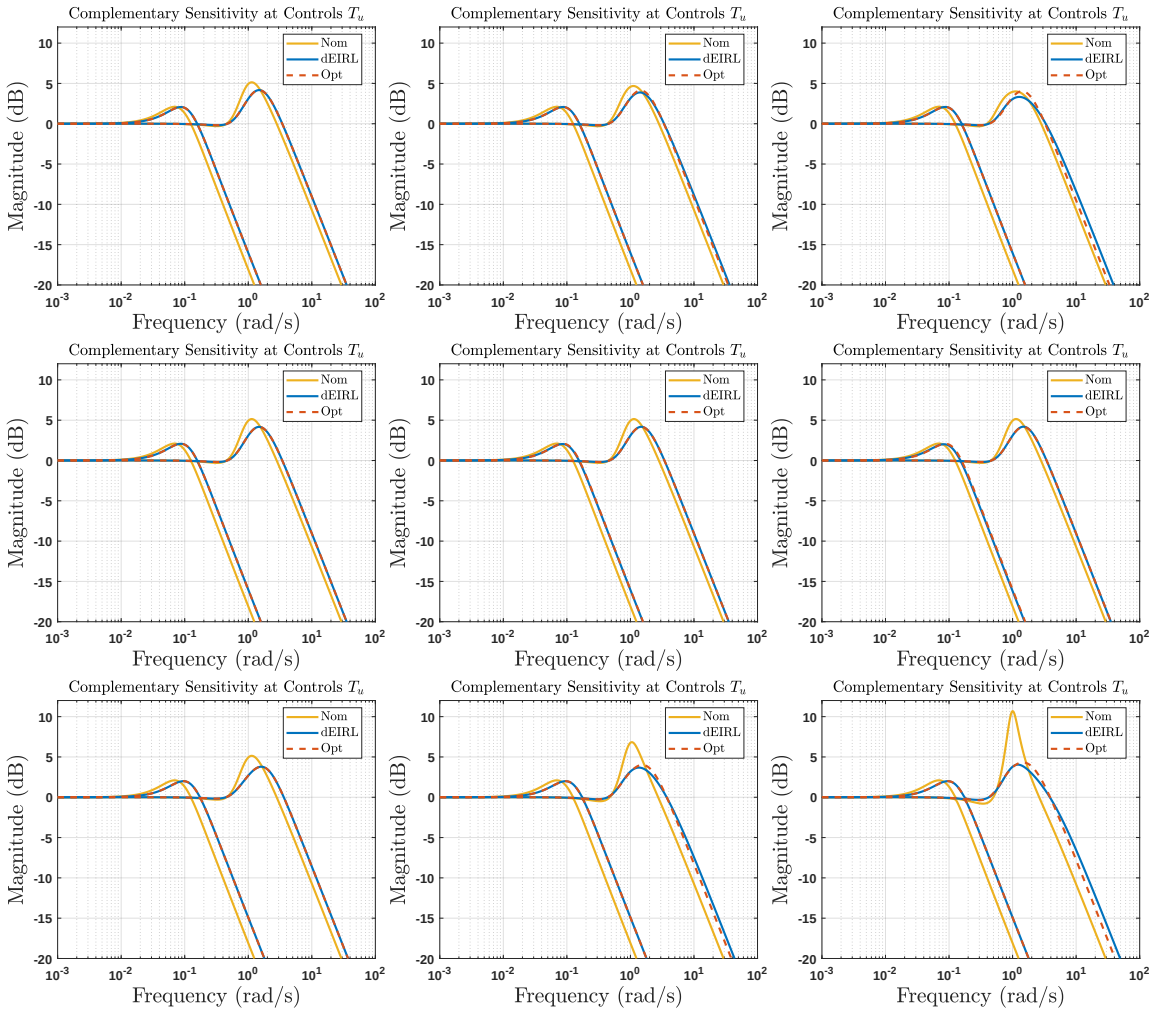


Figure 6.4: Complementary Sensitivity at Controls T_u Versus Modeling Error ν. First Row: Lift Coefficient Sweep ν_L. Second Row: Drag Coefficient Sweep ν_D. Third Row: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$. First Column: 0% Modeling Error. Second Column: 10% Modeling Error. Third Column: 25% Modeling Error.

6.3 Closed-Loop Step Response Performance Generalization to Modeling Error

In this study, we examine how closed-loop step response characteristics for the tested methods generalize with respect to increasing modeling error ν. Specifically, Table 6.3 displays the 1% settling time $t_{s,y_j,1\%}$, 90% rise time $t_{r,y_j,90\%}$, and percent overshoot M_{p,y_j} when issuing a step reference command in velocity $j = 1$ ($y_1 = V$) and FPA $j = 2$ ($y_2 = \gamma$) for the nominal LQR controller, dEIRL, the optimal LQR controller, and FBL [5, 6] (cf. Section 6.1.3). These step responses are issued at 0%, 10%, and 25% modeling errors in lift coefficient ν_L (B.4), drag coefficient ν_D (B.6), and pitch moment coefficient $\nu_{\mathcal{M}}$ (B.8). The system output and control responses to a step velocity command V are plotted for varying modeling errors in Figures 6.5–6.8 corresponding to the velocity $j = 1$ data in Table 6.3. The system output and control responses to a step FPA command γ are plotted for varying modeling errors in Figures 6.9–6.12 corresponding to the FPA $j = 2$ data in Table 6.3.

Table 6.3: Step Response Performance Metrics Versus Modeling Error ν

j	ν	$t_{s,y_j,1\%}$				$t_{r,y_j,90\%}$				M_{p,y_j}			
		Nom	dEIRL	Opt	FBL	Nom	dEIRL	Opt	FBL	Nom	dEIRL	Opt	FBL
1	0%	72.87	59.79	59.79	47.76	32.18	24.47	24.47	31.83	2.92	4.00	4.00	0.97
	10% L	73.06	59.68	60.16	47.78	32.72	24.43	24.49	31.77	2.94	4.00	4.00	0.97
	10% D	73.37	61.26	59.86	49.01	32.82	24.42	24.54	32.48	2.97	4.16	4.00	0.88
	10% $\mathcal{M}$	72.72	59.68	60.11	48.28	32.32	24.19	24.71	31.99	2.92	4.00	4.00	0.94
	25% L	73.50	59.64	60.20	47.83	32.84	24.19	24.65	31.84	2.99	4.00	4.01	0.96
	25% D	74.44	63.27	60.77	50.72	33.46	25.03	25.21	33.36	3.06	4.50	4.02	0.76
	25% $\mathcal{M}$	72.55	59.59	59.59	48.83	31.87	24.33	24.67	32.43	2.93	4.00	4.00	0.89
2	0%	9.55	9.81	9.81	9.23	5.35	5.38	5.38	4.62	3.43	1.93	1.93	3.74
	10% L	15.11	10.82	10.85	10.52	5.51	5.33	5.27	4.78	6.56	3.53	3.06	6.57
	10% D	9.55	9.82	9.815	9.30	5.36	5.39	5.38	4.61	3.42	1.89	1.93	3.74
	10% $\mathcal{M}$	12.02	9.99	10.16	7.39	4.94	5.13	5.23	4.46	1.91	4.15	2.32	2.56
	25% L	19.81	11.85	12.17	15.71	5.71	5.37	5.36	5.06	11.92	7.00	5.10	11.32
	25% D	9.56	9.84	9.83	9.25	5.36	5.40	5.39	4.65	3.40	1.84	1.92	3.75
	25% $\mathcal{M}$	16.75	10.03	10.93	9.95	4.73	5.22	5.30	4.19	0.95	4.51	2.97	2.11

Step Velocity Command. Overall, the velocity closed-loop step response performance fares well with respect to varying modeling errors. All methods maintain a 1% settling time in velocity $t_{s,V,1\%}$ of less than 75 s and a 90% rise time in velocity

$t_{r,V,90\%}$ of less than 35 s regardless of the modeling error type and severity. Percent overshoot also remains low at less than 5% for all methods, the lowest being FBL at $\approx 1\%$, followed by the nominal at $\approx 3\%$, then dEIRL and the optimal at $\approx 4\%$. Crucially, dEIRL recovers the closed-loop velocity command following properties of the optimal controller. Regardless of the modeling error introduced, dEIRL's 1% rise time remains within 2.50 s of the optimal (a 4.1% change), 90% settling time within 0.52 s of the optimal (a 2.0% change), and percent overshoot within 0.48% of the optimal (an 11.9% change).

Deviations in FPA due to step velocity commands are minimal for all methods, remaining less than 0.04° at max (Figure 6.6), and peak elevator deflection deviation δ_E from trim remains less than 1° (Figure 6.8). dEIRL, the optimal controller, and FBL all use similar throttle control effort δ_T, whose peaks reach on the order of 0.35–0.4 depending on the modeling error and remain within ± 0.02 of each other between the three methods (Figure 6.7). The nominal LQ design uses less control effort, peaking between 0.31–0.36. This comes at the cost of increased settling time (≈ 73 s for the nominal design versus ≈ 60 s for dEIRL and the optimal and ≈ 50 s for FBL), thus giving rise to a tradeoff between settling time and control effort. However, all methods remain within the 75 s velocity settling time Specification B.3.1.

Step FPA Command. Comparatively speaking, closed-loop performance degradation is more pronounced in the FPA response, dEIRL and the optimal exhibiting a performance edge over the nominal and FBL. Nominally, all methods achieve the original performance Specifications B.3.1 of a 1% FPA settling time $t_{s,\gamma,1\%} \leq 10$ s and percent overshoot $M_{p,\gamma} \leq 5\%$. The 90% FPA rise time $t_{r,\gamma,90\%}$ is also low at less than 5.5 s for all methods. Intuitively, the closed-loop FPA performance degrades less for modeling errors in the drag coefficient (which primarily affects the velocity dynamics); however, lift and pitching moment coefficient errors significantly impact performance

(Figure 6.9). For instance, from 0% to 25% lift coefficient modeling error, the 1% settling time $t_{s,\gamma,1\%}$ increases to 19.81 s (a +75% change) for the nominal LQR and 15.71 s (+70%) for FBL, taking these methods well out of the 10 s design specification. Meanwhile, degradation for dEIRL and the optimal LQR is less pronounced at 11.85 s (+21%) and 12.17 s (+24%), respectively. From this same 0% to 25% lift coefficient modeling error, percent overshoot in FPA $M_{p,\gamma}$ increases to 11.92% for the nominal LQR and 11.32% for FBL. Meanwhile, dEIRL increases to only 7.00%, and the optimal LQR to 5.10%.

Elevator control effort to a step FPA command is comparable among all methods, typically remaining within ±2 deg (Figure 6.12). For the nominal system, FBL exhibits virtually-zero deviations in velocity in its response to a step FPA command; meanwhile, the nominal LQR, dEIRL, and optimal LQR controllers all feature a velocity dip transient of 25–30 ft/s in their responses (Figure 6.10). The near-zero velocity deviations achieved by FBL are a direct result of its decoupling inversion of the system dynamics [5, 6], which guarantees that the output in the velocity channel remains unaffected by commands issued in the FPA channel. However, when modeling error is introduced, the FBL controller no longer achieves exact dynamical inversion, resulting in velocity dips of up to 15 ft/s in amplitude (Figure 6.10). Furthermore, this decoupling inversion of the velocity dynamics requires large control effort in the throttle channel δ_T (Figure 6.11), a phenomenon well-understood of FBL generally [14] and observed on this HSV model previously [5, 6]. Peak throttle setting for the nominal, dEIRL, and optimal controllers as a result of issuing a step FPA command is comparable at 0.35–0.4. Meanwhile, FBL's throttle peaks at 0.75 nominally, and by up to 1.05 when modeling error is introduced.

It should be noted that, when a severe 25% pitch moment coefficient modeling error $\nu_{\mathcal{M}}$ is introduced, percent overshoot of the nominal LQR (0.95%) and FBL

(2.11%) outperforms that of dEIRL (4.51%) and the optimal LQR (2.97%). However, Examination of Figure 6.9 (bottom right) shows the reason for the lower percent overshoot achieved by the nominal LQR and FBL: Both of these controllers exhibit an undesirable inverse FPA response occurring after the overshoot, resulting in FPA undershoot before the response settles. On the other hand, dEIRL and the optimal LQR do not exhibit such inverse behavior and maintain responses qualitatively similar to the nominal model response.

6.3.1 Plots: Step Velocity Command

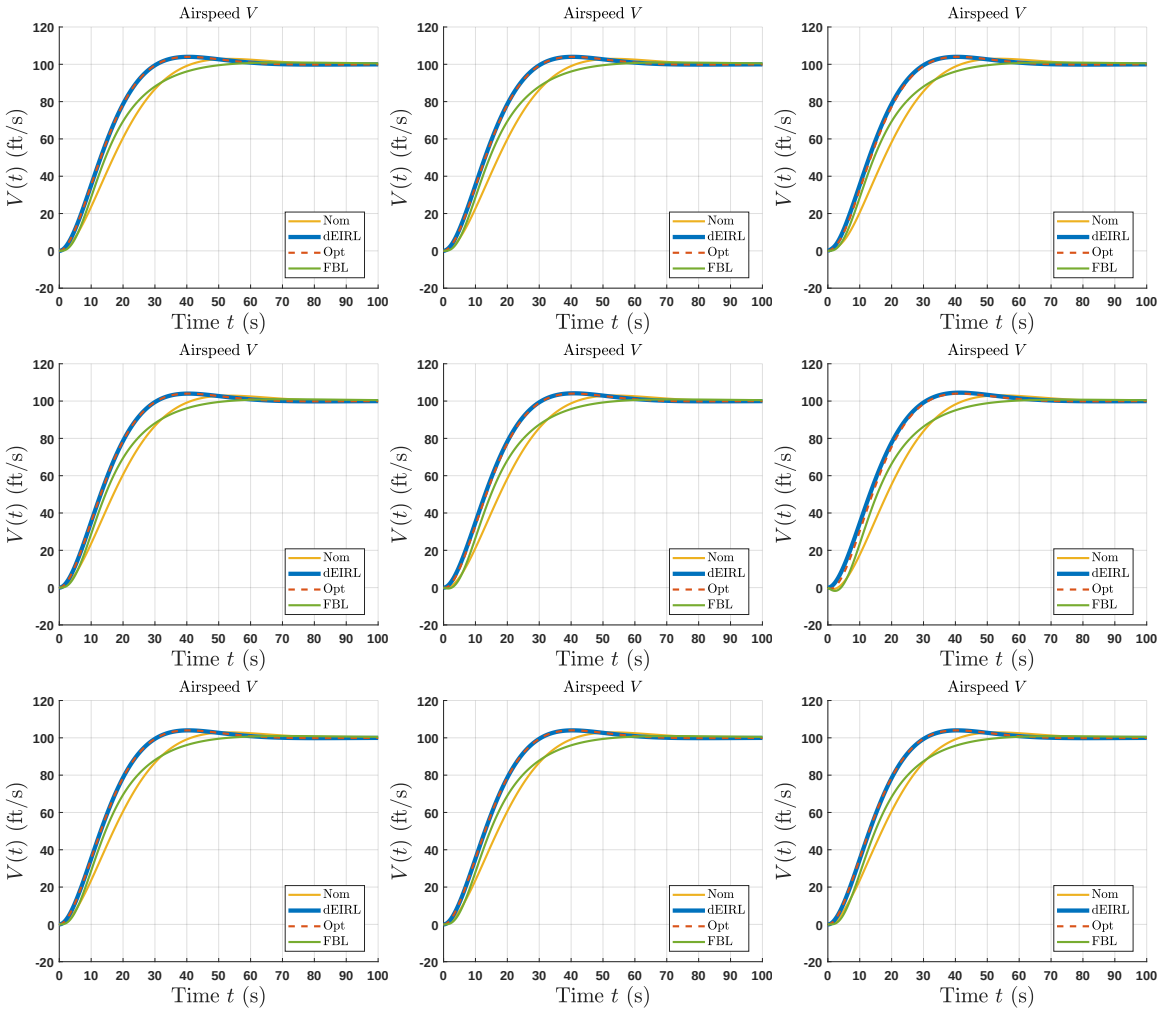


Figure 6.5: Velocity V Response to 100 ft/s Step-Velocity Command. First Row: Lift Coefficient Sweep ν_L. Second Row: Drag Coefficient Sweep ν_D. Third Row: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$. First Column: 0% Modeling Error. Second Column: 10% Modeling Error. Third Column: 25% Modeling Error.

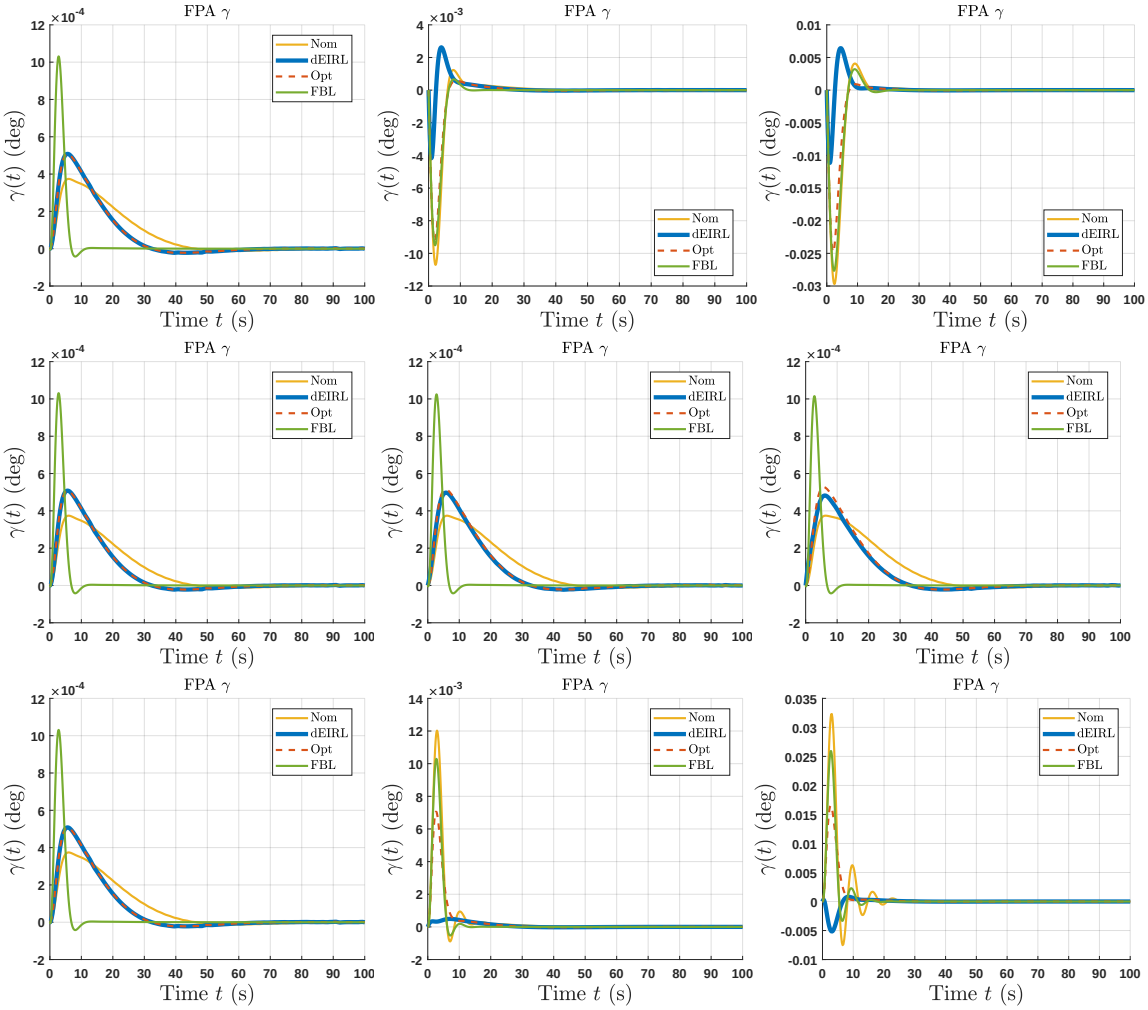


Figure 6.6: FPA γ Response to 100 ft/s Step-Velocity Command. First Row: Lift Coefficient Sweep ν_L. Second Row: Drag Coefficient Sweep ν_D. Third Row: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$. First Column: 0% Modeling Error. Second Column: 10% Modeling Error. Third Column: 25% Modeling Error.

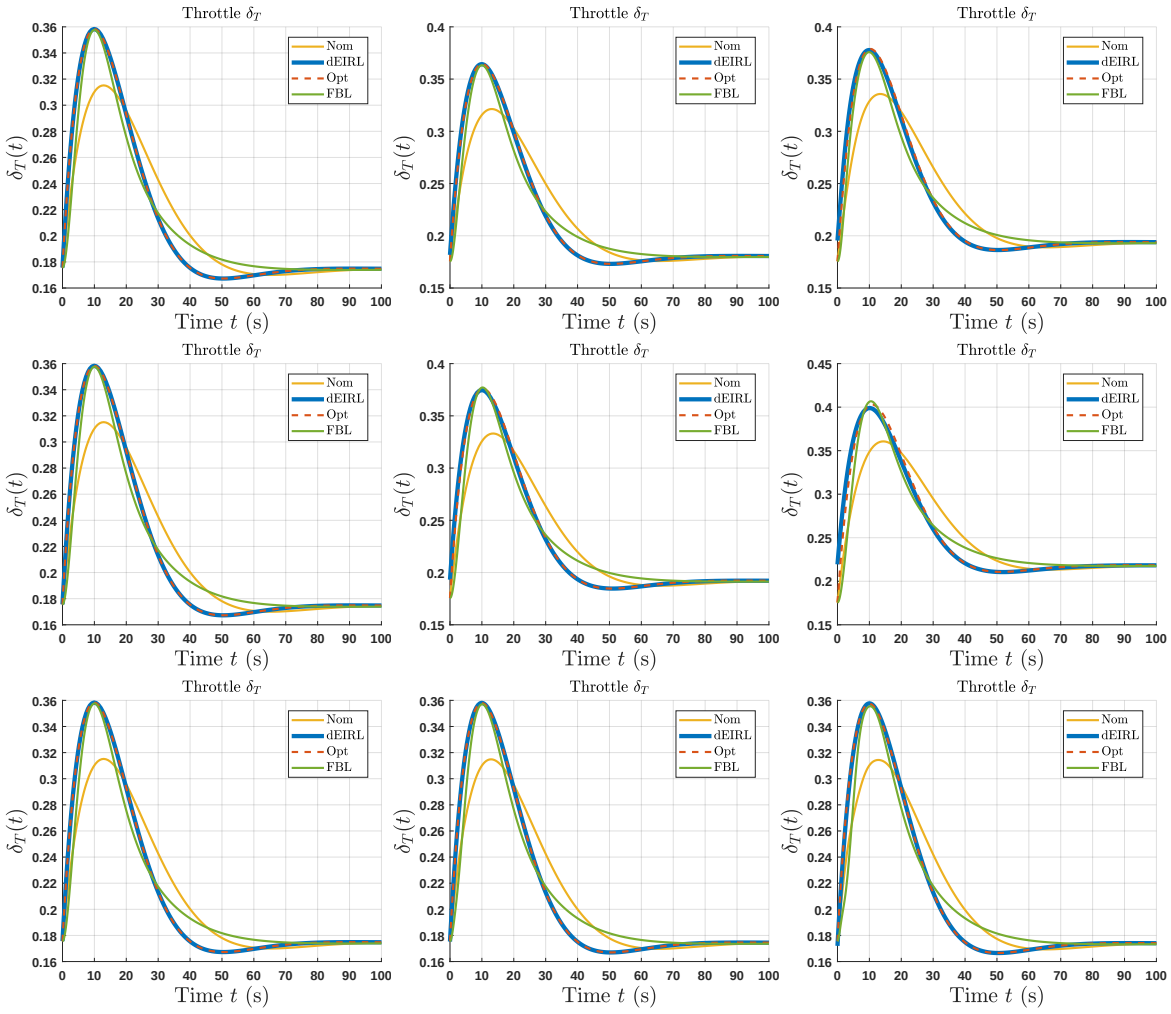


Figure 6.7: Throttle Setting δ_T Response to 100 ft/s Step-Velocity Command. First Row: Lift Coefficient Sweep ν_L. Second Row: Drag Coefficient Sweep ν_D. Third Row: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$. First Column: 0% Modeling Error. Second Column: 10% Modeling Error. Third Column: 25% Modeling Error.

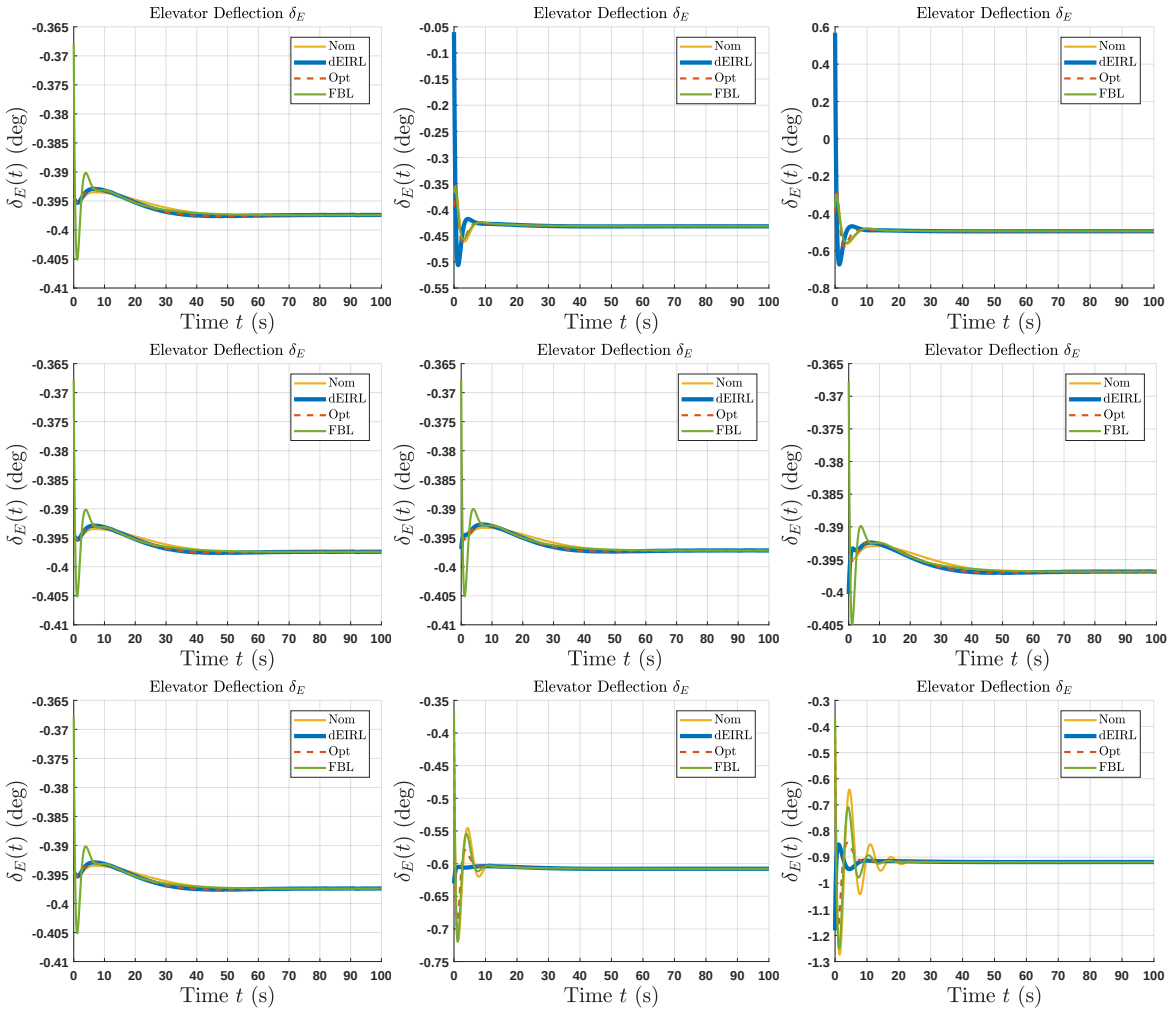


Figure 6.8: Elevator Setting δ_E Response to 100 ft/s Step-Velocity Command. First Row: Lift Coefficient Sweep ν_L. Second Row: Drag Coefficient Sweep ν_D. Third Row: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$. First Column: 0% Modeling Error. Second Column: 10% Modeling Error. Third Column: 25% Modeling Error.

6.3.2 Plots: Step FPA Command

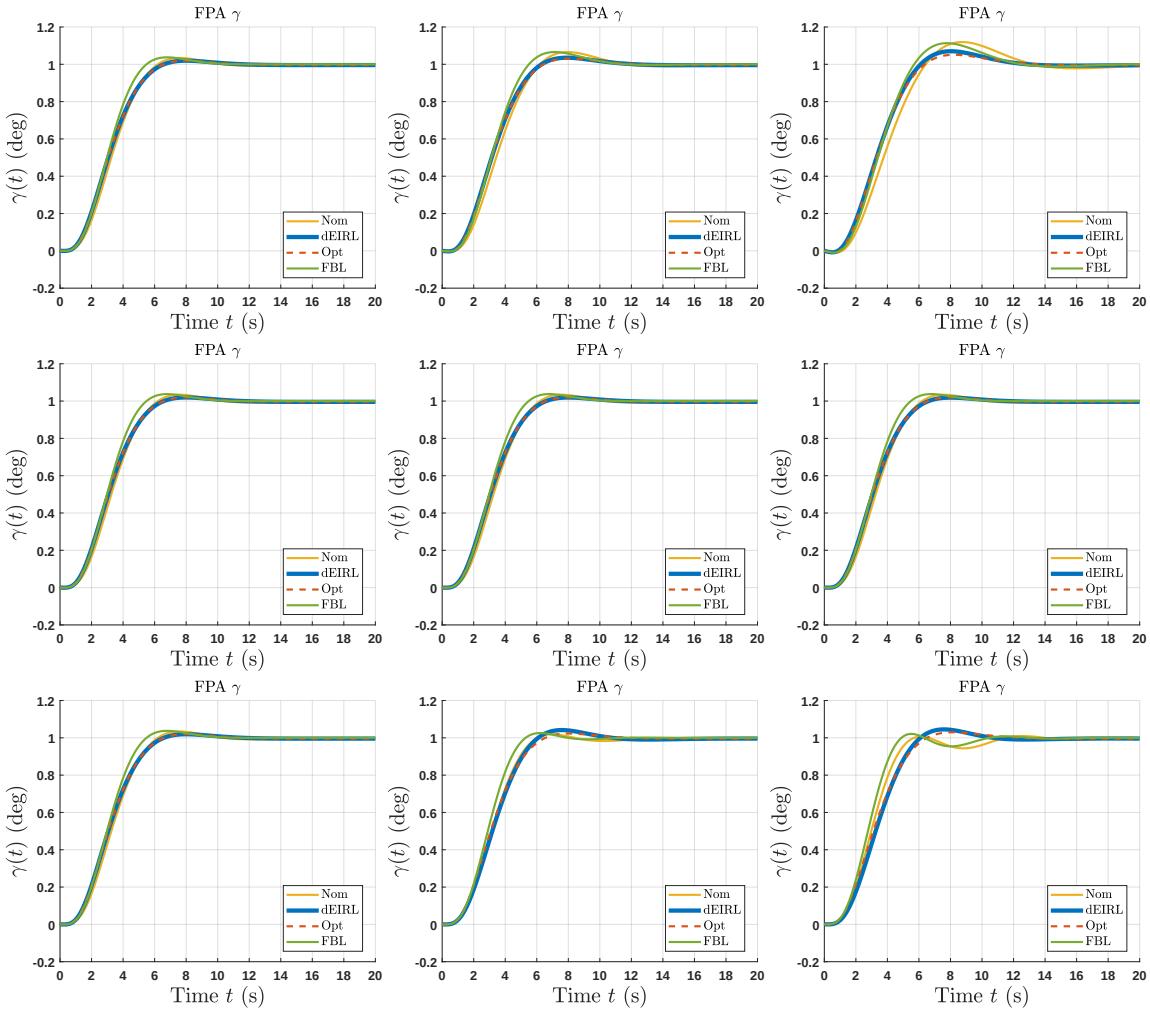


Figure 6.9: FPA γ Response to 1 deg Step-FPA Command. First Row: Lift Coefficient Sweep ν_L. Second Row: Drag Coefficient Sweep ν_D. Third Row: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$. First Column: 0% Modeling Error. Second Column: 10% Modeling Error. Third Column: 25% Modeling Error.

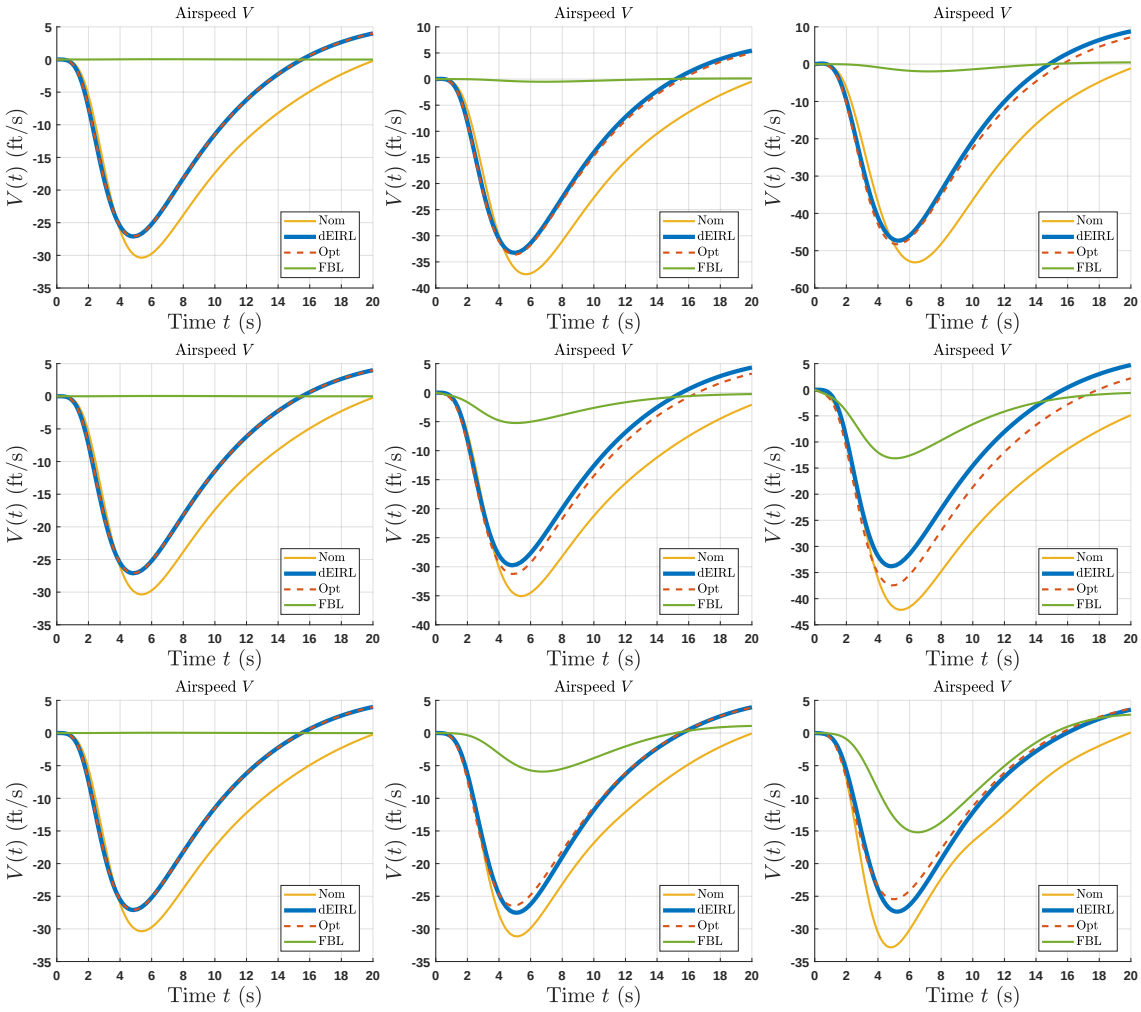


Figure 6.10: Velocity V Response to 1 deg Step-FPA Command. First Row: Lift Coefficient Sweep ν_L. Second Row: Drag Coefficient Sweep ν_D. Third Row: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$. First Column: 0% Modeling Error. Second Column: 10% Modeling Error. Third Column: 25% Modeling Error.

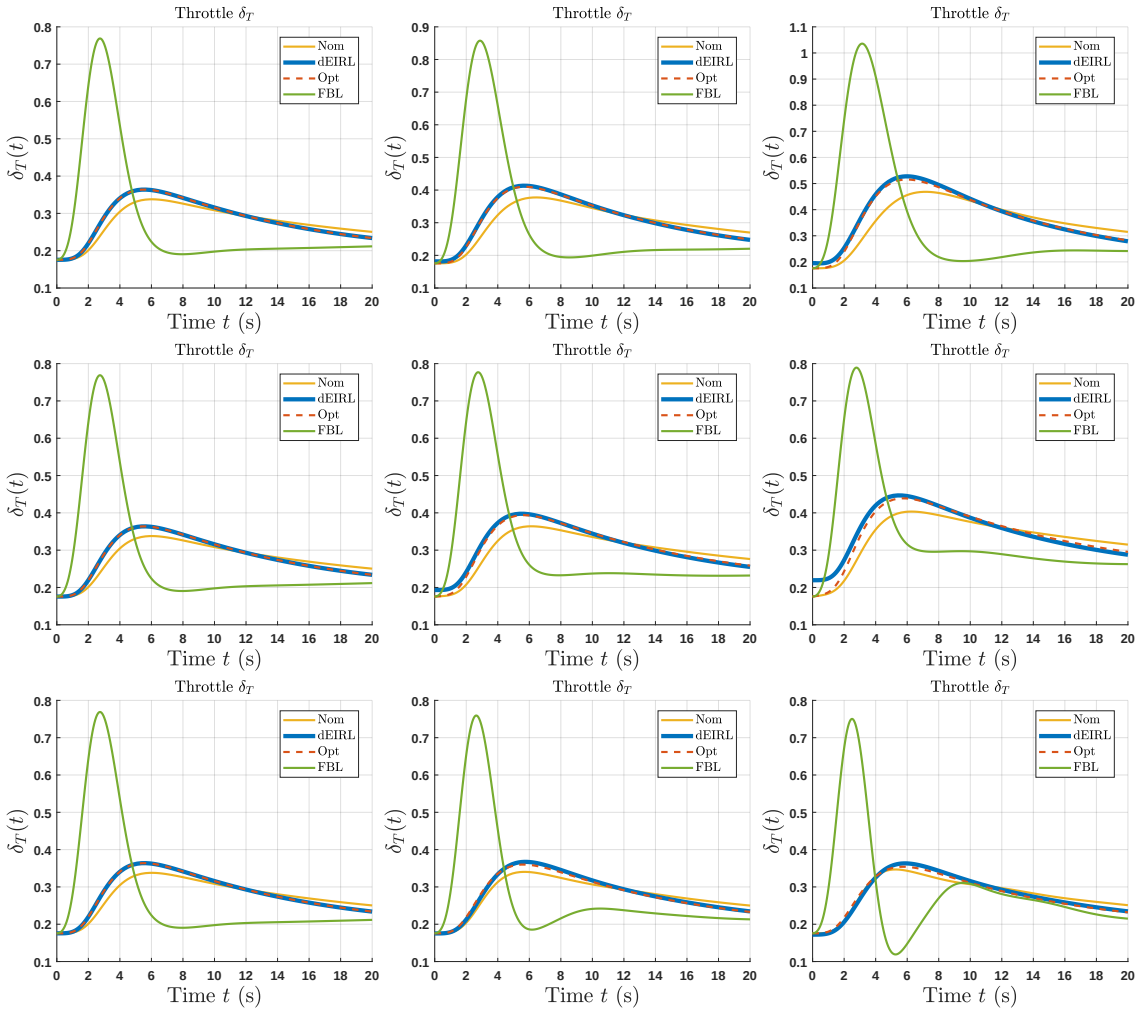


Figure 6.11: Throttle Setting δ_T Response to 1 deg Step-FPA Command. First Row: Lift Coefficient Sweep ν_L. Second Row: Drag Coefficient Sweep ν_D. Third Row: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$. First Column: 0% Modeling Error. Second Column: 10% Modeling Error. Third Column: 25% Modeling Error.

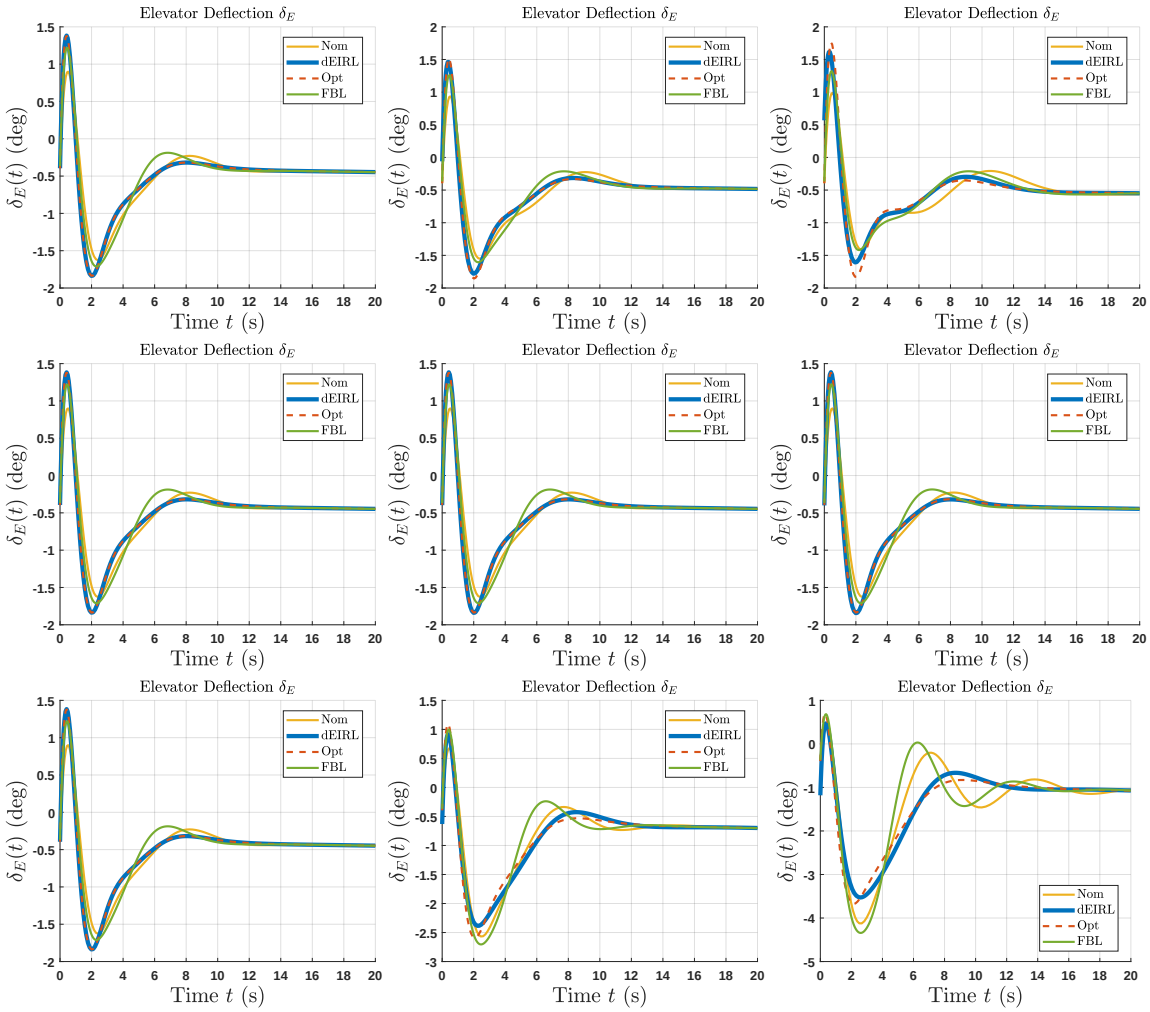


Figure 6.12: Elevator Setting δ_E Response to 1 deg Step-FPA Command. First Row: Lift Coefficient Sweep ν_L. Second Row: Drag Coefficient Sweep ν_D. Third Row: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$. First Column: 0% Modeling Error. Second Column: 10% Modeling Error. Third Column: 25% Modeling Error.

6.4 dEIRL Initial Condition Ablation Study

In this study, we run dEIRL for each initial condition over the IC grid $x_0 \in G_{x_0}$ (6.2), and at varying modeling errors 0%–25% in each of the modeling error grids G_{ν_L}, G_{ν_D}, and $G_{\nu_\mathcal{M}}$ (6.1). This section hence comprises a total of 2,511 independent learning trials. Table 6.4 displays the nominal controller optimality error $\left\|K_{0,j} - K_j^*\right\|$, dEIRL's optimality error $\left\|K_{i^*,j} - K_j^*\right\|$, and the percent reduction in optimality error from nominal $\rightarrow$ dEIRL (i.e., $i = 0 \rightarrow i^*$) in each loop $j = 1$ (velocity V) and $j = 2$ (FPA γ) for the IC sweep. Table 6.4 also includes dEIRL's iteration-wise max learning regression conditioning $\max_i \kappa(\mathbf{A}_{i,j})$, $j = 1, 2$. All performance measures include worst, average, and standard deviation data (each taken over the IC grid $x_0 \in G_{x_0}$). The controller optimality error and conditioning data presented in Table 6.4 is visually plotted in Figures 6.13 and 6.14, respectively.

Table 6.4: Initial Condition x_0 Ablation Optimality Error and Conditioning Data

ν		Over G_{x_0} (6.2)	$\|K_{i,1} - K_1^*\|$ Nom $i=0$	$\|K_{i,1} - K_1^*\|$ dEIRL $i=i^*$	$\|K_{i,1} - K_1^*\|$ % Reduc $i=0 \rightarrow i^*$	$\|K_{i,2} - K_2^*\|$ Nom $i=0$	$\|K_{i,2} - K_2^*\|$ dEIRL $i=i^*$	$\|K_{i,2} - K_2^*\|$ % Reduc $i=0 \rightarrow i^*$	$\max_i \kappa(\mathbf{A}_{i,1})$ dEIRL	$\max_i \kappa(\mathbf{A}_{i,2})$ dEIRL
0%		worst		1.01e-09	99.99		9.30e-06	99.99	460.01	293.50
		avg	1.23	5.64e-10	99.99	11.59	6.21e-06	99.99	170.35	288.02
		std		2.51e-10	2.04e-08		9.39e-07	8.10e-06	89.01	3.32
10%	L	worst		2.24e-05	99.99		0.09	99.29	457.42	259.64
		avg	1.23	2.24e-05	99.99	12.46	0.08	99.36	168.24	248.90
		std		1.63e-10	1.32e-08		0.01	0.04	86.28	5.27
	D	worst		0.46	62.75		0.01	99.90	491.95	294.09
		avg	1.23	0.23	81.05	11.58	0.01	99.91	174.51	288.56
		std		0.10	8.14		4.00e-04	3.48e-03	95.61	3.32
	$\mathcal{M}$	worst		1.11e-06	99.99		0.84	92.89	473.65	265.48
		avg	1.23	1.11e-06	99.99	11.81	0.75	93.63	171.16	260.18
		std		4.31e-10	3.50e-08		0.05	0.42	90.58	3.08
25%	L	worst		7.32e-05	99.99		0.38	97.31	456.93	268.01
		avg	1.23	7.32e-05	99.99	14.08	0.28	97.98	164.94	255.64
		std		2.77e-10	2.25e-08		0.06	0.41	82.39	6.62
	D	worst		1.15	6.84		0.029	99.74	545.80	294.99
		avg	1.23	0.57	54.00	11.57	0.027	99.77	181.20	289.41
		std		0.25	20.25		1.01e-03	0.01	107.31	3.33
	$\mathcal{M}$	worst		1.34e-05	99.99		1.52	87.58	538.49	728.94
		avg	1.23	1.34e-05	99.99	12.24	0.64	94.74	178.69	702.17
		std		1.59e-09	1.29e-07		0.37	3.02	103.61	8.68

Solution Optimality Generalization. Table 6.4 and Figure 6.4 show that, regard-

less of the modeling error type tested (in lift ν_L, drag ν_D, or pitching moment $\nu_{\mathcal{M}}$), and regardless of the severity of the modeling error (0%–25%), dEIRL successfully recovers optimality of the controller in each loop $j = 1, 2$ for all initial conditions tested in the grid $x_0 \in G_{x_0}$; i.e., dEIRL achieves small optimality error $\left\|K_{i^*,j} - K_j^*\right\|$. Indeed, regardless of the IC, modeling error type, and modeling error value tested, dEIRL's controller optimality error $\left\|K_{i^*,j} - K_j^*\right\|$ remains within 1.52 in both loops $j = 1, 2$. It is intuitive that the worst-case of 1.52 occurs in the higher-dimensional, unstable, nonminimum phase FPA loop $j = 2$ at the most severe 25% pitch moment coefficient $\nu_{\mathcal{M}}$ modeling error tested. By contrast, the nominal LQR controller's respective optimality error is $\|K_{0,2} - K_2^*\| = 12.24$, almost a factor of ten larger.

Importantly, dEIRL achieves significant percent reductions in controller optimality error relative to the nominal LQR design, even for severe modeling errors. For example, at 25% modeling error in the more dynamically-challenging FPA loop $j = 2$, dEIRL achieves a worst-case percent reduction from nominal to dEIRL over the IC grid $x_0 \in G_{x_0}$ of 97.31% for lift coefficient modeling error ν_L, 99.74% for drag ν_D, and 87.58% for pitch moment $\nu_{\mathcal{M}}$. Thus, dEIRL exhibits excellent learning generalization with respect to varying system initial conditions x_0, even in the face of severe model uncertainty. As a result, for recovery of controller optimality, a designer is at least a factor of ten times better off from running dEIRL than opting for a nominal classical LQR design.

The exception observed to this rule is in examining drag coefficient modeling error ν_D in the velocity loop $j = 1$; intuitively, drag modeling error is observed to have the greatest effect on dEIRL's performance in the velocity loop of the types tested (Figure 6.13, top middle). At 10% drag coefficient modeling error, dEIRL reduces optimality error by 62.75% relative to the nominal at worst-case, 81.05% on average. At 25% drag coefficient modeling error, dEIRL reduces optimality error by only 6.84% at worst-

case. Even so, dEIRL achieves an average reduction of 54% for this modeling error (a factor of two reduction), still a marked improvement in closed-loop performance relative to the nominal classical design.

Algorithm Conditioning Generalization. dEIRL's conditioning remains highly consistent with respect to varying system initial conditions $x_0 \in G_{x_0}$, demonstrating good IC learning generalization. In the velocity loop $j = 1$, conditioning maxes on the order of 460-470 at worst-case over the IC grid for all modeling error types ν and averages on the order of 170-180. Meanwhile, in the higher-dimensional FPA loop $j = 2$, conditioning remains relatively unchanged for varying initial conditions $x_0 \in G_{x_0}$ when lift ν_L and drag ν_D coefficient modeling errors are introduced, maxing in the range 260-300 and averaging in the range 240-290 regardless of the modeling error severity. Meanwhile, conditioning degradation in this loop $j = 2$ is more pronounced with respect to pitching moment coefficient modeling error $\nu_{\mathcal{M}}$, the worst-case over the IC grid increasing from 293.50 nominally to 728.94 at 25% modeling error. However, conditioning on this order ($< 10^3$) is a significant improvement from existing ADP-based CT-RL control algorithms, which our studies of Section 4 exhibit conditioning on the order of 10^{16} for this HSV system (cf. Table 4.2) and in our studies of Section 2.6 exhibit conditioning on the order of 10^{11} for academic second-order single-input examples (cf. Table 2.7). Lastly, even though the conditioning degradation is more pronounced in the FPA loop $j = 2$, this loop exhibits the lowest numerical sensitivity with respect to varying initial conditions $x_0 \in G_{x_0}$, as IC standard deviations for conditioning in this loop remain less than 10 regardless of the modeling error tested.

6.4.1 Plots: dEIRL Controller Optimality Error versus Initial Condition x_0

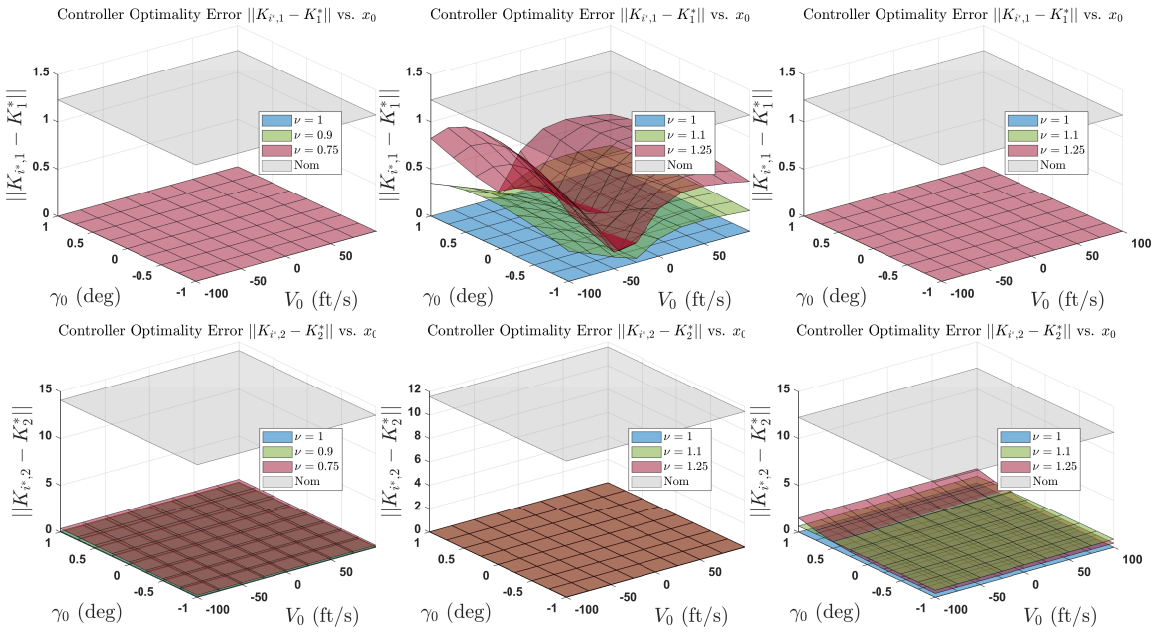


Figure 6.13: dEIRL Controller Optimality Error $\left\|K_{i^*,j} - K_j^*\right\|$ Versus Initial Condition x_0 for 0%, 10%, and 25% Modeling Errors ν. First Row: Velocity Loop $j = 1$. Second Row: FPA Loop $j = 2$. Gray: Nominal Classical Design. First Column: Lift Coefficient Sweep ν_L. Second Column: Drag Coefficient Sweep ν_D. Third Column: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$.

6.4.2 Plots: dEIRL Worst-Case Algorithm Condition Number versus Initial Condition x_0

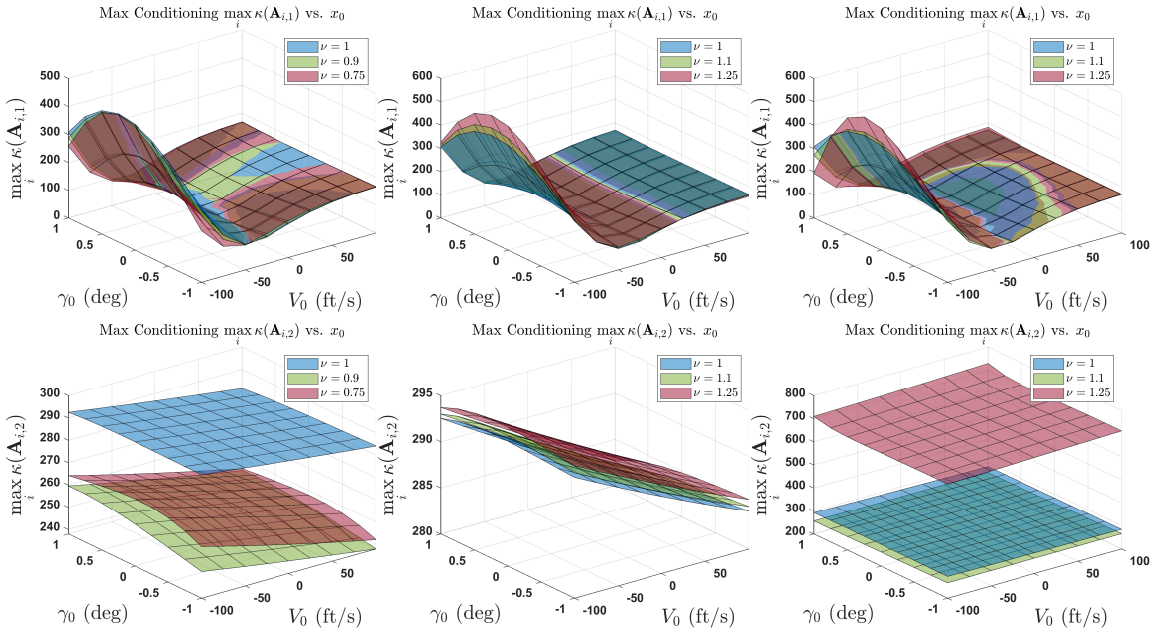


Figure 6.14: dEIRL Iteration-Wise Max Algorithm Condition Number $\max_i \kappa(\mathbf{A}_{i,j})$ Versus Initial Condition x_0 for 0%, 10%, and 25% Modeling Errors ν. First Row: Velocity Loop $j = 1$. Second Row: FPA Loop $j = 2$. First Column: Lift Coefficient Sweep ν_L. Second Column: Drag Coefficient Sweep ν_D. Third Column: Pitch Moment Coefficient Sweep $\nu_{\mathcal{M}}$.

6.5 dEIRL Modeling Error Ablation Study

In this study, we run dEIRL for simultaneous modeling errors ranging from 0%–25% in lift/drag over the grid $G_\nu = G_{\nu_L} \times G_{\nu_D}$ (6.1), lift/pitch moment $G_\nu = G_{\nu_L} \times G_{\nu_\mathcal{M}}$, and drag/pitch moment $G_\nu = G_{\nu_D} \times G_{\nu_\mathcal{M}}$ when initialized at trim ICs $x_0 = x_e$. This section hence comprises a total of 361 independent learning trials. Table 6.5 displays the nominal controller optimality error $\left\|K_{0,j} - K_j^*\right\|$, dEIRL's optimality error $\left\|K_{i^*,j} - K_j^*\right\|$, and the percent reduction in optimality error from nominal $\rightarrow$ dEIRL in each loop $j = 1$ (velocity V), $j = 2$ (FPA γ), as well as dEIRL's iteration-wise max learning regression conditioning $\max_i \kappa(\mathbf{A}_{i,j})$, $j = 1, 2$. All performance measures include worst, average, and standard deviation data (each taken over the respective 0%–25% modeling error grids tested $\nu \in G_\nu$). The controller optimality error and conditioning data presented in Table 6.4 is visually plotted in Figures 6.13 and 6.14, respectively.

Table 6.5: Modeling Error ν Ablation Optimality Error and Conditioning Data

ν	Over	$\|K_{i,1} - K_1^*\|$			$\|K_{i,2} - K_2^*\|$			$\max_i \kappa(\mathbf{A}_{i,1})$	$\max_i \kappa(\mathbf{A}_{i,2})$
	G_ν (6.1)	Nom $i = 0$	dEIRL $i = i^*$	% Reduc $i = 0 \rightarrow i^*$	Nom $i = 0$	dEIRL $i = i^*$	% Reduc $i = 0 \rightarrow i^*$	dEIRL	dEIRL
L/D	worst	1.23	0.14	88.29	14.08	1.11	92.09	101.25	289.66
	avg	1.23	0.05	95.63	12.75	0.123	99.05	98.29	258.39
	std	3.27e-04	0.04	3.49	0.79	0.15	1.09	1.81	13.81
$L/\mathcal{M}$	worst	1.23	7.32e-05	99.99	14.96	3.94	73.67	95.48	698.40
	avg	1.23	2.03e-05	99.99	13.11	1.00	92.42	93.78	231.06
	std	2.90e-04	1.85e-05	1.50e-03	0.86	0.68	4.90	1.48	101.66
$D/\mathcal{M}$	worst	1.23	0.12	90.17	12.25	1.42	88.26	100.85	793.94
	avg	1.23	0.04	96.90	11.89	0.61	94.92	96.20	365.64
	std	9.33e-05	0.03	2.52	0.21	0.41	3.41	2.64	148.36

Solution Optimality Generalization. dEIRL's learning generalizes robustly with respect to severe and simultaneous modeling errors, achieving a percent reduction in controller optimality error relative to the nominal LQR design of at least 88.29% in the velocity loop $j = 1$ and at least 73.67% in the FPA loop $j = 2$ regardless of the modeling error type and severity. For simultaneous lift/drag modeling errors,

optimality error from $\left\|K_{0,j} - K_j^*\right\| \rightarrow \left\|K_{i^*,j} - K_j^*\right\|$ (i.e., from nominal $\rightarrow$ dEIRL) averages $1.23 \rightarrow 0.05$ (95.63% reduction) in the velocity loop $j = 1$, and $12.75 \rightarrow 0.123$ (99.05% reduction) in the FPA loop $j = 2$. Similar average reductions are observed for the simultaneous lift/pitch moment and drag/pitch moment modeling error ablations. Meanwhile, the worst-case (i.e., smallest) reduction in optimality error across the board occurs in the higher-dimensional, unstable, nonmimimum phase FPA loop $j = 2$ for simultaneous lift/pitch moment modeling error, at 73.67%. This still represents a significant reduction by a factor of 3/4. Furthermore, the reduction averages 92.42% for this modeling error ablation with a standard deviation of only 4.90%, so the worst-case 73.67% is an outlier.

Algorithm Conditioning Generalization. Conditioning performance in the velocity loop $j = 1$ exhibits little variation with respect to modeling error, varying from 95–101 at worst case with standard deviation 2.64 or less for all ablations. Conditioning in the FPA loop $j = 2$ is more volatile, which given the higher regression dimensionality and dynamical features is to be expected. For the lift/drag ablation, conditioning remains low at a max of 289.66. Meanwhile, conditioning degradation is more pronounced for both of the ablations involving the pitch moment coefficient; i.e., the lift/pitch moment and drag/pitch moment sweeps. For the lift/pitch moment ablation, average conditioning remains low at 231.06; however, it reaches a worst-case of 698.40. Conditioning fares the worst for the drag/pitch moment ablation, averaging 365.64 and reaching 793.94 at max. However, relative to the existing ADP-based performance of existing ADP-based CT-RL algorithms (cf. Sections 4 and 2.6) conditioning less than 10^3 for this system with simultaneous modeling errors is a significant result.

.

6.5.1 Plots: dEIRL Controller Optimality Error versus Modeling Error ν

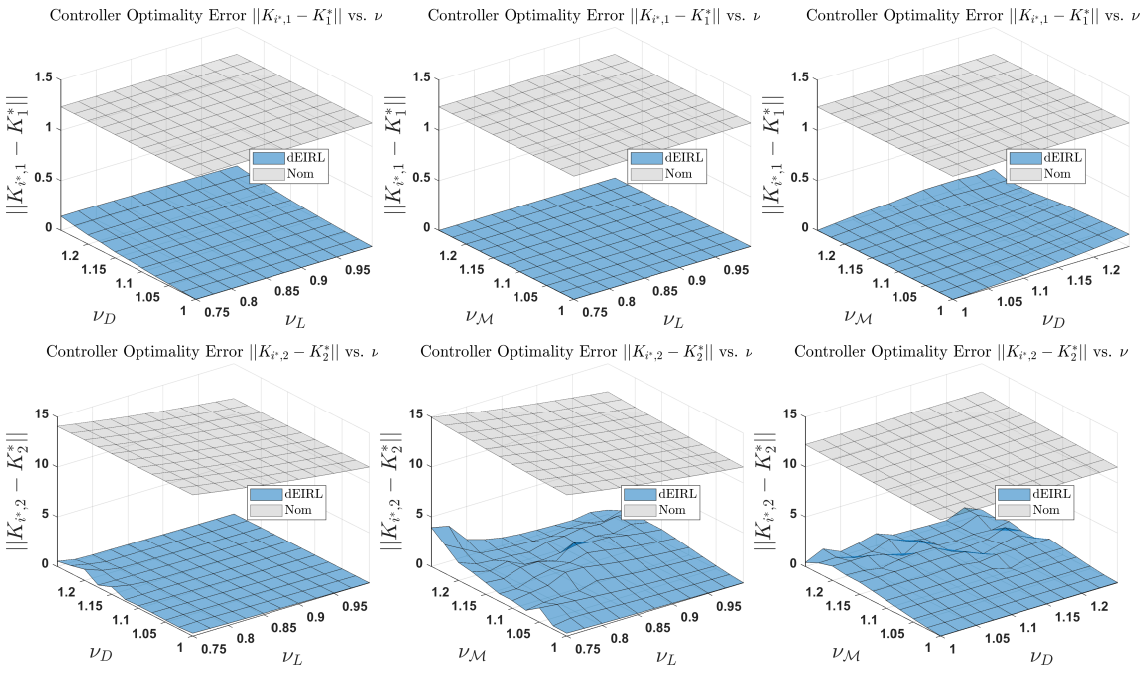


Figure 6.15: dEIRL Controller Optimality Error $\left\|K_{i^*,j} - K_j^*\right\|$ Versus Modeling Error ν of up to 25%. First row: Velocity Loop $j = 1$. Second Row: FPA Loop $j = 2$. Gray: Nominal Classical Design. First Column: Lift/Drag ν_L/ν_D Sweep. Second Column: Lift/Pitch Moment $\nu_L/\nu_{\mathcal{M}}$ Sweep. Third Column: Drag/Pitch Moment $\nu_D/\nu_{\mathcal{M}}$ Sweep.

6.5.2 *Plots: dEIRL Worst-Case Algorithm Condition Number versus Modeling Error ν*

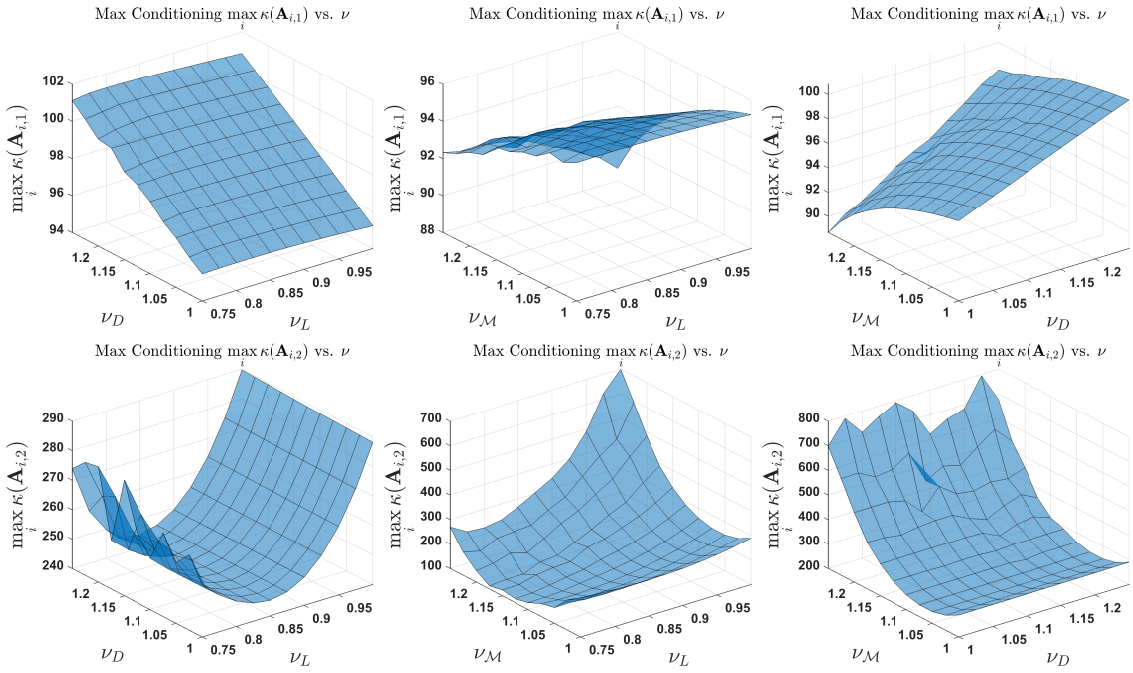


Figure 6.16: dEIRL Iteration-Wise Max Algorithm Condition Number $\max_i \kappa(\mathbf{A}_{i,j})$ Versus Modeling Error ν of up to 25%. First Row: Velocity Loop $j = 1$. Second Row: FPA Loop $j = 2$. First Column: Lift/Drag ν_L/ν_D Sweep. Second Column: Lift/Pitch Moment $\nu_L/\nu_{\mathcal{M}}$ Sweep. Third Column: Drag/Pitch Moment $\nu_D/\nu_{\mathcal{M}}$ Sweep.

6.6 Closed-Loop Performance Robustness with Respect to Random Modeling Error

In this study, we statistically examine how often the methods meet the 29 closed-loop step response performance metrics defined in Table 6.1 when random modeling error is introduced simultaneously in each parameter: lift coefficient ν_L (B.4), drag coefficient ν_D (B.6), and pitch moment coefficient $\nu_{\mathcal{M}}$ (B.8) (cf. Section 6.1 for complete numerical setup). We test 10,000 random trials of modeling error and assemble the failure percentages of each of the metrics in in Table 6.6 and Figure 6.17.

Table 6.6: Closed-Loop Performance Metrics Failure Percentage

Metric Number	Indicator Function	Failure Percentage (%)			
		Nom	dEIRL	Opt	FBL
1	I_S	0	0	0	0
2 (3)	$I_{V,t_{s,10\%}25}$ ($I_{V,t_{s,10\%}50}$)	100 (0)	0.3 (0)	2.7 (0)	100 (0)
4 (5)	$I_{V,t_{s,1\%}75}$ ($I_{V,t_{s,1\%}100}$)	0 (0)	0 (0)	0 (0)	17.3 (0)
6 (7)	$I_{V,t_{r,90\%}25}$ ($I_{V,t_{r,90\%}30}$)	100 (100)	0.3 (0)	2.7 (0)	100 (100)
8 (9)	I_{V,M_p5} (I_{V,M_p10})	0 (0)	0 (0)	0 (0)	0 (0)
10 (11)	I_{V,δ_T20} (I_{V,δ_T25})	0 (0)	0 (0)	4.9 (0)	5.9 (0)
12 (13)	$I_{V,\delta_E0.25}$ ($I_{V,\delta_E0.5}$)	19.0 (0)	40.0 (0.7)	16.6 (0)	17.4 (0)
14 (15)	$I_{V,\Delta\gamma0.01}$ ($I_{V,\Delta\gamma0.05}$)	27.7 (0)	0.4 (0)	13.5 (0)	21.0 (0)
16 (17)	$I_{\gamma,t_{s,10\%}7.5}$ ($I_{\gamma,t_{s,10\%}10}$)	0 (0)	0 (0)	0 (0)	0 (0)
18 (19)	$I_{\gamma,t_{s,1\%}10}$ ($I_{\gamma,t_{s,1\%}15}$)	73.4 (10.4)	42.6 (0)	44.7 (0)	30.0 (0)
20 (21)	$I_{\gamma,t_{r,90\%}5}$ ($I_{\gamma,t_{r,90\%}7.5}$)	95.0 (0)	99.9 (0)	100 (0)	0 (0)
22 (23)	I_{γ,M_p5} (I_{γ,M_p10})	25.3 (0)	3.4 (0)	0 (0)	28.4 (0)
24 (25)	I_{γ,δ_T25} (I_{γ,δ_T50})	0 (0)	0.2 (0)	2.3 (0)	100 (100)
26 (27)	I_{γ,δ_E5} ($I_{\gamma,\delta_E7.5}$)	0 (0)	0 (0)	0 (0)	0 (0)
28 (29)	$I_{\gamma,\Delta V0.15}$ ($I_{\gamma,\Delta V0.25}$)	52.9 (0)	22.5 (0)	25.5 (0)	0 (0)

Step Velocity Command. Firstly, all designs successfully stabilize the closed-loop system for the 10,000 random trials; i.e., each exhibits a failure rate of 0% in the stability metric I_S (metric 1). In comparison to the nominal LQR and FBL, dEIRL and the optimal LQR are 97% more likely to meet the tight 10% settling time (metric 2), while all designs achieve the less stringent 10% settling time (metric 3), and similar results hold for the 90% settling time (metrics 6, 7). Meanwhile, for the 1% velocity settling time (metrics 4, 5), all designs meet specification with the exception of FBL

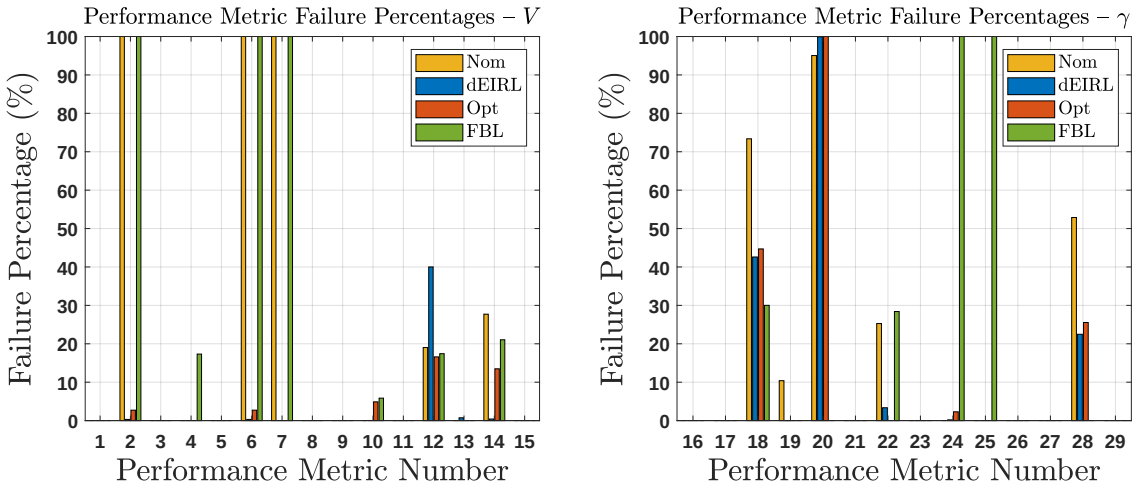


Figure 6.17: Closed-Loop Performance Metrics Failure Percentage (cf. Table 6.1 for Definitions).

at a 17% failure rate on the tighter metric 4. All designs meet the percent overshoot specifications (metrics 8, 9). For throttle control effort in metrics 10, 11, all methods meet the specifications except for failure rates in the optimal LQR and FBL of 4.9% and 5.9%, respectively. The area where dEIRL struggles the most is in the more stringent elevator control effort specification ($I_{V,\delta_E 0.25}$ metric 12, or a max 0.25 deg elevator deflection deviation), with a failure rate of 40%. By comparison, this is 21% higher than the nominal LQR (19%), 23.4% higher than the optimal LQR (16.6%), and 22.6% higher than FBL (17.4%). However, elevator deflections of 0.25 deg are small, and dEIRL meets the less stringent specification of 0.5 deg (metric 13) with only a 0.7% failure rate. Meanwhile, in FPA deviations as a result of issuing a step velocity command (metrics 14, 15), dEIRL has a 27% less likelihood of failure than the nominal LQR, 13% less than the optimal LQR, and 21% less than FBL.

Step FPA Command. All designs perform well in the 10% FPA settling time specifications (metrics 16, 17), each achieving a 0% failure rate. Meanwhile for the 1% settling time specifications (metrics 18, 19), dEIRL and the optimal LQR perform comparably in the stringent metric 18 ($I_{\gamma,t_{s,1\%}10}$), failing at similar percentages of

42.6% and 44.7%, respectively. Comparatively, dEIRL is 31% less likely to fail metric 18 than the nominal LQR (73.4%), and 13% more likely than FBL (30%). Similarly, FBL far outperforms the nominal LQR, dEIRL, and the optimal in the stringent 90% FPA rise time metric 20. However, as a consequence of fast rise/settling time, FBL exhibits the highest overshoot of the methods tested (metrics 22, 23) with a failure rate of 28.4% in metric 22, compared to dEIRL and optimal LQR failure rates of 3.4% and 0%, respectively. This points to a statistical tradeoff between meeting rise/settling time and overshoot specifications when modeling error is introduced.

Another distinct tradeoff emerges between deviations in velocity due to a step FPA command (metrics 28, 29) and the max throttle control exerted to mitigate the velocity deviation (metrics 24, 25). On one hand, FBL achieves superior velocity deviation performance, with a failure rate of 0% in the more stringent deviation metric 28. This is followed by dEIRL (22.5%), the optimal LQR (25.5%, similar to dEIRL), and the nominal LQR (52.9%, highest). This performance characteristic of FBL was observed in the step response trials of Section 6.3 (cf. Figure 6.10); fundamentally, they are a direct result of FBL's decoupling inversion of the system dynamics [5, 6]. However, FBL requires applying large throttle control δ_T in order to minimize the velocity dip transient caused by the FPA command (cf. Figure 6.11). As a result, it fails both throttle setting metrics 24, 25 at a rate of 100%. By comparison, the largest failure rate for these metrics between the nominal LQR, dEIRL, and the optimal LQR is only 2.3% (by the optimal LQR on metric 24). Intuitively, allowable velocity deviations and throttle control effort must be traded off for issued FPA commands.

6.7 Discussion

This study presents a newly-developed quantitative performance evaluation framework for RL algorithms in aerospace applications. These studies reveal that the

proposed dEIRL algorithm successfully recovers the optimal controller in the face of significant model uncertainty and initial conditions. dEIRL thus also recovers optimal closed-loop reference command following performance, results which hold statistically when the modeling error is randomly distributed. The proposed evaluation suite tests 37 learning/closed-loop design metrics on over 12,872 independent learning trials, a significant improvement from the tests performed in previous RL-based control works for HSVs (cf. Section 1). Our developed evaluation framework may be used for future RL control works, providing a comprehensive empirical valuation which is essential for real-world flight implementation.

Chapter 7

CONCLUSION, LIMITATION, AND DIRECTIONS OF FUTURE RESEARCH

The field of CT-RL owes great acknowledgment to the seminal works in ADP, which have achieved substantial theoretical results transformative to the development of learning control in the continuous-time setting. ADP's fundamental learning and approximation ideas have inspired a growing body of results in CT-RL, and the principles developed by these ADP works were instrumental in developing the methods presented in this dissertation. Nevertheless, in this work we develop a first-of-its-kind CT-RL algorithm performance diagnosis framework to quantitatively identify the key performance limitations facing ADP-based CT-RL algorithms. This framework's in-depth, designer-focused quantitative analyses reveals gaps between CT-RL theoretical promises and practical synthesis. These analyses show that CT-RL algorithms face fundamental performance limitations due dimensional scaling issues and a lack of persistence of excitation (PE) resulting in algorithm numerical issues. Furthermore, these CT-RL methods exhibit significant design complexity, which make them difficult for real-world designers to use.

In response to these algorithm learning issues, we develop a designer-centric approach to CT-RL via our suite of excitable integral reinforcement learning (EIRL) algorithms. We propose a three-prong numerical approach of 1) decentralization to break down the optimal control problem into lower-dimensional subproblems along physically-motivated dynamical partitions, 2) multi-injection (MI) to realign RL excitation with classical input/output insights for improved PE, and 3) modulation-enhanced excitation (MEE) to further improve scaling and numerics through design-motivated nonsingular state transformations.

In order to substantively demonstrate (d)EIRL's performance enhancements over existing ADP-based CT-RL methods [1–4], we conduct in-depth numerical evaluations on a real-world unstable, nonminimum phase hypersonic vehicle (HSV) system. Our evaluations show that the EIRL suite exhibits numerical conditioning improvements of many orders of magnitude relative to leading ADP-based CT-RL results. We also demonstrate that the flagship dEIRL method achieves recovery of the optimal controller in the face of significant HSV modeling errors and initial conditions. Altogether, these evaluations show that our two new RL design paradigms enable the proposed EIRL suite to become one of the first CT-RL methods to deliver both theoretical and real-world synthesis guarantees. dEIRL also recovers optimal closed-loop reference command following performance, results which hold statistically when the modeling error is randomly distributed.

We also evaluate dEIRL's performance relative to the leading deep CT-RL FVI methods [7, 8] on three pendulum, jet aircraft, and ground robot environments. These studies reveal advantages for dEIRL over deep RL FVI with respect to time/data complexity, dEIRL requiring three orders of magnitude less time and seven orders of magnitude less simulations to train. dEIRL also exhibits a clear cost, approximation, and time-domain performance advantage for the higher-order, multi-loop systems studied in this work, empirically validating our physics-based decentralization paradigm developed for these systems.

This dissertation presents fundamental performance improvements to the CT-RL state of the field. Nevertheless, future extensions of this work will likely achieve yet further enhancements. Additional numerical studies on a variety of dynamical systems will also be crucial to cementing the numerical performance of the proposed methods. In particular, benchmarking against the recently-developed fitted value iteration (FVI) CT-RL works in deep RL [7, 8] on the flagship HSV system will be

highly illustrative, as these new methods show great numerical promise. Given the empirically-demonstrated results on the challenging HSV system, we are confident that EIRL will produce highly competitive closed-loop performance for these newly-considered systems and control methods.

Furthermore, this work focuses on the control of affine nonlinear systems. Methods to substantively address general nonlinear systems require innovative new ideas, analyses of performance limitations, and substantial evaluations in meaningful applications in order to achieve success in what we have demonstrated is a uniquely-challenging CT-RL control context. These methods face yet further challenges from those seen for affine nonlinear systems. A fundamental hurdle arises in effectively approximating the general nonlinear dynamical structure in these problems. Furthermore, for general nonlinear systems, there does not exist a closed-form relation between the optimal policy and the gradient of the HJB solution, as is the case in the affine-nonlinear context. New ideas in learning and approximation must be developed to address these considerations. Fortunately, the principles developed in this work (both in algorithm numerical conditioning analysis, and in the physics-based, classical-control approach to improving learning quality) will likely serve as great starting points to address these new challenges, as these ideas have been successful in tackling general nonlinear control problems outside the RL field.

REFERENCES

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

[2] 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.

[3] Y. Jiang and Z.-P. Jiang, "Robust adaptive dynamic programming and feedback stabilization of nonlinear systems," *IEEE TNNLS*, vol. 25, no. 5, pp. 882–893, Jan. 2014.

[4] T. Bian and Z.-P. Jiang, "Reinforcement learning and adaptive optimal control for continuous-time nonlinear systems: A value iteration approach," *IEEE TNNLS*, vol. 33, no. 7, pp. 2781–2790, Jul. 2022.

[5] Q. Wang and R. F. Stengel, "Robust nonlinear control of a hypersonic aircraft," *AIAA J. Guid., Contr., & Dyn.*, vol. 23, no. 4, pp. 577–585, Jul. 2000.

[6] C. I. Marrison and R. F. Stengel, "Design of robust control systems for a hypersonic aircraft," *AIAA J. Guid., Contr., & Dyn.*, vol. 21, no. 1, pp. 58–63, Jan. 1998.

[7] M. Lutter, S. Mannor, J. Peters, D. Fox, and A. Garg, "Value iteration in continuous actions, states and time," in *Proc. 38th Int. Conf. Mach. Learn. (ICML)*, vol. 139, Jul. 2021, pp. 7224–7234.

[8] M. Lutter *et al.*, "Continuous-time fitted value iteration for robust policies," *IEEE Trans. Patt. Anal. Mach. Intel.*, vol. 45, no. 5, pp. 5534–5548, May 2023.

[9] H. W. Bode, *Network Analysis and Feedback Amplifier Design*. New York, NY, USA: D. Van Nostrand, 1945.

[10] H. Nyquist, "Regeneration theory," *Bell System Tech. J.*, vol. 11, pp. 126–147, 1932.

[11] K. Ogata, *Modern Control Engineering*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 1997.

[12] A. A. Rodriguez, *Analysis and Design of Feedback Control Systems*. Tempe, AZ, USA: CONTROL3D, 2003.

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

[14] A. Isidori, *Nonlinear Control Systems: An Introduction*. Berlin, Germany: Springer Verlag, 1985.

[15] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*, 3rd ed. John Wiley & Sons, 2012.

[16] M. Athans and P. L. Falb, *Optimal control: An introduction to the theory and its applications*, 1st ed. Mineola, NY, USA: McGraw-Hill, 1966.

[17] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Upper Saddle River, NJ, USA: Prentice Hall, 1995.

[18] F. Lin, *Robust Control Design: An Optimal Control Approach*. West Sussex, England: John Wiley & Sons, 2007.

[19] Z.-P. Jiang, A. R. Teel, and L. Praly, "Small-gain theorem for ISS systems and applications," *Math. Contr., Sig., Syst.*, vol. 7, no. 4, p. 95–120, Jun. 1994.

[20] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, vol. 13, no. 5, pp. 835–846, Oct. 1983.

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

[22] R. E. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press, 1957.

[23] J. Si, A. G. Barto, W. B. Powell, and D. C. Wunsch, *Handbook of Learning and Approximate Dynamic Programming*. Piscataway, NJ, USA: Wiley, 2004.

[24] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Contr. Syst. Mag.*, vol. 32, pp. 76–105, 2012.

[25] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE TNNLS*, vol. 29, no. 6, pp. 2042–2062, Jun. 2018.

[26] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Ann. Rev. Control, Robot., Auton. Syst.*, vol. 2, no. 1, pp. 253–279, 2019.

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

[28] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE TNNLS*, vol. 25, no. 3, pp. 621–634, Mar. 2014.

[29] Q. Wei, D. Liu, Q. Lin, and R. Song, "Discrete-time optimal control via local policy iteration adaptive dynamic programming," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3367–3379, Oct. 2016.

[30] W. Guo, J. Si, F. Liu, and S. Mei, "Policy approximation in policy iteration approximate dynamic programming for discrete-time nonlinear systems," *IEEE TNNLS*, vol. 29, no. 7, pp. 2794–2807, Jul. 2018.

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

[32] X. Gao, J. Si, Y. Wen, M. Li, and H. Huang, "Reinforcement learning control of robotic knee with human-in-the-loop by flexible policy iteration," *IEEE TNNLS*, vol. 33, no. 10, pp. 5873–5887, May 2021.

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

[34] D. Wang, D. Liu, Q. Wei, D. Zhao, and N. Jin, "Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming," *Automatica*, vol. 48, no. 8, pp. 1825–1832, Aug. 2012.

[35] F. Liu, J. Sun, J. Si, W. Guo, and S. Mei, "A boundedness result for the direct heuristic dynamic programming," *Neur. Net.*, vol. 32, pp. 229–235, Aug. 2012.

[36] D. Liu and Q. Wei, "Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 779–789, Apr. 2013.

[37] Q. Wei, F.-Y. Wang, D. Liu, and X. Yang, "Finite-approximation-error-based discrete-time iterative adaptive dynamic programming," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2820–2833, Sep. 2014.

[38] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, Jan. 2016.

[39] V. Minh *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[40] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[41] ——, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[42] F. Farahnakian, P. Lijeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *2014 22nd Euromicro Int. Conf. Par., Dist., Net.-Based Proc.*, Feb. 2014, pp. 500–507.

[43] K. Mondal, A. A. Rodriguez, S. S. Manne, N. Das, and B. A. Wallace, "Comparison of kinematic and dynamic model based linear model predictive control of non-holonomic robot for trajectory tracking: Critical trade-offs addressed," in *Proc. IASTED Int. Conf. Mech. Contr.*, Dec. 2019, pp. 9–17.

[44] K. Mondal, B. A. Wallace, and A. A. Rodriguez, "Stability versus maneuverability of non-holonomic differential drive wheeled robot: Focus on aggressive position control strategies," in *2020 IEEE CCTA*, Aug. 2020, pp. 388–395.

[45] C. Lu, J. Si, and X. Xie, "Direct heuristic dynamic programming for damping oscillations in a large power system," *IEEE Trans. Syst., Man, Cybern. – B: Cybern.*, vol. 38, no. 4, pp. 1008–1013, Aug. 2008.

[46] W. Guo, F. Liu, J. Si, D. He, R. Harley, and S. Mei, "Approximate dynamic programming based supplementary reactive power control for DFIG wind farm to enhance power system stability," *Neurocomp.*, vol. 170, pp. 417–427, Dec. 2015.

[47] ——, "Online supplementary ADP learning controller design and application to power system frequency control with large-scale wind energy integration," *IEEE TNNLS*, vol. 27, no. 8, pp. 1748–1761, Aug. 2016.

[48] Q. Wei and D. Liu, "Data-driven neuro-optimal temperature control of water–gas shift reaction using stable iterative adaptive dynamic programming," *IEEE Trans. Indus. Electr.*, vol. 61, no. 11, pp. 6399–6408, Jan. 2014.

[49] Y. Jiang, J. Fan, T. Chai, and F. L. Lewis, "Dual-rate operational optimal control for flotation industrial process with unknown operational model," *IEEE Trans. Indus. Electr.*, vol. 66, no. 6, pp. 4587–4599, Jun. 2019.

[50] R. Enns and J. Si, "Apache helicopter stabilization using neural dynamic programming," *AIAA J. Guid., Contr., & Dyn.*, vol. 25, no. 1, pp. 19–25, Jan. 2002.

[51] ——, "Helicopter trimming and tracking control using direct neural dynamic programming," *IEEE TNN*, vol. 14, no. 4, pp. 929–939, Jul. 2003.

[52] ——, "Helicopter flight-control reconfiguration for main rotor actuator failures," *AIAA J. Guid., Contr., & Dyn.*, vol. 26, no. 4, pp. 572–584, Jul. 2003.

[53] Q. Yang, W. Cao, W. Meng, and J. Si, "Reinforcement-learning-based tracking control of waste water treatment process under realistic system conditions and control performance requirements," *IEEE Trans. Syst., Man, Cybern. – A: Syst.*, vol. 52, no. 8, pp. 5284–5294, Aug. 2022.

[54] Y. Wen, M. Liu, J. Si, and H. Huang., "Adaptive control of powered transfemoral prostheses based on adaptive dynamic programming," in *2016 38th Ann. Int. Conf. IEEE Engin. Med. Bio. Soc.*, Aug. 2016, pp. 500–507.

[55] Y. Wen, J. Si, X. Gao, S. Huang, and H. Huang, "A new powered lower limb prosthesis control framework based on adaptive dynamic programming," *IEEE TNNLS*, vol. 28, no. 9, pp. 2215–2220, Sep. 2017.

[56] Y. Wen, J. Si, A. Brandt, X. Gao, and H. Huang, "Online reinforcement learning control for the personalization of a robotic knee prosthesis," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2346–2356, Jan. 2019.

[57] R. Wu, M. Li, Z. Yao, W. Liu, J. Si, and H. Huang, "Reinforcement learning impedance control of a robotic prosthesis to coordinate with human intact knee motion," *IEEE Robo. & Auto. Lett.*, vol. 7, no. 3, pp. 7014–7020, Jun. 2022.

[58] M. Li, Y. Wen, X. Gao, J. Si, and H. Huang, "Toward expedited impedance tuning of a robotic prosthesis for personalized gait assistance by reinforcement learning control," *IEEE Trans. Robo.*, vol. 38, no. 1, pp. 407–420, Feb. 2022.

[59] R. Wu, J. Zhong, B. A. Wallace, X. Gao, H. Huang, and J. Si, "Human-robotic prosthesis as collaborating agents for symmetrical walking," in *Adv. Neur. Info. Proc. Syst. (NeurIPS)*, vol. 36, Nov. 2022, pp. 1–15.

[60] J. J. Craig, *Introduction to Robotics: Mechanics and Control.*, 3rd ed. Upper Saddle River, NJ, USA: Pearson Education, 2005.

[61] R. Abraham and J. E. Marsden, *Foundations of Mechanics*, 2nd ed. Redwood City, CA, USA: Addison-Wesley, 1978.

[62] M. Morari and E. Zafiriou, *Robust Process Control*. Englewood Cliffs, NJ, USA: Prentice Hall, 1989.

[63] Y. Zhu and D. Zhao, "Comprehensive comparison of online ADP algorithms for continuous-time optimal control," *Artif. Intel. Rev.*, vol. 49, pp. 531–547, 2018.

[64] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circ. Syst. Mag.*, vol. 9, no. 2, pp. 32–50, Aug. 2009.

[65] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA, USA: Athena Scientific, 2005.

[66] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: Wiley, 1994.

[67] R. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA, USA: MIT Press, 1960.

[68] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA, USA: Athena Scientific, 1996.

[69] F. Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *IEEE Comput. Intel. Mag.*, vol. 4, no. 2, pp. 39–47, May 2009.

[70] P. J. Werbos, "Neural networks for control and system identification," in *Proc. 28th IEEE CDC*, Dec. 1989, pp. 260–265.

[71] ——, *A Menu of Designs for Reinforcement Learing Over Time*. Cambridge, MA, USA: MIT Press, 1991.

[72] ——, "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. Sofge, Eds. New York, NY, USA: Van Nostrand, 1992.

[73] D. Wang, D. Liu, H. Li, and H. Ma, "An approximate optimal control approach for robust stabilization of a class of discrete-time nonlinear systems with uncertainties," *IEEE Trans. Syst., Man, Cybern. – A: Syst.*, vol. 46, no. 5, pp. 713–717, May 2016.

[74] P. He and S. Jagannathan, "Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints," *IEEE Trans. Syst., Man, Cybern. – B: Cybern.*, vol. 43, no. 2, pp. 779–789, Apr. 2013.

[75] P. Zhang, Y. Yuan, and L. Guo, "Fault-tolerant optimal control for discrete-time nonlinear system subjected to input saturation: A dynamic event-triggered approach," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 2956–2968, Jun. 2021.

[76] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," *IEEE TNN*, vol. 12, no. 2, pp. 264–276, Mar. 2001.

[77] R. Hafner and M. Riedmiller, "Reinforcement learning in feedback control: Challenges and benchmarks from technical process control," *Mach. Learn.*, vol. 84, no. 1, pp. 137–169, Feb. 2011.

[78] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn. (ICML)*, Jan. 2014, pp. 387–395.

[79] R. Lillicrap, P. Timothy, 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.

[80] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel DDPG method with prioritized experience replay," in *2017 IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2017, pp. 316–321.

[81] A. E. Bryson and W. F. Denham, "A steepest-ascent method for solving optimum programming problems," *ASME J. Appl. Mech.*, vol. 29, pp. 247–257, Jun. 1962.

[82] D. E. Kirk, *Optimal Control Theory.* Englewood Cliffs, NJ, USA: Prentice-Hall, 1970.

[83] A. P. Sage and C. C. White III, *Optimum Systems Control*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1970.

[84] K. A. Wise and J. L. Sedwick, "Successive approximation solution of the hji equation," in *Proc. 33rd IEEE CDC*, vol. 2, Dec. 1994, pp. 1387–1391.

[85] J. Huang and C.-F. Lin, "Numerical approach to computing nonlinear $\mathcal{H}^\infty$ control laws," *AIAA J. Guid., Contr., & Dyn.*, vol. 18, pp. 989–994, 1995.

[86] M. G. Grandall, H. Ishii, and P.-L. Lions, "User's guide to viscosity solutions of second order partial differential equations," *Bulletin of the AMS*, vol. 27, pp. 1–67, 1992.

[87] H. J. Kushner, "Numerical methods for stochastic control problems in continuous time," *SIAM J. Control Opt.*, vol. 28, pp. 999–1048, 1990.

[88] W. H. Fleming and H. M. Soner, *Controlled Markov Processes and Viscosity Solutions*. Berlin, Germany: Springer Verlag, 1993.

[89] R. W. Beard and T. T. McLain, "Successive galerkin approximation algortihms for nonlinear optimal and robust control," *Int. J. Contr.*, vol. 71, pp. 717–743, 1998.

[90] R. W. Beard, "Improving the closed-loop performance of nonlinear systems," Ph.D. thesis, Rensselaer Poly. Inst., Troy, NY, USA, Oct. 1995.

[91] R. W. Beard, G. N. Saridis, and J. T. Wen, "Galerkin approximations of the generalized Hamilton-Jacobi-Bellman equation," *Automatica*, vol. 33, no. 12, pp. 2159–2177, Jun. 1997.

[92] M. Abu-Khalaf and F. L. Lewis, "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network hjb approach," *Automatica*, vol. 41, no. 5, pp. 779–791, May 2005.

[93] F. L. Lewis, M. Abu-Khalaf, and J. Huang, "Hamilton-Jacobi-Isaacs formulation for constrained input systems: Neural network solution," *IFAC*, vol. 37, no. 21, pp. 1–6, Dec. 2004.

[94] M. Abu-Khalaf and F. L. Lewis, *Nonlinear $\mathcal{H}_2/\mathcal{H}_\infty$ Constrained Feedback Control*. Berlin, Germany: Springer Verlag, 2006.

[95] O. Kuljaca and F. L. Lewis, "Fuzzy logic/neural network adaptive critic controller design," in *Proc. 41st IEEE CDC*, Dec. 2002, pp. 3356–3361.

[96] X. Ren, A. B. Rad, and F. L. Lewis, "Neural network-based compensation control of robot manipulators with unknown dynamics," in *Proc. 2007 IEEE ACC*, Jul. 2007, pp. 13–18.

[97] O. Kuljaca, F. L. Lewis, and S. Tesnjak, "Neural network frequency control for thermal power systems," in *Proc. 43rd IEEE CDC(IEEE Cat. No. 04CH37601)*, vol. 5, Dec. 2004, pp. 3509–3514.

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

[99] 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 TNNLS*, vol. 24, no. 10, pp. 1513–1525, Oct. 2013.

[100] Y. Jiang and Z.-P. Jiang, "Robust adaptive dynamic programming for large-scale systems with an application to multimachine power systems," *IEEE Trans. Circ. and Sys. – II: Express Briefs*, vol. 59, no. 10, pp. 693–697, Oct. 2012.

[101] ——, "Robust adaptive dynamic programming with an application to power systems," *IEEE TNN*, vol. 24, pp. 1150–1156, Jul. 2013.

[102] ——, "Adaptive dynamic programming as a theory of sensorimotor control," *Bio. Cybern.*, vol. 108, no. 4, pp. 459–473, 2014.

[103] W. Gao, Y. Jiang, Z.-P. Jiang, and T. Chai, "Output-feedback adaptive optimal control of interconnected systems based on robust adaptive dynamic programming," *Automatica*, vol. 72, pp. 37–45, Oct. 2016.

[104] Y. Jiang and J. Z.-P., "Global adaptive dynamic programming for continuous-time nonlinear systems," *IEEE TAC*, vol. 60, no. 11, pp. 2197–2929, Nov. 2015.

[105] D. Wang, D. Liu, and H. Ma, "Neural-network-based robust optimal control design for a class of uncertain nonlinear systems via adaptive dynamic programming," *Info. Sci.*, vol. 282, pp. 167–179, Oct. 2014.

[106] X. Yang, D. Liu, and D. Wang, "Reinforcement learning for adaptive optimal control of unknown continuous-time nonlinear systems with input constraints," *Int. J. Contr.*, vol. 87, no. 3, pp. 553–566, 2014.

[107] Y. Yang, D. Wunsch, and Y. Yin, "Hamiltonian-driven adaptive dynamic programming for continuous nonlinear dynamical systems," *IEEE TNNLS*, vol. 28, no. 8, pp. 1929–1940, Feb. 2017.

[108] F. A. Yaghmaie and D. J. Braun, "Reinforcement learning for a class of continuous-time input constrained optimal control problems," *Automatica*, vol. 99, pp. 221–227, Jan. 2019.

[109] Y. Li, K. Sun, and S. Tong, "Observer-based adaptive fuzzy fault-tolerant optimal control for SISO nonlinear systems," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 649–661, Feb. 2019.

[110] J. Zhao, J. Na, and G. Gao, "Adaptive dynamic programming based robust control of nonlinear systems with unmatched uncertainties," *Neurocomp.*, vol. 395, pp. 56–65, Jun. 2020.

[111] H. Liu, W. Zhao, F. L. Lewis, Z.-P. Jiang, and H. Modares, "Attitude synchronization for multiple quadrotors using reinforcement learning," in *2019 IEEE CCC*, Jul. 2019, pp. 2480–2483.

[112] L. Cui, S. Wang, J. Zhang, D. Zhang, J. Lai, Y. Zheng, Z. Zhang, and Z.-P. Jiang, "Learning-based balance control of wheel-legged robots," *IEEE Robo. & Auto. Lett.*, vol. 6, no. 4, pp. 7667–7674, 2021.

[113] J. N. Tsitsiklis and B. Van Roy, "Analysis of temporal-difference learning with function approximation," *Adv. Neur. Info. Proc. Sys.*, vol. 9, pp. 1075–1081, 1996.

[114] ——, "Analysis of temporal-difference learning with function approximation," *IEEE TAC*, vol. 42, no. 5, pp. 674–690, May 1997.

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

[116] C. Possieri and M. Sassano, "Q-learning for continuous-time linear systems: A data-driven implementation of the Kleinman algorithm," *IEEE Trans. Syst., Man, Cybern. – A: Syst.*, vol. 52, no. 10, pp. 6487–6497, Oct. 2022.

[117] D. Kleinman, "On an iterative technique for Riccati equation computations," *IEEE TAC*, vol. 13, no. 1, pp. 114–115, Feb. 1968.

[118] J. Lee and R. S. Sutton, "Policy iterations for reinforcement learning problems in continuous time and space–Fundamental theory and methods," *Automatica*, vol. 126, p. 109421, Apr. 2021.

[119] J. Kim, J. Shin, and I. Yang, "Hamilton-Jacobi deep Q-learning for deterministic continuous-time systems with Lipschitz continuous controls," *J. Mach. Learn. Res.*, vol. 22, no. 1, pp. 9363–9396, Sep. 2021.

[120] C. Yildiz, M. Heinonen, and H. Lähdesmäki, "Continuous-time model-based reinforcement learning," in *Proc. 38th Int. Conf. Mach. Learn. (ICML)*, Jul. 2021, pp. 12 009–12 018.

[121] J. D. Shaughnessy, S. Z. Pinckney, J. D. McMinn, C. I. Cruz, and M.-L. Kelley, "Hypersonic vehicle simulation model: Winged-cone configuration," *NASA TM-102610*, Nov. 1990.

[122] R. F. Stengel, *Flight Dynamics*, 2nd ed. Princeton, NJ, USA: Princeton University Press, 2022.

[123] S. Zhao, J. Wang, H. Xu, and B. Wang, "Composite observer-based optimal attitude-tracking control with reinforcement learning for hypersonic vehicles," *IEEE Trans. Cybern.*, vol. 53, no. 2, pp. 913–926, Feb. 2023.

[124] B. Xu, D. Wang, F. Sun, and Z. Shi, "Direct neural control of hypersonic flight vehicles with prediction model in discrete time," *Neurocomp.*, vol. 115, pp. 39–48, Sep. 2013.

[125] B. Xu, C. Yang, and Y. Pan, "Global neural dynamic surface tracking control of strict-feedback systems with application to hypersonic flight vehicle," *IEEE TNNLS*, vol. 26, no. 10, pp. 2563–2575, Aug. 2015.

[126] X. Bu, Y. Xiao, and H. Lei, "An adaptive critic design-based fuzzy neural controller for hypersonic vehicles: Predefined behavioral nonaffine control," *IEEE/ASME Trans. Mechatron.*, vol. 24, no. 4, pp. 1871–1881, Aug. 2019.

[127] X. Bu and Q. Qi, "Fuzzy optimal tracking control of hypersonic flight vehicles via single-network adaptive critic design," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 1, pp. 270–278, Jan. 2022.

[128] Q. Qi and X. Bu, "Adaptive dynamic programing design for the neural control of hypersonic flight vehicles," *J. Franklin. Inst.*, vol. 358, no. 16, pp. 8169–8192, Oct. 2021.

[129] X. Tao, J. Yi, X. Pu, and T. Xiong, "State-estimator-integrated robust adaptive tracking control for flexible air-breathing hypersonic vehicle with noisy measurements," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 11, pp. 4285–4299, Nov. 2019.

[130] C. Mu, Z. Ni, C. Sun, and H. He, "Air-breathing hypersonic vehicle tracking control based on adaptive dynamic programming," *IEEE TNNLS*, vol. 28, no. 3, pp. 584–598, Mar. 2017.

[131] H.-Y. Qiao, H. Meng, M.-J. Wang, W. Ke, and J.-G. Sun, "Adaptive control for hypersonic vehicle with input saturation and state constraints," *Aero. Sci. Tech.*, vol. 84, pp. 107–119, Jan. 2019.

[132] J. Dickeson, A. A. Rodriguez, S. Sridharan, J. Benevides, and D. Soloway, "Decentralized control of an airbreathing scramjet-powered hypersonic vehicle," in *AIAA Guid., Nav., Contr. Conf. Exhib.*, Aug. 2009, pp. 1–25, AIAA 2009-6281.

[133] J. Dickeson, A. A. Rodriguez, S. Sridharan, D. Soloway, A. Korad, J. Khatri, J. Benavides, A. Kelkar, and J. Vogel, "Control-relevant modeling, analysis, and design for scramjet-powered hypersonic vehicles," in *AIAA/DLR/DGLR Int. Space Planes Hyper. Syst. Tech. Conf.*, Oct. 2009, pp. 1–45, AIAA 2009-7287.

[134] J. J. Dickeson, "Control relevant modeling and design of scramjet-powered hypersonic vehicles," Ph.D. thesis, Arizona State University, Tempe, AZ, USA, 2012.

[135] J. T. Parker, A. Serrani, S. Yurkovich, M. A. Bolender, and D. B. Doman, "Approximate feedback linearization of an air-breathing hypersonic vehicle," in *AIAA Guid., Nav., Contr. Conf. Exhib.*, Aug. 2006, pp. 1–16, AIAA 2006-6556.

[136] A. A. Rodriguez, *Analysis and Design of Multivariable Feedback Control Systems*. Tempe, AZ, USA: CONTROL3D, 2004.

[137] B. A. Wallace and J. Si, "Continuous-time reinforcement learning control: A review of theoretical results, insights on performance, and needs for new designs," *IEEE TNNLS*, Feb. 2023.

[138] R. Dhaouadi and A. Abu Hatab, "Dynamic modelling of differential-drive mobile robots using lagrange and newton-euler methodologies: A unified framework," *Adv. Robo. Auto.*, vol. 2, no. 2, pp. 1–7, Jan. 2013.

[139] K. Mondal, "Dynamics, directional maneuverability and optimization based multivariable control of nonholonomic differential drive mobile robots," Ph.D. thesis, Arizona State University, Department of Electrical Engineering, Tempe, AZ, USA, Dec. 2021.

[140] P. Wang, Z. Man, Z. Cao, J. Zheng, and Y. Zhao, "Dynamics modelling and linear control of quadcopter," in *IEEE Int. Conf. Adv. Mech. Syst. (ICAMechS)*, Nov. 2016, pp. 498–503.

[141] M. A. Bolender and D. B. Doman, "Flight path angle dynamics of air-breathing hypersonic vehicles," in *AIAA Guid., Nav., Contr. Conf. Exhib.*, Aug. 2006, AIAA 2006-6692.

[142] ——, "Nonlinear longitudinal dynamical model of an air-breathing hypersonic vehicle," *J. Spacecraft Rockets*, vol. 44, no. 2, pp. 374–387, Mar. 2007.

[143] https://lewisgroup.uta.edu/code/Software%20from%20Research.htm, Accessed: 2022-04-08.

[144] K. G. Vamvoudakis, D. Vrabie, and F. L. Lewis, "Online policy iteration based algorithms to solve the continuous-time infinite horizon optimal control problem," *IEEE Symp. ADPRL*, pp. 36–41, Mar. 2009.

[145] V. Lakshmikantham, S. Leela, and A. A. Martynyuk, *Practical Stability of Nonlinear Systems*. Singapore: World Scientific Publishing Co., 1990.

[146] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. Philadelphia, PA, USA: SIAM, 2002.

[147] B. A. Wallace and J. Si, "TNNLS-2022 – CT-RL Optimal Control," https://github.com/bawalla2/TNNLS-2022--CT-RL-Optimal-Control.git, Accessed: 2022-12-01.

[148] ——, "Physics-based integral reinforcement learning: New control design algorithms with theoretical insights and performance guarantees," *IEEE TNNLS*, May 2023, in Review. arXiv preprint available: https://arxiv.org/abs/2307.08920.

[149] Y. Jiang and Z.-P. Jiang, "Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics," *Automatica*, vol. 48, no. 10, pp. 2699–2704, Oct. 2012.

[150] R. A. Horn, *Topics in Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1991.

[151] B. A. Wallace and J. Si, "TNNLS 2023 – dEIRL," Available: https://github.com/bawalla2/TNNLS-2023---dEIRL, Accessed: 2023-05-25.

[152] ——, "RCI (JMLR 2024)," https://github.com/bawalla2/JMLR-2024, 2024.

[153] M. Lutter *et al.*, "Continuous & robust fitted value iteration," https://github.com/milutter/value_iteration, 2023, Accessed: 2023-01-12.

[154] B. A. Wallace and J. Si, "AIAA JGCD 2024 RL HSV," Available: https://github.com/bawalla2/AIAA-JGCD-2024-RL-HSV, Accessed: 2023-11-02.

[155] F. Alizadeh, J. A. Haeberly, and M. Overton, "Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results," *SIAM J. Opt.*, vol. 8, no. 3, pp. 746–768, Aug. 1998.

[156] M. J. Todd, K. C. Toh, and R. H. Tütüncü, "On the Nesterov–Todd direction in semidefinite programming," *SIAM J. Opt.*, vol. 8, no. 3, pp. 769–796, Aug. 1998.

[157] A. Tunçel and H. Wolkowicz, "Strengthened existence and uniqueness conditions for search directions in semidefinite programming," *Lin. Alg. Apps.*, vol. 400, pp. 31–60, May 2005.

[158] N. Kalantarova and A. Tunçel, "On the spectral structure of jordan-kronecker products of symmetric and skew-symmetric matrices," *Lin. Alg. Apps.*, vol. 608, pp. 343–362, Jan. 2021.

[159] N. Kalantarova, "Spectral properties of structured kronecker products and their applications," Ph.D. thesis, University of Waterloo, Department of Combinatorics and Optimization, Waterloo, Ontario, CA, May 2019.

[160] J. W. Brewer, "Kronecker products and matrix calculus in system theory," *IEEE Circ. Syst.*, vol. 25, no. 9, pp. 772–781, Sep. 1978.

[161] J. M. Lee, *Introduction to Smooth Manifolds*, 2nd ed. New York, NY, USA: Springer, 2013.

[162] H. Xu, M. D. Mirmirani, and P. A. Ioannou, "Robust neural adaptive control of a hypersonic aircraft," in *AIAA Guid., Nav., Contr. Conf. Exhib.*, Aug. 2003, pp. 1–8, AIAA 2003-5641.

[163] ——, "Adaptive sliding mode control design for a hypersonic flight vehicle," *AIAA J. Guid., Contr., & Dyn.*, vol. 27, no. 5, pp. 829–838, Sep. 2004.

[164] M. A. Bolender and D. B. Doman, "A non-linear model for the longitudinal dynamics of a hypersonic air-breathing vehicle," in *AIAA Guid., Nav., Contr. Conf. Exhib.*, Aug. 2005, pp. 1–22, AIAA 2005-6255.

[165] P. T. Soderman and T. N. Aiken, "Full-scale wind-tunnel tests of a small unpowered jet aircraft with a T-tail," *NASA-TN-D-6573*, Nov. 1971.

[166] A. A. Rodriguez, J. Dickeson, O. Cifdaloz, A. Kelkar, J. Vogel, D. Soloway, R. McCullen, J. Benavides, and S. Sridharan, "Modeling and control of scramjet-powered hypersonic vehicles: Challenges, trends, and tradeoffs," in *AIAA Guid., Nav., Contr. Conf. Exhib.*, Aug. 2008, pp. 1–40, AIAA 2008-6793.

[167] J. Hauser, S. Sastry, and G. Meyer, "Nonlinear control design for slightly non-minimum phase systems: Application to V/STOL aircraft," *Automatica*, vol. 28, no. 4, pp. 665–679, Jul. 1992.

APPENDIX A

PRELIMINARIES: THE SYMMETRIC KRONECKER PRODUCT AND SYMMETRIC KRONECKER SUM

In this section, we first provide an overview of the symmetric Kronecker product, summarizing the notable developments to-date. We then derive a construction of the map and prove new key properties necessary for the development of the proposed modulation-enhanced excitation (MEE) framework (cf. Section 3.5).

A.1 Overview of Developments to-Date

The symmetric Kronecker product was originally devised in [155] for application to semidefinite programming as an operation on square-symmetric matrices. In this context, it was shown that the symmetric Kronecker product $\underline{\otimes}$ is symmetric as a bilinear form: $A \underline{\otimes} B = B \underline{\otimes} A$, and that $(A \underline{\otimes} A)^{-1} = A^{-1} \underline{\otimes} A^{-1}$ in the case A is invertible. The spectrum of $A \underline{\otimes} B$ was identified in the case that A, B are symmetric and commute. The symmetric Kronecker product was then extended in [156] to an operation on arbitrary square matrices. [156] identified many key algebraic properties analogous to those of the standard Kronecker product, including the usual transposition, mixed product, and mixed vector product identities. The spectrum of $A \underline{\otimes} A$ was identified in the general square matrix case. [157] then identified eigenpair relationships and definiteness characterizations: that positive (semi)definiteness of $A \underline{\otimes} B$ is equivalent to that of $A \otimes B$. More recently, the works [158, 159] provide spectral interlacing properties of the related Jordan-Kronecker product.

Notably, prior works to date have treated the symmetric Kronecker product as an operation only on square matrices $A, B \in \mathbb{R}^{n \times n}$, which we here generalize to rectangular matrices $A, B \in \mathbb{R}^{m \times n}$. Among other advantages, this allows us to identify the eigenstructure of $A \underline{\otimes} B$ as relating to the symmetric Kronecker products $x \underline{\otimes} y$ of eigenvectors x, y of A and B – a critical parallel to the well-known result of the standard Kronecker product. We also prove new properties in the square case which will be essential to the development of MEE. Importantly, we introduce the concept of the symmetric Kronecker sum $\underline{\oplus}$, proving algebraic, spectral, and exponentiation properties, as well as its role in characteriing existence/uniqueness of solutions to ALEs.

A.2 Construction

Notation. We denote $\langle \cdot, \cdot \rangle_F$ as the Frobenius inner product on $\mathbb{R}^{m \times n}$. Let $\otimes$, vec denote the usual Kronecker product and vectorization operations, respectively, and $\text{mat} = \text{vec}^{-1}$ [160]. For any concepts pertaining to differential geometry, this work follows the notational conventions of the standard text [161]. For $n \in \mathbb{N}$, let $\text{GL}(n) \subset \mathbb{R}^{n \times n}$ denote the (real) general linear group of square invertible $n \times n$ matrices. Let $\mathbb{S}^n \subset \mathbb{R}^{n \times n}$ denote the subspace of symmetric matrices, and let $\underline{n} \triangleq \frac{n(n+1)}{2}$ denote the dimension of $\mathbb{S}^n$.

Prior formulations of the symmetric Kronecker product [155–159] first define the product implicitly, but here we move straight to an explicit construction. For $n \in \mathbb{N}$, let $\{E_i\}_{i=1}^{\underline{n}}$ denote the orthonormal basis for $(\mathbb{S}^n, \langle \cdot, \cdot \rangle_F)$ enumerated as follows. Define

$s : \{0, \dots, n\} \to \{0, \dots, \underline{n}\}$, $r, c : \{1, \dots, \underline{n}\} \to \{1, \dots, n\}$ by

$$s(j) = \sum_{i=1}^{j} (n - (i-1)), \tag{A.1}$$

$$r(j) = p, \qquad s(p-1) < j \leq s(p), \tag{A.2}$$

$$c(j) = (r(j) - 1) + \big(j - s\big(r(j) - 1\big)\big)\,. \tag{A.3}$$

When necessary, we will add subscripts s_n, r_n, c_n to these maps to make their associated dimension n explicit. Note that $\{(r(j), c(j))\}_{j=1}^{\underline{n}}$ is given by $(1,1), (1,2)$, $\dots$, $(1,n)$, $(2,2), (2,3)$, $\dots, (2,n)$, $(3,3), \dots, (n-1,n), (n,n)$. This associates the index $1 \leq j \leq \underline{n}$ with its corresponding row/column index $(r(j), c(j))$ on/above the diagonal, beginning at the first row/column and moving left to right, up to down (cf. Figure A.1). These maps have not been defined explicitly in the constructions of prior works [155–159]; however, subsequently they will show great utility in indexing operations for proving properties of the symmetric Kronecker product, especially in developing our results for the rectangular-matrix case. Letting $\{e_i\}_{i=1}^{n}$ denote the standard basis on $\mathbb{R}^n$, we are now ready to enumerate the orthonormal basis $\{E_j\}_{j=1}^{\underline{n}}$ as

$$E_j = \begin{cases} e_{r(j)} e_{c(j)}^T, & r(j) = c(j), \\ \frac{\sqrt{2}}{2}\left(e_{r(j)} e_{c(j)}^T + e_{c(j)} e_{r(j)}^T\right), & r(j) < c(j). \end{cases} \tag{A.4}$$

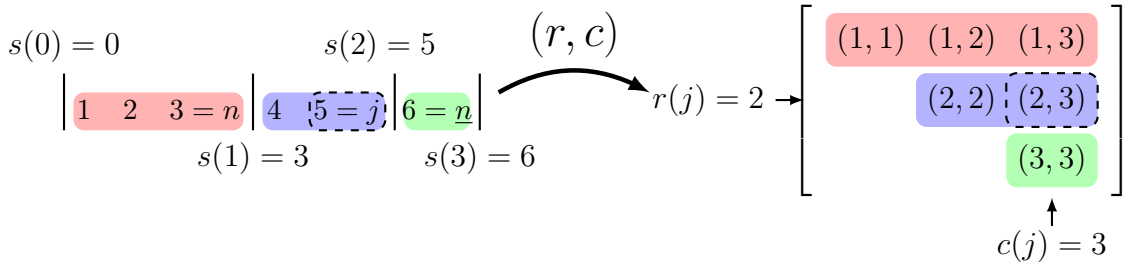


Figure A.1: Visualization of the Sum, Row, and Column Indexing Maps s (A.1), r (A.2), and c (A.3), Respectively, for $n = 3$.

Define $W \in \mathbb{R}^{\underline{n} \times n^2}$ as

$$W = \begin{bmatrix} \mathrm{vec}^T(E_1) \\ \vdots \\ \mathrm{vec}^T(E_{\underline{n}}) \end{bmatrix}. \tag{A.5}$$

Whenever necessary, we will also add a subscript $W_n \in \mathbb{R}^{\underline{n} \times n^2}$ to this matrix to make its dimensions explicit.

Definition A.2.1 (Symmetric Vectorization, Orthogonal Projection). Define

$\operatorname{svec} : \mathbb{S}^n \to \mathbb{R}^{\underline{n}}$ and $\pi : \mathbb{R}^{n\times n} \to \mathbb{S}^n$ by

$$\begin{aligned}\operatorname{svec}(P) &= \left[p_{1,1},\, \sqrt{2}p_{1,2}, \ldots,\, \sqrt{2}p_{1,n},\, p_{2,2},\, \sqrt{2}p_{2,3}, \ldots,\, \sqrt{2}p_{n-1,n},\, p_{n,n}\right]^T \\ &= \left[\, \langle P, E_1\rangle_F, \ldots, \langle P, E_{\underline{n}}\rangle_F \,\right]^T, \qquad \text{(A.6)} \\ \pi(A) &= \frac{A + A^T}{2}, \qquad \text{(A.7)}\end{aligned}$$

and define $\operatorname{smat} = \operatorname{svec}^{-1} : \mathbb{R}^{\underline{n}} \to \mathbb{S}^n$. We will discuss the properties of these operators shortly (cf. Proposition A.3.1).

Definition A.2.2 (The Symmetric Kronecker Product). Define the symmetric Kronecker product $\underline{\otimes} : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{\underline{m}\times\underline{n}}$ as

$$A \,\underline{\otimes}\, B = W_m \,(A \otimes B)\, W_n^T. \qquad \text{(A.8)}$$

Definition A.2.3 (The Symmetric Kronecker Sum). Define the symmetric Kronecker sum $\underline{\oplus} : \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n} \to \mathbb{R}^{\underline{n}\times\underline{n}}$ as

$$A \,\underline{\oplus}\, B = A \,\underline{\otimes}\, I + I \,\underline{\otimes}\, B = (A + B) \,\underline{\otimes}\, I. \qquad \text{(A.9)}$$

A.3 Properties

We begin this section by outlining the interaction of the vectorization operations vec, svec with the Frobenius inner product on matrix spaces.

Proposition A.3.1 (Vectorization and Frobenius Hilbert Space Structure).

1) $\operatorname{vec} : (\mathbb{R}^{m\times n}, \langle\cdot,\cdot\rangle_F) \to (\mathbb{R}^{mn}, \langle\cdot,\cdot\rangle)$ is a Hilbert space isomorphism; i.e., a linear bijection for which $\operatorname{vec}^T(A)\operatorname{vec}(B) = \langle A, B\rangle_F$, $A, B \in \mathbb{R}^{m\times n}$.

1S) $\operatorname{svec} : (\mathbb{S}^n, \langle\cdot,\cdot\rangle_F) \to (\mathbb{R}^{\underline{n}}, \langle\cdot,\cdot\rangle)$ is a Hilbert space isomorphism; i.e., a linear bijection for which $\operatorname{svec}^T(A)\operatorname{svec}(B) = \langle A, B\rangle_F$, $A, B \in \mathbb{S}^n$.

2) In the square-matrix case, the operators vec, svec interact with the Hilbert space structure of $(\mathbb{R}^{n\times n}, \langle\cdot,\cdot\rangle_F)$ via the following commutative diagram:

$$\begin{array}{ccccc}
(\mathbb{R}^{n\times n}, \langle\cdot,\cdot\rangle_F) & \xrightarrow[\perp \text{ proj.}]{\pi} & (\mathbb{S}^n, \langle\cdot,\cdot\rangle_F) & & \\
\operatorname{vec}\downarrow\cong\uparrow\operatorname{mat} & & \operatorname{vec}\downarrow\cong\uparrow\operatorname{mat} & \overset{\operatorname{smat}\ \cong}{\underset{\operatorname{svec}}{\swarrow\!\searrow}} & \\
\left(\mathbb{R}^{n^2}, \langle\cdot,\cdot\rangle\right) & \xrightarrow[\perp \text{ proj.}]{W^TW} & (\operatorname{vec}(\mathbb{S}^n), \langle\cdot,\cdot\rangle) & \underset{W^T}{\overset{W}{\underset{\cong}{\rightleftarrows}}} & (\mathbb{R}^{\underline{n}}, \langle\cdot,\cdot\rangle)
\end{array} \qquad \text{(A.10)}$$

where $W \in \mathbb{R}^{\underline{n}\times n^2}$ is given by (A.5), and $\cong$ denotes Hilbert space isomorphism.

In (A.10), $\operatorname{vec}(\mathbb{S}^n) \subset \mathbb{R}^{n^2}$ is often called the space of symmetric vectors [158, 159]. The operator π (A.7) and the matrix $W^TW \in \mathbb{R}^{n^2\times n^2}$ are the orthogonal projections onto the symmetric matrices and symmetric vectors, respectively. Furthermore,

(A.10) shows that the order of vectorization and projection may be swapped by interchanging these two projections.

We recall the general result from linear algebra that, given an n-dimensional real vector space V, any basis $\{x_i\}_{i=1}^n$ for V establishes a linear isomorphism $V \leftrightarrow \mathbb{R}^n$ between Euclidean space and elements of V via their (unique) representation in the basis. In the case considered, the symmetric vectors $\mathrm{vec}(\mathbb{S}^n) \subset \mathbb{R}^{n^2}$ comprise an $\underline{n}$-dimensional subspace spanned by the (orthonormal) basis $\{\mathrm{vec}(E_i)\}_{i=1}^{\underline{n}}$ (A.4). $W \in \mathbb{R}^{n^2 \times \underline{n}}$ is the matrix representation of the linear isomorphism $\mathrm{vec}(\mathbb{S}^n) \leftrightarrow \mathbb{R}^{\underline{n}}$. Indeed, invoking the Hilbert space structure of $\mathrm{vec}(\mathbb{S}^n)$, orthonormality of $\{\mathrm{vec}(E_i)\}_{i=1}^{\underline{n}}$ implies the i-th coefficient representation ($i = 1, \ldots, \underline{n}$) of an element $\mathrm{vec}(P) \in \mathrm{vec}(\mathbb{S}^n)$ is given by $\langle \mathrm{vec}(P), \mathrm{vec}(E_i) \rangle = \langle P, E_i \rangle_F$, which is precisely the action of W (A.5). Similarly, examination of (A.6) immediately shows that the symmetric vectorization svec is the Euclidean space correspondence $\mathbb{S}^n \leftrightarrow \mathbb{R}^{\underline{n}}$ associated with the orthonormal basis $\{E_i\}_{i=1}^{\underline{n}}$. In all, (A.10) illustrates that $\mathrm{svec} = W \circ \mathrm{vec}$, a composition of Hilbert space isomorphisms, is itself a Hilbert space isomorphism. Meanwhile, vec viewed as a map $\mathbb{S}^n \to \mathbb{R}^{n^2}$ "vectorizes" elements of $\mathbb{S}^n$ into the Euclidean space $\mathbb{R}^{n^2}$ isometrically, but vec is not onto. Thus, svec is the natural Hilbert space isomorphism of study on the symmetric matrices, as vec is to $\mathbb{R}^{n \times n}$.

Having formally related the properties of the vectorization operations vec and svec, it is perhaps of no surprise that svec algebraically interacts with the symmetric Kronecker product $\underline{\otimes}$ in an entirely analogous fashion to the interaction between vec and the Kronecker product $\otimes$.

Proposition A.3.2 (Kronecker Product Properties). For the sake of completeness and for comparison to the newly-developed results for the symmetric Kronecker product, we list the following well-known properties of the standard Kronecker product:

1) $\otimes : \mathbb{R}^{m \times n} \times \mathbb{R}^{p \times q} \to \mathbb{R}^{mp \times nq}$ is bilinear.

2) $\otimes$ is *not* symmetric; i.e., $A \otimes B \neq B \otimes A$, in general.

3) $(A \otimes B)\,\mathrm{vec}(C) = \mathrm{vec}(BCA^T)$, $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{p \times q}$, $C \in \mathbb{R}^{q \times n}$.

4) $(A \otimes B)^T = A^T \otimes B^T$, $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{p \times q}$.

5) $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$, $A \in \mathrm{GL}(n)$, $B \in \mathrm{GL}(m)$.

6) $(A \otimes B)(C \otimes D) = AC \otimes BD$, $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{p \times q}$, $C \in \mathbb{R}^{n \times r}$, $D \in \mathbb{R}^{q \times s}$.

7) For square matrices $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{m \times m}$, if $\sigma(A) = \{\lambda_i \mid i = 1, \ldots, n\}$ and $\sigma(B) = \{\mu_j \mid j = 1, \ldots, m\}$, then $\sigma(A \otimes B) = \{\lambda_i \mu_j \mid i = 1, \ldots, n,\ j = 1, \ldots, m\}$. Furthermore, if $x_i \in \mathbb{C}^n$, $y_j \in \mathbb{C}^m$ are eigenvectors corresponding to the eigenvalues λ_i of A and μ_j of B, respectively, then $x_i \otimes y_j$ is an eigenvector corresponding to the eigenvalue $\lambda_i \mu_j$ of $A \otimes B$.

8) $A \otimes I$ is symmetric if and only if A is, $A \in \mathbb{R}^{n \times n}$.

9) If $A \in \mathbb{S}^m, B \in \mathbb{S}^n$ are symmetric positive definite, then so is $A \otimes B$.

10) $A \otimes B = 0$ if and only if at least one $A, B = 0$, $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$.

11) $\det(A \otimes B) = \det(A)^m \det(B)^n$, $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{n\times n}$.

12) For $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{n\times n}$, if A, B are diagonal, then $A \otimes B$ is diagonal. If $A, B \neq 0$ and $A \otimes B$ is diagonal, then A, B are diagonal.

13) For $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{n\times n}$, $A \otimes B = I_{mn}$ if and only if $A = \lambda I_m$, $B = \frac{1}{\lambda} I_n$ for some $\lambda \neq 0$.

14) The map $\Phi : \mathrm{GL}(n) \to \mathrm{GL}^+(n^2)$,

$$\Phi(A) = A \otimes A, \qquad A \in \mathrm{GL}(n), \tag{A.11}$$

is a Lie group homomorphism with $\ker \Phi = \{\pm I\}$. $\Phi|_{\mathrm{GL}^+(n)} : \mathrm{GL}^+(n) \to \mathrm{GL}^+(n^2)$ is an injective Lie group homomorphism if and only if n is odd. In the case n is odd, $\Phi(\mathrm{GL}(n)) = \Phi(\mathrm{GL}^+(n)) \hookrightarrow \mathrm{GL}^+(n^2)$ is connected. In the case n is even, $\Phi(\mathrm{GL}(n))$ has two connected components $\Phi(\mathrm{GL}^+(n)), \Phi(\mathrm{GL}^-(n)) \hookrightarrow \mathrm{GL}^+(n^2)$.

Proof: 1))–13)) are standard results; see, e.g., [150, 160]. Enumerating $A = \{a_{i,j}\}_{i,j=1}^m$, $B = \{b_{k,l}\}_{k,l=1}^n$, 12)) and 13)) follow from the Kronecker product indexing identity:

$$\begin{aligned}(A \otimes B)_{(i-1)n+k,(j-1)n+l} &= a_{i,j} b_{k,l}, \\ i, j = 1, \ldots, m, &\quad k, l = 1, \ldots, n.\end{aligned} \tag{A.12}$$

For 14)), that Φ is a group homomorphism follows from 6)), and that $\ker \Phi = \{\pm I\}$ follows from 13)). For smoothness, identifying $\mathbb{R}^{n\times n} \cong \mathbb{R}^{n^2}$, $\mathbb{R}^{n^2\times n^2} \cong \mathbb{R}^{n^4}$, the map $A \mapsto A \otimes A : \mathbb{R}^{n\times n} \to \mathbb{R}^{n^2\times n^2}$ is polynomial in its coordinates, hence smooth. Thus, since $\mathrm{GL}(n) \hookrightarrow \mathbb{R}^{n\times n}$ is an open subset, it follows that $\Phi : \mathrm{GL}(n) \to \mathbb{R}^{n^2\times n^2}$ is smooth by restriction of the domain [161, Theorem 5.27]. But that $\Phi(\mathrm{GL}(n)) \subset \mathrm{GL}^+(n^2)$ follows from 11)), so since $\mathrm{GL}^+(n^2) \hookrightarrow \mathrm{GL}(n^2) \hookrightarrow \mathbb{R}^{n^2\times n^2}$, we may then restrict the codomain as well [161, Theorem 5.29], yielding $\Phi : \mathrm{GL}(n) \to \mathrm{GL}^+(n^2)$ is smooth. The remaining claims are straightforward, noting that $-I \in \mathrm{GL}^-(n)$ if and only if n is odd. ■

Proposition A.3.3 (Symmetric Kronecker Product Properties). The symmetric Kronecker product has the following properties developed previously in the in the square-matrix case [156], generalized here to rectangular matrices:

1S) $\underline{\otimes} : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{\underline{m}\times\underline{n}}$ is bilinear.

2S) $\underline{\otimes}$ is symmetric; i.e., $A \underline{\otimes} B = B \underline{\otimes} A$, $A, B \in \mathbb{R}^{m\times n}$.

3S) $(A \underline{\otimes} B)\,\mathrm{svec}(\pi(C)) = \mathrm{svec}(\pi(B\pi(C)A^T))$, $A, B \in \mathbb{R}^{m\times n}$, $C \in \mathbb{R}^{n\times n}$.

4S) $(A \underline{\otimes} B)^T = A^T \underline{\otimes} B^T$, $A, B \in \mathbb{R}^{m\times n}$.

5S) $(A \underline{\otimes} A)^{-1} = A^{-1} \otimes A^{-1}$, $A \in \mathrm{GL}(n)$. However, $(A \underline{\otimes} B)^{-1} \neq A^{-1} \underline{\otimes} B^{-1}$ for $A, B \in \mathrm{GL}(n)$, in general. Indeed, $A, B \in \mathrm{GL}(n)$ does not imply $A \underline{\otimes} B \in \mathrm{GL}(\underline{n})$.

6S) a) $(A \underline{\otimes} B)(C \underline{\otimes} D) = \frac{1}{2}(AC \underline{\otimes} BD + AD \underline{\otimes} BC)$, $A, B \in \mathbb{R}^{m \times n}$, $C, D \in \mathbb{R}^{n \times p}$.

b) $(A \underline{\otimes} B)(C \underline{\otimes} C) = AC \underline{\otimes} BC$, $A, B \in \mathbb{R}^{m \times n}$, $C \in \mathbb{R}^{n \times p}$.

c) $(C \underline{\otimes} C)(A \underline{\otimes} B) = CA \underline{\otimes} CB$, $A, B \in \mathbb{R}^{m \times n}$, $C \in \mathbb{R}^{p \times m}$.

7S) a) For a square matrix $A \in \mathbb{R}^{n \times n}$, if $\sigma(A) = \{\lambda_i \mid i = 1, \ldots, n\}$, then $\sigma(A \underline{\otimes} A) = \{\lambda_i \lambda_j \mid 1 \leq i \leq j \leq n\}$. Furthermore, if $x_i, x_j \in \mathbb{C}^n$ are eigenvectors corresponding to the eigenvalues λ_i, λ_j of A, respectively, then $x_i \underline{\otimes} x_j$ is an eigenvector corresponding to the eigenvalue $\lambda_i \lambda_j$ of $A \underline{\otimes} A$.

b) Suppose that $A, B \in \mathbb{R}^{n \times n}$ are simultaneously diagonalizable with common basis of eigenvectors $\{x_i\}_{i=1}^n$. If $\sigma(A) = \{\lambda_i \mid i = 1, \ldots, n\}$ and $\sigma(B) = \{\mu_j \mid j = 1, \ldots, n\}$ are the eigenvalues of A and B corresponding to the respective eigenvectors $\{x_i\}_{i=1}^n$, then
$\sigma(A \underline{\otimes} B) = \left\{\frac{1}{2}(\lambda_i \mu_j + \lambda_j \mu_i) \mid 1 \leq i \leq j \leq n\right\}$. Furthermore, $x_i \underline{\otimes} x_j$ is an eigenvector corresponding to the eigenvalue $\frac{1}{2}(\lambda_i \mu_j + \lambda_j \mu_i)$ of $A \underline{\otimes} B$.

c) Suppose that $A, B \in \mathbb{R}^{n \times n}$ share two eigenvectors $x, y \in \mathbb{C}^n$. If $Ax = \lambda_1 x$, $Bx = \mu_1 x$, $Ay = \lambda_2 y$, $By = \mu_2 y$, then $x \underline{\otimes} y$ is an eigenvector of $A \underline{\otimes} B$ corresponding to the eigenvalue $\frac{1}{2}(\lambda_1 \mu_2 + \lambda_2 \mu_1)$.

8S) $A \underline{\otimes} I$ is symmetric if and only if A is, $A \in \mathbb{R}^{n \times n}$.

9S) If $A, B \in \mathbb{S}^n$ are symmetric positive definite, then so is $A \underline{\otimes} B$.

10S) $A \underline{\otimes} B = 0$ if and only if at least one $A, B = 0$, $A, B \in \mathbb{R}^{m \times n}$.

In addition, the following newly-proved results are essential for the dEIRL MEE framework developed subsequently:

11S) $\det(A \underline{\otimes} A) = \det(A)^{n+1}$, $A \in \mathbb{R}^{n \times n}$.

12S) For $A, B \in \mathbb{R}^{n \times n}$, if A, B are diagonal, then $A \underline{\otimes} B$ is diagonal. If A, B are nonzero on each diagonal entry and $A \underline{\otimes} B$ is diagonal, then A, B are diagonal.

13S) For $A, B \in \mathbb{R}^{n \times n}$, $A \underline{\otimes} B = I_{\underline{n}}$ if and only if $A = \lambda I_n$, $B = \frac{1}{\lambda} I_n$ for some $\lambda \neq 0$.

14S) The map $\underline{\Phi} : \mathrm{GL}(n) \to \mathrm{GL}(\underline{n})$,

$$\underline{\Phi}(A) = A \underline{\otimes} A, \qquad A \in \mathrm{GL}(n), \tag{A.13}$$

is a Lie group homomorphism with $\ker \underline{\Phi} = \{\pm I\}$. $\underline{\Phi}|_{\mathrm{GL}^+(n)} : \mathrm{GL}^+(n) \to \mathrm{GL}^+(\underline{n})$ is an injective Lie group homomorphism if and only if n is odd. In the case n is odd, $\underline{\Phi}(\mathrm{GL}(n)) = \underline{\Phi}(\mathrm{GL}^+(n)) \hookrightarrow \mathrm{GL}^+(\underline{n})$ is connected. In the case n is even, $\underline{\Phi}(\mathrm{GL}(n))$ has two connected components $\underline{\Phi}(\mathrm{GL}^+(n)) \hookrightarrow \mathrm{GL}^+(\underline{n})$, $\underline{\Phi}(\mathrm{GL}^-(n)) \hookrightarrow \mathrm{GL}^-(\underline{n})$.

Proof: Aside from 7S)c), 1S)–9S) were proved in [156] in the square-matrix case. Here, we generalize 1S)–4S), 6S) to the rectangular-matrix case, and the arguments are similar. 3S), in particular, follows from the commutative diagram (A.10).

7S)Sa), 7S)Sb) were originally proved in [156] and are well-understood, but because prior works on the symmetric Kronecker product define it only as an operation on square matrices, they have missed that $x_i \underline{\otimes} x_j$ constitute the eigenvectors of $A \underline{\otimes} B$ – an important and intuitive result paralleling that of the usual Kronecker product (cf. Proposition A.3.2 7))). 7S)Sb) was proved in [156] in the case of commuting square matrices $A, B \in \mathbb{S}^n$, but simultaneous diagonalizability is the key property enabling this result. Underpinning the arguments in 7S)Sa) and 7S)Sb) is 7S)Sc), which we prove here because it will be illustrative subsequently. With all terms as in the hypotheses of 7S)Sc), we first note the subtlety that $x, y \neq 0$ implies $x \underline{\otimes} y \neq 0$, by 10S) (proven below, independently of this result). Next, applying the *now-generalized* mixed product identity 6S), we have

$$\begin{aligned}(A \underline{\otimes} B)(x \underline{\otimes} y) &= \frac{1}{2}\left(Ax \underline{\otimes} By + Ay \underline{\otimes} Bx\right) \\ &= \frac{1}{2}(\lambda_1\mu_2 + \lambda_2\mu_1)\, x \underline{\otimes} y. \end{aligned} \tag{A.14}$$

The authors are unaware of 11S)–14S) being proved previously. 11S) follows from 7S)Sa). For 10S), 12S)–14S), we employ the indexing maps r (A.2) and c (A.3), which together with the mixed product identity 6S) yield the symmetric Kronecker product indexing identity (A.15). Straightforward application of (A.15) yields 10S), 12S), and 13S).

$$(A \underline{\otimes} B)_{i,j} = \begin{cases} a_{r_m(i),r_n(j)} b_{r_m(i),r_n(j)}, & r_m(i) = c_m(i),\ r_n(j) = c_n(j), \\ \frac{\sqrt{2}}{2}\left(a_{r_m(i),r_n(j)} b_{r_m(i),c_n(j)} + a_{r_m(i),c_n(j)} b_{r_m(i),r_n(j)}\right), & r_m(i) = c_m(i),\ r_n(j) < c_n(j), \\ \frac{\sqrt{2}}{2}\left(a_{r_m(i),r_n(j)} b_{c_m(i),r_n(j)} + a_{c_m(i),r_n(j)} b_{r_m(i),r_n(j)}\right), & r_m(i) < c_m(i),\ r_n(j) = c_n(j), \\ \frac{1}{2}\big(a_{r_m(i),r_n(j)} b_{c_m(i),c_n(j)} + a_{r_m(i),c_n(j)} b_{c_m(i),r_n(j)} & \\ \quad + a_{c_m(i),r_n(j)} b_{r_m(i),c_n(j)} + a_{c_m(i),c_n(j)} b_{r_m(i),r_n(j)}\big), & r_m(i) < c_m(i),\ r_n(j) < c_n(j), \end{cases}$$

$$i = 1, \ldots, \underline{m},\ j = 1, \ldots, \underline{n}. \tag{A.15}$$

Finally, 14S) follows from 12S) and 13S) in an analogous argument to the one presented in the proof of Proposition A.3.2 14)). ■

Remark A.3.1 (On the Eigenstructure of the Symmetric Kronecker Product). Equation (A.14) elucidates a key issue surrounding the eigenstructure of the symmetric Kronecker product: In general, given eigenvectors $Ax = \lambda_1 x$, $By = \mu_2 y$ of $A, B \in \mathbb{R}^{n\times n}$, the first term in the expansion $Ax \underline{\otimes} By = \lambda_1\mu_2\, x \underline{\otimes} y$ always factors in the desired fashion. Yet, the second term $Ay \underline{\otimes} Bx = Bx \underline{\otimes} Ay$ need not be a scalar multiple of $x \underline{\otimes} y$, since x is not an eigenvector of B and y is not an eigenvector of A, in general. Naturally, this makes the eigenstructure of the symmetric Kronecker product a significantly more complicated object of study than that of the usual Kronecker product, cf. [156, 158, 159].

As a note, the eigenstructure results of Proposition 7S) require the symmetric Kronecker product as a map on complex matrices (specifically, when eigenvectors are

complex-valued). As is the case with the standard Kronecker product, the necessary results may developed for the complex case. Following the practice of previous works [155–159], we avoid carrying out this process explicitly here to maintain scope.

Remark A.3.2. For a counterexample illustrating the point of Proposition A.3.3 5S), consider $A = \texttt{diag}(1, -1)$, $B = I_2 \in \mathrm{GL}(2)$. Then $A \underline{\otimes} B = \frac{1}{2} A \underline{\oplus} A = \texttt{diag}(1, 0, -1) \notin \mathrm{GL}(\underline{2})$. The key here is that A possesses eigenvalues $\sigma(A) = \{\pm 1\}$ symmetric with respect to the origin (cf. Proposition A.3.6). Note further on this point that $\sigma(A \underline{\oplus} A) = \{1 + 1, 1 - 1, -1 - 1\}$.

Remark A.3.3. The strengthened hypotheses for the converse direction of Proposition A.3.3 12S) in relation to Proposition A.3.2 12)) are necessary. Indeed, in the case $n = 2$, consider $A = e_2 e_1^T \in \mathbb{R}^{2\times 2}$. Then $A, A^T \neq 0$, and neither of these matrices are diagonal, yet $A \underline{\otimes} A^T = \texttt{diag}(0, \frac{1}{2}, 0)$ is diagonal. Note that A, A^T are zero on their diagonals.

Remark A.3.4 (Lie Group Homomorphisms Φ, $\underline{\Phi}$). The Lie Group homomorphism $\underline{\Phi}$ in Proposition A.3.3 14S) is relevant to the MEE framework developed in Section 3.5. To maintain subsequent emphasis on the symmetric Kronecker product algebra, we will after this section avoid labeling this map explicitly. We have included construction of its Kronecker product counterpart Φ in Proposition A.3.2 14)) for completeness. By virtue of the bilinearity of the (symmetric) Kronecker product, these homomorphisms are homogeneous of degree two. For intuition, consider the case $n = 1$. Then $\underline{n} = 1$, $r_1 \equiv 1$ (A.2), $c_1 \equiv 1$ (A.3), $\{E_i\}_{i=1}^{\underline{1}} = \{1\}$ (A.4), and $W_1 = 1$ (A.5). In all, $\otimes = \underline{\otimes}$ are both given by scalar multiplication, and $\Phi(a) = \underline{\Phi}(a) = a^2$ (we will thus focus on $\underline{\Phi}$). Here, $\underline{\Phi} : \mathrm{GL}(1) = \mathbb{R}\backslash\{0\} \to \mathrm{GL}^+(1) = (0, \infty)$. This is a group homomorphism: $\underline{\Phi}(ab) = abab = aabb = \underline{\Phi}(a)\underline{\Phi}(b)$, which is polynomial in the global coordinate on $\mathbb{R}\backslash\{0\} \hookrightarrow \mathbb{R}$ and on $(0, \infty) \hookrightarrow \mathbb{R}$, hence smooth. Note also that $\underline{\Phi}(a) = a^2 = 1$ if and only if $a \in \{\pm 1\}$. Finally, $\underline{\Phi}|_{\mathrm{GL}^+(1)} : \mathrm{GL}^+(1) = (0, \infty) \to \mathrm{GL}^+(\underline{1}) = (0, \infty)$ is a Lie group isomorphism onto its image $\underline{\Phi}|_{\mathrm{GL}^+(1)}((0, \infty)) = \underline{\Phi}((0, \infty)) = (0, \infty)$ (a connected subgroup of $\mathrm{GL}^+(\underline{1}) = (0, \infty)$); in particular, the map $a \mapsto a^2 : (0, \infty) \to (0, \infty)$ is a diffeomorphism.

In the above, $\underline{\Phi}|_{\mathrm{GL}^+(1)} : \mathrm{GL}^+(1) \to \mathrm{GL}^+(\underline{1})$ is a Lie group isomorphism in its own right. However, in the case $n > 1$, $\underline{\Phi}|_{\mathrm{GL}^+(n)} : \mathrm{GL}^+(n) \to \mathrm{GL}^+(\underline{n})$ is not a Lie group isomorphism. In the case n is even, it fails to be injective. Meanwhile, for all $n > 1$, $\underline{\Phi}$ fails to be onto $\mathrm{GL}^+(\underline{n})$. For otherwise $\underline{\Phi}$ would be a surjective map of constant rank, hence a submersion by the global rank theorem [161, Theorem 4.14]; i.e., $\mathrm{rank}(\underline{\Phi}) = \underline{n}$ – a contradiction of the fact that $\mathrm{rank}(\underline{\Phi}) \leq \min\{n, \underline{n}\} = n < \underline{n}$ always. A similar argument prevails for Φ.

Having discussed the (symmetric) Kronecker product, we now move on to the (symmetric) Kronecker sum. We first recall the spectral result in the standard case:

Proposition A.3.4. (Eigenstructure of The Kronecker Sum [150, Theorem 4.4.5]). For square matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{m\times m}$, if $\sigma(A) = \{\lambda_i \mid i = 1, \ldots, n\}$ and $\sigma(B) = \{\mu_j \mid j = 1, \ldots, m\}$, then $\sigma(A \oplus B) = \{\lambda_i + \mu_j \mid i = 1, \ldots, n, j = 1, \ldots, m\}$. Furthermore, if $x_i \in \mathbb{C}^n$, $y_j \in \mathbb{C}^m$ are eigenvectors corresponding to the eigenvalues λ_i of A and μ_j of B, respectively, then $x_i \otimes y_j$ is an eigenvector corresponding to the eigenvalue $\lambda_i + \mu_j$ of $A \oplus B$.

While the eigenstructure of the Kronecker sum is quite intuitive, the eigenstructure of the symmetric Kronecker sum is more complicated, owing to the complications inherited from the symmetric Kronecker product (cf. Remark A.3.1). In the simultaneously-diagonalizable case, the result of Proposition 7S)b), developed originally in [156], may be applied to the symmetric Kronecker sum as follows:

Proposition A.3.5 (Eigenstructure of The Symmetric Kronecker Sum (Simultaneously Diagonalizable Case)). Suppose that $A, B \in \mathbb{R}^{n\times n}$ are simultaneously diagonalizable with common basis of eigenvectors $\{x_i\}_{i=1}^n$. If $\sigma(A) = \{\lambda_i \mid i = 1, \ldots, n\}$ and $\sigma(B) = \{\mu_j \mid j = 1, \ldots, n\}$ are the eigenvalues of A and B corresponding to the respective eigenvectors $\{x_i\}_{i=1}^n$, then
$\sigma(A \underline{\oplus} B) = \left\{\frac{1}{2}(\lambda_i + \mu_i + \lambda_j + \mu_j) \mid 1 \leq i \leq j \leq n\right\}$. Furthermore, $x_i \underline{\otimes} x_j$ is an eigenvector corresponding to the eigenvalue $\frac{1}{2}(\lambda_i + \mu_i + \lambda_j + \mu_j)$ of $A \underline{\oplus} B$.

For our purposes, Proposition A.3.5 is too restrictive. The following property will be useful shortly:

Lemma A.3.1 (Partial Eigenstructure of The Symmetric Kronecker Sum). Suppose that $A, B \in \mathbb{R}^{n\times n}$ share two eigenvectors $x, y \in \mathbb{C}^n$. If $Ax = \lambda_1 x$, $Bx = \mu_1 x$, $Ay = \lambda_2 y$, $By = \mu_2 y$, then $x \underline{\otimes} y$ is an eigenvector of $A \underline{\oplus} B$ corresponding to the eigenvalue $\frac{1}{2}(\lambda_1 + \mu_1 + \lambda_2 + \mu_2)$.

Proof: Follows from Proposition A.3.3 7S)Sc). ■

Lemma A.3.1 allows us to enumerate the eigenstructure of $A \underline{\oplus} A$, a special case relevant to ALEs.

Proposition A.3.6 (Eigenstructure of The Symmetric Kronecker Sum $A \underline{\oplus} A$). For a square matrix $A \in \mathbb{R}^{n\times n}$, if $\sigma(A) = \{\lambda_i \mid i = 1, \ldots, n\}$, then $\sigma(A \underline{\oplus} A) = \{\lambda_i + \lambda_j \mid 1 \leq i \leq j \leq n\}$. Furthermore, if $x_i, x_j \in \mathbb{C}^n$ are eigenvectors corresponding to the eigenvalues λ_i, λ_j of A, respectively, then $x_i \underline{\otimes} x_j$ is an eigenvector corresponding to the eigenvalue $\lambda_i + \lambda_j$ of $A \underline{\oplus} A$.

Proof: Follows from Lemma A.3.1. ■

Having discussed eigenstructure, we move on to the key exponentiation identity involving the (symmetric) Kronecker sum:

Proposition A.3.7 (Exponentiation of the Kronecker Sum [150]). Let $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{n\times n}$ be given.

1) $(A \otimes I)^k = A^k \otimes I$, and $(I \otimes B)^k = I \otimes B^k$, $k \geq 0$.

2) $\exp(A \oplus B) = \exp(A) \otimes \exp(B)$.

The analogue holds for the symmetric Kronecker sum in the case $A = B$:

Proposition A.3.8 (Exponentiation of the Symmetric Kronecker Sum). Let $A, B \in \mathbb{R}^{n\times n}$ be given.

1S) $(A \underline{\otimes} I)^k = (I \underline{\otimes} A)^k$ is given by the following binomial expansion

$$(A \underline{\otimes} I)^k = \frac{1}{2^k} \sum_{i=0}^{k} \binom{k}{i} A^{k-i} \underline{\otimes} A^i, \qquad k \geq 0. \tag{A.16}$$

2S) $\exp(A \underline{\oplus} A) = \exp(A) \underline{\otimes} \exp(A)$. However, in general $\exp(A \underline{\oplus} B) \neq \exp(A) \underline{\otimes} \exp(B)$.

Proof: Proving that (A.16) holds is a quick algebraic check following from the mixed product identity of Proposition A.3.3 6S). 2S) follows from (A.16) after examining the partial sums of $\exp(A \underline{\oplus} A)$ and $\exp(A) \underline{\otimes} \exp(A)$. ■

Remark A.3.5. For a counterexample illustrating the point of Proposition A.3.8 2S), consider the same matrices as in Remark A.3.2: $A = \texttt{diag}(1, -1)$, $B = I_2$. Then

$$\begin{aligned} \exp(A \underline{\oplus} B) &= \texttt{diag}(e^2, e, 1), \\ \exp(A) \underline{\otimes} \exp(B) &= \texttt{diag}\left(e^2, \frac{e^2+1}{2}, 1\right). \end{aligned} \tag{A.17}$$

A.4 Symmetric Kronecker Products in Algebraic Lyapunov Equations (ALEs)

As is well-known, the Kronecker product plays an important role in characterizing existence and uniqueness of solutions to ALEs [150]. We illustrate in this section that the symmetric Kronecker product algebra developed above also provides this same characterization under symmetric conditions. Substantively, the algebra is structurally identical to the standard case.

Definition A.4.1 (Algebraic Lyapunov Equation (ALE)). Given $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, consider the following algebraic Lyapunov equation (ALE)

$$A^T X + XA + B = 0. \tag{A.18}$$

Proposition A.4.1 (ALE Existence and Uniqueness of Solutions). Let $\sigma(A) = \{\lambda_i \mid i = 1, \ldots, n\}$. There exists a unique solution $X \in \mathbb{R}^{n \times m}$ of the ALE (A.18) if and only if $\lambda_i + \lambda_j \neq 0$ for all $1 \leq i, j \leq n$.

Proof: This proof is quite standard; see, e.g., [150, 160]. However, we include it here to illustrate structural parallels to the analogous results developed shortly for the symmetric Kronecker product. Applying the identities in Proposition A.3.2, we see that (A.18) is equivalent to

$$\operatorname{vec}(A^T X + XA) = (A \oplus A)^T \operatorname{vec}(X) = -\operatorname{vec}(B). \tag{A.19}$$

Thus, the ALE (A.18) has a unique solution if and only if $(A \oplus A)^T \in \mathrm{GL}(n^2)$. Applying Proposition A.3.4, $\sigma((A \oplus A)^T) = \{\lambda_i + \lambda_j \mid i, j = 1, \ldots, n\}$, from which the result follows. ■

Proposition A.4.2 (ALE Existence and Uniqueness of Solutions: Stable Systems [136, Proposition 5.2.1]). Suppose $A \in \mathbb{R}^{n \times n}$ is Hurwitz, and $Q \in \mathbb{S}^n$. Consider the ALE

$$A^T P + PA + Q = 0. \tag{A.20}$$

1) The unique solution is the symmetric matrix

$$P = \int_0^\infty e^{A^T t} Q e^{At} dt. \tag{A.21}$$

2) If Q is positive (semi)definite, then P is positive (semi)definite.

3) If Q is positive semidefinite, then P is positive definite if and only if $(Q^{1/2}, A)$ is detectable.

Remark A.4.1 (Symmetric Kronecker Algebra of the ALE (A.20)). Consider the ALE (A.20). Applying Proposition A.4.2, we know $P \in \mathbb{S}^n$. We may then apply the symmetric Kronecker product algebra in Proposition A.3.3, yielding

$$\operatorname{svec}(A^T P + PA) = -\operatorname{svec}(Q). \tag{A.22}$$

Now, applying Proposition A.3.3 3S), the left-hand-side of (A.22) becomes,

$$2\operatorname{svec}(\pi(PA)) = 2(A^T \underline{\otimes} I)\operatorname{svec}(P) = (A \underline{\oplus} A)^T \operatorname{svec}(P). \tag{A.23}$$

Altogether, the ALE (A.20) is equivalent to the following:

$$\operatorname{svec}(A^T P + PA) = (A \underline{\oplus} A)^T \operatorname{svec}(P) = -\operatorname{svec}(Q). \tag{A.24}$$

The reader is encouraged to compare Equations (A.19) and (A.24), which precisely motivates our definition of the symmetric Kronecker sum $\underline{\oplus}$ as the natural analogue to the Kronecker sum $\oplus$. The structural parallels extend further: Note by Proposition A.3.6 that $\sigma((A \underline{\oplus} A)^T) = \{\lambda_i + \lambda_j \mid 1 \leq i \leq j \leq n\}$. Thus, in the case $Q \in \mathbb{S}^n$, the symmetric Kronecker sum may be used to characterize existence and uniqueness of solutions to the ALE (A.20) in an entirely similar argument to the one used in the proof of Proposition A.4.1. Here, the square-symmetric nature of the matrix $Q \in \mathbb{S}^n$ has enabled an effective reduction is dimensionality of the problem from n^2 to $\underline{n}$.

APPENDIX B

HSV MODEL AND DECENTRALIZED DESIGN FRAMEWORK

B.1 HSV Model

The HSV model used in this study is the standard Wang and Stengel model developed in [5, 6] based on NASA Langley's winged-cone tabular aeropropulsive data [121]. This model has served as a standard testbed for HSV control development and has since been used in seminal classically-based works such as [162, 163], and simplified variants of it have been used in state-of-the-art RL-based control works such as [123–125]. The model presented here is identical to the original [5, 6], with two exceptions: First, we add an elevator-lift increment coefficient C_{L,δ_E} (B.5) from data in [121] to capture nonminimum phase behavior. Second, we remove angle of attack (AOA) dependence from the thrust coefficient C_T (B.11), as thrust coefficient AOA dependencies were considered negligible in the original propulsion model [121, pp. 12], and it was removed in subsequent studies [162, 163]. Consider the following HSV longitudinal model

$$\begin{aligned}
\dot{V} &= \frac{T\cos\alpha - D}{m} - \frac{\mu\sin\gamma}{r^2},\\
\dot{\gamma} &= \frac{L + T\sin\alpha}{mV} - \frac{(\mu - V^2 r)\cos\gamma}{Vr^2},\\
\dot{\theta} &= q,\\
\dot{q} &= \frac{\mathcal{M}}{I_{yy}},\\
\dot{h} &= V\sin\gamma,
\end{aligned} \tag{B.1}$$

where V is the vehicle airspeed, γ is the flightpath angle (FPA), α is the angle of attack (AOA), $\theta \triangleq \alpha + \gamma$ is the pitch attitude, q is the pitch rate, and h is the vehicle altitude. Here $r(h) = h + R_E$ is the total distance from the earth's center to the vehicle, $R_E = 20,903,500$ ft is the radius of the earth, and $\mu = Gm_E = 1.39 \times 10^{16}$ ft^3/s^2, where G is Newton's gravitational constant and m_E is the mass of the earth. $L, D, T, \mathcal{M}$ are the lift, drag, thrust, and pitching moment, respectively, and are given by

$$L = \frac{1}{2}\rho V^2 S C_L, \quad D = \frac{1}{2}\rho V^2 S C_D, \quad T = \frac{1}{2}\rho V^2 S C_T, \quad \mathcal{M} = \frac{1}{2}\rho V^2 S \bar{c} C_{\mathcal{M}}, \tag{B.2}$$

where ρ is the local air density, $S = 3603$ ft^2 is the wing planform area, and $\bar{c} = 80$ ft is the mean aerodynamic chord of the wing. Air density ρ and speed of sound a are modeled as functions of altitude h by $\rho = 0.00238 e^{-\frac{h}{24,000}}$, $a = 8.99 \times 10^{-9} h^2 - 9.16 \times 10^{-4} h + 996$, and Mach number $M \triangleq \frac{V}{a}$. C_L, C_D, $C_{\mathcal{M}}$, and C_T are given by

$$C_L = C_{L,\alpha} + C_{L,\delta_E}, \tag{B.3}$$

$$C_{L,\alpha} = \nu_L\, \alpha \left(0.493 + \frac{1.91}{M}\right), \tag{B.4}$$

$$C_{L,\delta_E} = \left(-0.2356\alpha^2 - 0.004518\alpha - 0.02913\right)\delta_E, \tag{B.5}$$

$$C_D = \nu_D\, 0.0082 \left(171\alpha^2 + 1.15\alpha + 1\right)\left(0.0012M^2 - 0.054M + 1\right), \tag{B.6}$$

$$C_{\mathcal{M}} = C_{\mathcal{M},\alpha} + C_{\mathcal{M},q} + C_{\mathcal{M},\delta_E}, \tag{B.7}$$

$$C_{\mathcal{M},\alpha} = \nu_{\mathcal{M}}\, 10^{-4} \left(0.06 - e^{-\frac{M}{3}}\right) \left(-6565\alpha^2 + 6875\alpha + 1\right), \tag{B.8}$$

$$C_{\mathcal{M},q} = \left(\frac{q\bar{c}}{2V}\right) (-0.025M + 1.37) \left(-6.83\alpha^2 + 0.303\alpha - 0.23\right), \tag{B.9}$$

$$C_{\mathcal{M},\delta_E} = 0.0292(\delta_E - \alpha), \tag{B.10}$$

$$C_T = \begin{cases} 0.0105 \left(1 + \frac{17}{M}\right) (1 + 0.15)\delta_T, & \delta_T < 1 \\ 0.0105 \left(1 + \frac{17}{M}\right) (1 + 0.15\delta_T), & \delta_T \geq 1, \end{cases} \tag{B.11}$$

where δ_E is the elevator deflection, δ_T is the throttle setting, and ν_L, ν_D, $\nu_{\mathcal{M}} \in \mathbb{R}$ are unknown parameters (nominally 1) representing modeling error in the basic lift increment coefficient $C_{L,\alpha}$ (B.4), drag coefficient C_D (B.6), and basic pitch moment coefficient $C_{\mathcal{M},\alpha}$ (B.8), respectively. The system (B.1) is fifth-order, with states $x = [V,\, \gamma,\, \theta,\, q,\, h]^\intercal$. The controls are $u = [\delta_T,\, \delta_E]^\intercal$, and we examine the outputs $y = [V,\, \gamma]^\intercal$. As in [5, 6], we study a steady level flight cruise condition $q_e = 0, \gamma_e = 0^\circ$, at $M_e = 15$, $h_e = 110,000$ ft, which corresponds to an equilibrium airspeed $V_e = 15,060$ ft/s. At this flight condition, the vehicle is trimmed at $\alpha_e = 1.7704^\circ$ by the controls $\delta_{T,e} = 0.1756$ ($T_e = 4.4966 \times 10^4$ lb), $\delta_{E,e} = -0.3947^\circ$.

HSV Dynamical Challenges: Instability, Nonminimum Phase. The HSV model studied here encompasses a variety of dynamical challenges facing real-world flight control designers. Firstly, the HSV is open-loop unstable. Linearization of the model about the equilibrium flight condition (x_e, u_e) has open-loop eigenvalues at $s = -0.8291, 0.7165$ (short-period modes), $s = -0.00001 \pm 0.0276j$ (phugoid modes), and $s = 0.0005$ (altitude mode). The dominant unstable short-period right half plane pole (RHPP) at $s = 0.7165$ is associated with the vehicle pitch-up instability (long vehicle forebody, aftward-set center of mass). As is commonplace with tail-controlled aircraft, the elevator-FPA map is nonminimum phase [164]. The linearized plant has transmission zeros at $s = 8.3938, -8.4620$, the right half plane zero (RHPZ) at $s = 8.3938$ being attributable to the elevator-FPA map (negative lift increment in response to pitch-up elevator deflections).

B.2 Analysis of Static and Dynamic Properties versus Modeling Error

Static Properties. Fig. B.1 plots the trim controls u_e (i.e., trim throttle setting $\delta_{T,e}$ and trim elevator setting $\delta_{E,e}$) for 0%–25% modeling errors in lift/drag ν_L/ν_D, lift/pitch moment $\nu_L/\nu_{\mathcal{M}}$, and drag/pitch moment $\nu_D/\nu_{\mathcal{M}}$.

Intuitively, the top row of Fig. B.1 shows that the trim throttle setting $\delta_{T,e}$ is most heavily influenced by increasing drag coefficient C_D caused by increases in the modeling error parameter ν_D $1 \to 1.25$, which generally causes increases from ≈ 0.18 nominally to ≈ 0.24 at 25% modeling error. Trim throttle $\delta_{T,e}$ varies the second most to decreasing lift coefficient C_L caused by decreases in the modeling error parameter ν_L $1 \to 0.75$ (qualitatively: Decreased lift efficiency $\Rightarrow$ increased trim AOA required $\Rightarrow$ increased drag at trim $\Rightarrow$ increased trim throttle $\delta_{T,e}$ required). Finally, trim throttle varies little with modeling errors in pitching moment coefficient $\nu_{\mathcal{M}}$, an intuitive result.

Meanwhile, the bottom row of Fig. B.1 shows that the trim elevator setting

$\delta_{E,e}$ is most heavily influenced by increasing pitch moment coefficient $C_{\mathcal{M}}$ caused by increases in the modeling error parameter $\nu_{\mathcal{M}}$ $1 \rightarrow 1.25$, which generally causes decreases from $\approx -0.4°$ nominally to $\approx -1.0°$ at 25% modeling error. Trim elevator $\delta_{E,e}$ varies the second most to decreasing lift coefficient C_L caused by decreases in the modeling error ν_L $1 \rightarrow 0.75$ (qualitatively: decreased lift efficiency $\Rightarrow$ increased trim AOA required $\Rightarrow$ increased pitch-up moment at trim $\Rightarrow$ more negative trim elevator deflection $\delta_{T,e}$ required). Finally, trim throttle varies little with modeling errors in drag moment coefficient ν_D, an intuitive result.

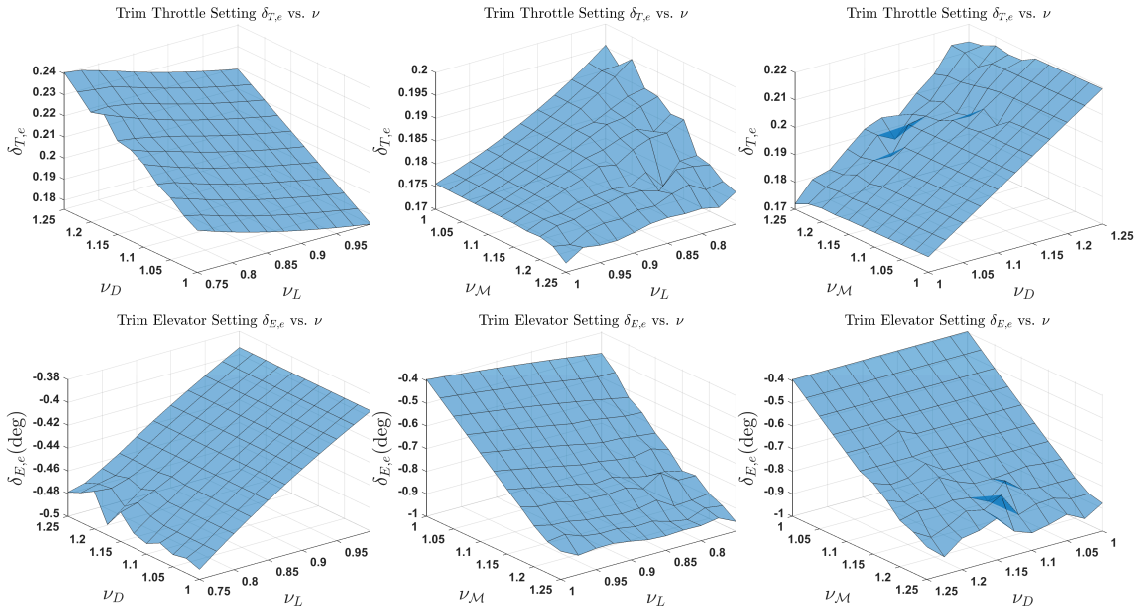


Figure B.1: Trim Controls u_e Versus Modeling Error ν. First Row: Trim Throttle Setting $\delta_{T,e}$ Versus ν. Second Row: Trim Elevator Setting $\delta_{E,e}$ Versus ν. First Column: Lift/Drag ν_L/ν_D Sweep. Second Column: Lift/Pitch Moment $\nu_L/\nu_{\mathcal{M}}$ Sweep. Third Column: Drag/Pitch Moment $\nu_D/\nu_{\mathcal{M}}$ Sweep.

Dynamic Properties. Fig. B.2 plots dynamic properties of the HSV model for the same sweeps of modeling error described in Fig. B.1, including the HSV right half plane pole (RHPP) location, right half plane zero (RHPZ) location, and the RHPZ/RHPP ratio (or simply the Z/P ratio). As a note, in this study we specifically chose the direction of the respective perturbation (i.e., $\nu > 1$ or $\nu < 1$) as the one which decreases the system Z/P ratio, thereby imposing a more difficult control problem on the proposed method.

As can be seen from the top row of Fig. B.2, the RHPP location is most heavily dependent on increasing pitch moment coefficient $C_{\mathcal{M}}$ caused by increases in the modeling error parameter $\nu_{\mathcal{M}}$ $1 \rightarrow 1.25$, which almost doubles the speed of the instability from ≈ 0.7 nominally to ≈ 1.2 at 25% modeling error. This result is intuitive, as an increased pitch moment coefficient will lead to larger pitching moments and hence a faster pitch-up instability. The RHPP location varies the second most to decreasing lift coefficient C_L caused by decreases in the modeling error ν_L $1 \rightarrow 0.75$ (qualita-

tively: decreased lift efficiency ⇒ increased trim AOA required ⇒ increased pitch-up moment sensitivity at trim ⇒ faster instability). Finally, the RHPP varies little with modeling errors in drag moment coefficient ν_D, an intuitive result. Meanwhile, the middle row of Fig. B.2 shows that the RHPZ location is most heavily influenced by modeling errors in lift coefficient ν_L (decreasing from ≈ 8.5 nominally to ≈ 7.25 at 25% modeling error) and by comparison varies relatively little with respect to modeling errors in drag coefficient ν_D and pitch moment coefficient $\nu_{\mathcal{M}}$; and intuitive result since the nonminimum phase behavior of the tail-controlled HSV model occurs in the elevator-FPA map.

Bringing this together, the bottom row of Fig. B.2 shows that the Z/P ratio is most heavily impacted by the near-doubling of the RHPP location due to increasing pitch moment modeling error $\nu_{\mathcal{M}}$ $1 \rightarrow 1.25$ (which causes a near-halving of the Z/P ratio from ≈ 12 nominally to a much more formidable ≈ 6 at 25% modeling error). Comparatively speaking, the variations in the Z/P ratio caused by lift coefficient modeling error are less pronounced (decreasing from ≈ 11.8 nominally to ≈ 10.6 at 25% modeling error). This decrease is caused primarily by the decreasing RHPZ location, which was observed to be less severe than the RHPP variations due to pitch moment error $\nu_{\mathcal{M}}$. Finally, since the RHPP and RHPZ are both little affected by drag moment coefficient modeling error ν_D, the Z/P ratio varies little with respect to this error, as well.

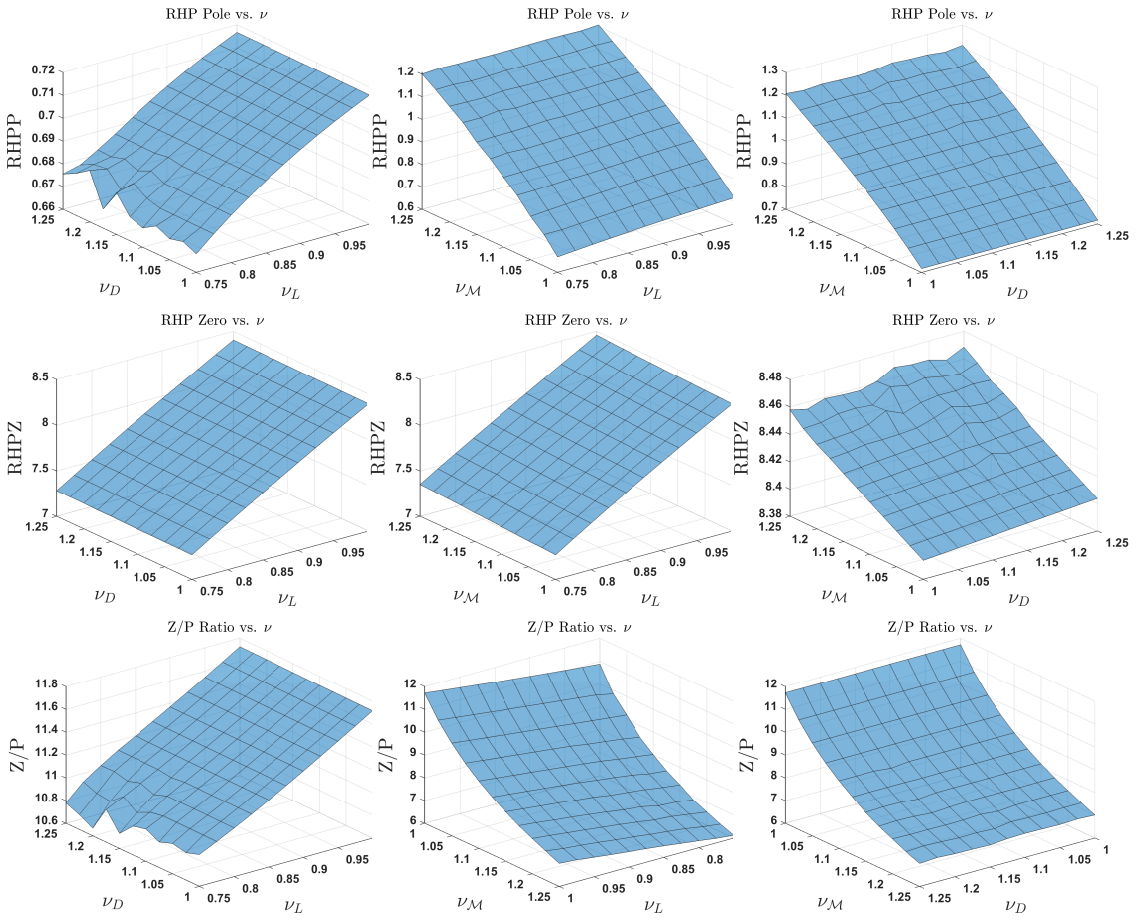


Figure B.2: Dynamic Properties Versus Modeling Error ν. First Row: RHPP Versus ν. Second Row: RHPZ Versus ν. Third Row: Z/P Ratio Versus ν. First Column: Lift/Drag ν_L/ν_D Sweep. Second Column: Lift/Pitch Moment $\nu_L/\nu_{\mathcal{M}}$ Sweep. Third Column: Drag/Pitch Moment $\nu_D/\nu_{\mathcal{M}}$ Sweep.

B.3 HSV Decentralized Hierarchical Inner-Outer Loop Control Framework

This work implements a decentralized design methodology structurally-identical to the framework [132] developed for HSVs and extensively tested on HSVs. Thus, our RL-based framework inherits significant advantages from classically-based performance guarantees. Here, controllers are designed separately for the weakly-coupled velocity subsystem (associated with the airspeed V and throttle control δ_T) and rotational subsystem (associated with the FPA γ, attitude θ, q, and elevator control δ_E). As in [132], for controllability reasons we do not feed back altitude h in the control design, though altitude is still included in the nonlinear simulation. In order to achieve zero steady-state error to step reference commands, we augment the plant at the output with the integrator bank $z = \int y \, d\tau = [z_V, \, z_\gamma]^\intercal = \left[\int V \, d\tau, \, \int \gamma \, d\tau\right]^\intercal$. For dEIRL, the state/control vectors are thus partitioned as $x_1 = [z_V, \, V]^\intercal$, $u_1 = \delta_T$ ($n_1 = 2$, $m_1 = 1$) and $x_2 = [z_\gamma, \, \gamma, \, \theta, \, q]^\intercal$, $u_2 = \delta_E$ ($n_2 = 4$, $m_2 = 1$). Applying the LQ

servo design framework [11, 136] to each of the loops yields an LQ-optimal decentralized controller $K = \texttt{diag}(K_1^*, K_2^*)$. We have depicted the decentralized hierarchical feedback structure [132] in Fig. B.3. Here, $x_r = [\theta,\, q]^T$ comprises the inner-loop feedback states, and the inner-loop controller K_{in} and outer-loop controller K_{out} are given by

$$K_{in}(s) = \begin{bmatrix} 0 & 0 \\ g_i z_i & g_i \end{bmatrix}, \qquad K_{out}(s) = \begin{bmatrix} K_V(s) & 0 \\ 0 & K_\gamma(s) \end{bmatrix} = \begin{bmatrix} \frac{g_1(s+z_1)}{s} & 0 \\ 0 & \frac{g_2(s+z_2)}{s} \end{bmatrix}. \tag{B.12}$$

This results in the following hierarchical control framework [132]:

- **Velocity Loop $j = 1$: Single-Loop PI Control.** Control of the velocity subsystem will consist of a single PI controller K_V (B.12). The velocity loop will be lower-bandwidth due to the natural low-bandwidth nature of the velocity loop $j = 1$.
- **Flightpath Loop $j = 2$: Hierarchical PD-Inner (Attitude), PI-Outer (FPA) Loop Control.** An inner-loop PD controller K_{in} (B.12) will be used for the pitch subsystem $x_r = [\theta,\, q]^T$. Fundamentally, this controller takes advantage of 1) the naturally high bandwidth of the elevator-pitch map, and 2) its minimum phase dynamics in order to provide sufficient closed-loop bandwidth to stabilize the natural pitch-up instability. The high bandwidth of the pitch inner loop will also facilitate design of an outer-loop PI FPA controller K_γ (B.12). Having stabilized the high-bandwidth inner pitch loop, this outer FPA loop can be sufficiently low-bandwidth to avoid excitation of the nonminimum phase elevator-FPA dynamics.

As in [132], reference command pre-filters are also introduced: $W_1 = \frac{z_1}{s+z_1}$ for velocity commands, $W_2 = \frac{z_2}{s+z_2}$ for FPA commands (not pictured in Fig. B.3). After simple block diagram algebra, it is immediate that the dEIRL control structure $K = \texttt{diag}(K_1^*, K_2^*)$ is identical to (B.12) with the identification $K_1^* = \begin{bmatrix} g_1 z_1 & g_1 \end{bmatrix}$, $K_2^* = \begin{bmatrix} g_2 z_2 & g_2 & g_i z_i & g_i \end{bmatrix}$. It is thus these optimal LQ controller parameters which the proposed dEIRL method will learn online.

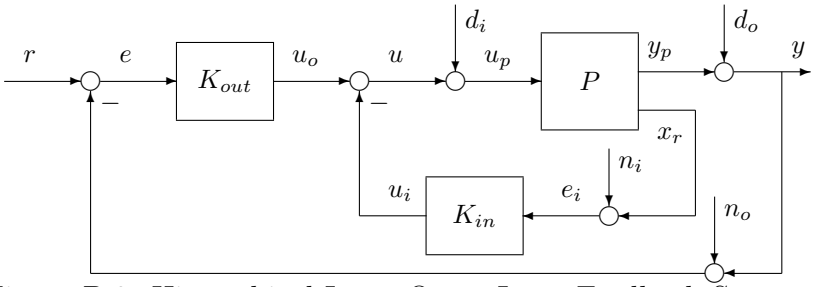


Figure B.3: Hierarchical Inner-Outer Loop Feedback Structure.

Associated with the feedback system in Fig. B.3, we define the following closed-loop maps. The sensitivity at the error signal is defined as $S_e \triangleq T_{r\to e}$, the complementary sensitivity as $T_e \triangleq T_{r\to y}$. The sensitivity at the control signal (plant input) is defined as $S_u \triangleq T_{d_i\to u_p}$, the complementary sensitivity as $T_u \triangleq T_{d_i\to y}$.

Specification B.3.1 (Closed-Loop Design Specifications). A design is termed "acceptable" when it meets the following:

- 0% steady-state error to step reference commands r.
- 0% steady-state error to step input disturbances d_i.
- Velocity: 1% settling time $t_{s,V,1\%} \leq 75$ s, overshoot $M_{p,V} \leq 5\%$ throttle $\delta_T \leq 0.4$ for $r_V \leq 100$ ft/s.
- FPA: 1% settling time $t_{s,\gamma,1\%} \leq 10$ s, overshoot $M_{p,\gamma} \leq 5\%$, elevator $|\delta_E| \leq 5^\circ$ for $r_\gamma \leq 1$ deg.
- Peak Closed-Loop Maps: $\|S_e\|_{\mathcal{H}^\infty}$, $\|T_e\|_{\mathcal{H}^\infty}$, $\|S_u\|_{\mathcal{H}^\infty}$, $\|T_u\|_{\mathcal{H}^\infty} \leq 6$ dB.

APPENDIX C

PENDULUM MODEL AND DESIGN FRAMEWORK

C.1 Pendulum Model

We consider the identical pendulum model used in the cFVI evaluations [7, 8] for this work, which has the following equations of motion

$$\begin{aligned} \dot{\theta} &= \omega, \\ \dot{\omega} &= \frac{mgL}{2I}\sin\theta + \frac{1}{I}\tau, \end{aligned} \tag{C.1}$$

where θ is the pendulum angle (measured zero pointing upward, positive counter-clockwise), ω is the pendulum angular velocity, and τ is the torque applied to the pendulum base. The numerical values of all model constants are chosen identical to the cFVI evaluations [7, 8] and are available in Table C.1.

The system (C.1) is second-order, with states $x = [\theta, \omega]^T$ and control $u = \tau$. We examine the output $y = \theta$; i.e., control of the pendulum angle θ. We examine the upright pendulum equilibrium $x_e = [\theta_e, \omega_e]^T = [0 \text{ rad}, 0 \text{ rad/s}]^T$. At this upright condition, the pendulum is trimmed by the control $\tau_e = 0$ N-m.

Table C.1: Pendulum Model Parameters

Definition	Symbol	Value
Pendulum length	L	$L_0 = 1$ m (nominal)
Pendulum mass	m	1 kg
Gravitational field constant	g	9.81 m/s^2
Pendulum moment of inertia	I	$\frac{1}{3}mL^2$

Remark C.1.1 (Pendulum Dynamical Structure). The pendulum length L is a central physical parameter in the dynamics (C.1). Firstly, increasing the pendulum length L increases its rotational inertia I in the square of the length; resultantly, (C.1) shows that the torque τ required to achieve the same angular acceleration increases with the square of the pendulum length L. The pendulum length L also determines the severity of the upright pendulum instability. Linearization of (C.1) about the upright equilibrium yields

$$\begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{mgL}{2I} & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{I} \end{bmatrix} \tau. \tag{C.2}$$

Examination the linearization (C.2) shows that the system has modes

$$s = \pm\sqrt{\tfrac{3}{2}}\sqrt{\tfrac{g}{L}}. \tag{C.3}$$

The real, imaginary-axis-symmetric pair of poles (C.3) is a common feature of inverted pendulum systems. We notice that the bandwidth of these modes are inversely proportional to the square root of the pendulum length L; i.e., a shorter pendulum increases the instability and system bandwidth. A longer pendulum reduces the instability, but it also reduces system bandwidth. Combined with the above discussion

of the reduced control effectiveness associated with increased pendulum length, these first-principles analyses have significant implications for practical robotics design. Generally speaking, taller robotic systems with inverted pendulum instabilities will require significantly more actuator effort to achieve control objectives than shorter systems.

In the studies conducted in this work, we will focus on how modeling errors in the pendulum length L affect the pendulum dynamics and learning performance. Specifically, we study modeling errors of the form

$$L = \nu\, L_0, \tag{C.4}$$

where $L_0 \in \mathbb{R}$ is a nominal value of the pendulum length, and $\nu \in \mathbb{R}$ is the modeling error parameter (nominally 1). As $\nu > 1$ increases, $L > L_0$ increases, and our prior discussion shows that the system becomes more sluggish and requires greater control effort. Table C.2 shows the inverted pendulum instability and control effectiveness constant $\frac{1}{I}$ as a function of the modeling error ν (C.4). As predicted in (C.3), the system instability reduces with increasing pendulum length. Control effectiveness is highly sensitive to changes in pendulum length, decreasing by 17% for a 10% modeling error $\nu = 1.1$ and by 36% for a 25% modeling error $\nu = 1.25$. Increases in the pendulum length will thus result in degraded closed-loop performance.

Table C.2: Pendulum Instability and Control Effectiveness Versus Modeling Error Parameter ν (C.4)

ν (C.4)	Unstable Mode Location (C.3)	Control Effectiveness $\frac{1}{I}$
1 (nom)	3.8360	3
1.1	3.6575	2.4793
1.25	3.4310	1.9200

C.2 Pendulum Control Framework

The pendulum system (C.1) is fundamentally a single-loop system $j = 1$. Thus, we do not employ the multi-loop decentralization techniques of dEIRL for this system. We do wish to highlight the great dynamical flexibility of dEIRL discussed in Section 3.1: dEIRL generalizes to any integer number of loops $j \in \mathbb{N}$, and this includes the single-loop case $j = 1$. The optimal LQ controller K_1^* is a function of the modeling error ν, so when necessary we will show explicit dependence by the notation $K_1^*(\nu)$. For the model parameters in Table C.1 and cost structure selections (5.19), the pendulum has the following optimal LQ controllers

$$K_1^*(1) = \begin{bmatrix} 10.0098 & 2.6217 \end{bmatrix}, \tag{C.5}$$

$$K_1^*(1.1) = \begin{bmatrix} 10.9733 & 3.0086 \end{bmatrix}, \tag{C.6}$$

$$K_1^*(1.25) = \begin{bmatrix} 12.4235 & 3.6251 \end{bmatrix}, \tag{C.7}$$

As can be seen, the optimal controller $K_1^*(\nu)$ is heavily dependent on the modeling error ν (C.4).

APPENDIX D

JET AIRCRAFT MODEL AND DECENTRALIZED DESIGN FRAMEWORK

D.1 Jet Aircraft Model

Consider the following T-tailed small jet airplane model [122, 165]

$$\begin{bmatrix} \dot{V} \\ \dot{\gamma} \\ \dot{q} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} -D_V & -g\cos\alpha_e & 0 & 0 \\ \frac{L_V}{V_e} & 0 & 0 & \frac{L_\alpha}{V_e} \\ 0 & 0 & M_q & M_\alpha \\ \frac{-L_V}{V_e} & 0 & 1 & \frac{-L_\alpha}{V_e} \end{bmatrix} \begin{bmatrix} V \\ \gamma \\ q \\ \alpha \end{bmatrix} + \begin{bmatrix} T_{\delta_T} & 0 \\ 0 & 0 \\ 0 & M_{\delta_E} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_T \\ \delta_E \end{bmatrix}, \quad \text{(D.1)}$$

where V is the vehicle airspeed, γ is the flightpath angle (FPA), q is the pitch rate, and α is the vehicle angle of attack (AOA). As is standard in aerospace circles, here a subscript denotes a partial derivative with respect to the particular variable (e.g., D_V denotes the dimensional aerodynamic derivative of drag D with respect to airspeed V). For definitions of the parameters and their numerical values, see Table D.1. This jet airplane model is a central example of the standard flight control text [122], and it was constructed from aerodynamic data obtained by full-scale wind tunnel tests conducted by NASA [165].

The jet (D.1) is fourth-order, with states $x = [V,\, \gamma,\, q,\, \alpha]^T$ and controls $u = [\delta_T,\, \delta_E]^T$. We examine a level steady flight condition $\gamma_e = 0, q_e = 0$ at a cruising airspeed $V_e = 100$ m/s and altitude $h_e = 1000$ m (Mach $M_e \approx 0.3$). At this flight condition, the vehicle is trimmed at an angle of attack $\alpha_e = 3.4006$ deg by the controls $u_e = [\delta_{T,e},\, \delta_{E,e}]^T = [0.2135,\, 0 \text{ deg}]^T$.

Table D.1: Jet Aircraft Model Parameters

Definition	Symbol	Value
Lift/AOA aero deriv	L_α	$L_{\alpha 0} = 127.9$ N/rad (nominal)
Lift/airspeed aero deriv	L_V	0.190 N/(m/s)
Drag/airspeed aero deriv	D_V	1.850 N/(m/s)
Moment/AOA aero deriv	M_α	-798.56 N-m/rad
Moment/pitch rate aero deriv	M_q	-127.94 N-m/(rad/s)
Thrust/throttle setting control deriv	T_{δ_T}	4.6645 N/-
Moment/elevator control deriv	M_{δ_E}	-9.069 N-m/rad
Gravitational field constant	g	9.81 m/s^2

Remark D.1.1 (Jet Aircraft Minimum Phase Behavior). We note in (D.1) that the $(2,2)$ element of the input gain matrix B is assumed zero; i.e., elevator deflections δ_E do not directly impact the FPA derivative $\dot{\gamma}$. As a result, the jet model (D.1) is minimum phase. However, in reality tail-controlled aircraft feature lift/elevator parasitic couplings in this location which cause them to be nonminimum phase [141, 166]. Nevertheless, the assumption made in the development of this model [122] is quite standard in modeling for flight control design [5, 6, 167].

Remark D.1.2 (Jet Aircraft Decentralized Dynamical Structure). The jet aircraft studied here is a multi-input system which naturally lends itself to a decentralized dynamical structure. The throttle δ_T is associated with the airspeed V in

the translational loop $j = 1$, and the elevator δ_E is associated with the FPA γ and attitude q, α in the rotational loop $j = 2$. Indeed, this decentralized structure is general to aviation systems [122], even high-performance hypersonic vehicles (HSVs) [5, 6, 132–135].

The lift/AOA derivative L_α determines the lift efficiency of the aircraft and is hence a central aerodynamic parameter in any aviation system. As with any aerodynamic modeling process, it is also subject to large modeling errors and sensitivity to changes in flight condition [122, 142]. Thus, in this work we study the effects of modeling error ν on the lift/AOA dimensional aerodynamic derivative as

$$L_\alpha = \nu \, L_{\alpha 0}, \tag{D.2}$$

where $L_{\alpha 0} \in \mathbb{R}$ is the nominal value of the lift/AOA aerodynamic derivative, and $\nu \in \mathbb{R}$ is the modeling error parameter (nominally 1). As $\nu < 1$ decreases, the vehicle exhibits decreased lift efficiency, leading to a more difficult control problem [122]. The jet aircraft (D.1) has the following characteristic equation and natural modes

$$\phi(s) = \left(s^2 + 2\zeta_{ph} s + \omega_{n_{ph}}\right)\left(s^2 + 2\zeta_{sp} s + \omega_{n_{sp}}\right), \tag{D.3}$$

$$s_{ph} = -\zeta_{ph}\omega_{n_{ph}} \pm j\omega_{n_{ph}}\sqrt{1-\zeta_{ph}^2}, \qquad s_{sp} = -\zeta_{sp}\omega_{n_{sp}} \pm j\omega_{n_{sp}}\sqrt{1-\zeta_{sp}^2}. \tag{D.4}$$

The first pair of modes s_{ph} (D.4) is the phugoid mode, associated with the translational dynamics via exchanges in kinetic and potential energy (i.e., with coupled oscillations between the airspeed V and FPA γ). They are generally slow and lightly damped. The second pair s_{sp} (D.4) is called the short-period mode and is associated with the rotational dynamics via exchanges between rotational energy and aeroelastic energy (i.e., with coupled oscillations between the pitch rate q and AOA α). If the vehicle is designed so the center of gravity (c.g.) lies forward the center of pressure (c.p.), they are generally fast, stable, and lightly damped (as is the case here). If the c.p. lies forward the c.g., they are generally real, imaginary-axis symmetric, one stable and the other unstable.

We have plotted these modes as a function of the modeling error paramer ν (D.2) in Table D.2. As can be seen, the damping of both modes decreases with decreased lift efficiency $\nu < 1$, and the short-period modes get closer to the imaginary axis; i.e., less stable.

Table D.2: Jet Aircraft Phugoid and Short-Period Modes Versus Modeling Error Parameter ν (D.2)

ν (D.2)	s_{ph}	ζ_{ph}	$\omega_{n_{ph}}$	s_{sp}	ζ_{sp}	$\omega_{n_{sp}}$
1 (nom)	$-0.00849 \pm j0.119$	0.0709	0.120	$-1.28 \pm j2.83$	0.412	3.10
0.9	$-0.00853 \pm j0.120$	0.0708	0.121	$-1.22 \pm j2.83$	0.395	3.08
0.75	$-0.00863 \pm j0.120$	0.0705	0.122	$-1.12 \pm j2.82$	0.369	3.04

D.2 Jet Aircraft Decentralized Control Framework

This work implements a decentralized design methodology, wherein controllers are designed separately for the weakly-coupled translational subsystem (associated with the airspeed V and throttle setting δ_T) and rotational subsystem (associated with the FPA γ, attitude q, α, and elevator δ_E). In order to achieve zero steady-state error to step reference commands, we augment the plant at the output with the integrator bank $z = \int y\, d\tau = [z_V,\, z_\gamma]^T = \left[\int V\, d\tau,\, \int \gamma\, d\tau\right]^T$. For dEIRL, the state/control vectors are thus partitioned as $x_1 = [z_V,\, V]^T$, $u_1 = \delta_T$ ($n_1 = 2$, $m_1 = 1$) and $x_2 = [z_\gamma,\, \gamma,\, q,\, \alpha]^T$, $u_2 = \delta_E$ ($n_2 = 4$, $m_2 = 1$). Applying the LQ servo design framework [136] to each of the loops yields a proportional-integral (PI) speed controller K_1^* and a PI/PD FPA/attitude inner-outer loop controller K_2^*. It is these optimal LQ controller parameters which dEIRL will learn online. In general, the optimal LQ controllers K_j^* ($j = 1, 2$) are functions of the modeling error ν, so when necessary we will show explicit dependence by the notation $K_j^*(\nu)$. For the model parameters in Table D.1 and cost structure selections (5.20), the jet aircraft has the following optimal LQ controllers

$$K_1^*(\nu) = \begin{bmatrix} 0.0316 & 0.1496 \end{bmatrix}, \tag{D.5}$$

$$K_2^*(1) = \begin{bmatrix} -0.7071 & -1.7089 & -0.1960 & -0.4251 \end{bmatrix}, \tag{D.6}$$

$$K_2^*(0.9) = \begin{bmatrix} -0.7071 & -1.7403 & -0.1858 & -0.3944 \end{bmatrix}, \tag{D.7}$$

$$K_2^*(0.75) = \begin{bmatrix} -0.7071 & -1.8014 & -0.1687 & -0.3450 \end{bmatrix}. \tag{D.8}$$

As can be seen from (D.5), the optimal controller K_1^* in the translational loop $j = 1$ is independent of the modeling error ν. This is because the lift/AOA derivative L_α enters dynamically into the FPA γ and AOA α equations in (D.1), so the modeling error only affects the dynamics in the rotational loop $j = 2$.

APPENDIX E

DIFFERENTIAL DRIVE MOBILE ROBOT (DDMR) MODEL AND DECENTRALIZED DESIGN FRAMEWORK

E.1 DDMR Model

Consider the following DDMR model [43, 44, 138, 139]

$$\begin{aligned}
\dot{V} &= \tfrac{-2\overline{\beta}}{\hat{m}r^2}V + \tfrac{m_c d}{\hat{m}}\omega^2 + \tfrac{k_t}{\hat{m}k_g r}i_{a_r} + \tfrac{k_t}{\hat{m}k_g r}i_{a_l}, \\
\dot{\omega} &= -\tfrac{\overline{\beta}d_w^2}{2\hat{I}r^2}\omega - \tfrac{m_c d}{\hat{I}}\omega V + \tfrac{d_w k_t}{2\hat{I}k_g r}i_{a_r} - \tfrac{d_w k_t}{2\hat{I}k_g r}i_{a_l}, \\
\dot{i}_{a_r} &= \tfrac{-k_g k_b}{l_a r}V - \tfrac{k_g k_b d_w}{2l_a r}\omega - \tfrac{r_a}{l_a}i_{a_r} + \tfrac{1}{2l_a}\overline{e}_a + \tfrac{1}{2l_a}\Delta e_a \\
\dot{i}_{a_l} &= \tfrac{-k_g k_b}{l_a r}V + \tfrac{k_g k_b d_w}{2l_a r}\omega - \tfrac{r_a}{l_a}i_{a_l} + \tfrac{1}{2l_a}\overline{e}_a - \tfrac{1}{2l_a}\Delta e_a
\end{aligned} \tag{E.1}$$

where V is the robot speed (measured positive forward), ω is the robot angular velocity (measured positive counterclockwise), and i_{a_r}, i_{a_l} are the right and left DC motor armature currents, respectively. We provide definitions and numerical values of all model constants in Table E.1. It should be noted that these parameter selections are standard and were obtained empirically from actual hardware [139].

The system (E.1) is fourth-order, with states $x = [V,\, \omega,\, i_{a_r},\, i_{a_l}]^T$. The controls are $u = [\overline{e}_a,\, \Delta e_a]^T$, where $\overline{e}_a = \frac{e_{a,r}+e_{a,l}}{2}$ is the average of the armature voltages $e_{a,r}, e_{a,l}$ applied to the right and left wheels, respectively, and $\Delta e_a = e_{a,r} - e_{a,l}$ is the difference of the armature voltages. We examine the outputs $y = [V,\, \omega]^T$; i.e., control of the DDMR speed V and angular velocity ω. We examine the equilibrium forward cruise condition $x_e = [V_e,\, \omega_e,\, i_{a_r,e},\, i_{a_l,e}]^T = [2 \text{ m/s},\, 0 \text{ rad/s},\, 0.68\text{A},\, 0.68\text{A}\,]^T$. At this cruise condition, the DDMR is trimmed by the controls $\overline{e}_{a,e} = 3.9115$ V, $\Delta e_{a,e} = 0$ V.

Remark E.1.1 (DDMR Dynamical Structure). Assuming that the motor armature inductance $l_a \approx 0$ (which has proven a reasonable approximation – if included, the motor dynamics have poles on the order of $s = -10^6$), then the DDMR model (E.1) reduces to [139]

$$\begin{aligned}
\dot{V} &= \frac{-2\overline{\beta}}{\hat{m}r^2}V + \frac{m_c d}{\hat{m}}\omega^2 + \frac{2k_t}{\hat{m}k_g r_a r}\overline{e}_a, \\
\dot{\omega} &= -\frac{\overline{\beta}d_w^2}{2\hat{I}r^2}\omega - \frac{m_c d}{\hat{I}}\omega V + \frac{d_w k_t}{2\hat{I}k_g r_a r}\Delta e_a.
\end{aligned} \tag{E.2}$$

We will use this model for purposes of first-principles analysis here. We also use it as the design model for the methods studied to improve numerics. Examination of the DDMR model (E.2) quickly reveals a natural dynamical partition of the form (3.21). The translational loop $j = 1$ consists of the speed state V and is associated with the average voltage control $\overline{e}_a$. The rotational loop $j = 2$ consists of the angular velocity state ω and is associated with the differential voltage control Δe_a.

The (signed) distance d that the vehicle center of gravity (c.g.) lies forward the wheelbase is a central physical parameter in the dynamics of the DDMR (E.2). Firstly, the c.g./wheelbase separation d determines the strength of the coupling terms in (E.2) (i.e., the second term in each state equation). Indeed, (E.2) shows that when $d = 0$, the translational and rotational dynamics of the DDMR decouple – why placing the robot c.g. on the wheel axis is a common design choice in the DDMR community

Table E.1: DDMR Model Parameters

Definition	Symbol	Value
c.g./wheelbase separation	d	d_0 = -6 cm (nominal)
Mass of robot chassis	m_c	3.963 kg
Mass of single wheel	m_w	0.659 kg
Wheel motor moment of inertia	I_w	570 μkg-m^2
Total vehicle moment of inertia	I	0.224 kg-m^2
Radius of wheels	r	3.85 cm
Length of robot chassis	l	44 cm
Width of robot chassis	w	34 cm
Distance between wheels at midpoint	d_w	34 cm
Motor armature inductance	l_a	13.2 μH
Motor armature resistance	r_a	3.01 Ohm
Motor gear up/down ratio	k_g	1
Motor back EMF constant	k_b	0.075 V/(rad/s)
Motor torque constant	k_t	0.075 (N-m)/A
Speed damping constant	β	7.4 μN-m-s
Total vehicle mass	m	$m_c + 2m_w$
Effective mass	$\hat{m}$	$m + \frac{2I_w}{r^2}$
Effective moment of inertia	$\hat{I}$	$I + \frac{d_w^2 I_w}{2r^2}$
Effective damping constant	$\overline{\beta}$	$\beta + \frac{k_t k_b}{r_a}$

[43, 44, 139].

The c.g./wheelbase separation d also determines the stability properties of the DDMR. Loosely speaking, placing the vehicle c.g. forward the wheelbase $d >> 0$ renders the rotational dynamics ω stable: Perturbations in the robot's rotational pose make the friction forces acting on the wheelbase induce torques on the vehicle which counter the direction of the perturbation. Conversely, placing the wheelbase forward the c.g. $d << 0$ results in directional instability for similar reasons. This stability behavior is entirely analogous to the longitudinal dynamics of aircraft, wherein pitch-up instabilities occur if and only if the vehicle center of pressure (playing the analogous role of the wheelbase as the center of forces acting on the vehicle) lies forward the c.g. [122] (see Appendix D).

More concretely, given an equilibrium $x_e = [V_e, \omega_e]^T$ of (E.2), the following controls $u_e = [\overline{e}_{a,e}, \Delta e_{a,e}]^T$ achieve equilibrium

$$\begin{aligned} \overline{e}_{a,e} &= -\frac{\hat{m} k_g r_a r}{2k_t}\left(-\frac{2\overline{\beta}}{\hat{m} r^2} V_e + \frac{m_c d}{\hat{m}}\omega_e^2\right), \\ \Delta e_{a,e} &= \frac{2\hat{I} k_g r_a r}{d_w k_t}\left(\frac{m_c d}{\hat{I}} V_e \omega_e + \frac{\overline{\beta} d_w^2}{2\hat{I} r^2}\omega_e\right), \end{aligned} \tag{E.3}$$

and linearization about the equilibrium (x_e, u_e) yields

$$\begin{bmatrix} \dot{V} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{-2\overline{\beta}}{\hat{m}r^2} & \frac{2m_c d\omega_e}{\hat{m}} \\ \frac{-m_c d\omega_e}{\hat{I}} & \left(\frac{-m_c dV_e}{\hat{I}} - \frac{\overline{\beta}d_w^2}{2\hat{I}r^2}\right) \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{2k_t}{\hat{m}k_g r_a r} & 0 \\ 0 & \frac{d_w k_t}{2\hat{I}k_g r_a r} \end{bmatrix} \begin{bmatrix} \overline{e}_a \\ \Delta e_a \end{bmatrix}. \quad \text{(E.4)}$$

Note from examination of the linearization (E.4) that, reaffirming our insights of the nonlinear dynamics (E.2), the DDMR decouples when $d = 0$; i.e., when the vehicle c.g. is placed on the wheelbase. Decoupling of the *linearized* model also occurs when $\omega_e = 0$ (studied here), in which case examination (E.4) shows that the DDMR has open-loop eigenvalues at $s = s_V, s_\omega$, where

$$s_V \triangleq \frac{-2\overline{\beta}}{\hat{m}r^2}, \quad \text{(E.5)}$$

$$s_\omega \triangleq -\left(\frac{m_c V_e}{\hat{I}} d + \frac{\overline{\beta}d_w^2}{2\hat{I}r^2}\right). \quad \text{(E.6)}$$

The first mode s_V (E.5) is a stable speed damping mode arising from wheel friction and the electro-mechanical damping characteristics of the motors. The second mode s_ω (E.6) determines the stability properties of the rotational dynamics. (E.6) shows that the following critical value d_{MS} of the c.g./wheelbase separation results in marginal stability:

$$d_{MS} = -\frac{\overline{\beta}d_w^2}{2m_c V_e r^2}. \quad \text{(E.7)}$$

For $d > d_{MS}$, the DDMR is stable. For $d < d_{MS}$, the DDMR is unstable. Note in the case of zero speed/motor damping $\overline{\beta} = 0$ that $d_{MS} = 0$; i.e., the DDMR is stable if and only the vehicle c.g. lies forward the wheel axis $d > 0$ – numerically reaffirming the physical intuitions discussed above. More generally, (E.6) shows that greater translational damping $\overline{\beta}$ increases the stability of the DDMR rotational dynamics. For the DDMR parameters studied (cf. Table E.1), $d_{MS} = -0.92$ cm, so the nominal c.g./wheelbase separation $d_0 = -6$ cm results in a directionally-unstable system.

In the studies conducted in this work, we will focus on how modeling errors in the c.g./wheelbase separation d affect the DDMR dynamics and learning performance. Specifically, we study modeling errors of the form

$$d = \nu\, d_0, \quad \text{(E.8)}$$

where $d_0 \in \mathbb{R}$ is a nominal value of the c.g./wheelbase separation, and $\nu \in \mathbb{R}$ is the modeling error parameter (nominally 1). As $\nu > 1$ increases, $d < d_0 < d_{MS} < 0$ decreases, and (E.6) shows that the system becomes more unstable. Table E.2 shows the effect of DDMR eigenvalues as a function of the modeling error parameter ν (E.8). The DDMR directional instability is highly sensitive to modeling errors ν in the c.g./wheelbase separation d. As predicted by (E.5), the speed mode s_V is stable and independent of the c.g/wheelbase separation d.

Table E.2: DDMR Eigenvalues Versus Modeling Error Parameter ν (E.8)

ν (E.8)	Speed Mode s_V (E.5)	Angular Velocity Mode s_ω (E.6)
1 (nom)	-0.4184	1.6343
1.1	-0.4184	1.8274
1.25	-0.4184	2.1171

E.2 DDMR Decentralized Control Framework

This work implements a decentralized design methodology, wherein controllers are designed separately for the weakly-coupled translational subsystem (associated with the speed V and average voltage control $\overline{e}_a$) and rotational subsystem (associated with the angular velocity ω and differential voltage control Δe_a). In order to achieve zero steady-state error to step reference commands, we augment the plant at the output with the integrator bank $z = \int y\, d\tau = [z_V,\, z_\omega]^T = \left[\int V\, d\tau,\, \int \omega\, d\tau\right]^T$. For dEIRL, the state/control vectors are thus partitioned as $x_1 = [z_V,\, V]^T$, $u_1 = \overline{e}_a$ ($n_1 = 2$, $m_1 = 1$) and $x_2 = [z_\omega,\, \omega]^T$, $u_2 = \Delta e_a$ ($n_2 = 2$, $m_2 = 1$). Applying the LQ servo design framework [136] to each of the loops yields a proportional-integral (PI) speed controller K_1^* and a PI angular velocity controller K_2^*. It is these optimal LQ controller parameters which dEIRL will learn online. For the model parameters in Table E.1 and cost structure selections (5.21), the DDMR has the following optimal LQ controllers

$$K_1^*(\nu) = [\ 3.6515 \quad 5.2062\ ], \tag{E.9}$$

$$K_2^*(1) = [\ 5.0000 \quad 10.2344\ ], \tag{E.10}$$

$$K_2^*(1.1) = [\ 5.0000 \quad 10.9164\ ], \tag{E.11}$$

$$K_2^*(1.25) = [\ 5.0000 \quad 11.9718\ ]. \tag{E.12}$$

As can be seen from (E.9), the optimal controller K_1^* in the translational loop $j = 1$ is independent of the modeling error ν. This is immediately seen from examination of the linearized dynamics (E.4), wherein we observe that the diagonal terms in A, B pertaining to the speed V are independent of the c.g./wheelbase separation d, hence of the modeling error ν. On the other hand, the optimal controller K_2^* in the rotational loop $j = 2$ is heavily dependent on the modeling error ν.