Towards Reducing Barriers to the Adoption of Reinforcement Learning
by Utilizing Closed-Form Feedback Control

Gerald Eaglin

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

University of Louisiana at Lafayette
Fall 2023

APPROVED:

Joshua E. Vaughan, Chair
Department of Mechanical Engineering

Alan A. Barhorst
Department of Mechanical Engineering

Terrence L. Chambers
Department of Mechanical Engineering

Afef Fekih
Department of Electrical and Computer Engineering

Raju Gottumukkala
Departement of Mechanical Engineering

Peter L. Wang
R&D Associate Staff
Oak Ridge National Laboratory

Mary Farmer-Kaiser
Dean of the Graduate School

Abstract

In order to utilize conventional control methods, it is necessary to have knowledge of both the system model and control theory. Even with this domain knowledge, there are many problems, particularly for optimal control, that are not amenable to analytical solution methods. Numerical optimal controllers and methods to iteratively learn optimal controllers are necessary for these cases. Reinforcement learning is one class of data-driven optimal control methods where an agent learns to control the system through iterative interaction. Although reinforcement learning can be implemented with no domain knowledge, the required iteration often makes training the agent time-consuming. Additionally, it is often difficult to generate an intuitive understanding of the control law that the agent learned and difficult to generate performance guarantees for the agent. These issues are addressed in this work by combining reinforcement learning with controllers designed using domain knowledge. Several combined controller architectures where developed and tested on multiple benchmark systems. It was shown that well-designed combined controllers often provided an improvement in pre-training performance, a reduction in required training time, and improved performance after training compared to using reinforcement learning agents alone. A method to improve the interpretability of the behavior of the agents by modeling the agent as a switching controller was also proposed. The stability and robustness of the combined controllers were also analyzed. Finally, an evaluation of the current commercializability of the proposed methods was presented along with a plan to increase its market readiness.

To those who have supported me during this endeavor,
especially my family, who have provided indispensable encouragement.
I could not have accomplished this without your continual support.

"The way to succeed is to double your failure rate."

—Thomas Watson

Acknowledgments

I would like to thank my advisor, Dr. Joshua Vaughan, who has provided guidance during this process and even stayed on as my advisor after transferring to Oak Ridge National Lab despite the extra effort of having a remote student.

I would also like to thank the members of the **C.R.A.W.LAB** and other students in our lab, namely my undergraduate research assistant, Thomas Poché, who assisted with experiments, and Andrew Albright and Nalaka Amarasiri for the numerous discussions and feedback that were vital to improving this work.

Finally, I cannot neglect to thank all of my family and friends who have provided copious amounts of support and encouragement that have so often reinvigorated me during this lengthy endeavor.

Table of Contents

List of Tables

List of Figures

I Introduction and Background

1.1 Introduction

Controllers allow systems to perform desired functions and reduce the need for human intervention to achieve the desired behavior. As technology advances, complex processes are being automated more often, which increases the need for more complex and sophisticated control systems. However, designing control systems to provide full autonomy for complex problems is often difficult using conventional control design methods. For this reason, control systems may be designed for a subsystem in order to provide partial autonomy. For instance, modern automobiles can have multiple levels of autonomy from cruise control to lane-stay assist to self-driving, each requiring control of multiple subsystems to provide the desired level of autonomy. Domain knowledge of the system used in the control application is necessary in order to design appropriate controllers. This domain knowledge includes how to model the system, how to choose and design appropriate performance metrics, as well as knowledge of how different controllers affect the behavior of the system.

Controllers are generally designed to optimize specific performance metrics. For the cruise control example, this may include minimizing fuel usage or minimizing settling time of the velocity of the vehicle with constraints on passenger comfort. For some simple systems, the optimal control problem can be solved analytically. For instance, Linear Quadratic Regulator (LQR) control is a common method for solving the optimal control problem for linear systems with quadratic cost functions. Methods such as this can also be used for nonlinear systems if they are linearized [1, 2]. For general nonlinear systems, especially those with discontinuous nonlinearities and those operating in dynamic, unstructured environments that are difficult to model, there are no general methods for analytically solving the optimal control problem. For these types of systems, optimal controllers are often approximated through recursive or iterative methods. One of these methods is reinforcement learning (RL).

Reinforcement learning is a class of methods in which control policies are learned through repeated interaction with the system [3, 4]. This allows RL to learn online without needing domain knowledge of the laws that govern the system. This is useful for applications that are not amenable to analytical solutions for optimal controllers. However, there are drawbacks that prevent widespread application of RL beyond simulation and laboratory settings. RL is sample inefficient, requiring many interactions with the system, meaning that significant training time is often required to learn a usable policy. RL policies are also black boxes, and it is often difficult to interpret the control laws governing their behavior. This is especially true for high degree-of-freedom systems or those with continuous-action spaces where it is necessary to utilize neural networks to learn the control policy. Additionally, it is usually impossible to generate performance guarantees for control policies generated using RL. RL policies for which there are no performance guarantees or that are not interpretable will not be utilized in safety-critical applications such as transportation or health care [5–7]. For these reasons, the use of RL is primarily reserved for simulation and laboratory settings where the risks from poorly performing policies can be minimized. Since conventional control methods do not have the same drawbacks as RL, it is beneficial to combine conventional control methods with RL to generate controllers that reduce the drawbacks of conventional control and RL when used separately.

There have been methods proposed to utilize RL agents along with conventional closed-form control methods [8–10]. However, there is no work that provides comprehensive design guidelines. The primary goal of this dissertation is to provide guidelines for designing controllers that combine RL and conventional control methods utilizing domain knowledge. These guidelines facilitate using domain knowledge of control theory to:

- design RL controllers that reduce required training time to achieve desirable performance compared to training RL without domain knowledge

- design RL controllers that provide an initial policy that is safe to use on a physical system before training the agent
- train policies that provide bounds on stability and can be analyzed using conventional stability analysis tools
- interpret policy behavior according to conventional control analysis methods

The above contributions will allow RL to be more readily applied for physical systems since shorter training time and good initial policies make training directly on physical systems more feasible. Additionally, generating policies that can be interpreted and that have performance and stability guarantees makes RL more attractive for safety-critical applications.

The remainder of this chapter introduces the background for analytical solutions for optimal control, as well as describing the limits of analytical solutions that motivate the use of approximate optimal control and RL methods. Section 1.2 describes conventional optimal control methods for dynamic systems, including analytical and approximation methods. Section 1.3 provides an overview of RL and describes the noteworthy developments leading to modern RL algorithms. Deep RL is then described in Section 1.4, followed by drawbacks of using RL in Section 1.5. An overview of the combined controller methods proposed in this work is presented in Section 1.6.

1.2 Optimal Control

The optimal control problem requires finding a control law that optimizes a performance metric subject to the system dynamics and other constraints. It is widely used in many applications ranging from robotics to economics [11]. Formulations for the optimal control problem can be categorized based on their applicability to discrete-time and continuous-time systems.

1.2.1 *The Discrete-Time Problem*

A discrete-time system is one in which the input and output signals do not vary continuously, but instead vary only at non-infinitesimal intervals. Although most systems vary continuously, continuous systems are typically controlled using discrete controllers. Discretizing the system can also simplify analysis. The state evolution of a discrete-time system is described by:

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

where x_{k+1} and x_k are the next state and current state, respectively, and u_k is the current input at time step k. The general form of a cost function defining the performance metric to be minimized is:

$$J_i(x_i) = \phi(N, x_N) + \sum_{k=i}^{N-1} L(x_k, u_k) \tag{2}$$

where $L(x_k, u_k)$ is the intermediate cost at time step k and is dependent on the current state and input. The fixed final cost, $\phi(N, x_N)$, is dependent on the final time step, N, and final state, x_N. Equation (2) defines the cost incurred during the interval $[i, N]$. The goal of solving the optimal control problem is to find a control policy, u_k^*, that minimizes (2) on the time interval $[i, N]$. For discrete-time systems, this can be accomplished by converting the optimal control problem into a static optimization problem and solving for the optimal control sequence, $[u_i^*, u_{i+1}^*, \ldots, u_{N-1}^*]$, that minimizes J subject to (1). This method is sometimes employed in approximate solutions methods such as Model Predictive Control (MPC), but the optimization must be performed for every initial condition since it does not result in a closed-form control law.

1.2.2 *The Continuous-Time Problem*

Continuous-time systems describe the continuous state evolution of the output based on a continuous input. The general form describing the state evolution of a

continuous-time system is:

$$\dot{x}(t) = f(x, u, t) \tag{3}$$

where x is the current state at time t, and $\dot{x}(t)$ is the time derivative of $x(t)$ dependent on the current state, x, current input, u, and current time, t. The general form of the cost function for the continuous-time problem is:

$$J(t_0) = \phi(x(T), T) + \int_{t_0}^{T} L(x(t), u(t), t)\,\mathrm{d}t \tag{4}$$

where $\phi(x(T), T)$ is the fixed final cost at the final time, T, and $L(x(t), u(t), t)$ is the intermediate cost. Equation (4) defines the cost from any initial time, t_0, to the final time, T. The goal of the continuous-time optimal control problem is to find a continuous control policy, $u^*(t)$, that minimizes the cost on the time interval. For simple or simplified optimal control problems, the discrete-time and continuous-time optimal control problems can be solved analytically. For instance, the process of solving (4) for linear continuous-time systems with quadratic intermediate cost, $L(x(t), u(t), t)$, results in LQR. In other cases, the solution must be approximated. The following sections describe common solution methods for the optimal control problem.

1.2.3 *Dynamic Programming*

Dynamic programming was originally developed by Richard Bellman during the 1950s. It relies on the principle that the optimal policy must have the property that no matter the previous control decisions, the remaining control decisions must be optimal [12]. This allows the optimal control problem to be solved recursively.

Using this notion to define the optimal control problem recursively, the cost function for discrete-time systems (2) can also be defined from any time step, k:

$$J_k(x_k) = L(x_k, u_k) + \phi(N, x_N) + \sum_{k=i}^{N-1} L(x_{k+1}, u_{k+1}) = L(x_k, u_k) + J_{k+1}(x_{k+1}) \tag{5}$$

where the cost of being in the current state, $J_k(x_k)$, is the intermediate cost in the current state, $L(x_k, u_k)$, plus the sum of all future cost from the next state, $J_{k+1}(x_{k+1})$.

If the optimal control sequence from state x_{k+1} is known, $J^*_{k+1}(x_{k+1})$, then the optimal cost in state x_k is:

$$J^*_k(x_k) = \min_{u_k}(L(x_k, u_k) + J^*_{k+1}(x_{k+1}, u_{k+1})) \tag{6}$$

For systems without constraints, (6) is used to analytically solve for the optimal control law. In the case where there are constraints that make analytical solutions of (6) impossible, the problem can still be solved numerically.

Dynamic programming can also be used to solve the optimal control problems for continuous-time systems. Following the principle analogous to (5), the cost from any state, x, at time, t, described by (4), can be reformulated as:

$$J(x,t) = \phi(x(T),T) + \int_t^{t+\Delta t} L(x,u,\tau)\,\mathrm{d}\tau + \int_{t+\Delta t}^{T} L(x,u,\tau)\,\mathrm{d}\tau \tag{7}$$

where the integral of the intermediate cost from $[t, t+\Delta t]$ is the cost from the current state at time t to a new state after a short time Δt. The final cost, $\phi(x(T),T)$, and the integral of intermediate cost from $[t+\Delta t, T]$ is the cost, $J(x+\Delta x, t+\Delta t)$, from the next state, $x+\Delta x$, at the next time, $t+\Delta t$, such that (7) can be written as:

$$J(x,t) = \int_t^{t+\Delta t} L(x,u,\tau)\,\mathrm{d}\tau + J(x+\Delta x, t+\Delta t) \tag{8}$$

If the optimal cost, $J^*(x+\Delta x, t+\Delta t)$, is known, then the optimal cost for all x and t is,

$$J^*(x,t) = \min_{u(\tau)} \left[\int_t^{t+\Delta t} L(x,u,\tau)\,\mathrm{d}\tau + J^*(x+\Delta x, t+\Delta t) \right] \tag{9}$$

Although it is possible to use (9) to numerically solve for optimal policies, it cannot be solved analytically. Instead, it is common to manipulate (9) such that it is more amenable to analytical solutions:

$$-\frac{\partial J^*}{\partial t} = \min_{u(t)} \left(L + \left(\frac{\partial J^*}{\partial x} \right)^T f \right) \tag{10}$$

This is known as the Hamilton-Jacobi-Bellman (HJB) equation. It is a partial differential equation for the optimal cost, and its solution provides the optimal control

policy for any system. For LTI systems with quadratic cost functions, the HJB equation can be used to derive the Linear Quadratic Regulator (LQR). However, it is usually impossible to solve (10) analytically for nonlinear systems with constraints and non-quadratic cost functions. This has motivated the creation of methods to numerically solve for the solution to the HJB equation [13–16].

Even for the cases in which the solution of the HJB equation can be approximated, it must be solved offline because it is recursive. The following section introduces online and forward-in-time methods to directly approximate solutions for optimal control problems.

1.2.4 *Forward-in-Time Solutions*

One commonly used online optimal control method is Model Predictive Control (MPC), which approximates solutions for the optimal control problem by directly solving for a control sequence that optimizes a cost function over a finite horizon [17]. It is able to approximate solutions for the general optimal control problem for nonlinear systems and for systems with constraints. To determine an optimal control sequence, a model of the system dynamics is necessary in order to predict the state evolution of the system with respect to the optimal input sequence. At each time step, the first input in this sequence is passed to the system, then the process is repeated at the next time from the next state. While, in theory, MPC can be implemented for any system, its use is limited by its computational complexity. Since MPC does not solve for a closed-form control law, the control sequence must always be solved online. Although online solutions are often not challenging for systems with low bandwidth, it is a challenge for high-bandwidth systems since the time to compute the optimal sequence over the finite horizon can cause significant delay.

Another approximation method is Iterative LQR (iLQR), which is an extension of LQR for optimal control of nonlinear systems. iLQR adapts standard LQR for the

general optimal control problem by linearizing the system at the current state. The cost function also needs to be approximated as a quadratic, often by using Taylor approximation taken to two terms. The optimal gains for the current linearization and approximated cost function are then solved using the formulation as in standard LQR. For the next state at the next time step, the linearization of the system and quadratic approximation of the cost function are repeated. This method works well for unconstrained nonlinear systems with general cost functions. Unlike MPC, however, since iLQR is based on LQR, it cannot be solved for systems with hard constraints on state and control inputs. However, there are modifications to iLQR proposed to solve constrained problems [18].

MPC provides a way to handle constraints while optimizing. However, it is necessary to perform computationally expensive numerical optimization at every time step to generate an optimal control sequence over a finite horizon. iLQR simplifies the control problem into one in which the optimal control input at the current time can be found analytically, which reduces computational complexity compared to MPC. However, the standard implementation does not account for systems with constraints. This motivates the development of optimal control methods to handle general constraints with low computational complexity during implementation. Reinforcement learning is one method that solves these problems.

1.3 Reinforcement Learning

1.3.1 *Reinforcement Learning Overview*

Reinforcement learning (RL) is an iterative forward-in-time method to solve for optimal control policies. RL combines concepts from psychological studies on trial-and-error learning with optimal control theory. Algorithms for RL are able to generate optimal policies through repeated trial-and-error interaction with an environment. This is illustrated graphically in the diagram of the basic learning process

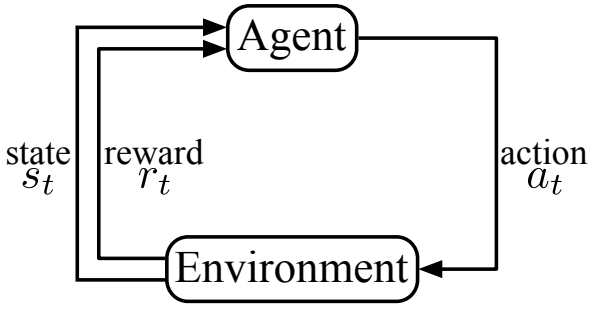


Figure 1. Basic reinforcement learning process

for RL shown in Figure 1. The learning process has two main components which exchange information. The agent is analogous to a controller in conventional controls terminology, and the environment is analogous to the plant. The agent generates an action, a_t, or control input, in accordance with the current policy. This action is passed to the environment. The state, s_t, of the environment changes as a result of the action. During training, a reward signal is used to quantify the value of the current action in the current state; this is similar to a cost function in conventional optimal control. When the reward signal is passed to the agent, the policy is updated in order to progressively maximize the reward. This is usually repeated for a set number of repetitions or episodes. Once training is complete, the approximated optimal control policy is implemented in closed form without further online optimization.

RL is based on the same principles as dynamic programming. While an optimal control policy is found in dynamic programming by minimizing a cost function, RL finds optimal policies by maximizing cumulative reward or return from the policy:

$$R(s_0) + \gamma^1 R(s_1) + \cdots + \gamma^N R(s_N) \tag{11}$$

where $R(s)$ is the incremental reward received from being in state s and γ is a discount rate between 0 and 1. This reward is used to evaluate the performance of the policy along a state trajectory $[s_0, s_N]$. It is useful to use cumulative reward to evaluate the policy from any state, s_n. This allows the agent to generate policies that are optimal

over a longer horizon since a state that produces low incremental reward may later lead to states that produce high reward when following the current policy. However, the cumulative reward that will be received after the current state is usually not known and is instead approximated. The expected cumulative reward or return is defined as the value of being in state, s_n, and following the current policy, π:

$$V^{\pi}(s_n) = \mathbb{E}\left[\sum_{n=0}^{N} \gamma^n R(s_n)\right] \tag{12}$$

where γ is the discount rate between $0 \leq \gamma \leq 1$ that adjusts the current value of future rewards. Because value discounting is exponential due to γ^n, (12) can be written recursively:

$$V^{\pi}(s_n) = \mathbb{E}\left[R(s_n) + \sum_{n=0}^{N} \gamma^{n+1} R(s_{n+1})\right] \tag{13}$$

This is analogous to the recursive Bellman equation from (5) since the current value at state s_n is a function of the expected future value from state s_{n+1}. These concepts are also applicable to stochastic systems. For stochastic systems, the probability of transitioning from state s to state s' given the action, a, is:

$$\text{Prob}[s'|s, a] = P_{ss'}[a] \tag{14}$$

Modifying (13) for use with stochastic systems results in:

$$V^{\pi}(s) = R(\pi(s)) + \gamma \sum_{s'} P_{ss'}[\pi(s)] V^{\pi}(s') \tag{15}$$

where $R(\pi(s))$ is the reward gained by performing an action according to the policy, π, in state s. The optimal value of the current state corresponding to the optimal policy, π^*, is:

$$V^*(s) = V^{\pi^*}(s) = \max_a \left[R(\pi^*(s)) + \gamma \sum_{s'} P_{ss'}[\pi^*(s)] V^{\pi^*}(s')\right] \tag{16}$$

Many algorithms that rely on using the value function to approximate future reward have been developed, each with distinct advantages and disadvantages. The following sections discuss algorithms that have been noteworthy contributions to the field of RL.

	a_1	a_2	a_3	$\cdots$
s_1	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$Q(s_1, a_3)$	$\cdots$
s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$Q(s_2, a_3)$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Figure 2. Example Q-table

1.3.2 *Q-Learning*

Q-learning is a tabular method for model-free RL [19–21]. The primary advantage of Q-learning compared to previous algorithms is the reduction in computational complexity of updating the policy during training. Since the goal of RL is to iteratively optimize the policy, multiple versions of the policy must be evaluated. The value of being in each state will depend on the policy being used, such that a new policy would require the value function under that policy to be reevaluated for every state. Q-learning modifies how the value of a state is calculated, such that modifying the policy does not require recalculating the value function for every state, thus reducing the computational complexity of evaluating new policies. The action-value function represents the value of applying any action, a, at the current state, s, then following the policy, π, thereafter:

$$Q^{\pi}(s, a) = R(a) + \gamma \sum_{s'} P_{ss'}[\pi(s)] V^{\pi}(s') \tag{17}$$

This is used to aggregate a Q-table that stores the evaluation of the action-value function for all combinations of states and actions, as illustrated in Figure 2. Q-learning allows the action-value function, Q, to approximate the optimal action-value function independently of the current policy. This provides a way to incrementally update the existing Q-table without needing to regenerate a new table for every policy

update. The success of Q-learning motivated the development of other algorithms that utilize action-value functions [22]. However, because Q-learning is a tabular method, it suffers from the "curse of dimensionality." The Q-learning algorithm has to experience every combination of state and action repeatedly. This is problematic for high dimensional problems such as multi-degree-of-freedom systems or multiple-input systems. This also makes it difficult to use for continuous problems since the system must be discretized to be solved by Q-learning. These challenges motivated the introduction of general function approximation methods to represent the value and action-value for any combination of state and action [23, 24].

1.4 Deep Reinforcement Learning

Although tabular RL methods often work well for problems with discrete action and observation spaces, tabular methods are restrictive for high-dimensional problems since the size of the tables need to increase with the size of the action and observation spaces, which increase computational complexity to find optimal policies. This restriction also prevents tabular RL from being used for continuous action and observation spaces. This issue is solved by using nonlinear function approximators. Neural networks are commonly employed as nonlinear function approximators for machine learning and have been used extensively in supervised learning and unsupervised learning [25]. RL is referred to as deep RL when a deep neural network is used to approximate the value function or policy. Neural networks in deep RL are able to mitigate the "curse of dimensionality" by directly approximating the nonlinear value function. Whereas tabular methods require every combination of observation and action to be visited, using a neural network to approximate the value function generates continuous estimates of the value. This eliminates the need to visit every observation and allows a continuous observation space to be used. One of the first successful deep RL algorithms was used to train agents to play Atari games, where the agents were

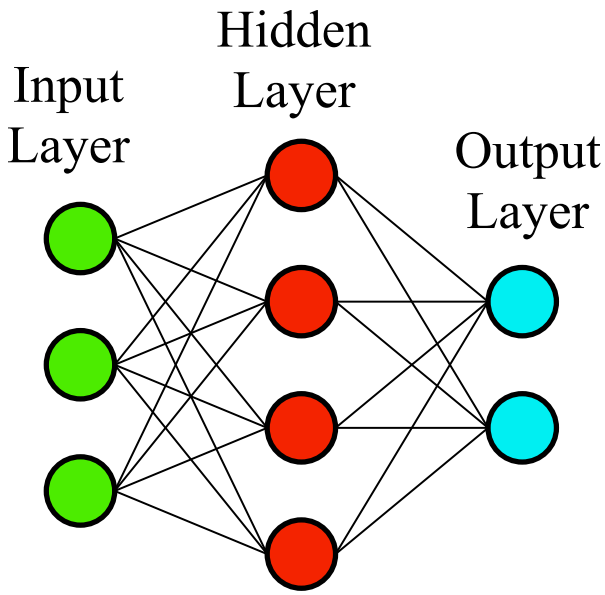


Figure 3. Neural network diagram

trained directly from pixel data. These agents trained using deep RL were often able to match or surpass human performance [26, 27]. These contributions have enabled the development of other RL algorithms to train agents for other high-dimensional problems.

An illustration of a neural network is shown in Figure 3. Neural networks are comprised of nodes which each have input and output signals. Nodes of a neural network are arranged in layers. Nodes within each layer are not connected to each other, but each node is connected to nodes in the previous and following layer. Each connection between the nodes is associated with a real-valued weight, and the value of the output signal from each node is computed from the value of the input signal and the weight of the connection. The green nodes represent the input layer. In the case of RL, the input layer accepts the observations. The blue nodes represent the output layer. In RL, the outputs of the network can be the value in a state or the actions in the case of an actor-critic algorithm. The red nodes represent the hidden layers. Although the number of nodes in the input and output layers are determined by the

application, the number of hidden layers and the number of nodes in each hidden layer of deep neural networks can vary. Networks with fewer nodes in the hidden layers have fewer parameters with which to approximate the value function. Increasing the size of the network generally allows the deep neural network to better approximate the value function due to having more parameters to adjust to get a good fit [28].

There are many deep RL algorithms that have been developed. The main categories include value-based methods and actor critic methods. Noteworthy algorithms from these categories are described in the following sections.

1.4.1 *Value-Based Methods*

Value-based deep RL algorithms replace the tabular evaluation of the value or action-value function with a deep neural network capable of approximating continuous functions. This enables learning policies for systems with many observations or continuous observation spaces. One of the first and most influential deep RL algorithms is the Deep Q-network (DQN) algorithm [26, 29]. DQN is based on the standard Q-learning algorithm with modifications that enable the use of deep neural networks to approximate the nonlinear action-value function. The use of deep neural networks allowed DQN to learn to play Atari games directly from raw pixel input. DQN produced performance comparable or superior to humans for four of the seven Atari games tested. Although DQN is able to operate on continuous observation spaces, it requires discrete action spaces since every action must be evaluated to find the optimal action from each state. This motivated the development of methods that can directly model the optimal policy.

1.4.2 *Actor-Critic Methods*

One method to directly learn policies is to model them using deep neural networks. Using a function approximator such as a neural network to model the policy

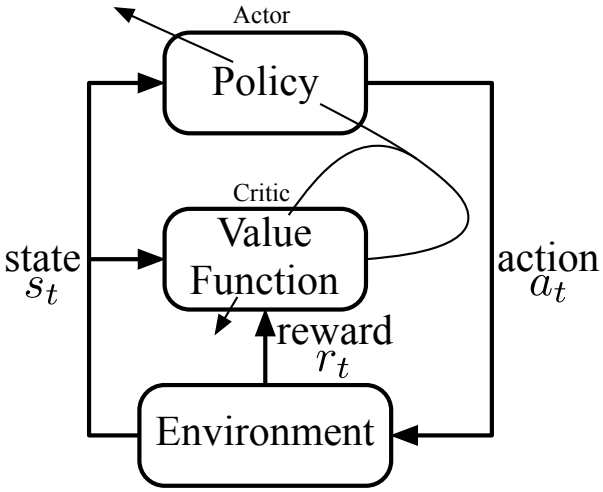


Figure 4. Actor-critic diagram

allows inputs from observations to produce actions in a continuous action space. One type of RL algorithm that takes advantage of neural networks to produce both continuous action spaces and continuous observation spaces is an actor-critic algorithm, where the value function and policy are modeled using separate neural networks [30, 31]. A diagram illustrating the basic function of an actor-critic algorithm is shown in Figure 4. The actor portion of the algorithm contains the policy and provides the action to the environment. The critic contains the network that approximates the value function. Both the actor and critic networks must be progressively updated to produce accurate estimates of the expected cumulative reward as well as the optimal policy that maximizes the reward.

1.4.3 *Deep Deterministic Policy Gradient*

A noteworthy development in actor-critic algorithms came from the development of the Deep Deterministic Policy Gradient (DDPG) algorithm [30]. The critic portion of DDPG is inspired by Deep Q Network (DQN) [26, 29]. The actor portion of DDPG is updated based on the Deterministic Policy Gradient (DPG), which allows the policy to be updated based on the gradient of the value function [32].

1.4.4 *Twin-Delay DDPG*

A common disadvantage of deep RL algorithms is the occurrence of overestimation. This occurs when the algorithm overestimates the value approximation, often resulting in suboptimal policies. The Twin-Delay DDPG (TD3) algorithm is a modification of the DDPG algorithm that limits the occurrence of overestimation [31]. This is accomplished by using two critic networks, where the policy is updated according to the critic with the lowest value estimate to avoid overly optimistic updates. This makes the action-value estimates more accurate to more reliably find optimal policies.

1.5 Problems with RL

Although RL has advantages over other methods for approximating optimal controllers, it also has drawbacks that have prevented its widespread adoption. One of the primary drawbacks is the iteration needed to train policies, which often requires significant training time. This is illustrated by Figure 5 showing reward during training. At the beginning of training, the policy performs poorly and produces low reward. This not only corresponds to a suboptimal policy but may also indicate that the policy causes unsafe behavior or does not perform the desired task, making it unusable. More training time is required before the policy achieves a suitable threshold of performance to allow it to be implemented for its intended purpose.

The tendency to perform poorly in the initial stage of training is a drawback that prevents training directly on physical systems. For instance, using a poorly performing policy on a robotic system may result in the system behaving in a way that causes it to damage itself or its surroundings. The iteration required during training also reduces the feasibility of training on physical systems since this may require significant human supervision to reset the system for each iteration or if a failure occurs due to the poor performance. Training in simulation is faster and can be more easily

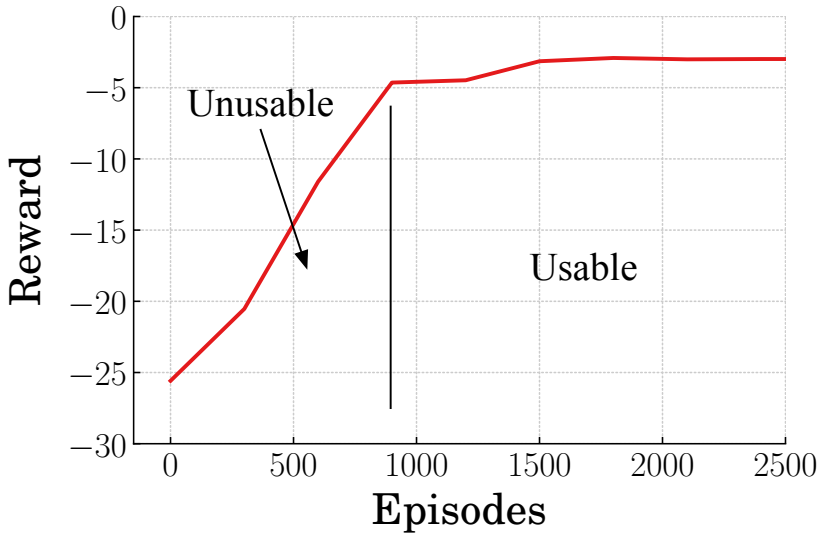


Figure 5. Example of initial unusable policy

automated to require less human supervision. However, agents trained in simulation often have degraded performance when transferred to physical systems due to modeling error and sensor noise. There has been a large amount of research performed to bridge the gap between simulation and physical applications, or sim-to-real [33, 34]. This often requires initially training in simulation then completing training on the physical system.

Another drawback to RL is the difficulty to develop performance guarantees for RL agents. In particular, the lack of stability guarantees prevents the adoption of RL in many applications. Although some applications such as games do not require stability guarantees, physical applications, such as robotics and autonomous navigation, require stability guarantees in order to provide trust in the agent before implementation. One method for training stable agents is to design a set of stabilizing controllers to serve as candidate controllers that the agent has to switch between [35]. This method uses discrete actions to choose the appropriate stabilizing controller resulting in a discontinuous controller that is stable but suboptimal. Another method for learning stable agents involves providing an initial stabilizing policy and constraining any modification to the policy to remain stable [36]. As the agent learns, it is able to

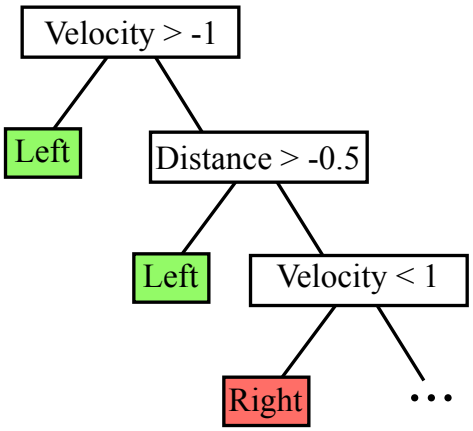


Figure 6. Example of a decision tree

increase its approximated region of attraction. While this method provides safe and stabilizing agents within the region of attraction, it sacrifices exploration at the expense of stability. Additionally, a stabilizing initial policy must be provided. It is beneficial to have agents that achieve stability without having to pre-design stable policies for the RL controllers.

RL agents also tend to be difficult to interpret, particularly deep RL agents. Because of the large number of nodes in a deep neural network required to model the policy, it is difficult to intuitively understand the control law learned by the agent, making the agent a black box [37]. As with the inability to generate stability guarantees, this prevents the adoption of RL for many applications since it is difficult to predict the behavior of the agents in a wide range of scenarios. Some methods used to interpret agents require manual summarization of the behavior of the agent [38, 39]. Other methods utilize decision trees, illustrated in Figure 6, which can help interpretability by reducing the complexity of the deep model into a concise graphical model [40]. Decision trees often require training a shallow model from a deep neural network model. However, agents with continuous action spaces or those that learn complex behaviors may require large decision trees that are still difficult to interpret.

1.6 Combining RL and Conventional Control

Although RL agents can learn policies without knowledge of the model of the system being controlled, a model is often available for use by the control systems designer. This is always the case for agents that are initially trained in simulation. Additionally, if a system model is known, an experienced controls engineer will also often have knowledge of how to design a controller that enables the system to accomplish the desired goal even if the controller is suboptimal. RL controllers can be trained in a way to take advantage of established domain knowledge of dynamics and control theory. This can include direct design of controllers that combine RL and conventional control elements that both contribute to the total system input [41, 42]. Domain knowledge may also be used indirectly through the design of the environment, such as reward function design. Domain knowledge can also be utilized in the analysis of agent performance after training to improve interpretability and generate performance guarantees.

The following chapters present contributions that the combination of RL and conventional control have to the control design processes:

- Chapter 2 introduces multiple controller architectures for combining RL with conventional control. The baseline performance of these combined controllers applied to multiple benchmark systems is then presented. The combined controllers are shown to generally reduce required training time to generate acceptable policies. Design guidelines are then presented based on the performance of the controllers.

- Chapter 3 presents stability evaluations of the combined controllers.

- Chapter 4 presents the effects of the combined controllers on robustness to modeling error.

- Chapter 5 proposes a method to use domain knowledge to improve interpretability of the learned controllers by approximating the agent with a piecewise set of control laws.
- Chapter 6 presents an analysis of the current commercializability of this work.

II Reinforcement Learning Control with Model-Based Elements

Although Reinforcement Learning (RL) can be implemented model-free, where an agent is trained without domain knowledge of control systems, it is common for the practitioner to have experience with dynamics and control. For systems for which a model is known, there is often some understanding of how to design a controller to accomplish the desired task, especially for linear systems or systems that can be approximated as linear. For instance, PID and LQR are well documented linear control methods that can be applied to many problems. Even though a controller may be found to accomplish the goal, such as remaining stable near a reference trajectory, optimal controllers cannot generally be found analytically for nonlinear systems. This challenge can be solved with RL, which is capable of iteratively generating optimal controllers for cases which cannot be solved analytically. However, it is often time-consuming to train RL agents due to the iteration required. Additionally, initial policies often perform poorly.

Controllers that combine conventional control methods with RL can address the drawbacks of both methods by allowing RL to be of benefit where it is most needed. Instead of using RL to learn what is already understood by the designer, it can be used to extend beyond the domain knowledge of the designer. The way in which this domain knowledge should be utilized depends on the system being controlled and the desired goal of the controller. The following section introduces classification categories for different control structures for this combined control approach that have been investigated in this work. The performance of these control structures is then described, followed by a summary of the performance and guidelines for designing combined controllers for a variety of systems.

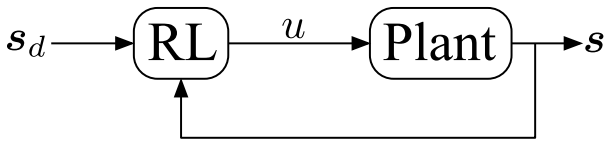


Figure 7. Block diagram for Pure RL control structure

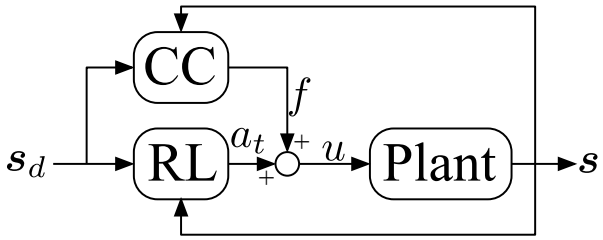


Figure 8. Block diagram for RL and fixed-gain control structure

2.1 Control Structures

2.1.1 *Pure RL*

Controllers utilizing an RL agent alone are used as a baseline for evaluating the effects of various combined controller architectures. The implementation of the trained agent is represented by the block diagram in Figure 7, where $\boldsymbol{s}_d$ is the desired state, u is the agent output, and $\boldsymbol{s}$ is the current state of the plant. Since the agent controller is acting on its own without any contribution from any model-based controller, the input to the plant is the agent action, $u = a_t$.

2.1.2 *RL and Fixed-Gain Control*

One proposed control architecture for combining domain knowledge with RL uses fixed-gain control components in parallel with the learned policy. The block diagram for this control architecture is shown in Figure 8, where the CC block represents Conventional Control, or a model-based controller designed using domain knowledge of dynamics and control. The plant input, u, for this control structure is:

$$u = f(\boldsymbol{s}) + a_t \tag{18}$$

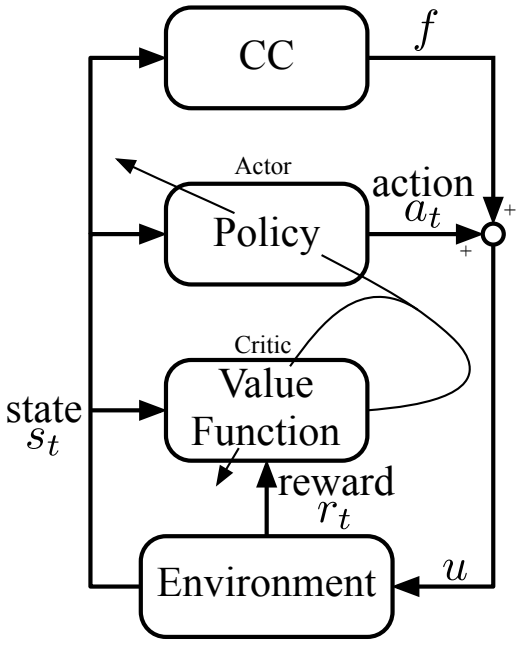


Figure 9. Learning diagram for RL with fixed-gain control

where $f(\boldsymbol{s})$ is a fixed-gain input that is a function of the state, $\boldsymbol{s}$. The specific design of $f(\boldsymbol{s})$ depends on the system and goal of the controller. For instance, it could be designed such that it could function on its own to provide satisfactory (though suboptimal) performance, or it can be designed to perform well for only a subset of the desired operation space for the system.

The RL component of the controller will still need to interact with the fixed-gain component and modify the controller output to optimize the combined control policy. This provides the designer with the ability to design a simple controller that can provide satisfactory performance while leaving the more difficult optimal control problem to be solved by the RL agent. The training process for this type of controller is illustrated by a modified actor-critic training diagram in Figure 9. The action output from the policy, a_t, is combined with the output from the conventional controller, f. The sum of the outputs, u, is the input to the environment. However, the value function only evaluates the action, a_t, not the total input to the environment. Since the

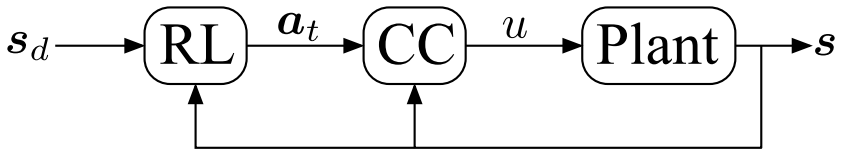


Figure 10. Block diagram for RL continuous gain-scheduling control structure

value function and policy are unaffected by the conventional control block, the addition of the fixed-gain term is equivalent to changing the environment.

2.1.3 *Agent-Driven Gain Scheduling*

In the fixed-gain and RL controller architecture, the agent is unable to modify the component of the controller designed using domain knowledge. In addition to this controller architecture, the combined controller can be designed to have the agent interact directly with the domain-knowledge controller. One way to realize this direct interaction is by using a gain-scheduled controller for which the agent learns the scheduling law. This is shown by the block diagram in Figure 10, where the controller output, u, is:

$$u = \boldsymbol{a}_t \boldsymbol{s} \tag{19}$$

and $\boldsymbol{a}_t$ is a vector of the agent outputs, $\boldsymbol{a}_t = [a_t^{(1)}, a_t^{(2)}, \ldots, a_t^{(n)}]$, corresponding to a feedback controller with n states. Implementation of the agent as a gain scheduling mechanism allows for domain knowledge to be used to design the feedback controller. It may also improve interpretability, since the response at each time step can be understood as a system subject to a linear feedback controller. Figure 11 shows the learning process for the gain-scheduled RL controller. The set of actions, a_t, is a set of gains that are used to update the gain-scheduled feedback controller in the CC block. The output, u, from the controller is passed to the environment and causes its state, s_t, to change. The reward is used to update the value function so that the gain update law can be optimized. This controller architecture provides a more integrated combination

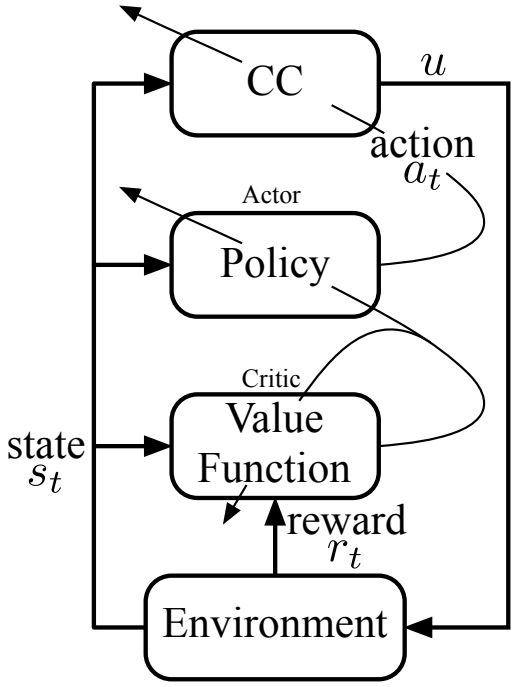


Figure 11. Learning diagram for RL with gain scheduled control

between the parts of the controller designed using domain knowledge and the parts learned by RL.

Training RL agents requires selecting training parameters such as the RL algorithm, neural network architecture, and episode length. The training parameters were tuned for each benchmark problem and will be introduced for each system alongside each set of results. Additional information about the mechanical parameters for each benchmark system and the hyperparameters used for training are found in Appendix A.

2.2 Duffing Oscillator Benchmark System

2.2.1 *Problem Description*

A duffing oscillator model was used as a simple benchmark model to illustrate the usefulness of combined RL and conventional control methods. The model is shown in Figure 12, where m is the mass, c is the damping coefficient, x is the displacement,

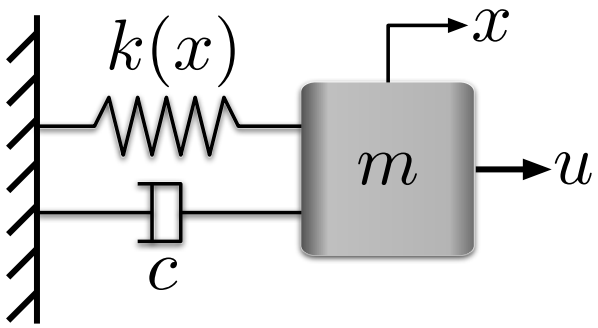


Figure 12. Duffing oscillator model

$k(x)$ is the nonlinear spring stiffness, and u is the force input. The equation of motion for this system is:

$$m\ddot{x} + c\dot{x} + k(x)x = u \tag{20}$$

where the nonlinear stiffness is $k(x) = \alpha + \beta x^2$. The displacement is constrained between $-1.5\text{m} \leq x \leq 1.5\text{m}$. The desired behavior of the system is to move the mass from rest at $x = 0$ to a desired displacement, x_d, while limiting oscillation amplitude. The reward function used to accomplish this was:

$$r = -(x_d - x)^2 \tag{21}$$

Although this reward function is simple, it accomplishes the desired task by requiring the position error to be quickly minimized in order to maximize reward.

The previously described combined control structures were applied to the duffing oscillator. The Pure RL controller uses a force output as the action applied to the system. For the combined controllers, a fixed-gain component was designed to provide a constant force that stabilizes the system about the desired displacement, x_d. The force which stabilizes the system at the desired equilibrium is:

$$f(\boldsymbol{s}) = \alpha x_d + \beta x_d^3 \tag{22}$$

The response of the duffing oscillator with this fixed-gain controller and desired displacement of $x_d = 1$ is shown in Figure 13. Although the fixed-gain controller causes

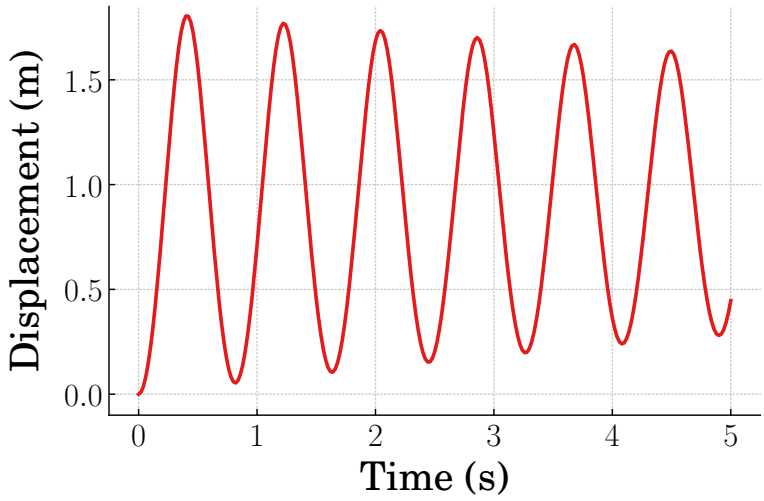


Figure 13. Duffing oscillator response with fixed-gain term

the system to move to the desired displacement, the settling time is high. In the combined controllers, the RL components are responsible for modifying the fixed-gain component to optimize the response. This fixed-gain controller was used for two combined controllers, one that uses a single scalar action and another that uses a gain-scheduled controller. In summary, the set of controllers tested on the duffing oscillator are:

$$\text{Pure RL:} \qquad u = a_t \tag{23}$$

$$\text{RL-PD:} \qquad u = \alpha x_d + \beta x_d^3 - a_t^{(1)}(x - x_d) - a_t^{(2)}\dot{x} \tag{24}$$

$$\text{RL-LA:} \qquad u = \alpha x_d + \beta x_d^3 + a_t \tag{25}$$

Pure RL is the baseline controller that uses the action, a_t, from RL as the only input to the system. RL-PD uses the fixed-gain controller in (22) in addition to a Proportional-Derivative (PD) controller, where the actions $a_t^{(1)}$ and $a_t^{(2)}$ are gains for the displacement error and velocity of the system. RL-Lumped Action (RL-LA) uses the fixed-gain controller with a lumped action, or single scalar action.

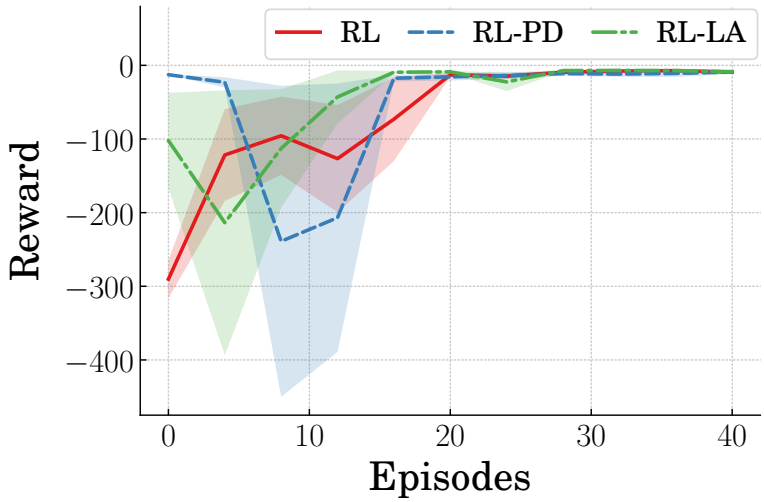


Figure 14. Mean duffing oscillator reward during training

2.2.2 *Baseline Performance*

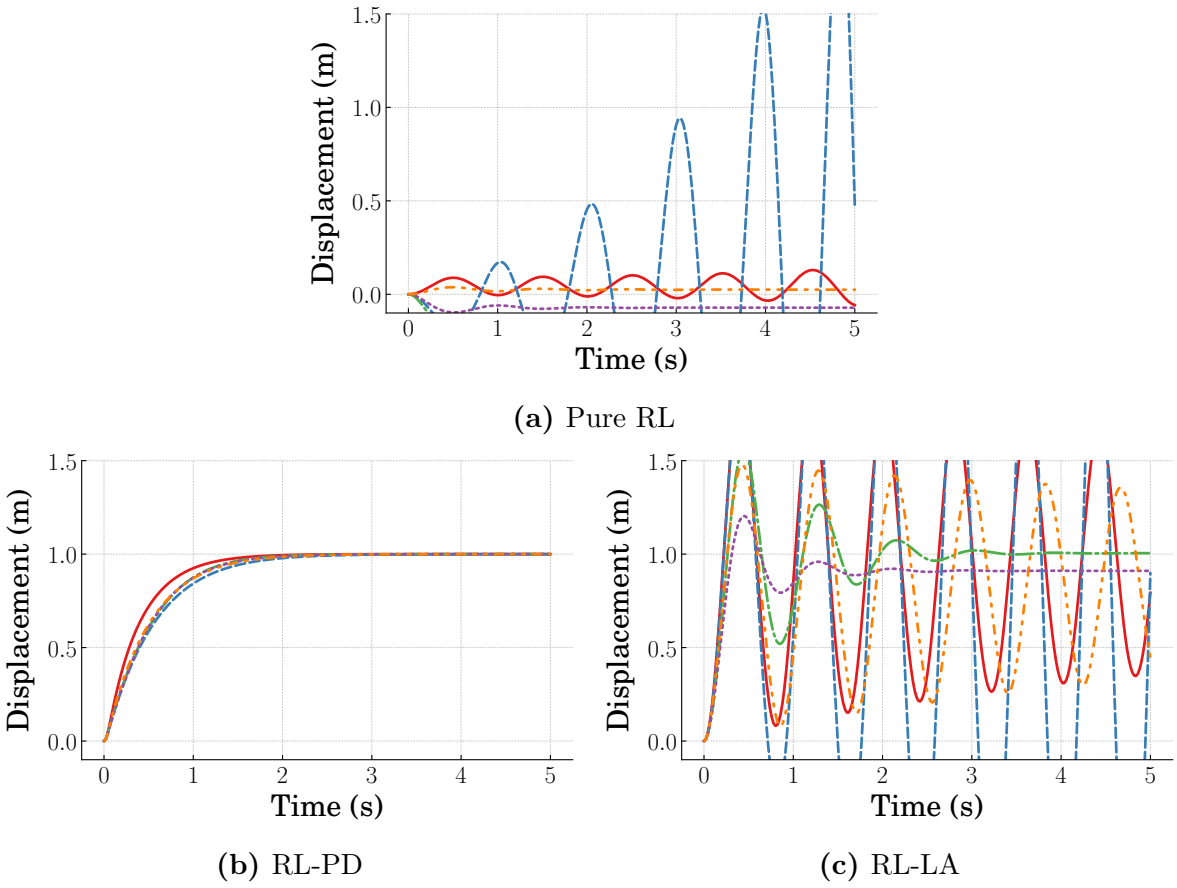
The agents for each controller were trained using the Twin Delayed DDPG (TD3) algorithm [31]. All neural networks had two hidden layers, containing 400 and 300 nodes, respectively, and used the rectified linear unit (RELU) activation function as was used in the original TD3 paper. The observations included measurements of the displacement, x, and velocity, $\dot{x}$, of the mass, analogous to full-state feedback for a conventional controller, as well as the desired displacement, x_d. Training was conducted over 40 episodes. The desired displacement was randomized at the initialization of each episode. Since RL relies on stochasticity, five agents were trained for each control structure. The same set of five initial seeds was used for the different control structures. After training the agents, they were tested to compare the performance of the controllers without model-based components to those with model-based and RL components.

The performance of the agents during training was evaluated at fixed intervals using the reward received from the system starting at rest with a desired displacement of $x_d = 1\text{m}$. The reward trends during training are shown in Figure 14, where the lines are the mean reward, and the shaded regions are standard deviation for the five agents

of each controller type. The Pure RL controllers begin training with the lowest reward whereas the RL-PD controllers begin with the highest reward. Although the reward from the RL and RL-LA controllers tend to increase after the beginning of training, the reward from RL-PD initially significantly decreases. This is due to one of the responses becoming unstable, which skews the mean and standard deviation. The mean rewards from the different controllers converge to approximately the same value after 20 episodes.

Time responses from the agents at the beginning of training are shown in Figure 15. The low reward at the beginning of training with Pure RL was caused by one of the responses being unstable and the steady-state error from the other responses resulting from the inability of the controller to force the system to reach the desired displacement, x_d, as shown in Figure 15a. Since the combined controllers have the fixed-gain component, (22), they cause the system to move near the desired displacement. The responses with RL-PD in Figure 15b do not have overshoot and tend to have shorter settling times than the other controllers. Since the controller output of the gain-scheduled PD component is zero at the desired displacement, the combination of the fixed-gain component and PD component in RL-PD results in zero steady-state error. Similar to RL-PD, the RL-LA responses in Figure 15c settle near the desired displacement. However, the responses have higher oscillation amplitude and some responses have steady-state error. These responses show that the combined controllers have better performance than the Pure RL controller even before training.

Time responses of the duffing oscillator after the agent controllers were trained are shown in Figure 16. The responses with Pure RL in Figure 16a settled near the desired displacement at $x_d = 1$m with low oscillation amplitude. However, all of the responses have steady-state error. The RL-PD responses in Figure 16b have zero steady-state error just as the responses before training. However, one response has overshoot and another response has a long rise time. The RL-LA responses in



(a) Pure RL

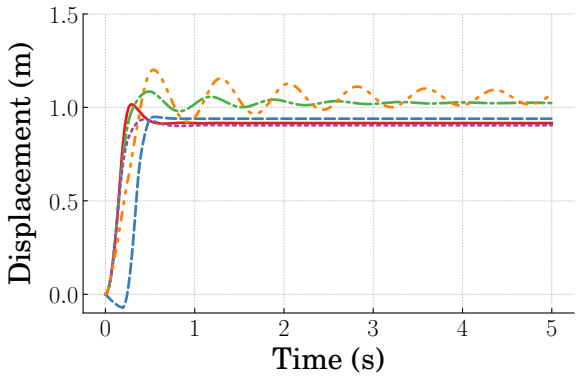
(b) RL-PD

(c) RL-LA

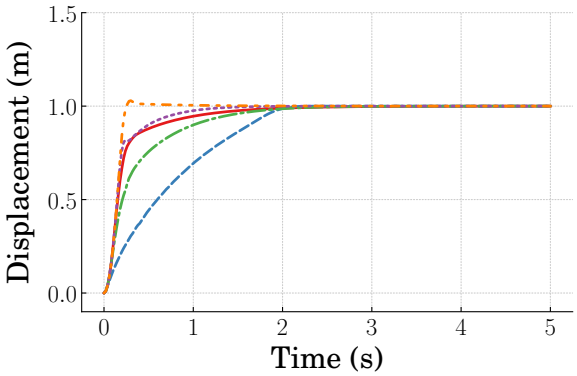
Figure 15. Duffing oscillator responses before training

Figure 16c have short rise times; however, they have steady-state error and two of the responses have long settling times due to oscillation. The response characteristics of the Pure RL and combined controllers are summarized in Table 1. The RL-PD controllers had the best performance in all categories. Pure RL has higher mean steady-state error than RL-LA, however, RL-LA has much longer settling time and higher overshoot.

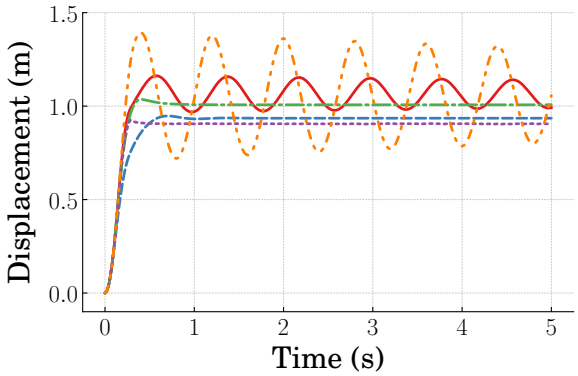
Although the Pure RL controller was not designed with any domain knowledge, it still provides good performance after training. RL-PD had the best performance overall. Its controller architecture also relied more heavily on domain knowledge since it included both a fixed-gain component and a state-feedback control component for which the agents only had to learn appropriate parameters. The responses from the RL-LA controllers have slightly lower steady-state error than the Pure RL controllers



(a) Pure RL



(b) RL-PD



(c) RL-LA

Figure 16. Duffing oscillator responses after training

due to the fixed-gain component; however, settling times for RL-LA were inconsistent.

2.3 Double-Pendulum Crane

2.3.1 *Problem Description*

A planar double-pendulum crane, shown by the model in Figure 17, was used as a second benchmark system for the combined RL controllers. The double-pendulum crane is a three degree-of-freedom underactuated system. This model approximates the dynamics of physical cranes where a payload is fastened to the crane by a hook with non-negligible mass. The payload is manipulated by moving a crane trolley on a horizontal track. The displacement of the trolley along the track is described by x. The angular displacements of the hook and payload pendulums are θ_1 and θ_2, respectively,

Table 1. Duffing Oscillator Response Characteristics

		Settling Time	Steady-state Error	Percent Overshoot
RL	Mean	2.212s	0.064m	6.047%
	SD	2.885s	0.025m	7.702%
RL-PD	Mean	**1.372**s	**0.0**m	**0.5**%
	SD	**0.598**s	**0.0**m	**1.174**%
RL-LA	Mean	14.62s	0.0587m	11.839%
	SD	18.672s	0.0283m	15.004%

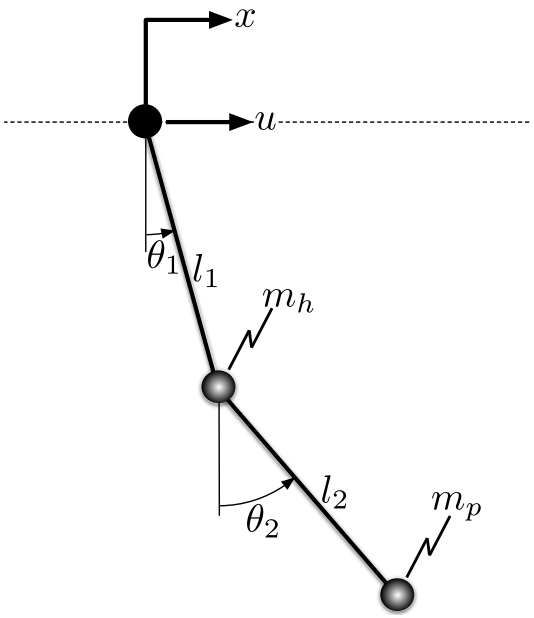


Figure 17. Double-pendulum planar crane model

both measured from the vertical axis of the Newtonian frame. The lengths of the pendulums, commonly called hoist length and rigging length, are l_1 and l_2. The hook mass is m_h, and the payload mass is m_p. The input, u, is desired acceleration of the trolley. The goal of the controllers for this system was to move the trolley to a desired displacement while minimizing oscillation of the pendulums. The equations of motion

for the double-pendulum crane are:

$$\begin{bmatrix} 1 & 0 & 0 \\ l_1\cos(\theta_1)(m_h+m_p) & l_1^2(m_h+m_p) & l_1 l_2 m_p\cos(\theta_1-\theta_2) \\ l_2 m_p\cos(\theta_2) & l_1 l_2 m_p\cos(\theta_1-\theta_2) & l_2^2 m_p \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}$$

$$= \begin{bmatrix} u \\ -l_1 l_2 m_p \sin(\theta_1-\theta_2)\dot{\theta}_2^2 - l_1 g \sin(\theta_1)(m_h+m_p) \\ -l_1 l_2 m_p \sin(\theta_2-\theta_1)\dot{\theta}_1^2 - l_2 g m_p \sin(\theta_2) \end{bmatrix} \quad (26)$$

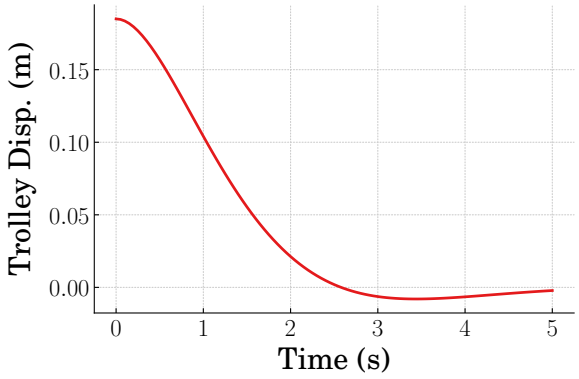
The acceleration of the crane trolley, $\ddot{x}$, is determined by only the control output, u. The dynamics of the pendulums are coupled with each other and the acceleration of the trolley. The incremental reward function for this system is:

$$r = -\omega_x x^2 - \omega_{\theta_1}\theta_1^2 - \omega_{\theta_2}\theta_2^2 \quad (27)$$

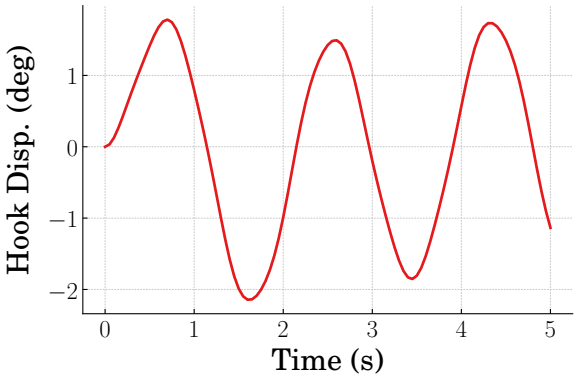
where ω_x, ω_{θ_1}, and ω_{θ_2} are weights designed to normalize the reward terms to have approximately equal importance during training.

The Pure RL, fixed-gain and RL, and gain-scheduled RL control elements were applied to this system. The fixed-gain term, $f(\boldsymbol{s}) = k_p x + k_d \dot{x}$, is a PD controller, where the gains, k_p and k_d, are designed to move the trolley to the desired displacement without input saturation and with a damping coefficient of $\zeta = \frac{\sqrt{2}}{2}$. Time responses of the crane subject to the fixed-gain controller are shown in Figure 18. This movement excites unwanted oscillation in the pendulums, shown in Figure 18b and Figure 18c. Since the fixed-gain controller does not account for motion of the pendulums, the agent must modify the control output to limit hook and payload oscillation.

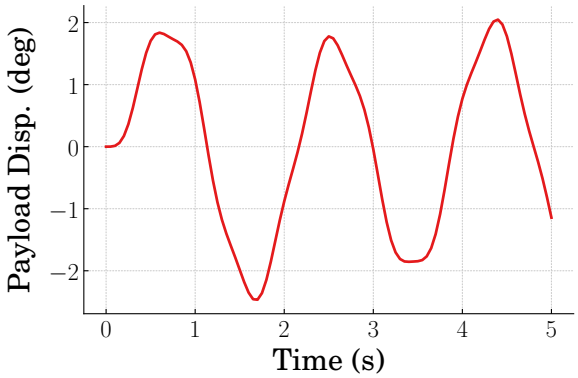
Multiple combined controllers consisting of fixed-gain, gain-scheduled, and lumped action terms were tested on the benchmark double-pendulum crane. While the fixed-gain term only accounts for the trolley position and velocity, the gain scheduling term only accounts for pendulum motion. Because of this, one combined controller uses the fixed-gain control element combined with the gain-scheduled control element.



(a) Trolley displacement response



(b) Hook displacement response



(c) Payload displacement response

Figure 18. Crane responses from fixed-gain term

Another combined controller uses the fixed-gain terms with a lumped action term learned by the agent. Additionally, a controller combining the features of all the above controllers was tested, where the control output is determined by a fixed-gain, gain-scheduled, and lumped-action control elements. In summary, the control laws for the crane are:

$$\text{Pure RL:} \qquad u = a_t \tag{28}$$

$$\text{RL-PD:} \qquad u = k_p x + k_d \dot{x} + \boldsymbol{a_t \theta} \tag{29}$$

$$\text{RL-LA:} \qquad u = k_p x + k_d \dot{x} + a_t \tag{30}$$

$$\text{RL-PD-LA:} \qquad u = k_p x + k_d \dot{x} + \boldsymbol{a_t \theta} + a_t \tag{31}$$

where Pure RL is the baseline RL controller that does not utilize any domain

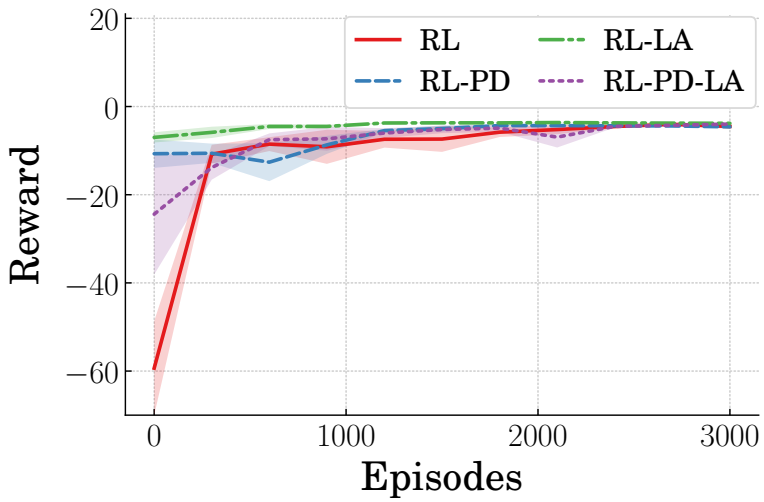


Figure 19. Mean crane reward during training for $x(0) = 0.185$m

knowledge. RL-PD is the controller with the fixed-gain term combined with a gain-scheduled term, where $\boldsymbol{a}_t$ are gains assigned by the agent and are updated at every time step, and $\boldsymbol{\theta}$ are the states describing the motion of the pendulums. RL-Lumped-Action (RL-LA) uses the fixed-gain control element with a single (lumped) action from the agent. RL-PD-LA combines fixed-gain, gain-scheduled, and lumped action control elements.

2.3.2 *Baseline Performance*

The TD3 algorithm [31] was used to train five agents for each controller using Pure RL, RL-PD, RL-LA, and RL-PD-LA. The same set of five initial seeds was used for the different controllers. Each episode during training was initialized with the crane at rest with a trolley displacement of $x(0) = 0.185$m and hook and payload angular displacements of $\theta_1 = \theta_2 = 0$. The desired final displacement of the trolley was $x(t_f) = 0$m. Trends during training are shown to evaluate the baseline performance of the controller types. Figure 19 shows the mean reward earned from the five trained agents of each controller type. The shaded region shows the standard deviation of the rewards at each sampled episode of training. The untrained performance of the combined controllers at 0 episodes is better than that of the controller using Pure RL.

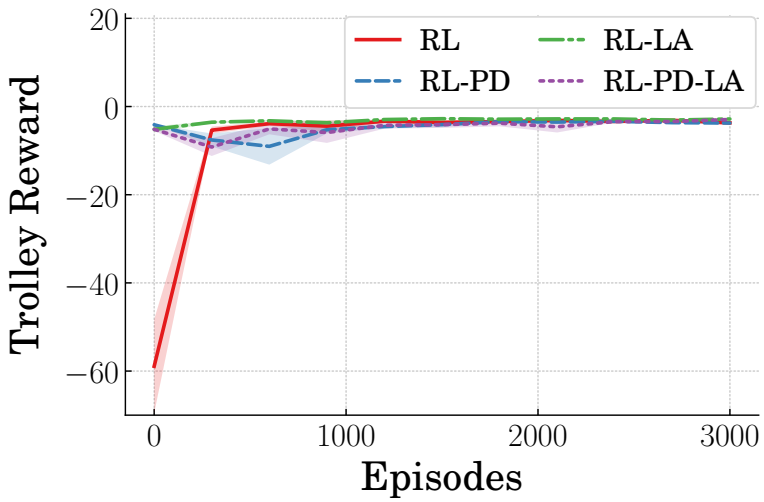


Figure 20. Mean reward from trolley during training

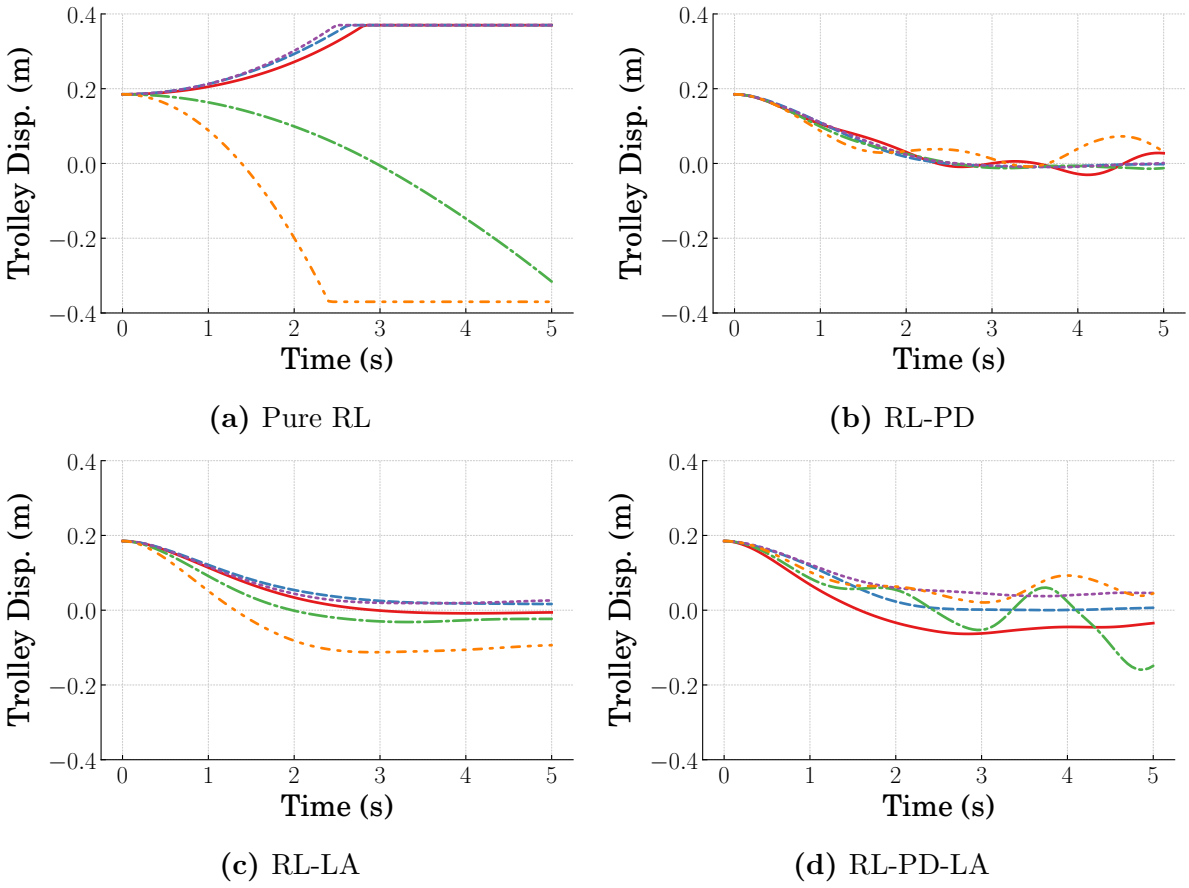
However, near the end of training, at 3000 episodes, the rewards earned by each controller are approximately equal.

The higher initial reward from the combined controllers is due to the fixed-gain term commanding the trolley to move to the desired location, whereas more training is necessary for the agent of the Pure RL controller to learn to move the trolley to the desired location. This can be seen by isolating the contribution of the trolley response to the total reward, where the reward contribution from the trolley is:

$$r_x = -\omega_x x^2 \tag{32}$$

The reward from the trolley is shown in Figure 20. Although the combined controllers start with higher reward from the trolley displacement, the agents for the Pure RL controller learn to increase the trolley reward after training for only a short time. The trolley reward for RL-LA tends to be the highest for the controllers tested.

Figure 21 shows the trolley displacement responses for the different controllers before training. The crane begins at rest with initial trolley displacement of $x(0) = 0.185$m and pendulum displacements $\theta_1 = \theta_2 = 0$. The trolley responses for Pure RL do not settle near the equilibrium and instead diverge towards the limits of the workspace, as shown in Figure 21a. The responses for RL-PD in Figure 21b have the



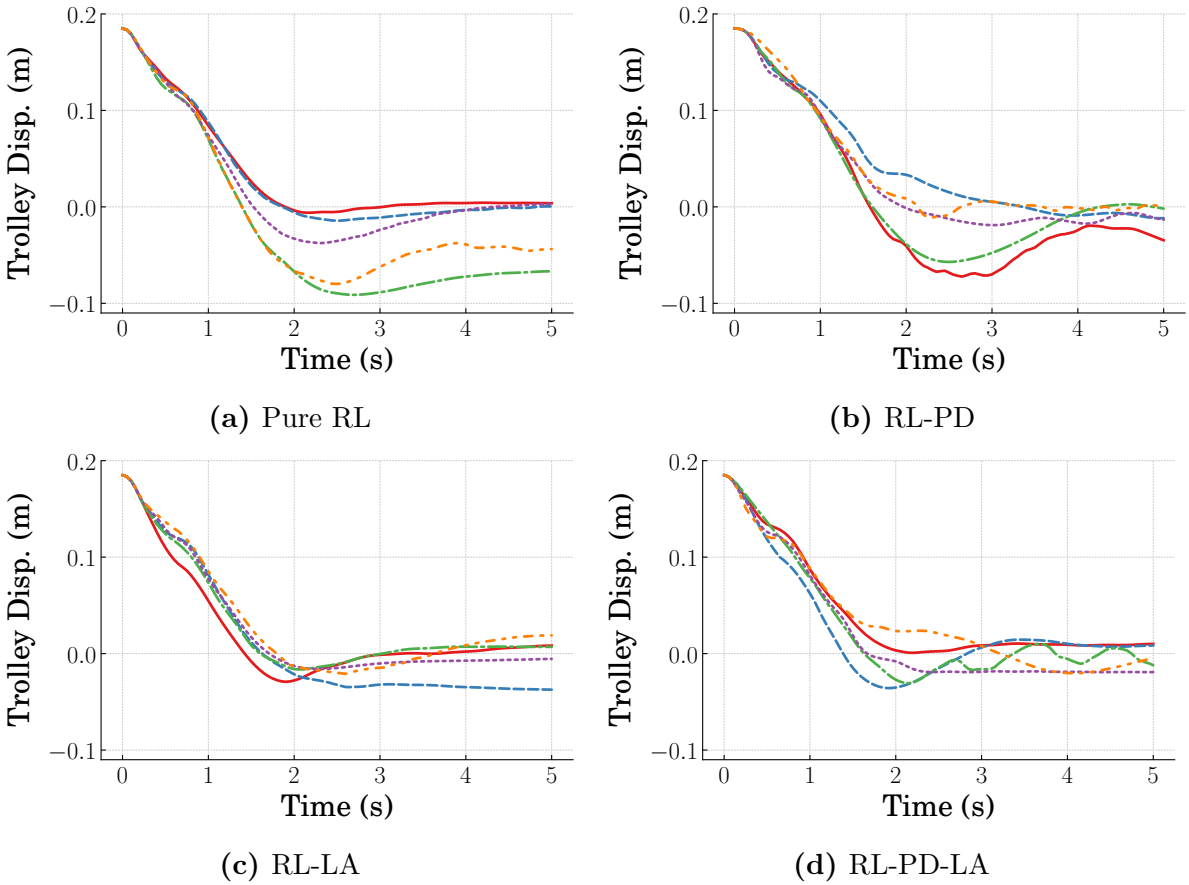
(a) Pure RL **(b)** RL-PD

(c) RL-LA **(d)** RL-PD-LA

Figure 21. Crane trolley responses for $x(0) = 0.185$m before training

best trolley performance, where most of the responses converge near the desired displacement with low oscillation amplitude due to the fixed-gain terms. The RL-LA responses in Figure 21c have higher error in trolley displacement than RL-PD but have lower oscillation amplitude. The RL-PD-LA responses in Figure 21d have less trolley displacement error than RL-LA, but some of the responses have high oscillation amplitude.

Figure 22 shows the trolley displacement responses for the different controllers after training, where the crane begins at rest with initial trolley displacement of $x(0) = 0.185$m and pendulum displacements $\theta_1 = \theta_2 = 0$. Three of the five Pure RL agents shown in Figure 22a can bring the trolley to the desired displacement within the allotted time, whereas two responses do not converge to zero displacement. The RL-PD



(a) Pure RL **(b)** RL-PD

(c) RL-LA **(d)** RL-PD-LA

Figure 22. Crane trolley responses for $x(0) = 0.185$m after training

responses in Figure 22b tend to have higher oscillation amplitude than the other controllers. Figure 22c shows that the RL-LA responses tend to be able to converge closer to zero displacement than Pure RL. The RL-PD-LA responses in Figure 22d are also able to converge to be close to the desired displacement at $x(t_f) = 0$m, but exhibit some oscillation. In both the cases before training and after training, the controllers utilizing a continuous gain scheduled term, RL-PD and RL-PD-LA, tend to exhibit oscillation near the desired displacement, whereas the controllers using a lumped action, Pure RL and RL-LA, have much lower oscillation amplitude. For the responses after training, the controllers that include a fixed-gain term and a lumped action, RL-LA and RL-PD-LA, tend to converge closer to the desired equilibrium and have lower steady state-error. The response characteristics of the trolley are summarized in Table 2.

Table 2. Trolley Response Characteristics for $x(0) = 0.185$m

		Amplitude	Steady-state Error	Percent Overshoot
RL	Mean	0.0005m	0.027m	24.8%
	SD	0.0007m	0.025m	18.5%
RL-PD	Mean	0.004m	0.024m	21.6%
	SD	0.003m	0.017m	17.4%
RL-LA	Mean	**0.0**m	**0.02**m	**13.9**%
	SD	**0.0**m	**0.014**m	**5.9**%
RL-PD-LA	Mean	0.007m	0.024m	28.2%
	SD	0.01m	0.029m	34%

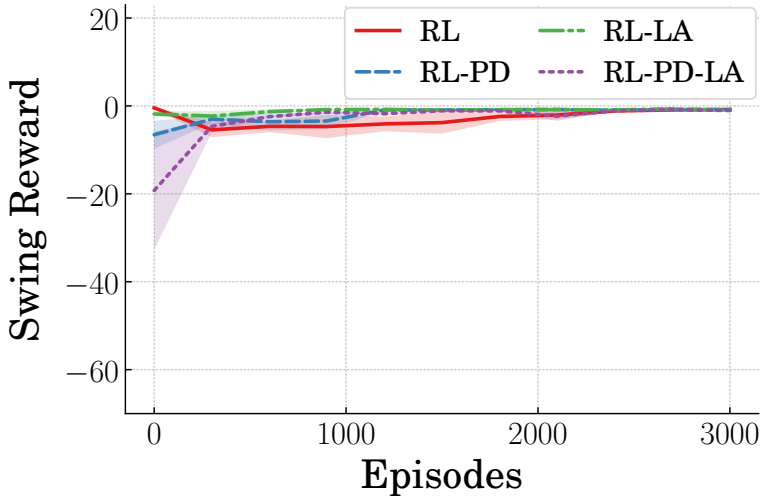
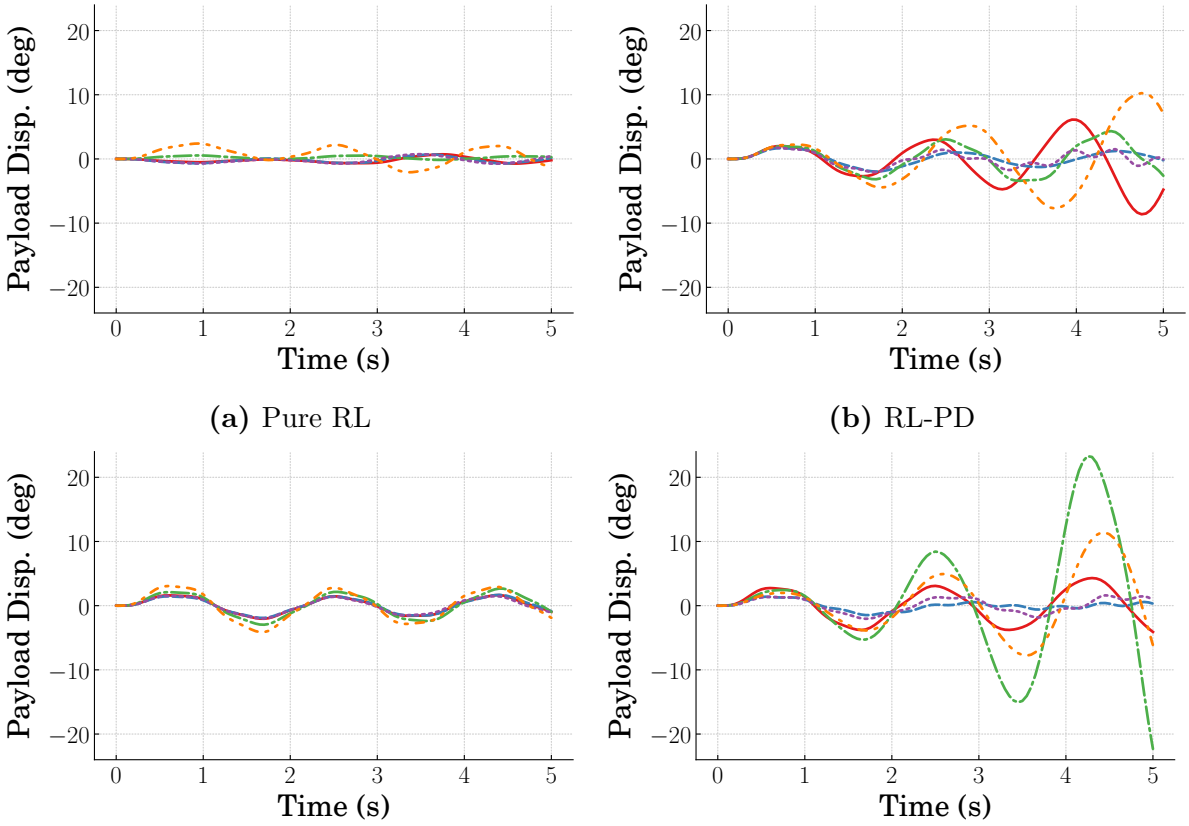


Figure 23. Mean reward from hook and payload during training

RL-LA has the lowest values for all of the performance metrics shown. Although RL-PD has lower steady-state error and overshoot than Pure RL, it has higher residual oscillation amplitude. RL-PD-LA has lower steady-state error than Pure RL, but has higher values for residual oscillation amplitude and overshoot. In general, the combined controllers tend to have better performance than Pure RL for steady-state error and percent overshoot. However, RL-PD and RL-PD-LA have higher residual oscillation amplitudes due to the gain scheduling terms on the pendulum states.

The previous evaluation of the contribution of the trolley responses to the reward is not adequate to understand the performance of the agents. Therefore, the



(a) Pure RL **(b)** RL-PD

(c) RL-LA **(d)** RL-PD-LA

Figure 24. Crane payload responses for $x(0) = 0.185$m before training

reward contribution from the pendulum responses also needs to be analyzed. The reward contribution from the oscillation of the hook and payload is:

$$r_{\theta_1,\theta_2} = -\omega_{\theta_1}\theta_1^2 - \omega_{\theta_2}\theta_2^2 \tag{33}$$

The reward trends from the pendulum contributions are shown in Figure 23. At the initialization of training, Pure RL begins with the highest reward followed by RL-LA with a slightly lower reward. The RL-PD and RL-PD-LA controllers have the lowest swing angle reward. The time responses of the payload before training are shown in Figure 24. The crane began at rest with an initial trolley displacement of $x(0) = 0.185$m. The responses with Pure RL in Figure 24a and RL-LA in Figure 24c have relatively low oscillation amplitude. The RL-PD responses in Figure 24b and RL-PD-LA responses in Figure 24d have higher amplitude oscillation, with some

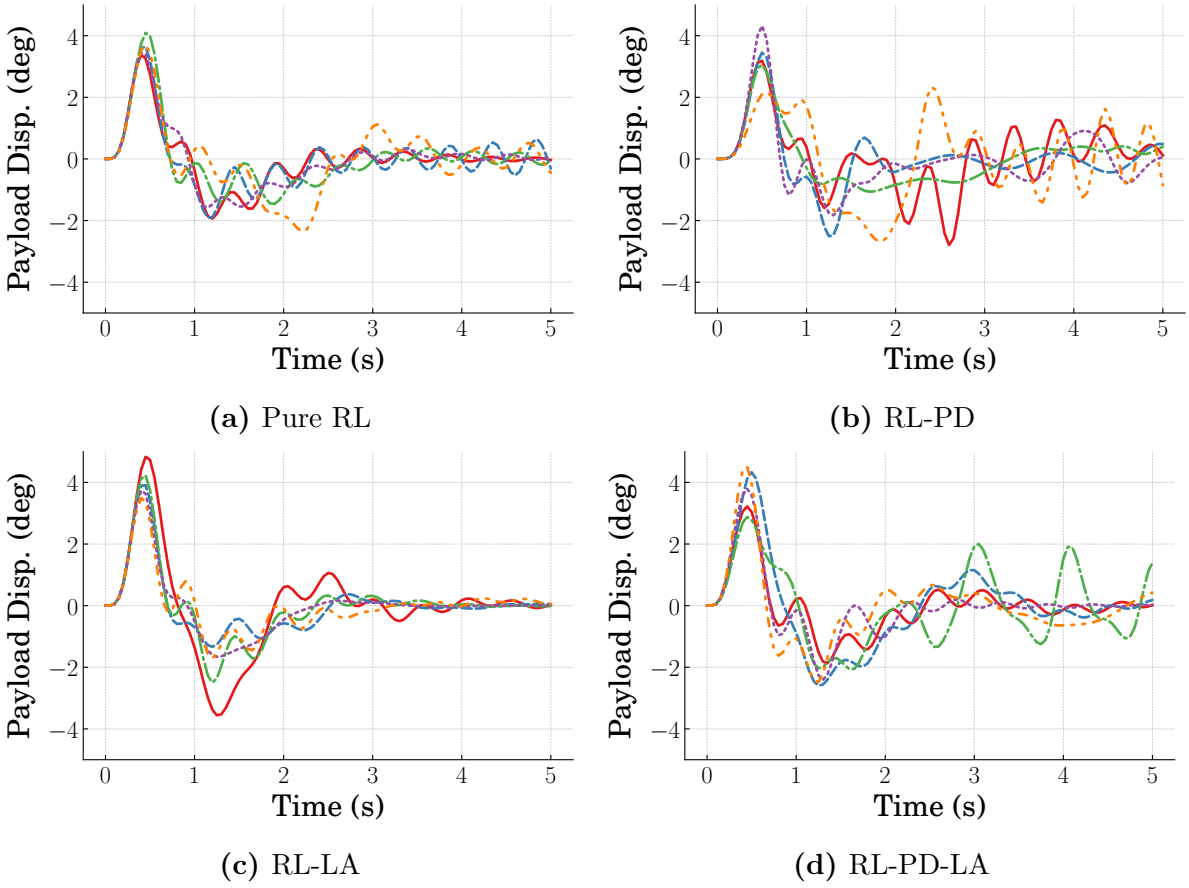
Table 3. Payload Residual Amplitude for $x(0) = 0.185$m

		Amplitude
RL	Mean	0.474°
	SD	0.481°
RL-PD	Mean	0.896°
	SD	0.23°
RL-LA	Mean	**0.015°**
	SD	**0.03°**
RL-PD-LA	Mean	0.756°
	SD	0.947°

responses increasing amplitude with time. This corresponds with what was seen for the trolley responses at the start of training, where the controllers using agent-driven gain scheduling tended to have higher oscillation amplitude.

Figure 25 shows the time responses of the payload after training has been completed. The controllers that include a single lumped action term tend to have lower oscillation amplitude. This can be seen for Pure RL in Figure 25a, RL-LA in Figure 25c and RL-PD-LA in Figure 25d. RL-LA has the lowest oscillation amplitude, whereas RL-PD in Figure 25b has the highest oscillation amplitude. This is consistent with the previous figures showing that the controllers that include a gain scheduling term tend to have higher oscillation amplitude. This is summarized in Table 3, where RL-LA had the lowest mean residual oscillation amplitude. Although the mean oscillation amplitude for RL-PD-LA is the second highest, this was primarily due to one response skewing the mean.

The RL-LA controllers and RL-PD-LA controllers have the best overall performance in terms of limiting steady-state error and oscillation amplitude. They both tend to have lower trolley displacement error, where RL-PD-LA has lower error than RL-LA. However, RL-PD-LA achieves lower trolley steady-state error at the expense of higher pendulum oscillation. The Pure RL controllers tend to have the most



(a) Pure RL **(b)** RL-PD

(c) RL-LA **(d)** RL-PD-LA

Figure 25. Crane payload responses for $x(0) = 0.185$m after training

trolley steady-state error since they do not have the benefit of the fixed-gain term to move the trolley towards zero displacement.

The results for the double-pendulum crane shown above were acquired from responses with initial trolley displacements equal to the one used for training. The fixed initial displacement of $x(0) = 0.185$m during training caused the agents to have the most experience for trolley displacements $0 \leq x \leq 0.185$m. To show how the agents perform for initial displacements other than the one used for training, the following results show how the agents that were trained with an initial displacement of $x(0) = 0.185$m perform for an initial trolley displacement of $x(0) = -0.185$m instead. The reward trends from this initial trolley displacement are shown in Figure 26. The initial rewards at the beginning of training are similar to those for initial displacements

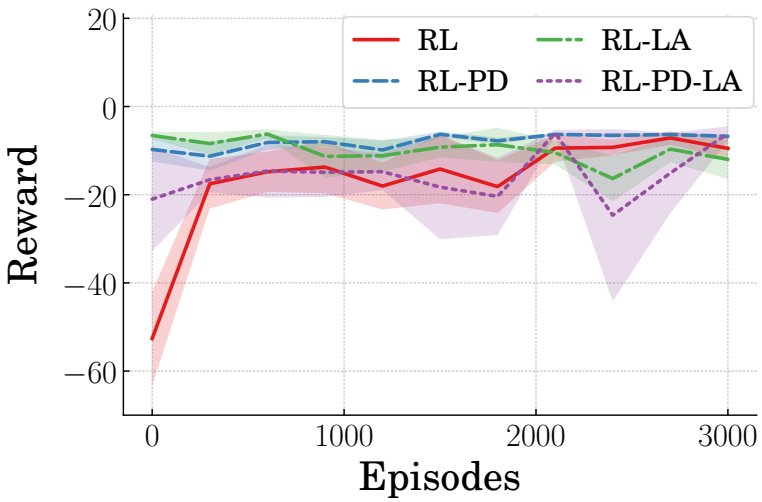
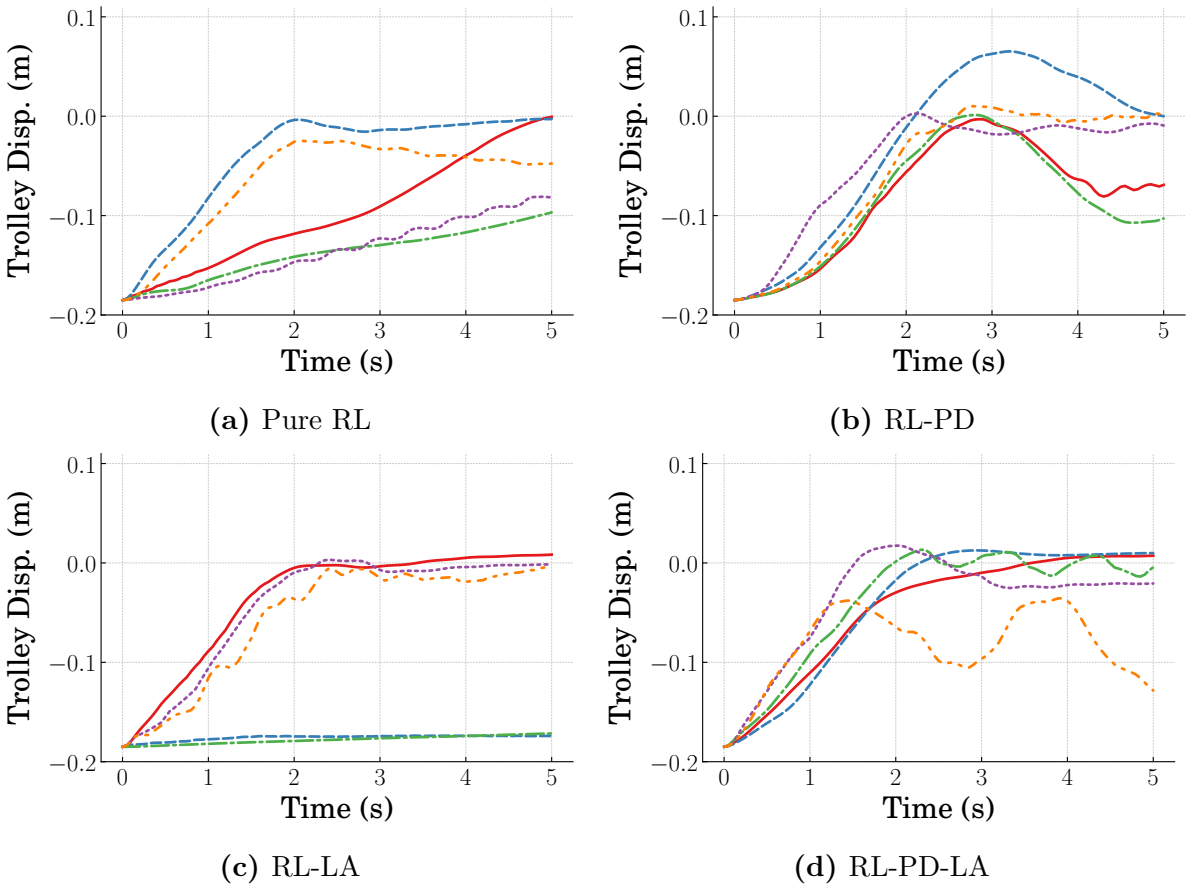


Figure 26. Mean crane reward during training for $x(0) = -0.185$m

of $x(0) = 0.185$m in Figure 19. The fixed-gain terms cause the rewards from the combined controllers to begin higher than that of Pure RL. However, while the performance of the Pure RL controllers showed improvement from the initial reward, the other controllers do not show clear trends in improved performance.

Figure 27 shows the trolley response from $x(0) = -0.185$m for the agents trained from $x(0) = 0.185$m. Two responses with the Pure RL controller in Figure 27a converge quickly towards the desired displacement at $x(t_f) = 0$m; however, three responses converge slowly. Although the agents have no experience starting from this initial displacement, the combined controller responses tend to have lower trolley error due to the contribution from the fixed-gain terms. Several of the RL-PD responses in Figure 27b quickly converge towards the desired displacement, but two responses move away from it again. Three of the RL-LA responses in Figure 27c converge to be near the desired displacement; however, two of the responses stay near the initial displacement. Similarly, most of the RL-PD-LA responses in Figure 27d converge and stay near the desired displacement.

The response characteristics of the performance of the trolley from $x(0) = -0.185$m are summarized in Table 4. The RL-LA responses have the lowest



(a) Pure RL **(b)** RL-PD

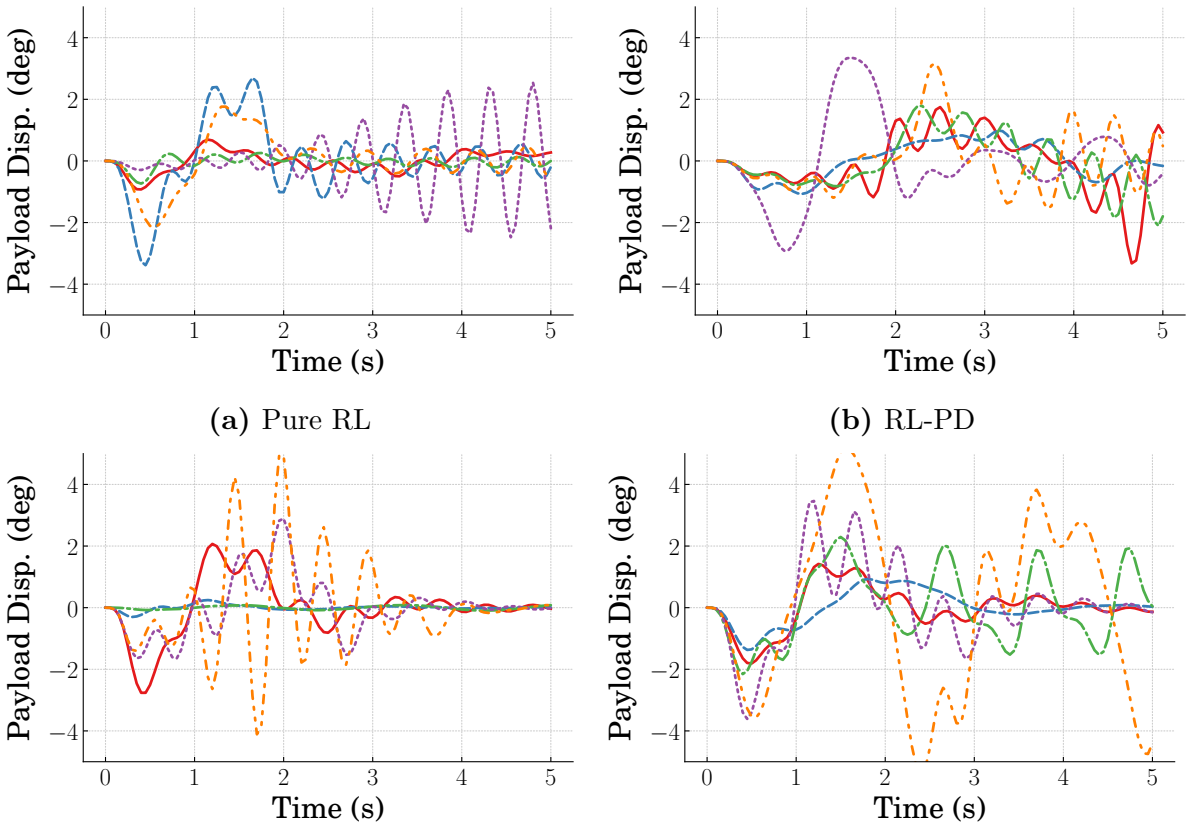
(c) RL-LA **(d)** RL-PD-LA

Figure 27. Crane trolley responses for $x(0) = -0.185$m after training

Table 4. Trolley Response Characteristics for $x(0) = -0.185$m

		Amplitude	Steady-state Error	Percent Overshoot
RL	Mean	0.00025m	0.028m	**3.05**%
	SD	0.0002m	0.025m	**2.67**%
RL-PD	Mean	0.004m	**0.025**m	8.57%
	SD	0.003m	**0.017**m	13.5%
RL-LA	Mean	**0.0**m	0.046m	6.27%
	SD	**0.0**m	0.064m	4.31%
RL-PD-LA	Mean	0.006m	0.028m	5.76%
	SD	0.008m	0.035m	3.16%

mean residual oscillation amplitude; however, they have the highest mean steady-state error. RL-PD has the lowest mean steady-state error, but it also has the highest mean percent overshoot. Although Pure RL and RL-PD-LA have the same mean steady-state error, Pure RL has the lowest mean percent overshoot. These results suggest that the



(a) Pure RL **(b)** RL-PD

(c) RL-LA **(d)** RL-PD-LA

Figure 28. Crane payload responses for $x(0) = -0.185$m after training

combined controllers do not have a clear advantage over the Pure RL controller when operating in a region of the state-space that was not experienced during training. This may be caused by the higher variance in the responses compared to the responses starting at $x(0) = 0.185$m.

The payload responses corresponding to initial trolley displacements of $x(0) = -0.185$m are shown in Figure 28. Although one of the Pure RL payload displacement responses in Figure 28a grows significantly, the other responses have low residual oscillation amplitude. All of the RL-PD responses in Figure 28b have residual oscillation. Although two of the trolley responses from RL-LA did not converge to the desired displacement, all of the trained agents in Figure 28c were able to significantly reduce the payload oscillation. This is similar for RL-PD-LA in Figure 28d, where the

Table 5. Payload Residual Amplitude for $x(0) = -0.185$m

		Amplitude
RL	Mean	0.412%
	SD	0.482%
RL-PD	Mean	0.906%
	SD	0.234%
RL-LA	Mean	**0.001**%
	SD	**0.002**%
RL-PD-LA	Mean	0.743%
	SD	0.925%

payload oscillation amplitude tends to settle to be lower than that of the other controllers. However, the payload response shown by the orange dashed line retains significant oscillation amplitude. This response corresponds to the trolley response that did not remain near the equilibrium in Figure 27d. The residual oscillation amplitudes of the responses are summarized in Table 5. The oscillation mitigation when starting at $x(0) = -0.185$m follows the same trend as when starting at $x(0) = 0.185$m. The RL-LA controllers have the lowest mean residual oscillation amplitude, whereas both controller architectures that use agent-driven gain scheduling, RL-PD and RL-PD-LA, have the highest mean residual oscillation amplitude.

2.3.3 *Planar Crane Experiments*

It is common to train agents in simulation primarily because it is faster and safer than training on physical systems. Since modeling error always exists between models and physical systems, the agents will often not perform as expected when implemented on physical systems. This section presents results from simulation-trained agents on a small-model physical crane. The performance of the Pure RL controllers were compared with the performances of the combined controllers.

The crane used for experimental verification is shown in Figure 29. The displacement range used for the trolley in simulation was modeled from the length of the

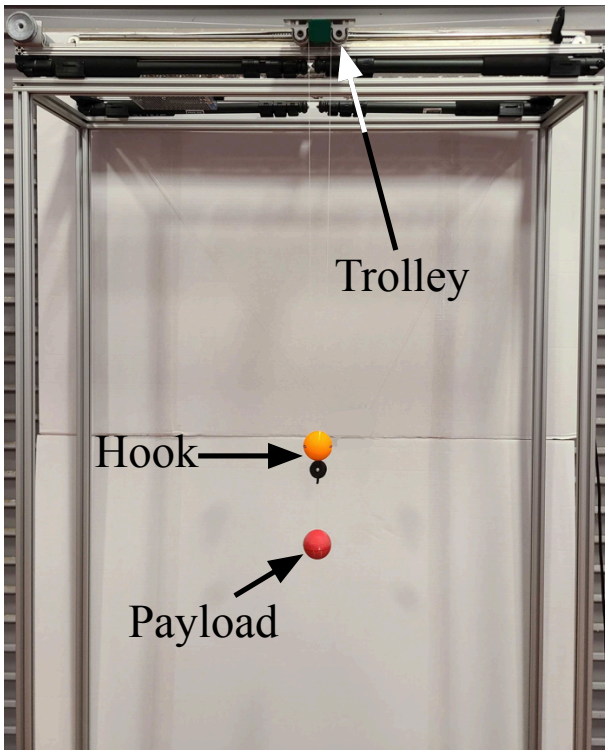


Figure 29. Physical double-pendulum crane

Table 6. Crane Implementation Parameters

Low Mode	**Accel. Limit**	**Vel. Limit**	**Sampling Rate**
0.55Hz	1.18m/s^2	0.45m/s	20Hz

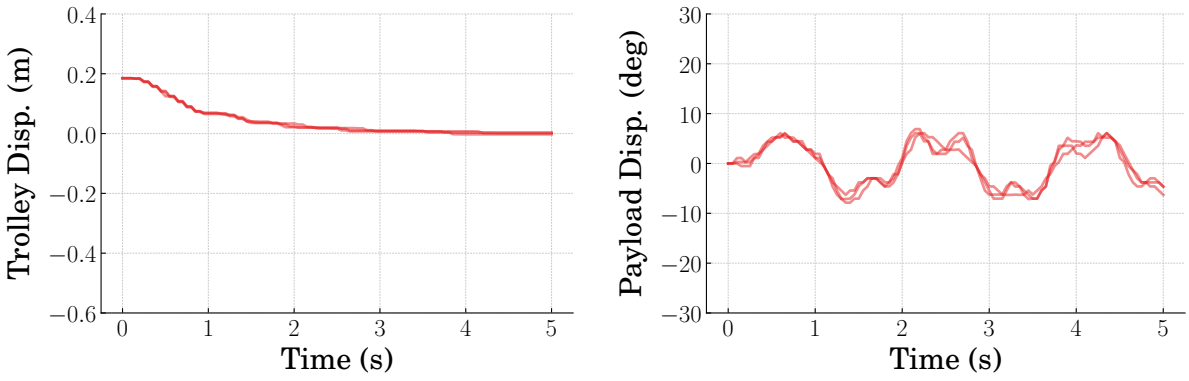
track on which the trolley rides for the physical crane. The displacement of the trolley was directly measured by encoders on the motors, whereas the angular displacements of the hook and payload where calculated from the relative positions of the trolley, hook, and payload, which had colored markers that where tracked using computer vision. The implementation parameters for the planar crane experiments are shown in Table 6. The simulated model and physical crane were configured to have approximately the same low mode of 0.55Hz. However, there are some unmodeled factors that exist on the physical crane that where not in the simulations used to train the agents, including friction acting on the trolley, damping at the hook, motor dynamics, feedback noise when using color tracking, and excitation of out-of-plane degrees-of-freedom.

Table 7. Experimental Crane Rewards

		Mean Reward	Standard Deviation
Simulation	PD	-5.51	-
	RL	-4.52	1.1
	RL-LA	**-3.8**	0.23
	RL-PD-LA	-3.98	**0.19**
Experiment	PD	**-3.36**	**0.02**
	RL	-25.6	25.6
	RL-LA	-5.75	2.2
	RL-PD-LA	-5.7	1.5

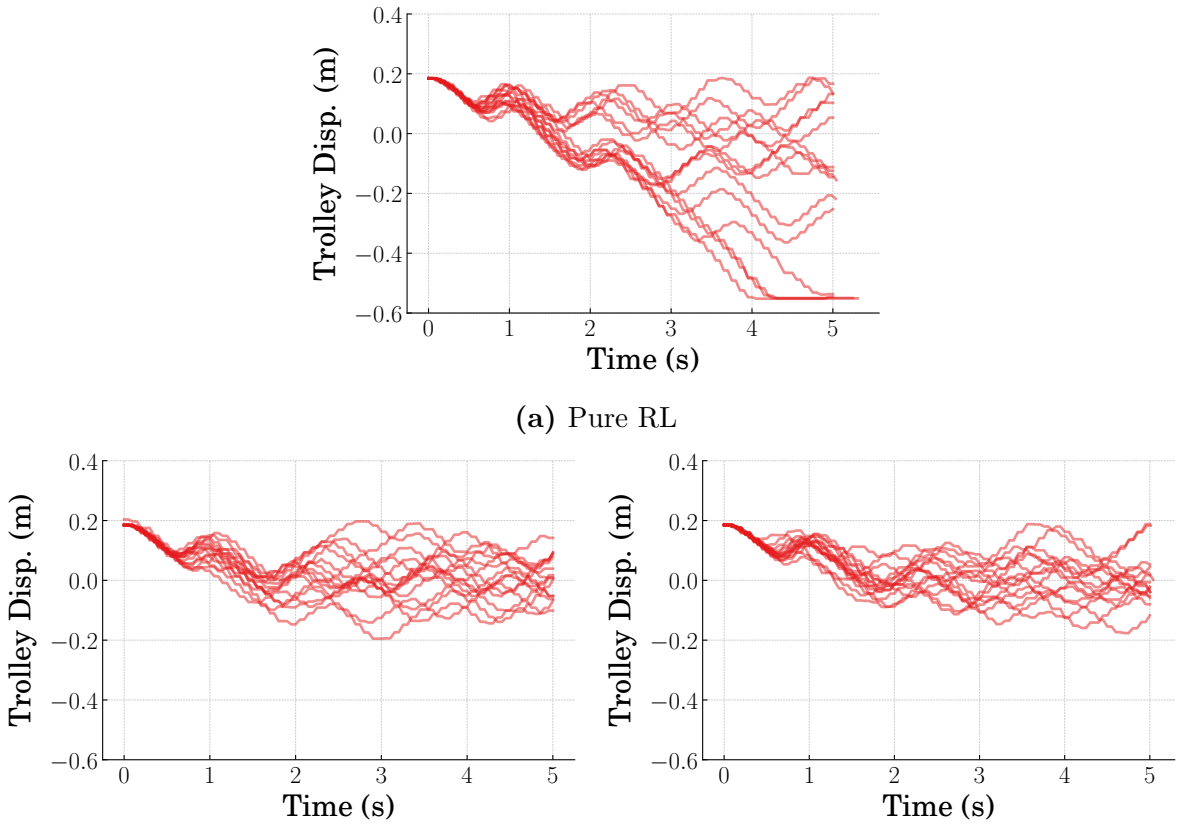
The baseline performance of the crane was determined using a PD controller for trolley position using the same gains as in the fixed-gain term in the combined controllers. This PD controller was implemented to acquire a baseline for the performance in simulation and on the physical crane. Table 7 summarizes the mean rewards and standard deviation of reward from simulation of all agents for each control structure. For the evaluation of the simulated results, the response when using PD resulted in the lowest reward. The Pure RL controller resulted in a higher reward than the PD controller; however, the combined controllers (RL-LA and RL-PD-LA) resulted in the highest reward. The simulated trials were repeated for the physical crane. Three experimental trials were run for each of the five agents trained for RL, RL-LA, and RL-PD-LA, including three trials from the baseline PD controller. Each experimental trial had a time limit of 5s, as was used for episode length during training. The results of the experimental trials are shown in Table 7. The experimental trials using PD alone resulted in the highest mean reward. The mean reward using Pure RL was significantly lower than that of the other controllers due to the controller being unable to cause the trolley to settle near the desired displacement.

Figure 30 shows the time responses of the crane experiments using the baseline PD controller. The responses are plotted with semi-opaque lines such that darker portions of the plot indicate overlapping responses. The trolley responses in Figure 30a



(a) Trolley Responses **(b)** Payload Responses

Figure 30. Experimental crane PD baseline responses

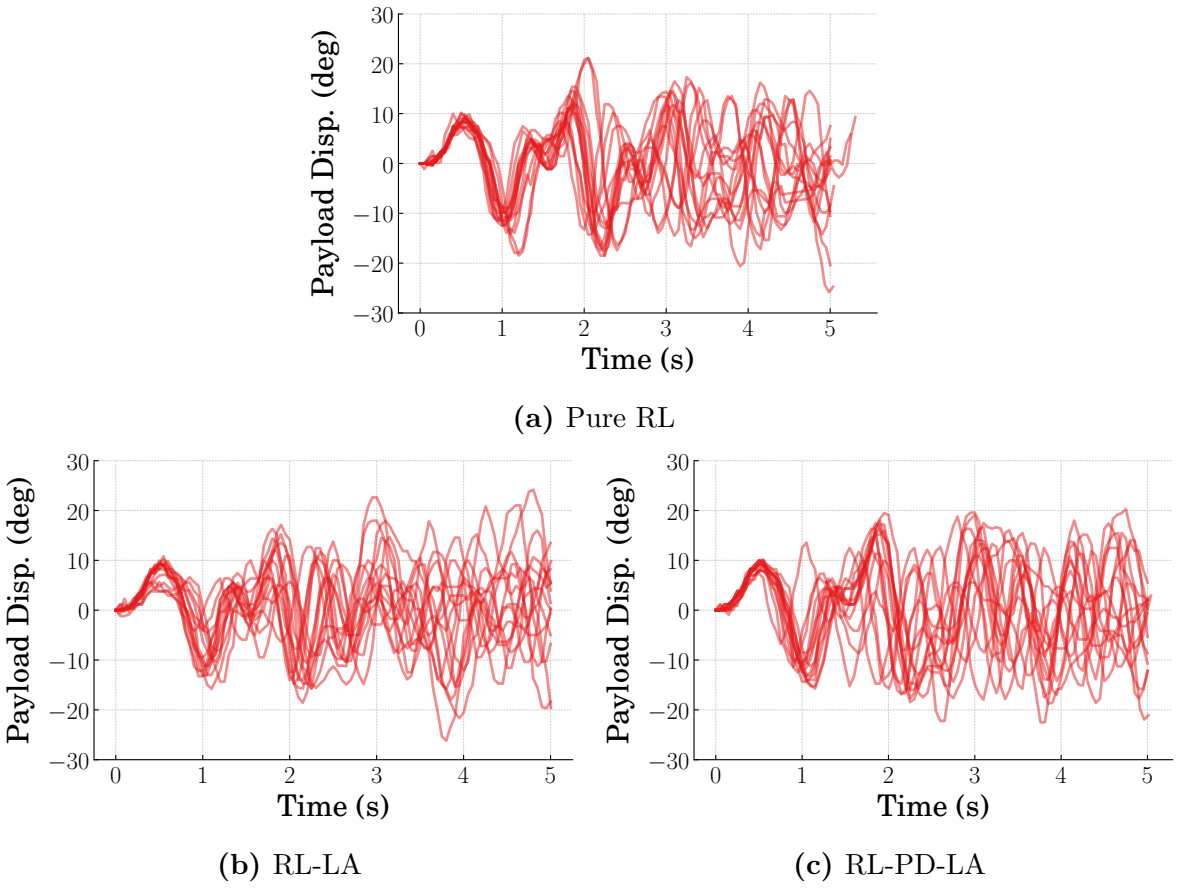


(a) Pure RL

(b) RL-LA **(c)** RL-PD-LA

Figure 31. Experimental crane trolley displacement responses

were able to quickly settle to zero displacement without steady-state oscillation, resulting in the high reward for PD from Table 7. The payload responses in Figure 30b have residual oscillation, but the amplitude is low.



(a) Pure RL

(b) RL-LA

(c) RL-PD-LA

Figure 32. Experimental crane payload displacement responses

The trolley time responses from all experimental trials are shown in Figure 31. The responses from the Pure RL controllers in Figure 31a tend to have significant error. This caused the high mean and standard deviation described in Table 7. The RL-LA and RL-PD-LA performance in Figure 31b and Figure 31c tend to be better than the Pure RL responses. Although there is still error in trolley response, the responses with the combined controllers tend to oscillate about zero displacement.

Figure 32 shows the responses of the payload for all experimental trials with the controllers. The responses all have similar residual oscillation amplitude, including RL-LA which previously had completely mitigated oscillation in simulation. Although the results using the RL controller are undesirable, they illustrate the benefit of the combined RL and conventional control methods. Even though the pendulum responses

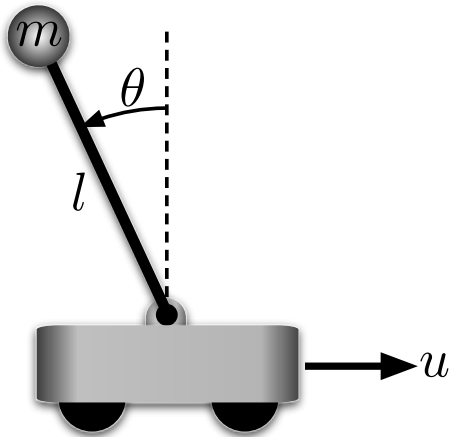


Figure 33. Inverted pendulum model

are similar for the learned controllers, the combined controllers performed significantly better for the trolley responses.

2.4 Inverted Pendulum

2.4.1 *Problem Description*

Another benchmark system used to evaluate the proposed control algorithms was an inverted pendulum model, as shown in Figure 33. The inverted pendulum model can serve as a simple analog for many systems with unstable equilibria, such as bipedal legged systems and rockets. The cart moves on a frictionless surface with cart acceleration as the input, u. The length and mass of the pendulum are l and m, respectively. The angular displacement of the pendulum, θ, is measured from the vertical axis. The equations of motion for this system are:

$$\begin{bmatrix} 1 & 0 \\ -l\cos(\theta) & l^2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} u \\ gl\sin(\theta) \end{bmatrix} \tag{34}$$

The input, u, is direct acceleration control of the cart. This system has two equilibrium positions: an unstable equilibrium at the upright position, $\theta = 0 \pm 2\pi n$, and a stable equilibrium at the hanging position, $\theta = \pi \pm 2\pi n$. The goal is to accelerate the cart to stabilize the pendulum about the upright equilibrium. The intermediate reward

function is:

$$r = -\omega\theta^2 \tag{35}$$

where $\omega = \left(\frac{1}{2\pi}\right)^2$ is a weighting factor used to normalize the reward for pendulum displacements $-2\pi \leq \theta \leq 2\pi$. Although the displacement of the cart has no influence on the reward, cart displacement limits are still enforced between $-25\text{m} \leq x \leq 25\text{m}$, and the agent must learn to avoid contact between the cart and these limits.

A controller using Pure RL and two combined controllers utilizing a fixed-gain term were trained for the inverted pendulum swing-up problem. The nonlinearity of the inverted pendulum prevents the swing-up problem from being solved with a linear controller alone. Some solution methods include Lyapunov-based or energy methods and sliding mode control [43–47]. However, linear controllers can be effective in stabilizing the system near the upright equilibrium at $\theta = 0$. A Linear Quadratic Regulator (LQR) was used as the fixed-gain term for the combined controllers for the inverted pendulum. LQR gains are generated for linear systems that minimize a cost function of the form:

$$J = \int_0^\infty \boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{u}^T R \boldsymbol{u} \, \mathrm{d}t \tag{36}$$

where Q and R are weighting matrices. The gains are found by solving the continuous-time algebraic Riccati equation:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{37}$$

where A and B are matrices from the linearized state-space equations of motion and P is a symmetric matrix. The P matrix that satisfies the Riccati equation is used to find the gain matrix, K, that minimizes (36) with respect to the linearized equations of motion:

$$\boldsymbol{u} = -K\boldsymbol{x}$$

$$K = R^{-1}B^T P$$

For the nonlinear inverted pendulum, the system must be linearized to apply LQR. The inverted pendulum model was linearized about the desired equilibrium, $\theta = \dot{\theta} = 0$:

$$\begin{bmatrix} \Delta\dot{\theta} \\ \Delta\ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{l} & 0 \end{bmatrix} \begin{bmatrix} \Delta\theta \\ \Delta\dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{l} \end{bmatrix} u \tag{38}$$

The R and Q matrices for the system were chosen to correspond to the weights in the reward function and acceleration command limit:

$$Q = \begin{bmatrix} \omega & 0 \\ 0 & 0 \end{bmatrix}, \qquad R = \frac{1}{u_{\max}^2} \tag{39}$$

where $u_{\max} = 10\text{m/s}^2$ is the maximum acceleration command.

Since LQR is not globally stabilizing for the inverted pendulum, LQR needs to be combined with RL in order to solve the swing-up problem. For one configuration, the RL and LQR components were applied simultaneously:

$$\text{RL-LQR:} \qquad u = -k_1\theta - k_2\dot{\theta} + a_t \tag{40}$$

where k_1 and k_2 are LQR gains. This configuration is simple and easy to design, but it requires that the LQR component operates outside the region for which it is stabilizing. This can hinder the training and performance of the RL component of the controller since the contributions from the LQR and RL control elements may not always coordinate well when the pendulum is far from the upright equilibrium. For this reason, a Switching-RL-LQR (S-RL-LQR) controller configuration was also tested:

$$\text{S-RL-LQR:} \qquad u = \begin{cases} -k_1\theta - k_2\dot{\theta}, & \forall(\theta,\dot{\theta}) \in S_{st} \\ a_t, & \forall(\theta,\dot{\theta}) \notin S_{st} \end{cases} \tag{41}$$

This controller implements LQR only when in a set of states for which it is stabilizing, S_{st}; the RL component is utilized alone for all other states. This allows RL to learn to swing up the pendulum, but does not require that it learn to stabilize near the equilibrium since this is already accomplished by the LQR term. This reduces the

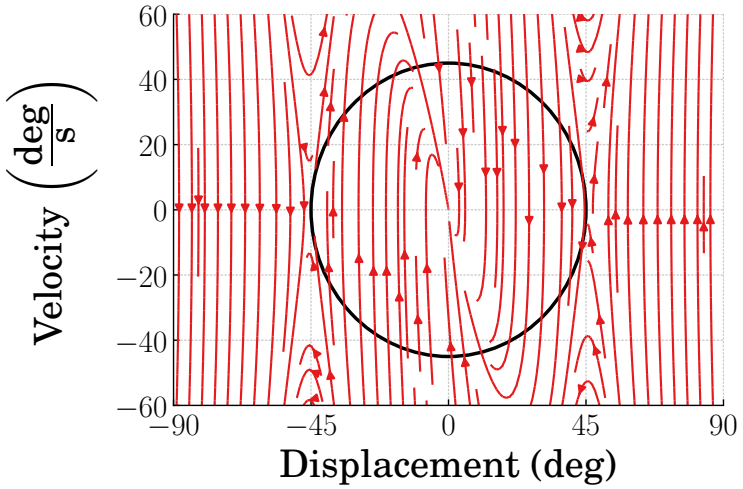


Figure 34. Inverted pendulum phase diagram using LQR

operating region for which the agents have to learn. Figure 34 shows the phase diagram of the inverted pendulum using LQR. Since all trajectories that enter the area indicated by the circle $(\theta^2 + \dot{\theta}^2) = 45^2$ remain stable, this circle is used as the switching criterion and all states that satisfy $\theta^2 + \dot{\theta}^2 \leq 45^2$ populate the set S_{st}.

2.4.2 *Baseline Performance*

The agents in this work were trained using the Twin Delayed Deep Deterministic policy gradient algorithm (TD3) [31]. Because RL often relies on stochasticity to add noise to the action and randomize initial state to aid in exploration, five agents were trained for each controller type. The same five seeds for the artificial random number generator were used for each controller type to train the agents for 3000 episodes each. The cart was at rest at the start of each episode, while the initial angular displacement was randomized between $\theta(0) = -180°$ and $\theta(0) = 180°$. The initial angular velocity of the pendulum was always zero, $\dot{\theta}(0) = 0$.

After training, the performance of the agents was evaluated from the lower stable equilibrium at $\theta(0) = 180°$. The mean reward during training for the agents of each controller type is shown in Figure 35. The shaded region shows the standard deviation of the rewards during training for the individual agents. The Pure RL

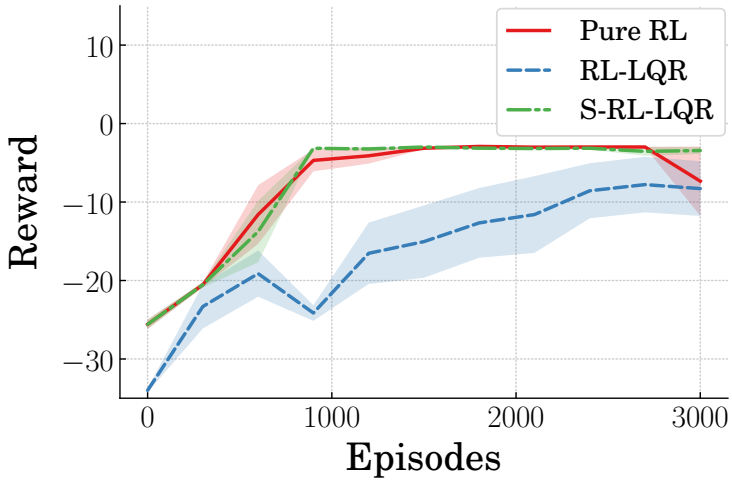
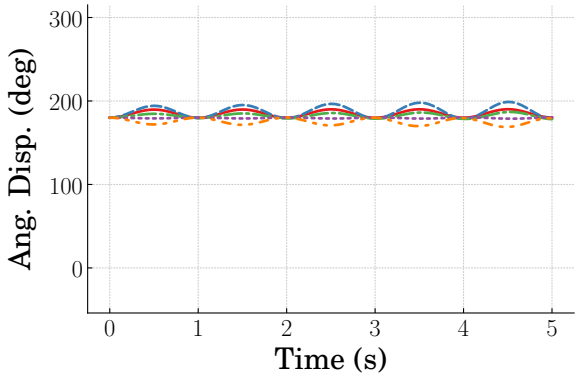


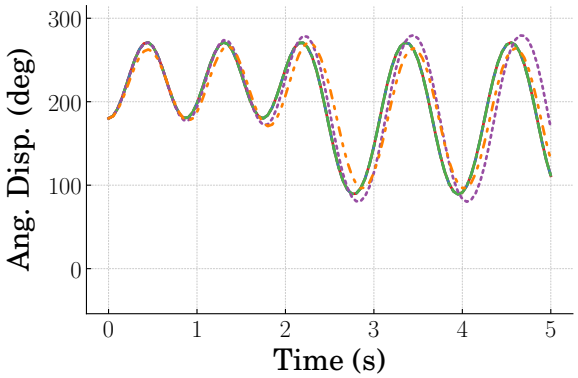
Figure 35. Mean inverted pendulum reward during training

controller and S-RL-LQR controller begin training with the same mean reward since they both use only the RL element of the controller during the swing-up phase. RL-LQR begins training with a lower mean reward due to the addition of the LQR element which can impede controller performance outside of the region of attraction. It takes more training for the RL element of the controller to learn to override the LQR contribution to the control output during the swing-up phase, which accounts for the lower mean reward of RL-LQR throughout training compared to the other controller types. Figure 36 shows the pre-training time responses for the inverted pendulum from $\theta(0) = 180°$ when the inverted pendulum is hanging down at the stable equilibrium. The behavior of the Pure RL controllers in Figure 36a is identical to that of the S-RL-LQR controllers in Figure 36c since both controllers only rely on the RL agent component outside of the switching region. The RL-LQR responses in Figure 36b have higher oscillation amplitude, resulting in the lower initial reward.

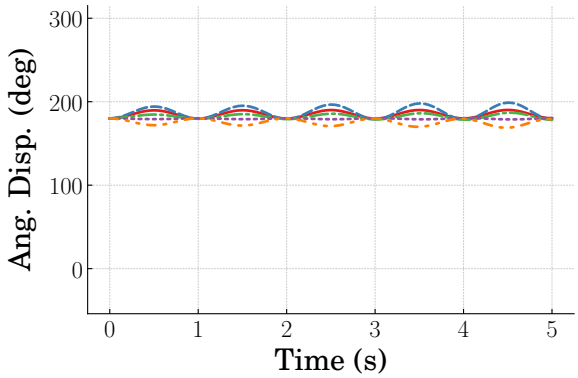
Figure 37 shows the number of agents that successfully stabilize the system from an initial angular displacement of $\theta = 180°$. The S-RL-LQR agents are able to learn to stabilize the system with the least training. All S-RL-LQR agents learned to stabilize the system before 900 episodes of training, whereas the Pure RL agents required



(a) Pure RL



(b) RL-LQR



(c) S-RL-LQR

Figure 36. Inverted pendulum responses for $\theta(0) = 180°$

between 1200 and 1500 episodes before all agents successfully stabilized the pendulum. Only four of the five RL-LQR agents had learned to swing up and stabilize the system by the end of training. Although the Pure RL controllers had learned to stabilize the system, one of the agents lost the ability to swing up and stabilize the inverted pendulum near the end of training between 2700 and 3000 episodes. This caused the reduction in reward for Pure RL when evaluated at the end of training.

Figure 38 shows the angular displacement response after training using the agents trained for the controller using Pure RL. The initial angular displacement was $\theta(0) = 180°$. The agents were successful in the swing-up and stabilization problem if the response settles in the stable region indicated by the gray shaded region in the figure. Four of the agents are able to successfully swing up and stabilize the inverted pendulum.

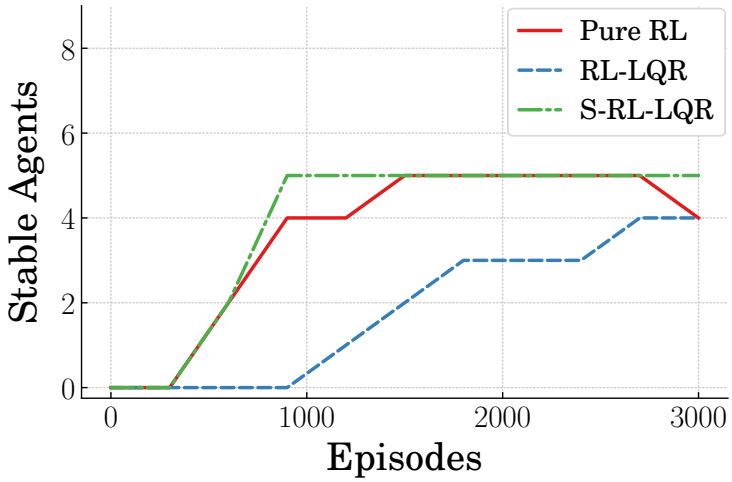


Figure 37. Number of stable agents during training

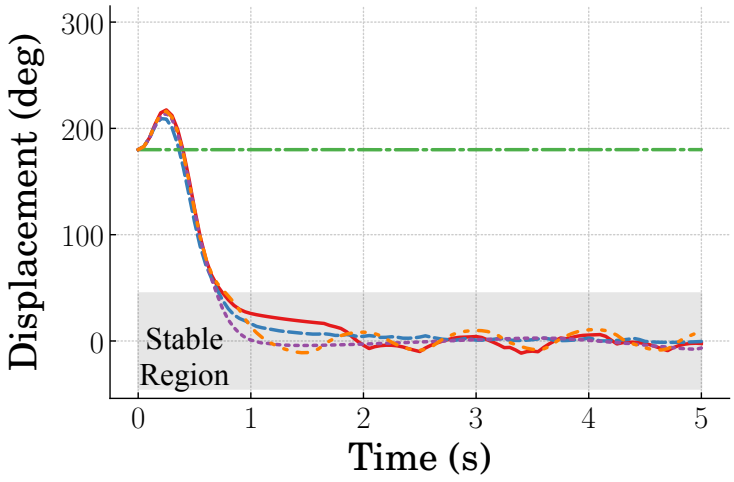


Figure 38. Angular displacement for Pure RL at 3000 episodes

However, although the responses remain near the equilibrium, the successful agents are not asymptotically stable. Additionally, the response shown in green converge to be within the bounds for success. However, this agent was stabilizing at 2700 episodes, as shown by the responses in Figure 39. This unstable training and reward oscillation often happens due to the "deadly triad" in reinforcement learning [3, 48].

Figure 40 shows the angular displacement response of the RL-LQR agents from an initial displacement of $\theta = 180°$. As in the Pure RL case, four of the agents successfully stabilized the system. However, while three of the successful agents had

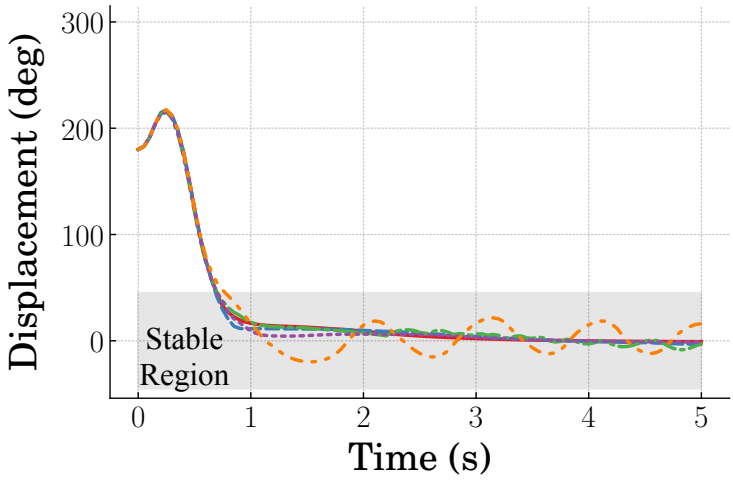


Figure 39. Angular displacement for Pure RL at 2700 episodes

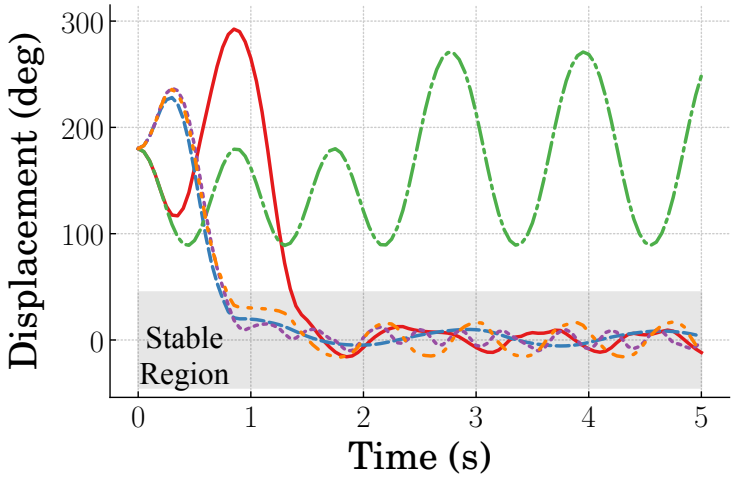


Figure 40. Angular displacement for RL-LQR at 3000 episodes

comparable settling times, one of the agents had a higher settling time to achieve stability. Just as in the Pure RL case, the agents with RL-LQR were not asymptotically stable. Additionally, one of the agents was unsuccessful in swinging up and stabilizing the pendulum. Unlike the Pure RL case, there is no significant differences between the time responses of the RL-LQR controllers at 3000 episodes and 2700 episodes.

Figure 41 shows the controller output contributions of the RL-LQR controllers. Ideally, the control output contribution from the RL element should be significant enough to overpower the LQR output during the swing-up phase then diminish during

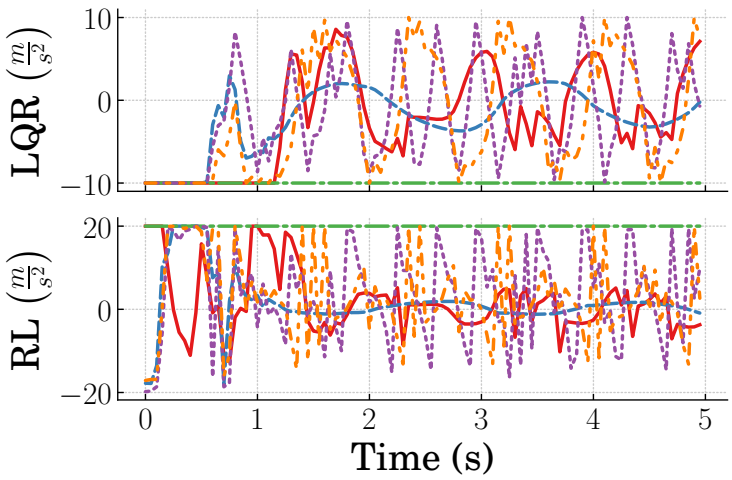


Figure 41. Outputs from RL and LQR controller elements

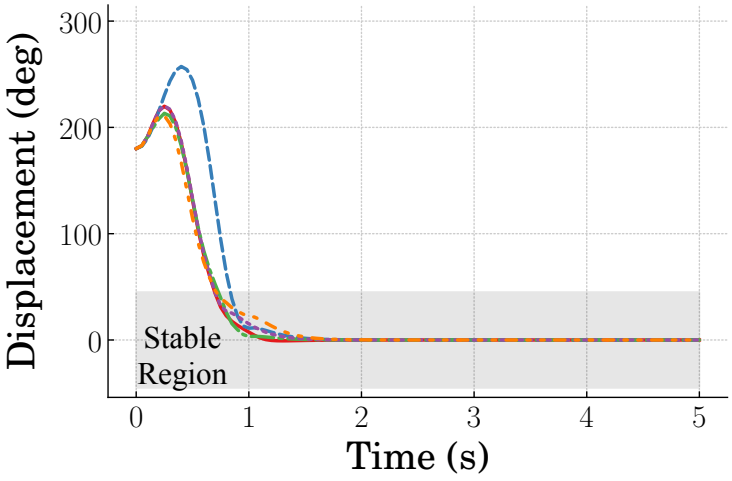


Figure 42. Angular displacement for S-RL-LQR at 3000 episodes

the stabilization phase. However, this constraint was not explicitly included in the design of the RL environment. Some of the agents reduce their RL output during the stabilization phase, but some continue to have a high frequency oscillatory output. However, this output oscillation does not significantly effect the response because its frequency is much higher than the bandwidth of the system near the equilibrium.

Figure 42 shows the angular displacement response of the agents using S-RL-LQR. By utilizing the RL element of the controller only during the swing-up phase of the movement and utilizing LQR only during the stabilization phase, the

Table 8. Inverted Pendulum Response Characteristics for $\theta(0) = 180°$

		Amplitude	Settling Time	Percent Overshoot
Pure RL	Mean	3.56°	0.763s	6.73%
	SD	3.2°	0.041s	4.33%
RL-LQR	Mean	8.81°	0.938s	7.43%
	SD	4.17°	0.268s	1.83%
S-RL-LQR	Mean	**0.0°**	**0.75**s	**0.089**%
	SD	**0.0°**	**0.055**s	**0.178**%

agents were able to learn controllers with less variation among the responses. All five S-RL-LQR agents successfully stabilized the system. Additionally, the LQR element of the controllers were able to provide asymptotic stability about $\theta = 0$ during the stabilization phase unlike the Pure RL and RL-LQR cases. Response characteristics for the responses are summarized in Table 8. The unstable responses from Pure RL and RL-LQR were not included in this analysis. The S-RL-LQR controller has the best mean performance in terms of residual oscillation amplitude, settling time, and percent overshoot.

The performance for the swing-up and stabilization problem was shown above for initial angular displacements of $\theta(0) = 180°$. To isolate the performance of the controllers for the stabilization phase near $\theta = 0$, responses were acquired with initial angular displacements of $\theta(0) = 45°$. Figure 43 shows the reward trends during training evaluated from this initial displacement. The trends for the stabilization problem from this initial displacement are similar to the reward trends for the swing-up and stabilization problem where S-RL-LQR tends to have the highest reward followed by Pure RL then RL-LQR. The S-RL-LQR reward trend is constant for this initial condition since only the LQR controller element is contributing to the controller output in this response range.

Figure 44 shows the time responses of the inverted pendulum from $\theta(0) = 45°$ before training. The Pure RL responses in Figure 44a are unstable about the upright

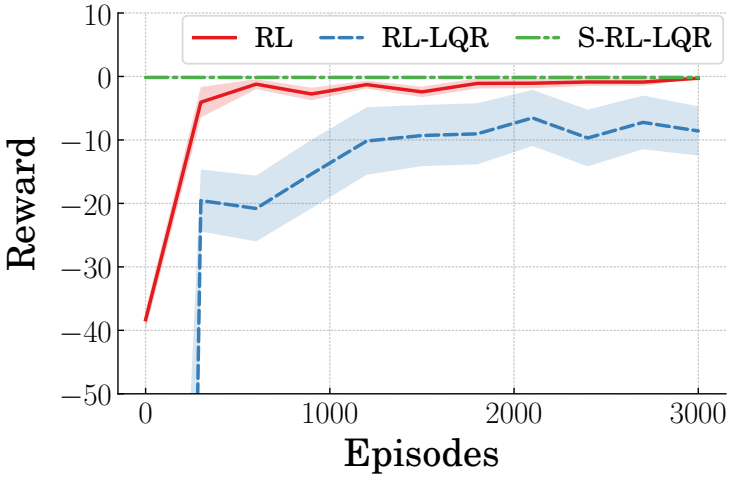
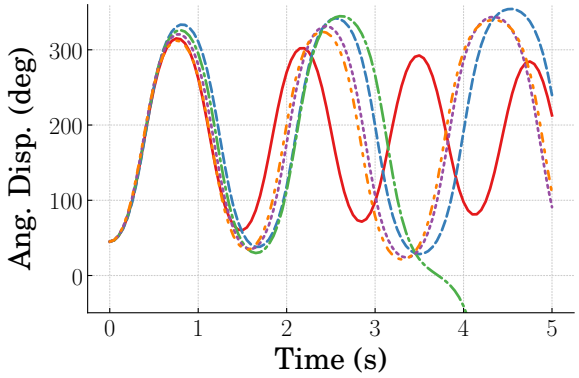
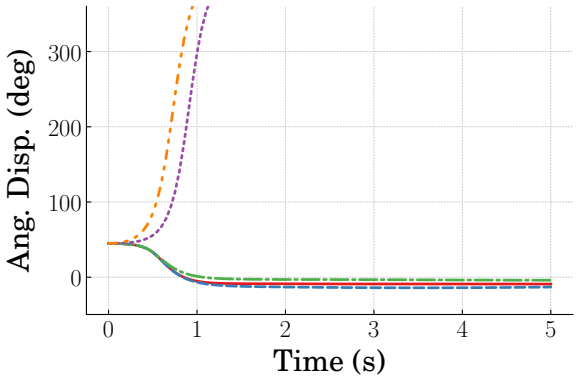


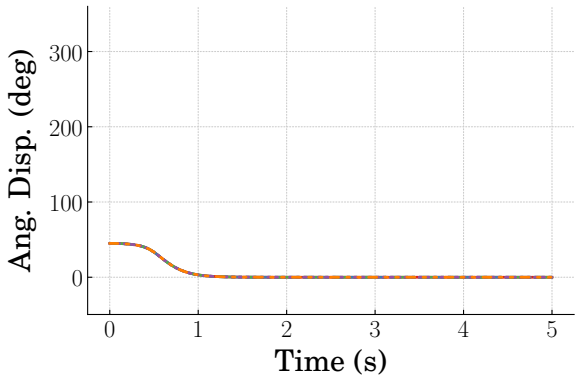
Figure 43. Mean inverted pendulum reward during training for $\theta(0) = 45°$



(a) Pure RL



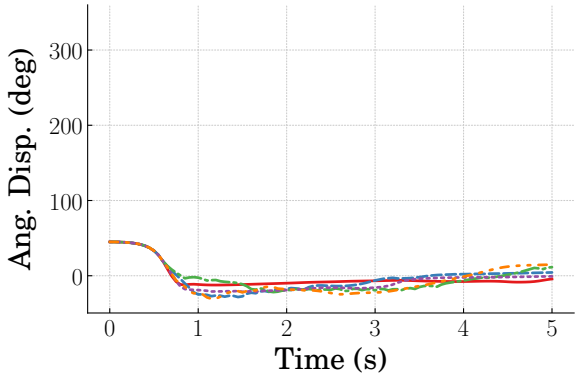
(b) RL-LQR



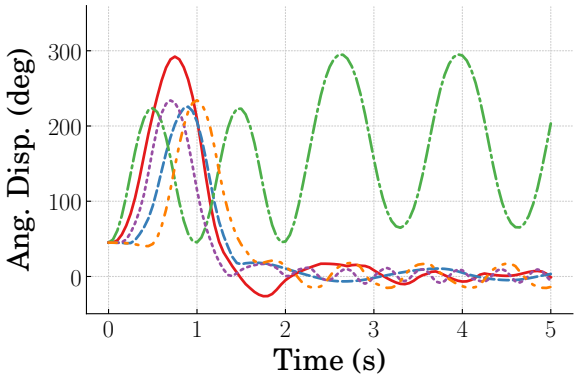
(c) S-RL-LQR

Figure 44. Inverted pendulum responses for $\theta(0) = 45°$ before training

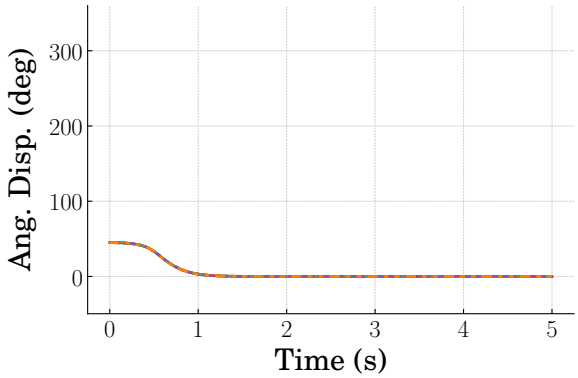
equilibrium and most of them fall and oscillate about the stable equilibrium at $\theta = 180°$. Figure 44b shows that three of the five responses settle near the equilibrium due to the contribution of the LQR element, however they retain some steady-state



(a) Pure RL



(b) RL-LQR



(c) S-RL-LQR

Figure 45. Inverted pendulum responses for $\theta(0) = 45°$ after training

error caused by the untrained agents. For two of the responses in this figure, the untrained RL element causes the responses to diverge. These diverging responses are responsible for the low reward at the beginning of training despite the good performance of the other agents. The S-RL-LQR responses in Figure 44c are a result of the LQR control element acting with no contribution from the agents.

Figure 45 shows the responses of the inverted pendulum for $\theta(0) = 45°$ after training. The Pure RL responses in Figure 45a all converge near the equilibrium. In Figure 45b, all of the RL-LQR responses swing far away from the equilibrium before swinging back up to $\theta = 0$. However, one of the responses does not converge to the upright equilibrium and instead oscillates about the stable equilibrium at $\theta = 180°$. Although most of the RL-LQR responses before training had better performance than

Table 9. Inverted Pendulum Response Characteristics for $\theta(0) = 45°$

		Amplitude	Percent Overshoot
Pure RL	Mean	3.08°	52.3%
	SD	3.17°	14.1%
RL-LQR	Mean	8.82°	34.3%
	SD	4.15°	15.1%
S-RL-LQR	Mean	**0.0°**	**0.0%**
	SD	**0.0°**	**0.0%**

Pure RL, the performance of RL-LQR after training is worse than that of Pure RL. This is due to contribution from the agents being low at the beginning of training such that LQR is the main contributor to the RL-LQR response. The response characteristics from $\theta(0) = 45°$ are summarized in Table 9. The settling time metric was removed since all responses begin within the stable region use previously. The residual oscillation amplitude was similar for both the responses starting near the equilibrium and those starting at $\theta(0) = 180°$. The mean percent overshoot was highest for the Pure RL responses. However, this metric does not account for RL-LQR initially swinging down away from the desired equilibrium.

2.5 Summary of Results

The results above presented methods in which RL and domain knowledge can be leveraged for different types of systems. The general categories of combined controller types tested were model-based fixed-gain controllers with RL and agent-driven gain-scheduled controllers. Each category of combined controller are summarized based on how they affect initial performance, required training time, the steady-state error generated after training, and the residual oscillation.

Initial Performance – The combined controllers typically provided better initial performance before training than Pure RL controllers since the conventional control elements provided performance that was closer to being optimal. One exception

was the RL-LQR combined controller for the inverted pendulum. While the other combined controllers for the other systems presented fixed-gain controllers that were stabilizing in the entire operating space used for training, the fixed-gain LQR controller for the inverted pendulum was not stabilizing for the operating region being used.

Training Time – Given adequate training time, agents often reach a local optimum during training where the rate of change of the reward with more training time approaches zero. Since the Pure RL controllers tended to begin training with lower reward, they also often required more training time to achieve performance comparable to the combined controllers. The agent-driven gain scheduling controllers reached near optimal performance more quickly than the fixed-gain combined controllers for the duffing oscillator. However, the combined controllers with gain-scheduled terms took longer to train for the double-pendulum crane. This was due to the oscillation excited in the multi-modal, underactuated crane. The fixed-gain term was more effective at mitigating oscillation, resulting in less training time needed to receive higher rewards. The controller that switched from RL to a fixed-gain LQR controller for the inverted pendulum required less training time than the other controllers due to the reduction in the operating area in which the agents had to learn.

Steady-State Error – The combined controllers tended to have lower steady-state error than the Pure RL controllers. This trend was consistent across all benchmark systems tested. However, the combined controller architecture that resulted in the lowest steady-state error varied by system. Whereas the gain-scheduling controller (RL-PD) provided the lowest steady-state error for the duffing oscillator, the fixed-gain and lumped action (RL-LA) controller provided the lowest steady-state error for the double-pendulum crane.

Residual Oscillation – Not all combined controller architectures showed a general improvement in mitigating residual oscillation. Whereas the duffing oscillator controller with agent-driven gain scheduling (RL-PD) completely mitigated residual

oscillation, the combined controllers for the double-pendulum crane with agent-driven gain scheduling elements tended to have higher residual oscillation amplitude than controllers that did not include this element. The switching controller for the inverted pendulum completely mitigated residual oscillation by only using the LQR terms near the equilibrium.

2.6 Design Guidelines

Since not every combined controller architecture provided better performance than RL for every benchmark system, guidelines are necessary to facilitate controller design. In general, fixed-gain model-based controllers improved performance compared to Pure RL when the fixed-gain controllers were stabilizing. If a globally stabilizing controller cannot be found, a locally stabilizing controller will still improve performance. The fixed-gain controller should only be used in regions for which it is stabilizing and should be deactivated outside of the region of attraction. This concept was illustrated for the inverted pendulum, where the controller that switched between the RL agents and LQR had the best performance.

Many of the agents in this work caused their respective systems to move near the desired equilibrium but still have steady-state error. Agent-driven gain scheduling can eliminate steady-state error since the controller output is proportional to the error. This was shown by the RL-PD controller for the duffing oscillator. However, agent-driven gain scheduling can make interaction with the environment more complex since the agent has multiple ways to influence the performance of the system. Agent-driven gain scheduling may also be a poor design choice for underactuated systems, which was illustrated by the high residual oscillation amplitude of the double-pendulum crane controllers that utilized gain scheduling. Therefore, lumped action terms should be used for underactuated systems.

2.7 Conclusion

This chapter presented methods for designing learning controllers that leverage domain knowledge. The basic controller architectures included agent-driven gain scheduled controllers and controllers that combined model-based fixed-gain controllers with RL elements. These control architectures where tested on three benchmark systems. Although there was no combined controller architecture that consistently resulted in the best performance for all systems, there was always a combined controller that outperformed Pure RL. The combined controllers typically resulted in improved initial performance and required less training to achieve desirable performance from the system. Additionally, the performances of the combined controllers often had less variance regardless of the initial seed. Since one of the primary obstacles to using RL on physical systems is the long training time required, these methods reduce the barrier to implementation of RL for physical systems.

III Agent Stability

As was previously mentioned, requiring long training time and having poor initial performance have prevented RL from being widely adopted beyond research settings. Additionally, lack of stability guarantees prevents application of trained agents to safety critical applications. This chapter presents a numerical stability analysis of the learned controllers and shows how the combined controller architectures affect stability. There are a variety of notions of stability that have been developed. Bounded-Input-Bounded-Output (BIBO) and Lyapunov stability were used to evaluate the performance of the controllers that were introduced in Chapter 2.

3.1 Stable in the Sense of Lyapunov

In order for a system to be stable in the sense of Lyapunov, a system that begins in the neighborhood of an equilibrium, x_e, must remain near the equilibrium. This notion of stability is defined formally as:

$$||x(t_0) - x_e|| < \delta \quad \text{and} \quad ||x(t) - x_e|| < \epsilon \quad \forall t > t_0 \tag{42}$$

where δ and ϵ are greater than zero. In order for this condition to hold for all time, the set contained within a distance δ from x_e must be a subset of that contained within a distance ϵ. This is a weak notion of stability since it claims that a system is Lyapunov stable as long as the system state does not diverge to infinity. However, this notion allows for some initial analysis of the behavior and stability of the combined controllers near equilibrium.

The following section presents a Bounded-Input-Bounded-Output (BIBO) analysis where the minimum output bound that satisfies BIBO is used to quantify stability for each system and controller. The definition of Lyapunov stability is also used to quantify behavior of the agents near equilibrium. The system starts at rest at the equilibrium such that $\delta = 0$. The response is then sampled for a time range of $0 \leq t \leq t_f$, where t_f is the final time of each recorded state trajectory. The minimum

value of ϵ that satisfies the Lyapunov stability conditions for the sampled states is used to quantify the stability of the agent.

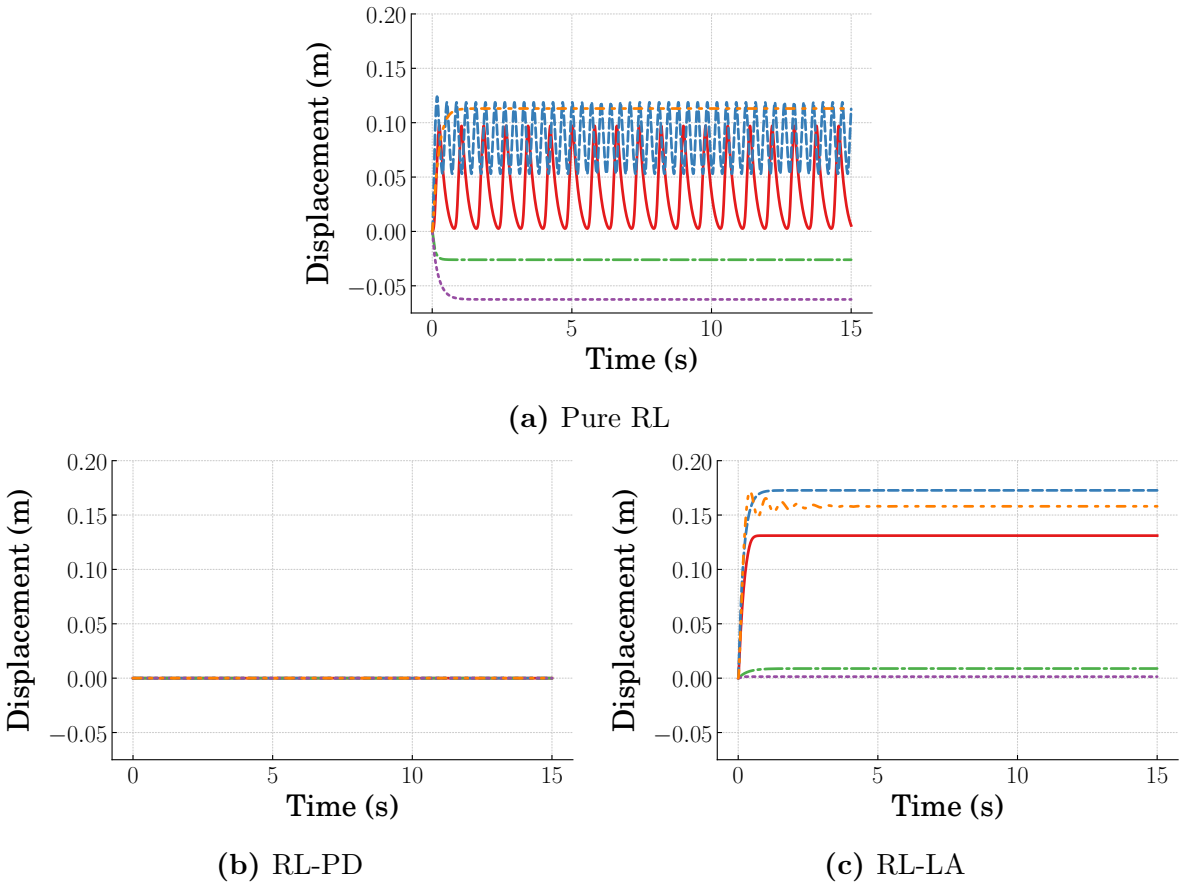
3.1.1 *Duffing Oscillator Stability*

This section presents a stability evaluation for the controllers trained for the duffing oscillator shown previously. The agents for this system were trained to generate a force input that moved the mass to the desired setpoint. As a reminder, the controllers trained for the duffing oscillator were:

$$\begin{aligned} &\text{Pure RL:} && u = a_t \\ &\text{RL-PD:} && u = \alpha x_d + \beta x_d^3 + a_t^{(1)} x + a_t^{(2)} \dot{x} \\ &\text{RL-LA:} && u = \alpha x_d + \beta x_d^3 + a_t \end{aligned}$$

where the force output, u, from Pure RL is the action, a_t, from the agent, RL-PD contains fixed-gain terms in addition to a PD controller with gains, $a_t^{(1)}$ and $a_t^{(2)}$, determined by the agent, and RL-LA has the fixed-gain terms in addition to a single lumped action, a_t.

Figure 46 shows the five time responses for each controller type for the duffing oscillator starting at rest with the desired setpoint and initial displacement being equal, $x = x_d = 0$. Although the system begins at the equilibrium, most of the responses move away from the equilibrium and settle at a non-zero displacement. The Pure RL case in Figure 46a has some responses that settle to a fixed non-zero displacement and some that have residual oscillation. The responses from RL-PD in Figure 46b have zero steady-state error and no residual oscillation since the initial displacement and desired setpoint are equal. Figure 46c shows the responses from RL-LA. Two of the responses have low steady-state error, while three of the responses have higher steady-state error. Although there are RL-LA responses with higher steady-state error than that of Pure RL, the RL-LA responses have no residual oscillation.



(a) Pure RL

(b) RL-PD

(c) RL-LA

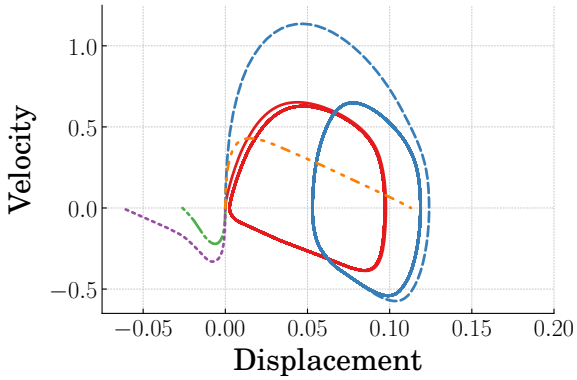
Figure 46. Duffing oscillator response near equilibrium

The minimum bounds on the displacement error for which the responses were BIBO stable were determined by the maximum displacement from each response. The mean of the absolute values and standard deviations of the bounds are summarized in Table 10. RL-PD has zero steady-state error. Although the responses from Pure RL have higher residual oscillation amplitude, the mean of the maximum displacements is lower than that of RL-LA. However, the standard deviation of the maximum displacements is slightly higher for Pure RL compared to RL-LA.

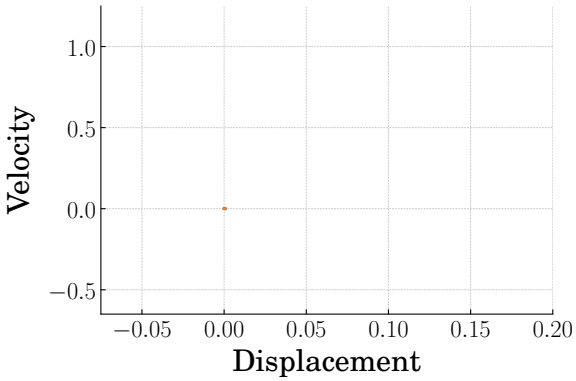
Phase plots of the displacement and velocity for the duffing oscillator responses are shown in Figure 47. These were used to determine the bounds, ϵ, for which the system with the different controllers was Lyapunov stable. The responses from Pure RL in Figure 47a show that the two responses with residual oscillation have stable limit

Table 10. Duffing Oscillator Stability

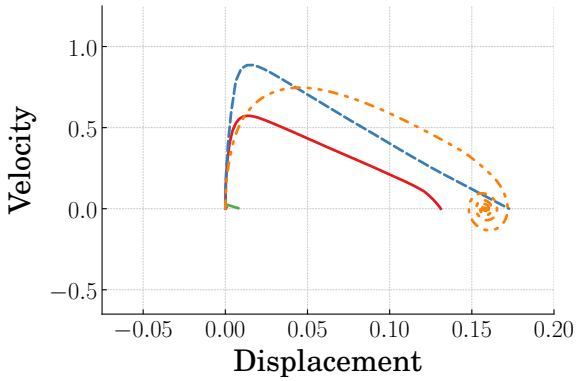
Displacement Error			
	Pure RL	RL-PD	RL-LA
Mean	0.085m	**0.0**m	0.097m
SD	0.078m	**0.0**m	0.077m
Phase Bounds			
	Pure RL	RL-PD	RL-LA
Mean	0.554	**0.0**	0.448
SD	0.324	**0.0**	0.365



(a) Pure RL



(b) RL-PD



(c) RL-LA

Figure 47. Duffing oscillator phase responses

cycles. The RL-PD responses in Figure 47b do not move from the initial displacement as was shown previously. Although some of the RL-LA responses in Figure 47c have higher maximum displacement than that of Pure RL, they have lower maximum

velocities and do not exhibit limit cycles. Table 10 which contained the summary of BIBO stability for the controllers also contains the mean and standard deviation of the minimum bounds for which the system satisfies Lyapunov stability. Although RL-LA has a higher mean displacement error, the mean Lyapunov bound for the Pure RL case has a larger radius due to the higher velocities of the responses.

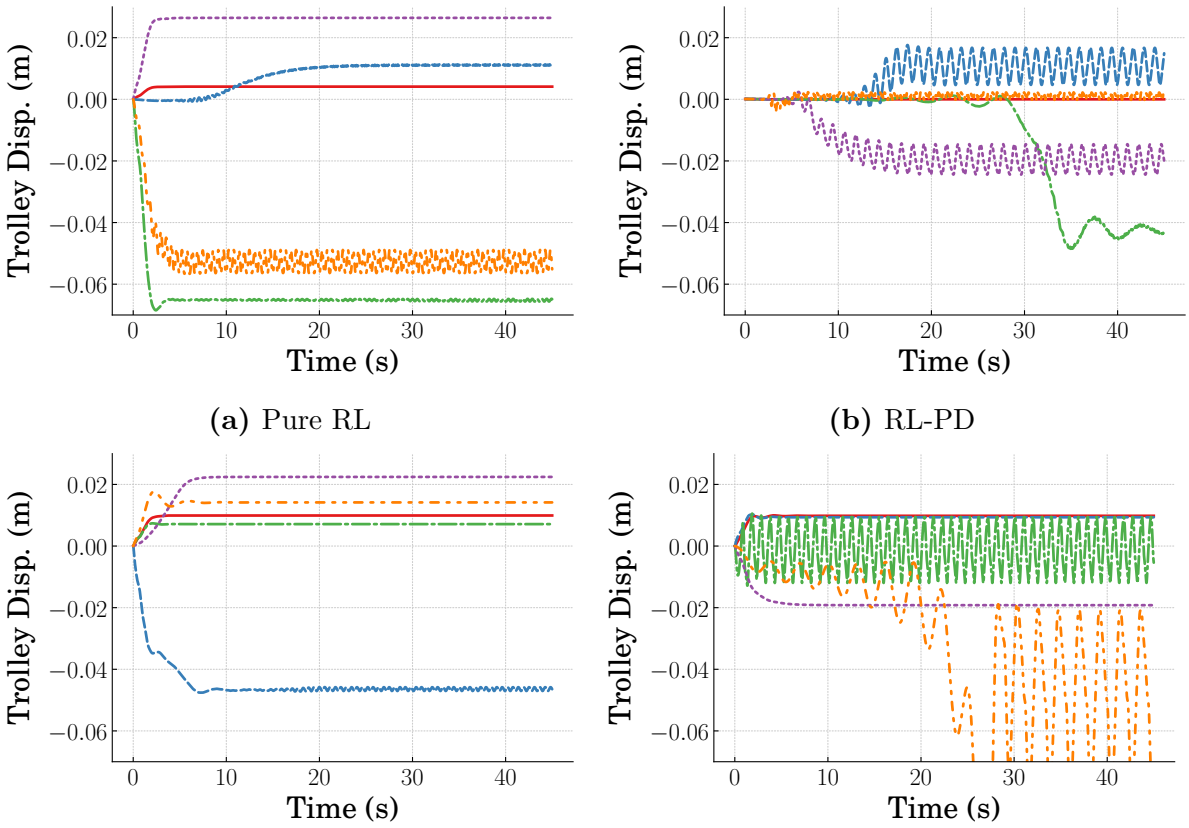
3.1.2 *Double-Pendulum Crane Stability*

This section presents the stability analysis of the double-pendulum crane near the equilibrium. The controllers for the crane that were introduced in the previous chapter were:

$$
\begin{aligned}
&\text{Pure RL:} && u = a_t \\
&\text{RL-PD:} && u = k_p x + k_d \dot{x} + \boldsymbol{a}_t \boldsymbol{\theta} \\
&\text{RL-LA:} && u = k_p x + k_d \dot{x} + a_t \\
&\text{RL-PD-LA:} && u = k_p x + k_d \dot{x} + \boldsymbol{a}_t \boldsymbol{\theta} + a_t
\end{aligned}
$$

where the acceleration command, u, for Pure RL is determined by the agent action, a_t. The other controllers utilize fixed-gain terms to bring the trolley to the desired displacement. RL-PD combines the fixed-gain terms with an agent-driven gain-scheduling controller for the pendulum states, where $\boldsymbol{a}_t$ is a vector of gains and $\boldsymbol{\theta}$ is a vector of hook and payload angular displacement and angular velocity. RL-LA uses the fixed-gain terms combined with a lumped action term and RL-PD-LA uses the fixed-gain terms in addition to both agent-driven gain scheduling and a lumped action. The stability was evaluated with the crane at rest and initial trolley and pendulum displacements of zero. The desired displacement of the trolley was also zero.

The time responses of the trolley are shown in Figure 48. The Pure RL case in Figure 48a tends to have the highest steady-state error and little to no residual oscillation. For the RL-PD case in Figure 48b, the error is low at the beginning of the



(a) Pure RL

(b) RL-PD

(c) RL-LA

(d) RL-PD-LA

Figure 48. Near equilibrium trolley behavior

response. However, three of the responses eventually diverge away from the equilibrium. Although RL-PD tends to have lower steady-state error, more of the responses have residual oscillation. The responses from RL-LA in Figure 48c have steady-state error, but they also tend to have little to no residual oscillation compared to RL-PD. For the RL-PD-LA responses in Figure 48d, four of the five responses remain close to the equilibrium. However, the response shown in green has high residual oscillation amplitude about the equilibrium. Additionally, the response in orange slowly diverges farther from the equilibrium than the other responses until reaching a fixed steady-state error with fixed residual oscillation amplitude. The minimum bounds for BIBO stability were determined by the maximum displacements of the responses. The mean and standard deviation of the bounds for the trolley are shown at the top of Table 11.

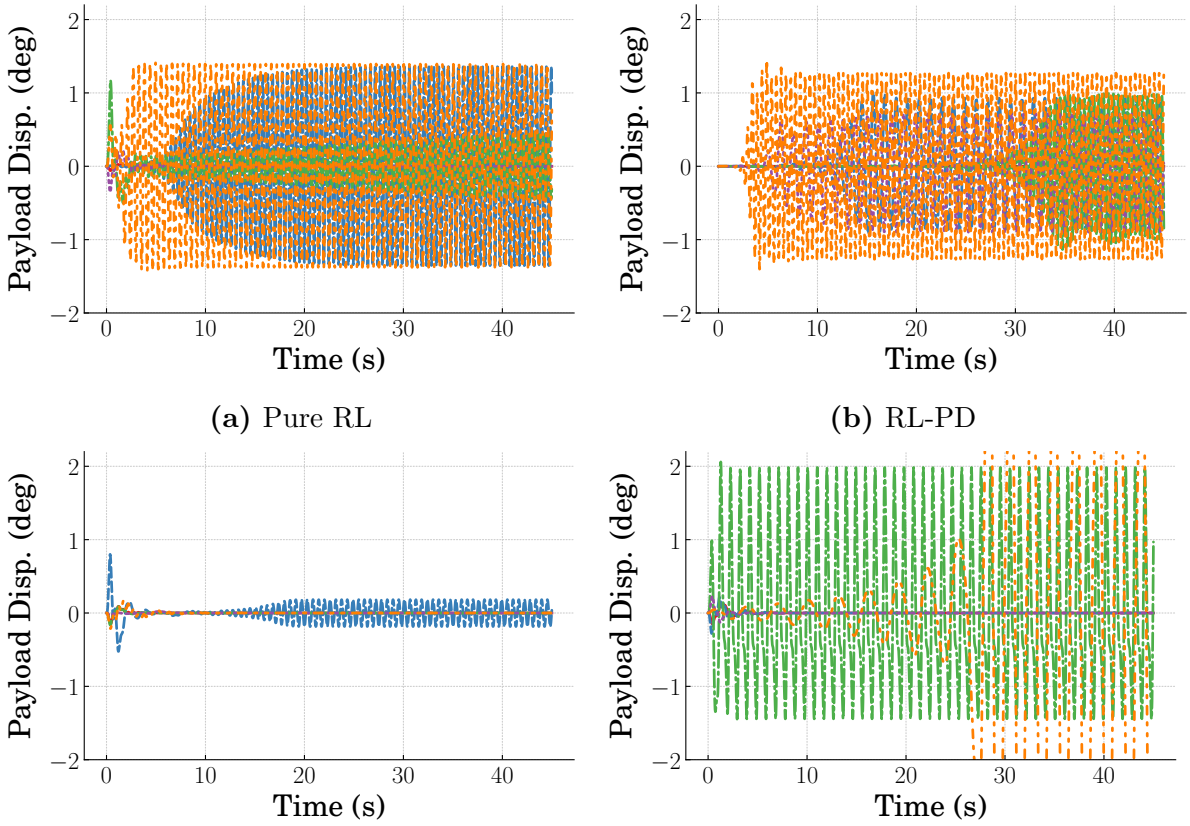
Table 11. Planar Crane BIBO Stability

Trolley Displacement Error				
	Pure RL	RL-PD	RL-LA	RL-PD-LA
Mean	0.032m	**0.019**m	0.021m	0.046m
SD	0.037m	**0.023**m	0.025m	0.072m
Payload Displacement Error				
	Pure RL	RL-PD	RL-LA	RL-PD-LA
Mean	0.874°	0.9°	**0.273**°	1.899°
SD	1.027°	1.019°	**0.381**°	2.662°

The maximum displacements from the RL-PD-LA controller have the highest mean and standard deviation primarily due to the response in orange in Figure 48d. The Pure RL case had the second-highest mean, followed by RL-LA with the second-lowest mean. RL-PD had the lowest mean bound to achieve BIBO stability.

The payload responses for the crane starting at rest with initial trolley and pendulum displacements of $x(0) = \theta(0) = 0$ are shown in Figure 49. The responses from the different controllers all tended to have constant-amplitude residual oscillation. Three of the five Pure RL responses in Figure 49a have residual oscillation, whereas four of the five RL-PD responses in Figure 49b have residual oscillation. The RL-LA responses in Figure 49c have significantly lower residual oscillation amplitude than the other controllers. Three of the five RL-PD-LA responses in Figure 49d have no residual oscillation, whereas two of the responses have much higher oscillation amplitude. The performance of the crane payload is summarized in the bottom half of Table 11. The mean and standard deviation of the bounds is highest for RL-PD-LA. This is primarily caused by the two responses with high residual oscillation amplitude. The Pure RL and RL-PD controllers have similar mean and standard deviation. RL-LA has the lowest mean and standard deviation.

The trolley time responses that were previously shown are presented as phase diagrams in Figure 50. The plots show that many of the responses have limit cycles. However, there is also variance in the responses for each controller. The Pure RL

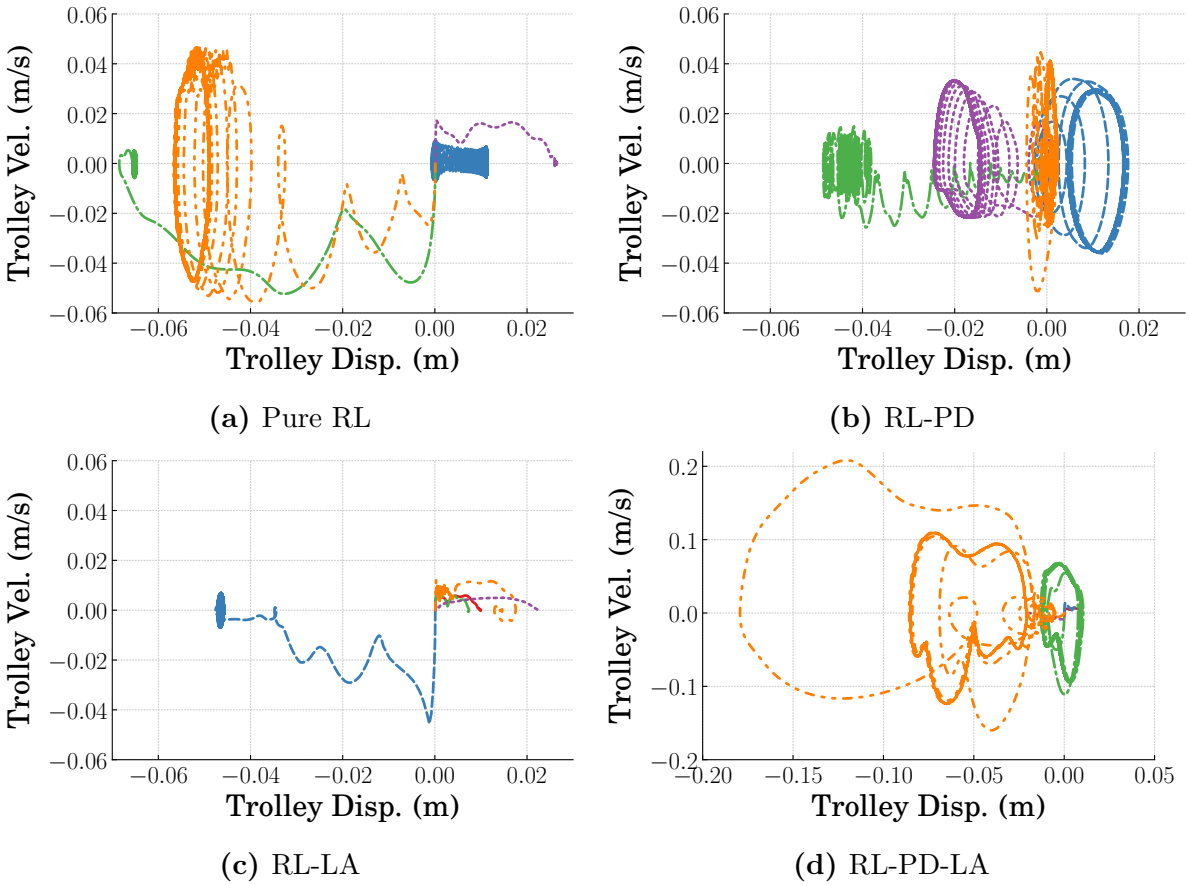


(a) Pure RL **(b)** RL-PD

(c) RL-LA **(d)** RL-PD-LA

Figure 49. Near equilibrium payload behavior

responses in Figure 50a tend to have low amplitude oscillation. However, the response in orange has a larger limit cycle with higher velocity. The RL-PD responses in Figure 50b tend to have larger limit cycles with higher amplitude in both trolley displacement and velocity. The RL-LA responses in Figure 50c do not have limit cycles and have less variance in displacement and velocity except for the response in blue, which travels in the opposite direction as the other responses. The RL-PD-LA phase plot responses are shown in Figure 50d. It should be noted that the plot window for this figure is larger than for the other figures. Although three of the RL-PD-LA responses have low displacement and velocity, the responses in green and orange have large and nonsmooth limit cycles. The bounds for which the trolley responses satisfy Lyapunov stability are summarized in Table 12. RL-LA has the smallest mean of the bounds that satisfy the stability conditions.



(a) Pure RL **(b)** RL-PD

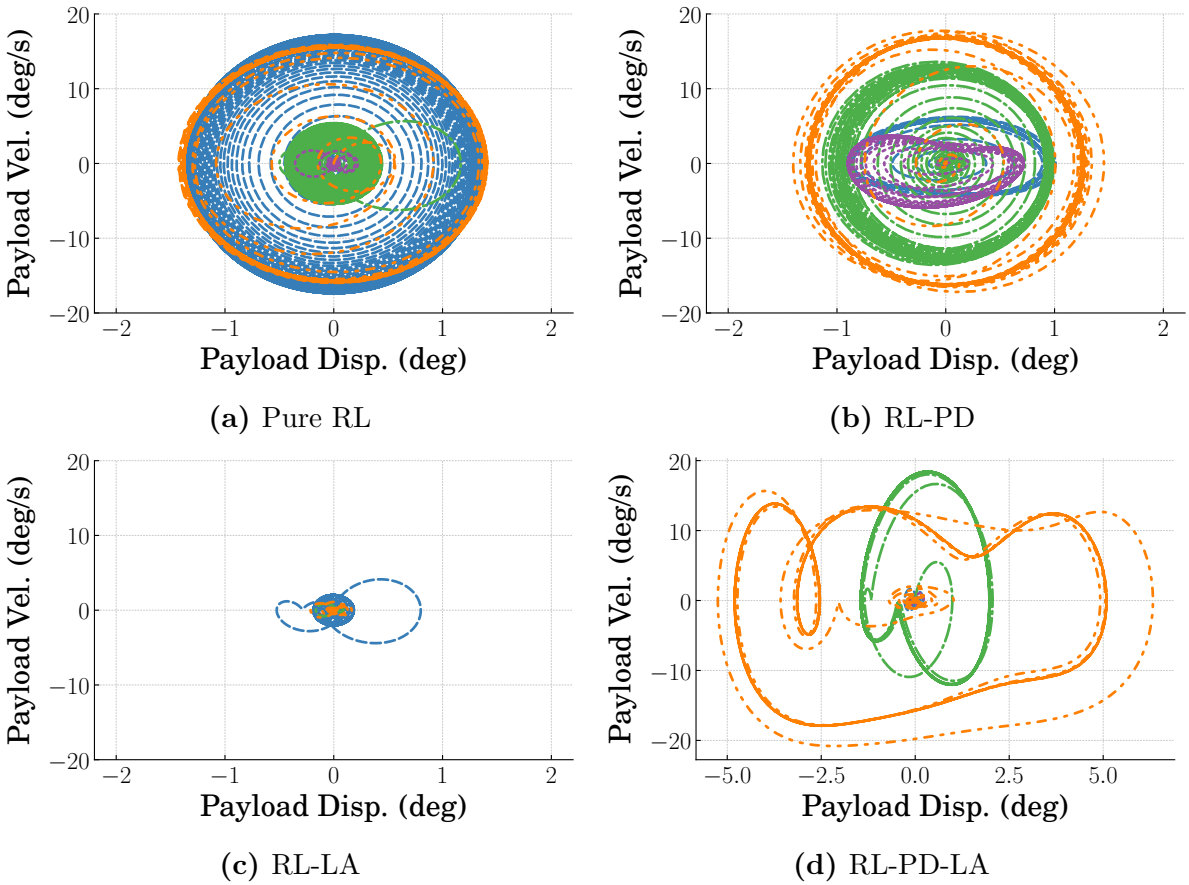
(c) RL-LA **(d)** RL-PD-LA

Figure 50. Phase diagrams for trolley response near equilibrium

Table 12. Planar Crane Lyapunov Stability

Trolley				
	Pure RL	RL-PD	RL-LA	RL-PD-LA
Mean	0.037	0.036	**0.021**	0.075
SD	0.028	0.019	**0.014**	0.082
Payload				
	Pure RL	RL-PD	RL-LA	RL-PD-LA
Mean	8.334	8.697	**1.461**	8.543
SD	7.092	6.259	**1.514**	9.124

Phase diagrams for the payload responses are shown in Figure 51. The trends seen previously for the trolley responses are similar for the payload responses. The phase diagram of the RL-LA responses in Figure 51c have low displacement and velocity compared to the other controllers, corresponding to the results shown above.



(a) Pure RL **(b)** RL-PD

(c) RL-LA **(d)** RL-PD-LA

Figure 51. Phase diagrams for payload response near equilibrium

The payload responses for RL-PD-LA are shown in Figure 51d. It should be noted that the horizontal range for the plotting window of this figure is larger than that of the other figures. Although three of the responses have low displacement and velocity, two responses have limit cycles with larger displacement amplitudes than the responses for the other cases. The bounds for which the payload responses are Lyapunov stable are shown at the bottom of Table 12. RL-LA has the lowest mean bound and standard deviation. The other controllers have similar mean bounds for Lyapunov stability.

In summary, RL-PD had the best trolley performance in terms of BIBO stability. However, due to the tendency of the controller to excite residual oscillation, it did not perform as well for either BIBO stability of the payload or for Lyapunov stability. RL-LA consistently had the best BIBO stability of the payload and the best

Lyapunov stability of the trolley and payload.

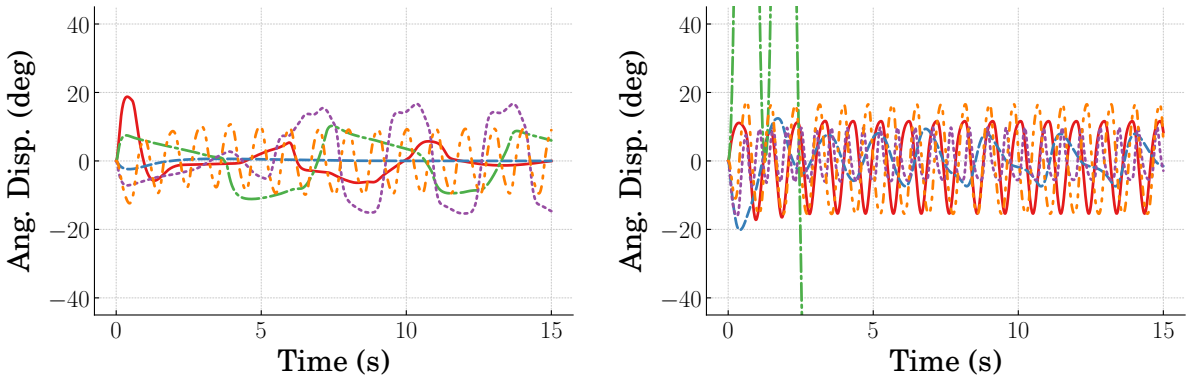
3.1.3 *Lyapunov Stability of Inverted Pendulum*

A stability evaluation of the inverted pendulum is presented in this section. The control laws for this system that were introduced in Chapter 2 are:

$$\begin{aligned}
&\text{Pure RL:} && u = a_t \\
&\text{RL-LQR:} && u = -k_1\theta - k_2\dot{\theta} + a_t \\
&\text{S-RL-LQR:} && u = \begin{cases} -k_1\theta - k_2\dot{\theta}, & \forall(\theta,\dot{\theta}) \in S_{st} \\ a_t, & \forall(\theta,\dot{\theta}) \notin S_{st} \end{cases}
\end{aligned}$$

where u is the acceleration input to the cart. RL-LQR combines an action, a_t, from the RL agent with a fixed-gain LQR controller that is stable about the upright equilibrium at $\theta = 0$. LQR and the agent operate concurrently in this controller. For S-RL-LQR, the agent controls the system for the swing up phase when the pendulum is far from the equilibrium, and LQR controls the system on its own when the pendulum is within the region of stability for LQR near the equilibrium.

Time responses for the inverted pendulum with initial conditions of $\theta(0) = \dot{\theta}(0) = 0$ are shown in Figure 52. Only the Pure RL and RL-LQR cases are shown since the S-RL-LQR controller does not use an RL agent term near the equilibrium. All of the Pure RL responses in Figure 52a remain near the equilibrium at $\theta = 0$. However, all except one of the responses have residual oscillation. One of the RL-LQR responses in Figure 52b diverges from the equilibrium. The other responses remain near the equilibrium, but have residual oscillation. The mean bounds for which the responses are BIBO stable are summarized in Table 13. The unstable response was removed from the evaluation of RL-LQR, as indicated by the asterisk on the values. Even without the unstable response, the mean BIBO bound for RL-LQR was still higher than that of Pure RL.



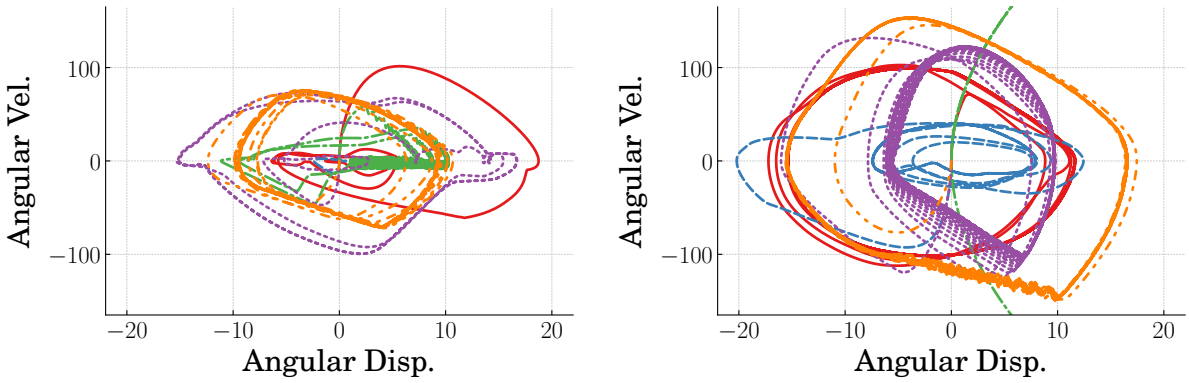
(a) Pure RL **(b)** RL-LQR

Figure 52. Inverted pendulum response with initial condition at equilibrium

Table 13. Inverted Pendulum Stability, * Indicates Unstable Responses Where Removed from the Evaluation

BIBO for Displacement			
	Pure RL	RL-LQR	S-RL-LQR
Mean	15.43°	18.83°*	**0.0°**
SD	7.45°	1.75°*	**0.0°**
Lyapunov Bound			
	Pure RL	RL-LQR	S-RL-LQR
Mean	70.1	123*	**0.0**
SD	32.2	22.6*	**0.0**

Figure 53 shows the phase diagrams for the inverted pendulum responses. The responses tend to have limit cycles corresponding to the residual oscillation seen in the time responses. The maximum velocities for Pure RL in Figure 53a tend to be lower than the maximum velocities for RL-LQR in Figure 53b. This corresponds to the higher frequency of the residual oscillation in the responses. The bounds for which the responses are Lyapunov stable are summarized in the bottom of Table 13. The unstable response was removed from the evaluation for RL-LQR, as indicated by the asterisk. The mean bound for the RL-LQR responses is higher than that of the Pure RL responses. Although the BIBO mean bound for RL-LQR is 22% higher than that of Pure RL, the Lyapunov bound for RL-LQR was 75% higher due to the higher velocity



(a) Pure RL **(b)** RL-LQR

Figure 53. Inverted pendulum phase diagram from equilibrium

of the responses.

S-RL-LQR had the best stability performance due to it only using LQR near the equilibrium. Pure RL had slightly better stability performance than RL-LQR when using BIBO, but it performed much better in terms of Lyapunov stability. The poor performance of RL-LQR near the equilibrium may be the result of the lack of experience of the agent near the equilibrium due to its poor performance for the swingup problem shown in Chapter 2.

3.2 Conclusion

The stability performance of the combined controllers for the benchmark systems was evaluated using BIBO and Lyapunov stability. The performance was summarized based on the combined controller architectures introduced in Chapter 2, which were Pure RL, agent-driven gain scheduling, and fixed-gain control with RL.

The controllers with Pure RL tended to have residual oscillation and steady-state error for all of the benchmark systems. The inverted pendulum was the only system for which there was no steady-state error. There was no benchmark system for which Pure RL was the best performing controller architecture, and it instead tended to provide moderate performance.

The stability of the agent-driven gain scheduling controllers was dependent on

the benchmark system employed. For the duffing oscillator, the RL-PD responses did not diverge from the equilibrium such that it had the best performance for both BIBO and Lyapunov stability. For the double-pendulum crane, however, the controllers with gain scheduling had good stability performance for the trolley but had moderate to poor performance for the stability bounds for the payload. This poor performance may be due to a poorly designed feedback controller rather than being a weakness of the agent-driven gain-scheduling architecture.

Combined controllers that used a fixed-gain term tended to have little to no residual oscillation; however, they often had steady-state error. For this reason, the RL-LA controller for the duffing oscillator had the largest BIBO stability bounds, but had smaller Lyapunov bounds than Pure RL. Similarly, the RL-LA stability bounds for the double-pendulum crane tended to be the smallest due to the vibration mitigation. However, RL-LQR for the inverted pendulum had the largest stability bounds, which may be related to the poor performance seen for the baseline results in Chapter 2.

Although none of the Pure RL controllers had the best stability performance, there was often a combined controller architecture that had worse performance. This suggests that combined controllers can improve stability compared to Pure RL provided that an appropriate controller architecture is used for the system.

IV Robustness

One of the primary requirements of reinforcement learning in robotics is to train agents that work safely and effectively on a physical system. Because RL agents are usually initially trained in simulation, agents must be robust to parametric uncertainty and unmodeled dynamics in order to be successfully transferred from simulation to reality (sim-to-real) [49, 50]. In the absence of physical systems to perform sim-to-real evaluation for this work, robustness was evaluated by simulating modeling error for the benchmark systems.

4.1 Duffing Oscillator Robustness

The performance of the duffing oscillator with modeling error is dependent on the sensitivity of the model-based elements and RL elements of the controllers. The model-based component of the controllers is:

$$f(t) = k_n x_d + \beta_n x_d^3 \tag{43}$$

where k_n is the nominal linear stiffness and β_n is the nominal nonlinear stiffness. Figure 54 shows the mean reward for a range in modeling error for k, β, and damping ratio, ζ. Since the fixed-gain components are dependent on the nominal stiffness values and were designed to result in zero steady-state error, modeling error in the stiffness values causes steady-state error in the RL-PD and RL-LA responses. This steady-state error can reduce the reward from the responses. However, the reward for modeling error of k in Figure 54a increases with positive modeling error. This is due to lower oscillation amplitude compared to responses with k lower than the nominal value. The reward trend with modeling error in k are similar for the β modeling error shown in Figure 54b. The reward is lower for β that is lower than nominal due to higher oscillation amplitude, but reward is higher for larger β due to lower oscillation amplitude. The rewards from the time responses with error in damping ratio are shown in Figure 54c. The nominal damping ratio used during training was $\zeta = 0.01$, indicated by the vertical

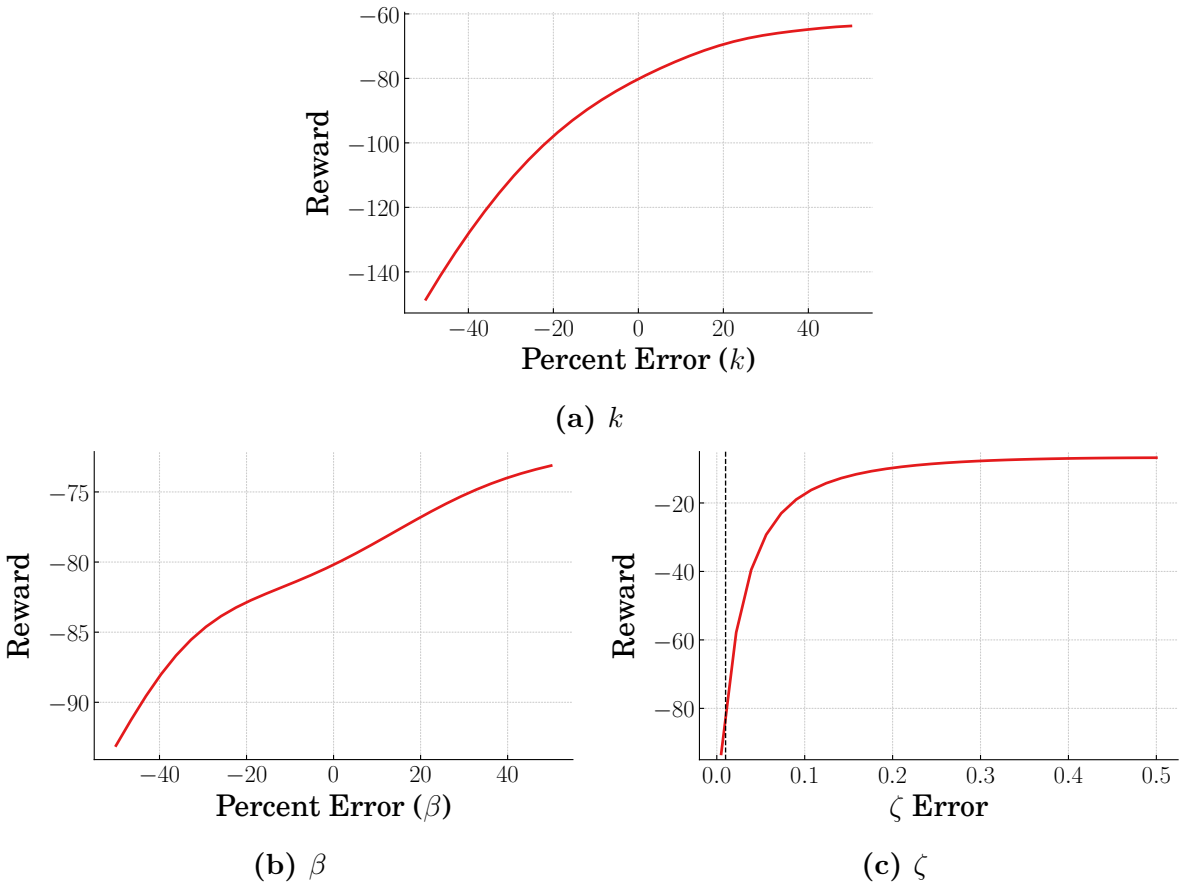
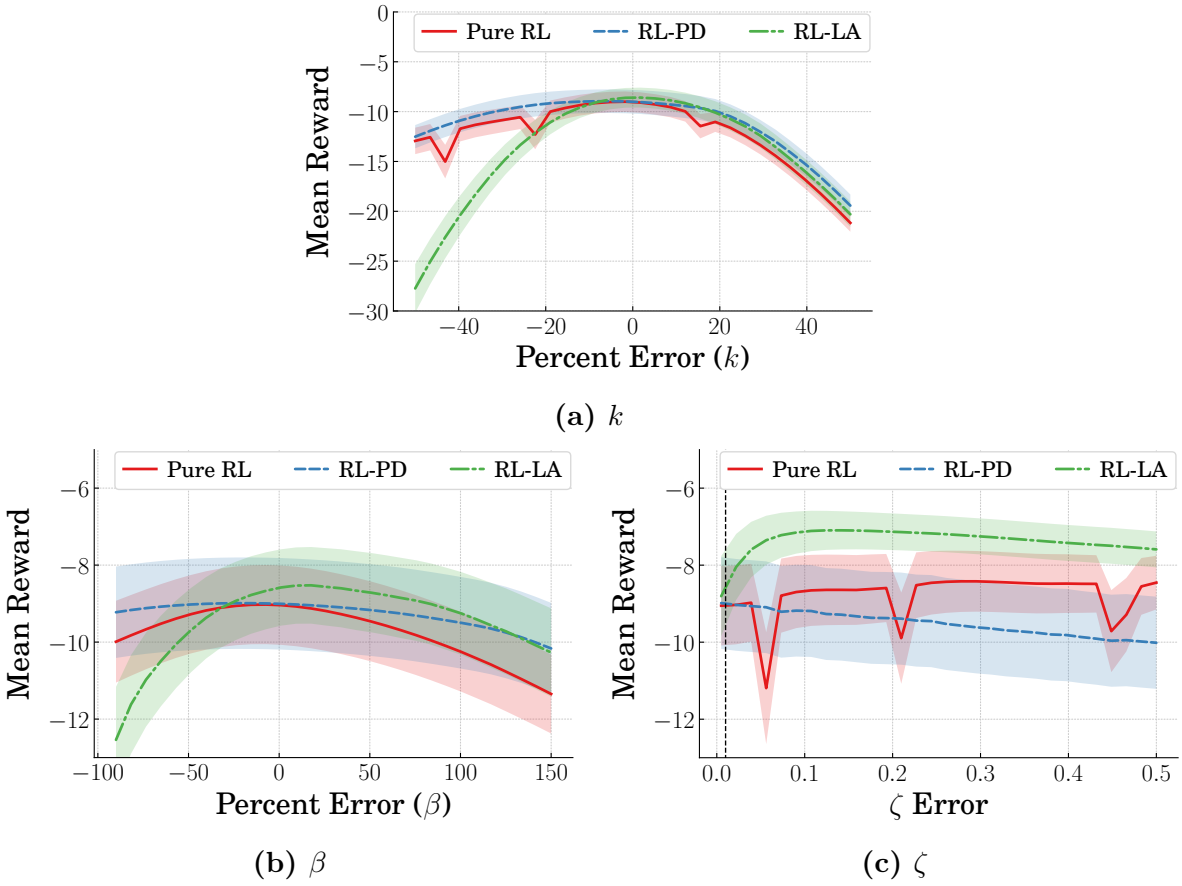


Figure 54. Robustness of model-based duffing oscillator controller

dashed line. Damping ratio was not used to design the fixed-gain controller and modeling error in damping ratio does not cause steady-state error. Therefore, the mean reward increases with increasing damping ratio due to faster oscillation decay.

The plots in Figure 55 show the sensitivity of the combined controllers to modeling error. The mean reward was generated from the cumulative reward for the time response with each agent and controller type. The lines in each figure represent the mean reward and the shaded region is one standard deviation. Figure 55a shows the mean reward and standard deviation for linear stiffness modeling error, k. The mean reward decreases for all controllers as the modeling error increases. For modeling error less than the nominal stiffness, Pure RL and RL-PD have smaller rates of decrease in reward compared to RL-LA. For stiffness modeling error greater than nominal



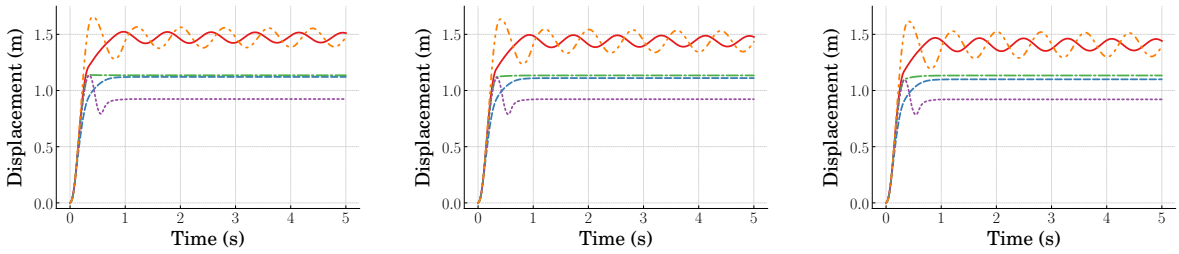
(a) k

(b) β

(c) ζ

Figure 55. Robustness of RL and combined controllers to modeling error

stiffness, the different controllers have similar rates of decrease in mean reward. However, RL-PD tends to have the highest reward across most of the range of stiffness modeling error shown. The rapid decrease in reward for RL-LA with lower than nominal stiffness is explained by Figure 56, which shows time responses for -50%, nominal, and $+50\%$ modeling error in stiffness. Two responses in Figure 56a for -50% modeling error have higher steady-state error than the other cases. For modeling error in β in Figure 55b, the trends are similar to those shown for k modeling error where RL-LA has the greatest sensitivity to β that is lower than the nominal value. However, Pure RL has the highest sensitivity to β greater than the nominal value. Figure 55c shows the mean reward for a range of damping ratios. The controllers have less variation in mean reward from damping modeling error compared to modeling error in



(a) −50% error **(b)** Nominal **(c)** +50% error

Figure 56. RL-LA time responses for error in stiffness, k

the stiffness parameters, k and β. The reward with Pure RL tends to increase with increased damping ratio whereas the reward from RL-LA initially increases before gradually decreasing. The reward with RL-PD only decreases with increasing damping ratio due to the responses already being highly damped for the nominal damping ratio.

In general for the duffing oscillator controllers, the RL-PD controller had the greatest insensitivity to modeling error compared to Pure RL and RL-LA. RL-LA tended to be most sensitive to k and β values that were below the nominal values. However, the controllers still tend to perform well even for large modeling error.

4.2 Double-Pendulum Crane Robustness

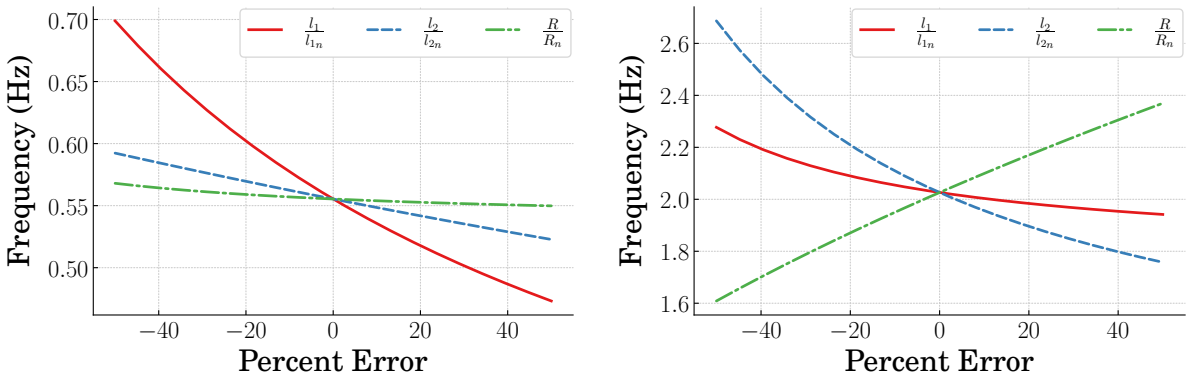
4.2.1 *Modal Contributions*

The robustness of the double-pendulum crane controllers were evaluated for a range in hoist length, l_1, rigging length, l_2, and payload mass, m_p. The performance degradation from parameter variation can be the result of changes to the modes of oscillation and response amplitudes. The two modes of a linearized double-pendulum are [51]:

$$\omega_{1,2} = \sqrt{\frac{g}{2}}\sqrt{(1+R)\left(\frac{1}{l_1}+\frac{1}{l_2}\right) \mp \beta} \tag{44}$$

where R is the payload-to-hook mass ratio and β is:

$$\beta = \sqrt{(1+R)^2\left(\frac{1}{l_1}+\frac{1}{l_2}\right)^2 - 4\left(\frac{1+R}{l_1 l_2}\right)} \tag{45}$$



(a) Low mode **(b)** High mode

Figure 57. Modes of crane vs. modeling error

Assuming the amplitude of oscillation is low, the total response amplitude from a unity magnitude impulse is the sum of the amplitude responses due to both angular displacements:

$$A_{Total} = A_1 + A_2 \tag{46}$$

where

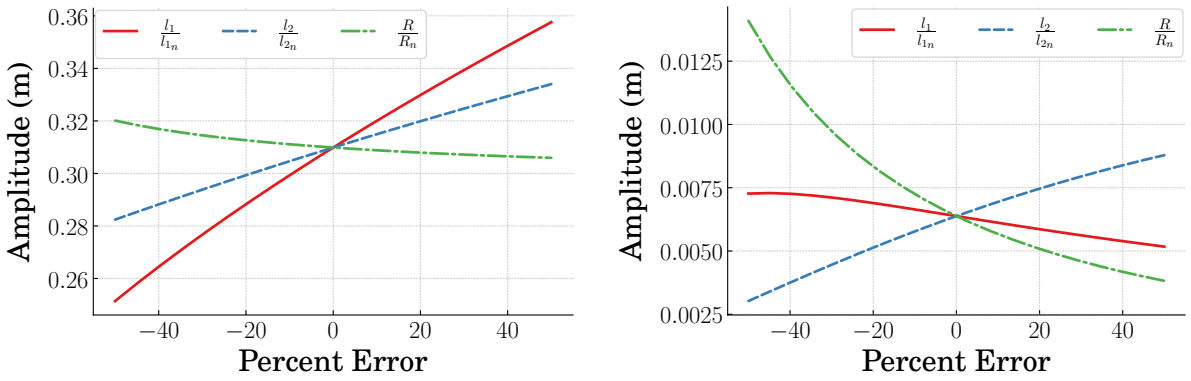
$$A_1 = \frac{\omega_1 l_1 (1 + {\omega_2}^2 \alpha (l_1 + l_2))}{k} \tag{47}$$

$$A_2 = \frac{\omega_2 l_1 (1 + {\omega_1}^2 \alpha (l_1 + l_2))}{k} \tag{48}$$

$$\alpha = \frac{-g(1+R)}{{\omega_1}^2 {\omega_2}^2 l_1 l_2} \tag{49}$$

$$k = \beta l_1 g \tag{50}$$

The change in the modes of the crane for modeling error is shown in Figure 57. The variation in the low mode in Figure 57a show that modeling error in hoist length, l_1 has the most significant effect on the low mode for the range of $\pm 50\%$ modeling error. However, hoist length has the lowest effect on the high mode shown in Figure 57b. Modeling error in rigging length, l_2, and payload-to-mass ratio, R, have the greatest effect on the high mode frequency. The response amplitude from a unity magnitude impulse on a linearized double-pendulum is shown in Figure 58. The



(a) Low mode contribution amplitude **(b)** High mode contribution amplitude

Figure 58. Amplitude contributions of crane modes

contribution from the low mode is most significantly affected by the hoist length, l_1, as shown in Figure 58a. This corresponds with the change in low-mode frequency. For the high-mode contribution in Figure 58b, payload-to-hook mass ratio, R, has the largest impact on the amplitude. Based on the changes in modal frequency and modal amplitude contributions, the most significant factors affecting performance degradation are modeling error in hoist length, l_1, and payload-to-hook mass ratio. The rest of this section discusses robustness of the double pendulum crane controllers to this modeling error.

4.2.2 *Crane Controller Robustness*

The agents discussed previously were trained with fixed parameters. The sensitivity of the controllers to modeling error was evaluated for the final weights during training at episode 3000. The controllers were evaluated over a range of ±50% modeling error. Figure 59 shows the mean and standard deviation of reward of the controllers for percent modeling error in hoist length, l_1. The modeling error does not significantly affect the mean reward for most of the range shown. However, the mean reward from RL-PD is much lower for modeling error 40% less than the nominal value. This decrease in reward is explained by the payload time responses with modeling error

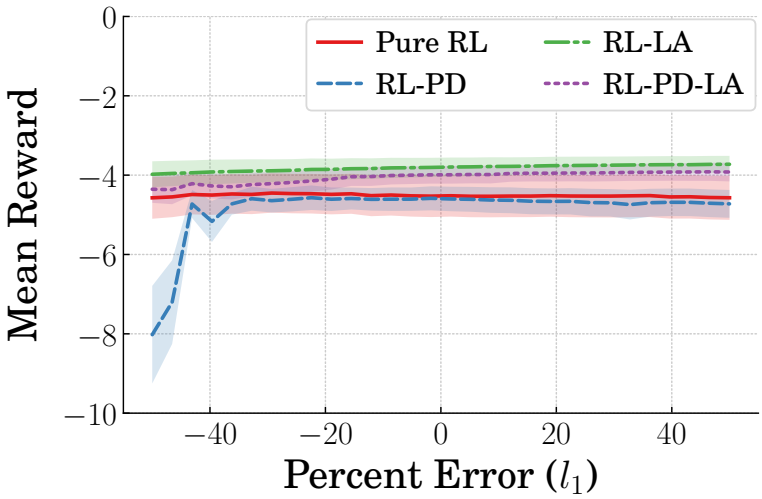
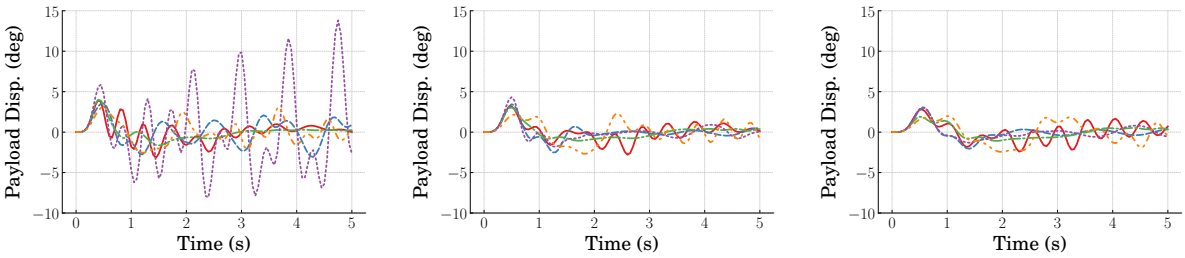


Figure 59. Mean reward for a range of hoist lengths



(a) −50% error **(b)** Nominal **(c)** +50% error

Figure 60. RL-PD payload time responses for error in hoist length, l_1

in hoist length shown in Figure 60. The response amplitude for nominal hoist length in Figure 60b is similar to the case with hoist length 50% longer than nominal in Figure 60c. The case with hoist length 50% shorter than nominal has one response with growing amplitude that skews the mean of the reward.

Although the controllers tend to have low sensitivity to hoist length modeling error, the reward from RL-LA and RL-PD-LA slightly increase with increasing hoist length whereas the reward from Pure RL and RL-PD slightly decrease. Similarly, Figure 61 shows the change in reward for percent modeling error in rigging length, l_2. All of the controllers show low sensitivity to rigging length. This was expected from the modal analysis, which showed that rigging length had less influence on modal contributions than other parameters. The mean reward for payload mass modeling

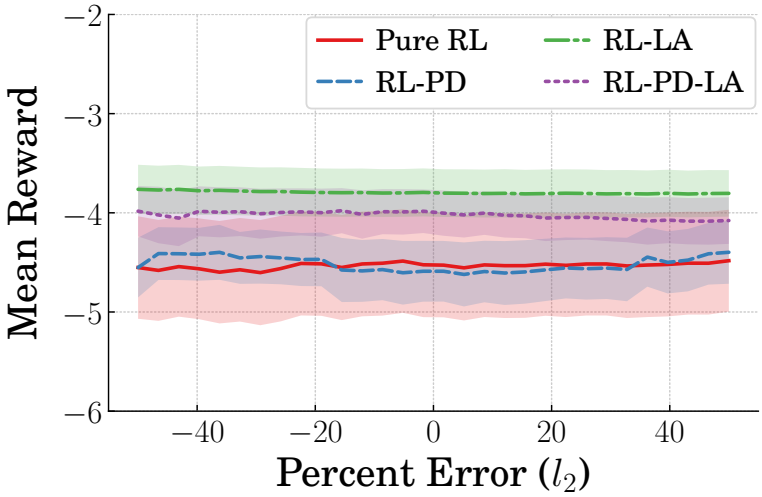


Figure 61. Mean reward for a range of payload lengths

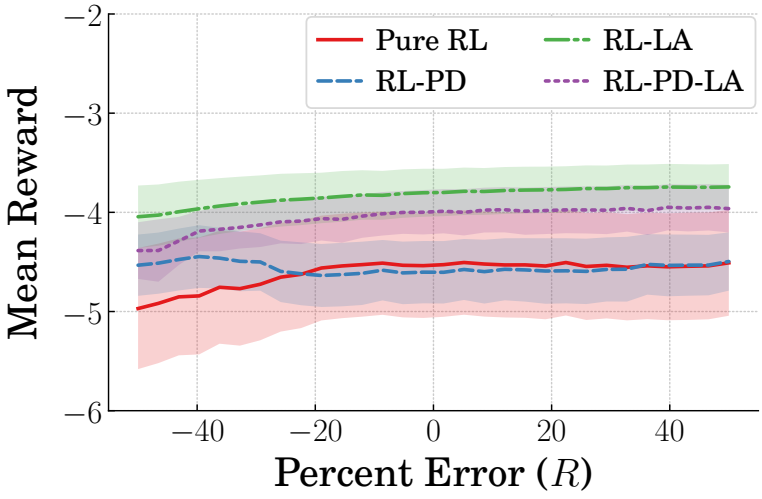


Figure 62. Mean reward for a range of mass ratios

error, R, is shown in Figure 62. The mean reward for most of the controllers tends to decrease as payload-to-hook mass decreases. However, the reward for RL-PD increased for payload-to-mass ratio less than the nominal value. The standard deviation remains insensitive to modeling error for all parameters, with standard deviation from RL being higher than that of the combined controllers.

The hoist length, l_1, and payload-to-hook mass ratio, R, had the most significant impact on the mean reward from the controllers. However, despite the changing modal characteristics of the double-pendulum crane, the controllers tended to maintain low

sensitivity to modeling error with RL-LA and RL-PD-LA having the lowest sensitivity.

4.3 Inverted Pendulum Robustness

This section presents a robustness analysis for the agents trained for the inverted pendulum. Since the previous sections have already presented robustness analysis for parametric modeling error, the robustness of inverted pendulum was evaluated for unmodeled dynamics. Agents that were not able to cause the system to settle in the region of attraction in the ideal case were removed for the following robustness evaluation.

4.3.1 *Friction*

The agents were trained without friction in the ideal model. However, friction is a part of any physical system, and it is beneficial to determine the robustness of a controller to unmodeled friction. Since the input of the trolley was modeled as an acceleration input instead of a force, the friction was normalized by mass and modeled as an acceleration disturbance. The equation of motion of the inverted pendulum with coulomb friction is:

$$\ddot{\theta} = (g \sin\theta + \cos\theta(\ddot{x}_d + f))/l \tag{51}$$

$$f = \begin{cases} -f_m \text{sgn}(\dot{x}) & \forall x \neq 0 \\ -\text{sat}_{-f_m}^{f_m}(\ddot{x}_d) & \dot{x} = 0 \end{cases} \tag{52}$$

where f is the acceleration disturbance from normalized friction, and f_m is the maximum disturbance magnitude. Stiction occurs when the velocity of the trolley is zero, $\dot{x} = 0$, and friction is constant when the trolley is moving. Figure 63 shows the mean reward and standard deviation of the agents for a range of coulomb friction acting on the cart. For a range of normalized friction from 0 to approximately 1.9N/kg the mean reward remains relatively unchanged as the agents are still capable of stabilizing the system. After this point, the mean reward begins decreasing more

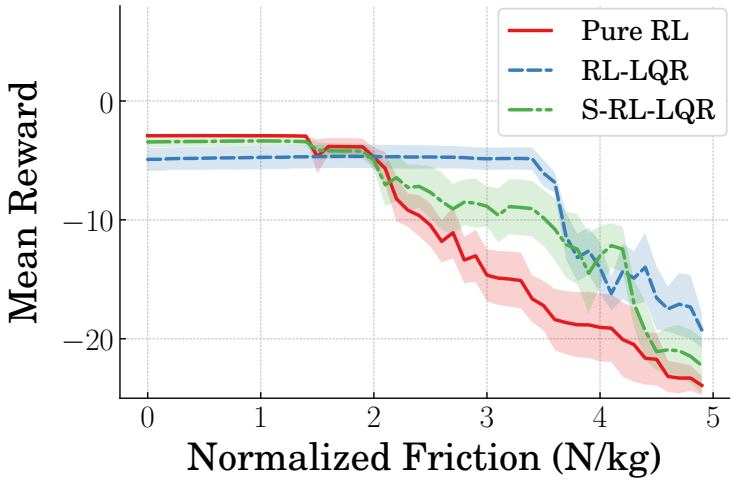


Figure 63. Reward with normalized friction

steeply for Pure RL and S-RL-LQR. At 2N/kg, some of the S-RL-LQR agents begin to fail to stabilize the inverted pendulum, resulting in the decrease in mean reward. At 2.2N/kg, some of the Pure RL agents begin failing to stabilize the system. After this, the mean reward for Pure RL is lower than that of the other controller types. However, all of the RL-LQR agents maintain the ability to stabilize the system for a larger range of normalized friction. The RL-LQR agents begin to fail to stabilize the system for normalized friction above 3.6N/kg.

4.3.2 *Feedback Noise*

Feedback noise is also a common occurrence when implementing sensors for feedback on a real system. Figure 64 shows the mean reward from the agents when subjected to feedback noise. Noise was applied using a Gaussian distribution with a mean of $\mu = 0$. Increasing the standard deviation of the distribution increases the severity of the added noise signal. Because the observation was normalized between -1 and 1 during training, the noise signal was added to the normalized feedback signal. The RL-LQR agents tend to have the lowest robustness to feedback noise compared to Pure RL and S-RL-LQR. The agents begin to fail to stabilize the system at noise standard deviation of $\sigma = 0.025$. The Pure RL and S-RL-LQR agents begin to fail soon

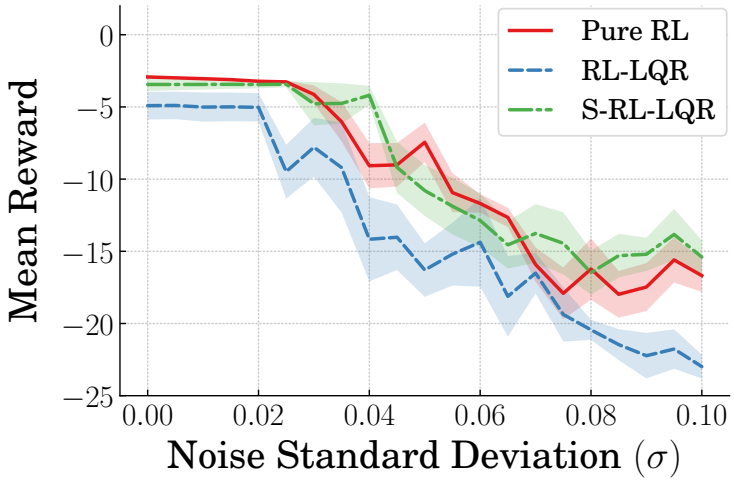


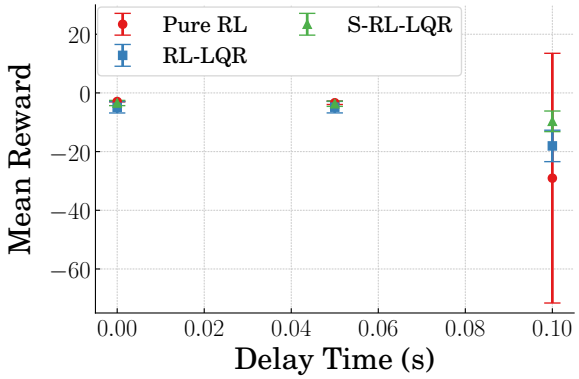
Figure 64. Reward with observation noise

after this at $\sigma = 0.03$, where two Pure RL agents and one S-RL-LQR agent fails during the simulation time. Between $\sigma = 0.03$ and $\sigma = 0.045$, only one of the S-RL-LQR agents have failed to achieve stability, whereas multiple of the Pure RL and RL-LQR agents have failed. Multiple S-RL-LQR agents begin to fail at $\sigma = 0.045$, after which all agents fail to maintain stability for increasing noise severity.

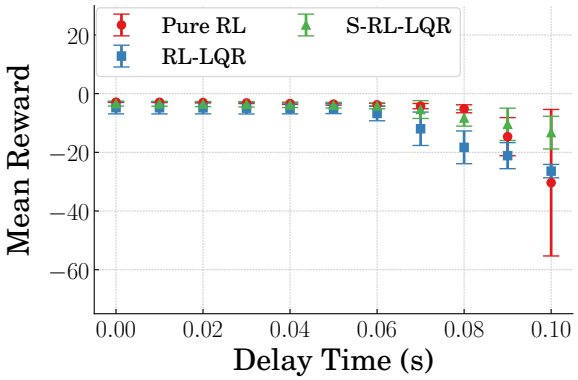
4.3.3 *Feedback Delay*

Delay is often introduced into a system as a result of required computation time for the controller. Since the controllers in this work were trained and implemented with a fixed sampling rate, the simulated delay was modeled as multiples of the sampling time. Therefore, robustness was evaluated for multiple sampling rates. Figure 65 shows the mean reward and standard deviation for each controller type with delay. The controllers with different sampling rates have similar trends in mean reward as delay increases. The controllers have low sensitivity to delays less than 0.05s, which was the sampling time used for training. The mean rewards begin to significantly decrease for delays greater than 0.05s. For longer delays, S-RL-LQR has the highest mean reward, whereas Pure RL has the lowest mean reward and highest standard deviation.

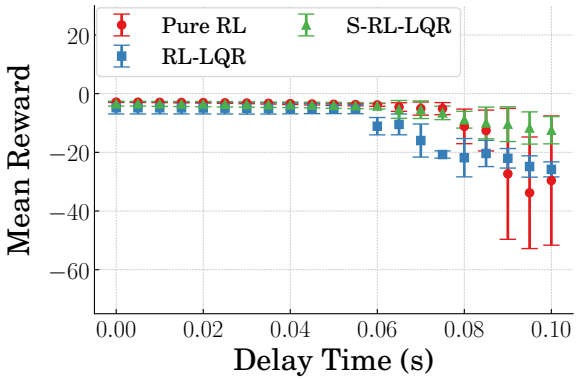
Angular displacement time responses with a delay of 0.1s and sampling time of



(a) 0.05s sampling time



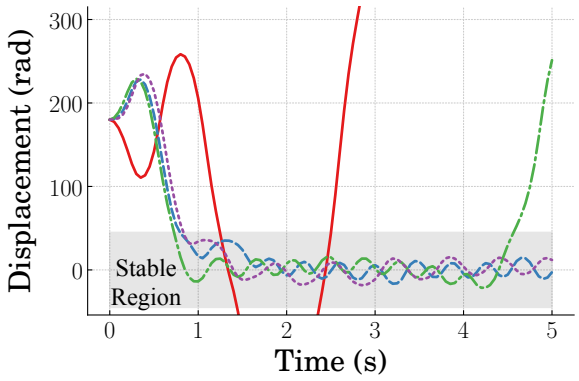
(b) 0.01s sampling time



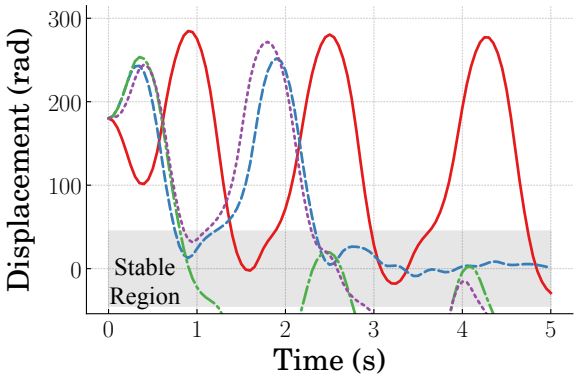
(c) 0.005s sampling time

Figure 65. Reward trends with time delays

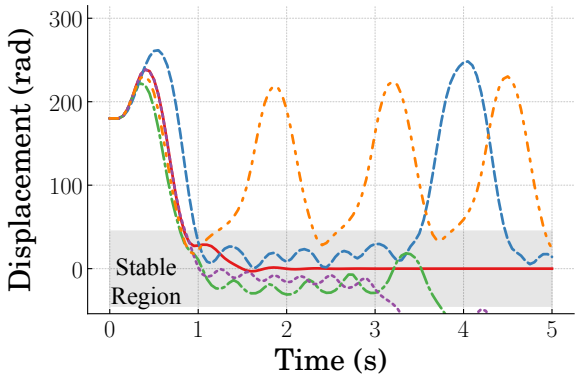
0.05s are shown in Figure 66. The decrease in the mean reward and increase in standard deviation for the Pure RL controllers is a result of two of the agents diverging far from the equilibrium at $\theta = 0$ as shown in Figure 66a. The remaining two agents successfully stabilize the inverted pendulum about the equilibrium. Although the mean reward with RL-LQR was higher than that of Pure RL with the delay, fewer of the agents were able to stabilize the system as shown in Figure 66b. However, the responses that did not enter the stable region also did not diverge to displacements as large as those of Pure RL, resulting in the higher mean reward seen previously. This is similar for the S-RL-LQR angular displacement responses for delay shown in Figure 66c. Only one of the agents was able to stabilize the system for the duration of the response. However, the unstable S-RL-LQR responses remained closer to the equilibrium than the



(a) Pure RL



(b) RL-LQR



(c) S-RL-LQR

Figure 66. Angular displacement with 0.1s delay

unstable Pure RL responses.

In summary, the combined controllers for the inverted pendulum tended to have the least sensitivity to unmodeled friction compared to Pure RL. S-RL-LQR and Pure RL had the least sensitivity to high frequency noise compared to RL-LQR. For the cases with feedback delay, although more Pure RL responses tended to remain stable for a longer time period, the unstable responses from the combined controllers tended to remain closer to the equilibrium.

4.4 Conclusion

Robustness of the combined controllers to modeling error was evaluated in this chapter. Each benchmark system had different places where modeling error could be introduced, and the modeling error was simulated for each system. The controllers for

the duffing oscillator were evaluated with modeling error in stiffness and damping ratio. The Pure RL and combined controllers had high robustness to modeling error compared to the robustness of the fixed-gain controller alone, where Pure RL and RL-PD were the least sensitive to modeling error. For the double-pendulum crane, modeling error in hoist length, rigging length, and payload-to-hook mass ratio were introduced. The controllers for the crane tended to have low sensitivity to modeling error in all parameters except for the sensitivity of RL-PD to short rigging lengths. For the inverted pendulum, friction, feedback noise, and feedback delay were introduced to the system. Both combined controllers had more robustness to friction than Pure RL, with RL-LQR being the most robust. S-RL-LQR had the highest robustness to feedback noise. Although the combined controllers tended to have less sensitivity to friction and feedback noise, Pure RL tended to have the most robustness to feedback delay.

V Interpreting Policies

5.1 Interpretability Background

Reinforcement Learning (RL) with deep neural networks has increasingly been used for learning applications due to the ability to solve high-dimensional problems with continuous observation and action spaces. However, the large number of parameters used in the network makes it difficult to develop intuitive interpretations of the logic that governs the policy. This is contrary to conventional controllers which typically describe the relationship between the control output and input with few parameters. In other words, the neural network is a black box which takes observations as its input and provides the actions as the output, but a detailed explanation of the relationship between the observations and corresponding actions is generally unknown.

Post-training methods of interpretation are necessary to provide explanations for the behavior of the agent. Many post-training interpretation methods that have been proposed for image-based agents, where the inputs to the neural networks for the agents consist of pixels from image data rather than the state of a dynamic system [37]. Another class of interpretation methods relies on developing high-level summaries of the behavior of the agent [52]. However, these methods are often time-consuming and require significant manual interpretation. There are not many general methods for providing detailed explanations for low-level agent behavior. One applicable method for low-level interpretation of agents is decision trees [40]. Decision trees can be used to distill the deep neural network model onto a shallower model with fewer nodes. The reduction of the model size facilitates developing an intuitive understanding of the behavior of the agent. However, decision trees must be trained from the deep model. Depending on the complexity of the deep neural network model, the decision tree may still contain a significant number of nodes, making it unamenable to interpretation.

This chapter proposes an interpretation method for state-based observations that relies on established control methods to interpret complex policies. The method

relies on approximating the agent as a switching controller. The viability of this method was tested on the double-pendulum crane and inverted pendulum benchmark systems.

5.2 Interpretable Switching

There are well established methods for designing linear controllers to provide a linear system with the desired performance characteristics. However, these methods are not directly applicable to nonlinear systems. One way to simplify controller design for nonlinear systems is to use a set of local controllers designed for subsets of the operating space of the system, where the behavior of the system in a subset of the operating space is easier to describe with linear approximations. A switching law is then used to combine the individual local controllers into a global control law. One method for designing the local controllers is gain scheduling control, where the form of each local controller is the same, and the gains are updated based on the switching or scheduling rule.

The two main types of gain scheduling include discrete scheduling and the Linear Parameter Varying method. For the discrete gain scheduling method, the system is linearized about multiple points in state-space; gains of a state feedback controller are then selected to provide desirable performance when the system is near that linearization point [53, 54]. This forms a type of piecewise controller for different regions of the workspace. Since this makes discrete regions in the state space where the gains are fixed, the gains are often interpolated between the different regions. It is also possible to use Linear Parameter Varying control to generate a continuous switching law for the gains [55, 56]. In gain scheduling control, a single scheduling variable is often used to indicate when to switch from one set of gains to another [57]. To be effective, this scheduling variable needs to be meaningful and capture the nonlinearity of the system.

Although gain-scheduling and switching control are usually used to simplify

controller design for nonlinear systems, this work proposes using local approximations of RL agents to improve interpretability. By fitting local behaviors of the black-box agent to local approximations with few parameters, the behavior of the agent in a subregion of state space can be more easily understood. A more general understanding of the behavior of the agent can then be developed by consolidating the interpretations of the local approximations. There are multiple challenges with representing an agent with a switching controller. The local controller will need to approximate the input-output law of the agent for a subset of the observation space. However, there can be error in the local approximations of the agent in each subset of state space, which may lead to significant error in the time response. Factors that contribute to the accuracy of the fit include using an appropriate scheduling variable, an appropriate fit function, and an appropriate number of local control functions.

The local approximations in the piecewise controllers were generated by minimizing the error between the agent's output and the output of the local controller. Each local controller approximates the output of the agent for a subset, s_l, of all possible states, $\mathbb{S} \subset \mathbb{R}^n$, where n is the system order. The subset, s_l, is defined for a range of the scheduling variable. A local approximation is then generated by:

$$\text{minimize} \quad u(s) - \hat{u}_l(s) \quad \forall s \in s_l \tag{53}$$

where $u(s)$ is the output from the agent controller for an input state, s, and $\hat{u}_l(s)$ is the output of the local approximation. The local approximations are then collected to form $\hat{u}(s)$, which approximates the agent $\forall s \in \mathbb{S}$. For the combined controllers, the conventional control components remain unchanged while the RL agent components are replaced with the piecewise local approximations.

5.3 Model Verification

After generating the switching controller approximations, it is necessary to verify the accuracy of the approximations in terms of time-based performance to

determine the trustworthiness of the approximation for interpreting the agents. This work uses Integral Square Error (ISE) of the time responses from the agent controllers and switching controller approximations:

$$ISE = \int_0^T (y(t) - \hat{y}(t))^2 \, \mathrm{d}t \tag{54}$$

where $y(t)$ is the output from the response with the agent controller, and $\hat{y}(t)$ is the responses from the approximate switching controller. ISE provides a single measurement to quantify the response error between the response with the agent and the responses with the switching controller. However, the ISE will depend on the initial condition, so the ISE must be evaluated for an adequate number of samples to evaluate the accuracy of the switching controller.

5.4 Interpreting Combined Controllers

5.4.1 *Double-Pendulum Crane Agent Approximation*

This section presents an analysis of switching controllers for the double-pendulum crane shown previously in Chapter 2. Multiple switching controllers with different numbers of local approximations were generated for the Pure RL and combined controllers. As a reminder, the controllers trained for the double-pendulum crane were:

Pure RL:	$u = a_t$
RL-PD:	$u = k_p x + k_d \dot{x} + \boldsymbol{a}_t \boldsymbol{\theta}$
RL-LA:	$u = k_p x + k_d \dot{x} + a_t$
RL-PD-LA:	$u = k_p x + k_d \dot{x} + \boldsymbol{a}_t \boldsymbol{\theta} + a_t$

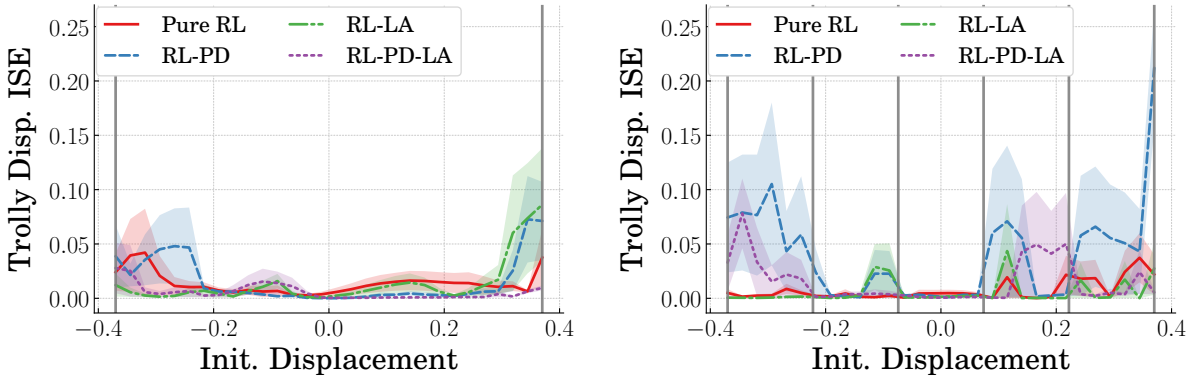
where u is the desired acceleration command for the crane trolley. Pure RL is a controller using just the action, a_t, as the output. RL-PD has fixed-gain terms on the trolley states as well as gain scheduled terms on pendulum states. RL-LA has the

fixed-gain component and a single, lumped action. RL-PD-LA has fixed-gain terms, gain scheduling, and lumped action. The inputs for the local controllers are the same as the neural network policies, which are the displacement and velocities of the trolley, hook, and payload of the crane. Polynomials are commonly used for general curve fitting applications and are used here to generate local approximations of the behavior of the agent:

$$\hat{a}_l = \sum_{i=1}^{p} a_i x^i + b_i \dot{x}^i + c_i \theta_1^i + d_i \dot{\theta_1}^i + e_i \theta_2^i + f_i \dot{\theta_2}^i \tag{55}$$

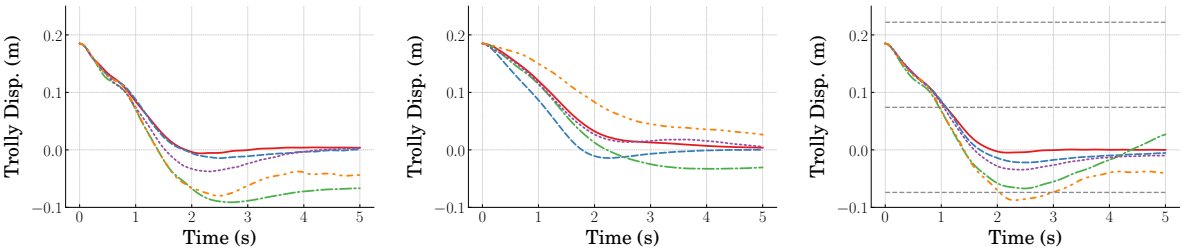
where l is the current local controller, and p is the degree of the polynomial, x and $\dot{x}$ are the trolley displacement and velocity, θ_1 and $\dot{\theta}_1$ are the angular displacement and angular velocity of the hook, and θ_2 and $\dot{\theta}_2$ are the angular displacement and angular velocity of the payload. Although polynomials of higher degree provide the potential to generate more accurate approximations, the increase in the number of parameters in each local controller can decrease interpretability. Therefore, the polynomial degree selected for the crane controller approximations was $p = 3$. The most vital function of the crane is to move the trolley to the desired displacement. Therefore, the scheduling variable used to switch between the local controllers is the displacement of the crane trolley.

Figure 67 shows the ISE of the trolley responses for two sets of switching controllers using one and five local approximations. The ISE is measured for a range of initial values of the scheduling variable. Since multiple agents were trained for each controller type, the lines in the figures show mean ISE and the shaded region shows the standard deviation of the ISE for the responses. The solid gray vertical lines show the switching boundaries for the local approximations. Both sets of approximations are able to maintain low response error. However, the error from the approximations with only 1 local controller, shown in Figure 67a, has smoother trends than the ISE from five local approximations in Figure 67b. Increasing the number of local approximations tends to increase the magnitude of the peaks in ISE, where each peak is also associated



(a) 1 local approximation **(b)** 5 local approximations

Figure 67. ISE of crane trolley for local controllers



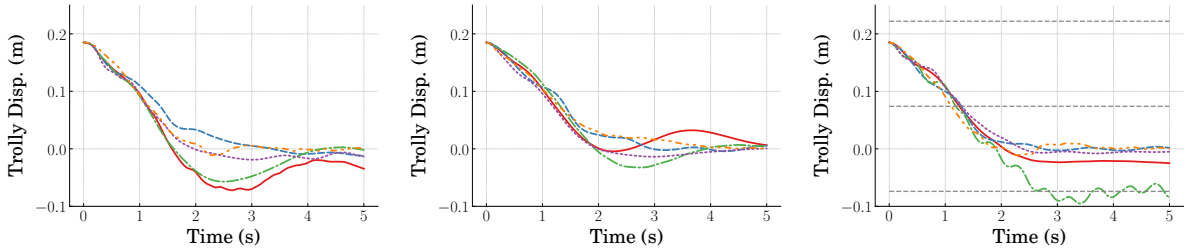
(a) Agent response **(b)** 1 local approximation **(c)** 5 local approximations

Figure 68. Pure RL curve fit responses for crane trolley for $x(0) = 0.185$m

with higher variance in ISE. Although generating local approximations from smaller subsets may provide less error between the approximation and the agent, switching sometimes creates instability in the responses.

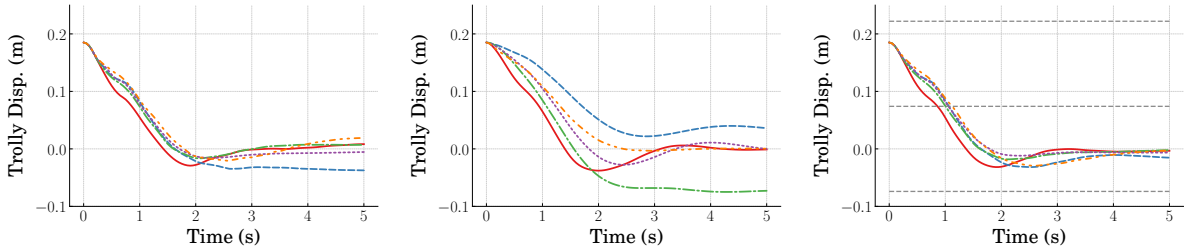
Time responses with the approximated switching controllers for Pure RL with an initial trolley displacement of 0.185m are shown in Figure 68. The horizontal gray dashed lines show the switching boundaries between the local approximations. The responses from the unmodified agent based Pure RL controller is shown in Figure 68a. The responses for the approximated local controllers were able to bring the trolley near the desired displacement, similar to the agent. The responses with five local approximations in Figure 68c have less error from their corresponding agent responses compared to the case with 1 local approximation in Figure 68b.

Figure 69 shows the time responses from the switching controllers for RL-PD.



(a) Agent response **(b)** Curve fit with 1 bin **(c)** Curve fit with 5 bins

Figure 69. RL-PD curve fit responses for crane trolley for $x(0) = 0.185$m

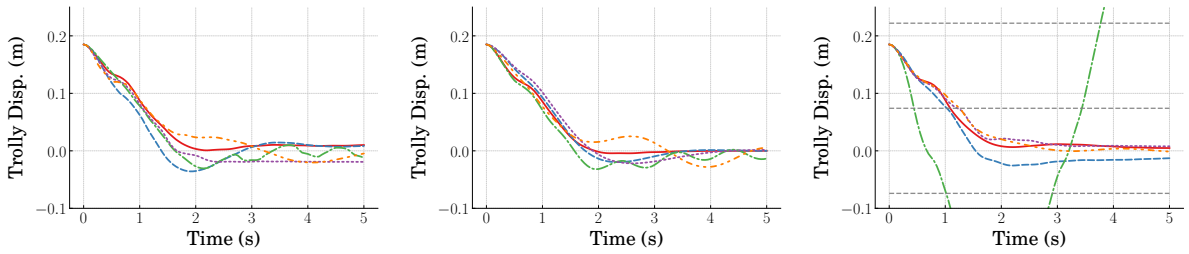


(a) Agent response **(b)** 1 local approximation **(c)** 5 local approximations

Figure 70. RL-LA curve fit responses for crane trolley for $x(0) = 0.185$m

For both the case with one local controller in Figure 69b and the case with five local controllers in Figure 69c, the switching controllers tend to converge close to the desired displacement at $x = 0$, similar to the agent controller. However, the response shown in green for the five local controller case does not satisfactorily approximate the corresponding agent response since it oscillates about the switching boundary indicated by the dashed gray horizontal line.

The RL-LA time responses for the switching controller approximations are shown in Figure 70. The agent responses in Figure 70a show that the trolley responses tend to converge near the desired displacement with only low steady-state error. The approximate controllers with one local controller in Figure 70b have three responses that settle to zero displacement. However, two of the responses have much higher steady-state error than the corresponding agent responses. The responses for the case with five local controllers in Figure 70c tend to converge to zero displacement with less steady-state error than both the agent responses and the one local approximation case.



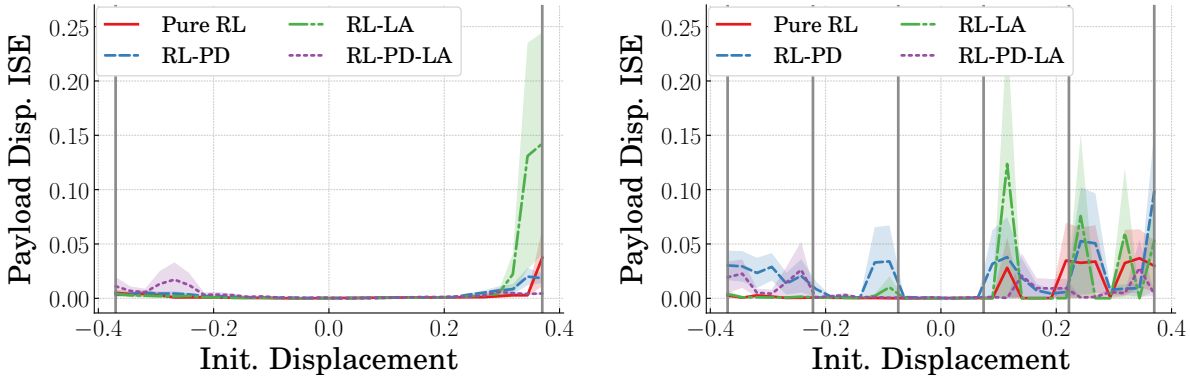
(a) Agent response **(b)** Curve fit with 1 bin **(c)** Curve fit with 5 bins

Figure 71. RL-PD-LA curve fit responses for crane trolley for $x(0) = 0.185$m

Figure 71 shows the responses of the switching controllers for RL-PD-LA. Similar to the responses for the other controller types for the crane, the switching controllers tend to satisfactorily approximate the responses of the agent controller by converging near the equilibrium. However, one of the responses for the five local approximation case in Figure 71c is unstable and diverges from the desired displacement.

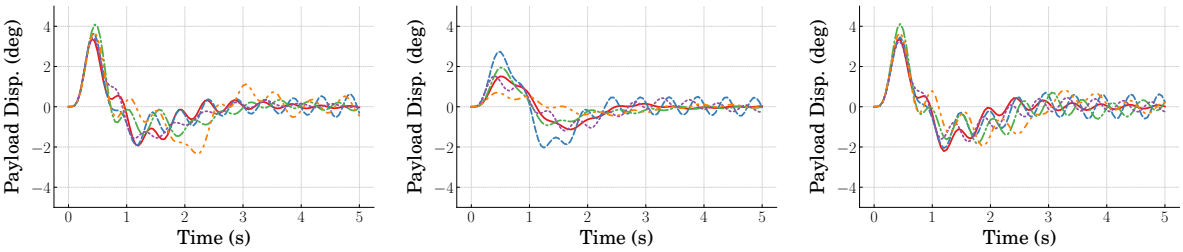
Figure 72 shows the mean ISE of the payload angular displacements for the switching controllers when the initial displacement of the trolley is $x(0) = 0.185$m. The ISE is plotted against the initial displacement of the trolley since that is the scheduling variable. Similar to the ISE of the trolley responses, the controllers using one local approximation of the agents in Figure 72a have smoother trends than the switching controllers using five local approximations in Figure 72b. The case with five local approximations has more peaks with higher variance. These peaks are explained by the time responses below.

The figures below show the payload time responses for trolley initial displacements of $x(0) = 0.185$m. These responses resulted from the same switching controllers shown previously. Figure 73 shows the payload time responses of the switching controllers for Pure RL. The crane started at rest with an initial trolley displacement of $x = 0.185$m and pendulum angular displacements of $\theta_1 = \theta_2 = 0$. The pendulum responses with the switching controllers were able to maintain low oscillation amplitude and closely approximate the agent responses. The responses with one local



(a) 1 local approximation **(b)** 5 local approximations

Figure 72. ISE of crane payload for local controllers



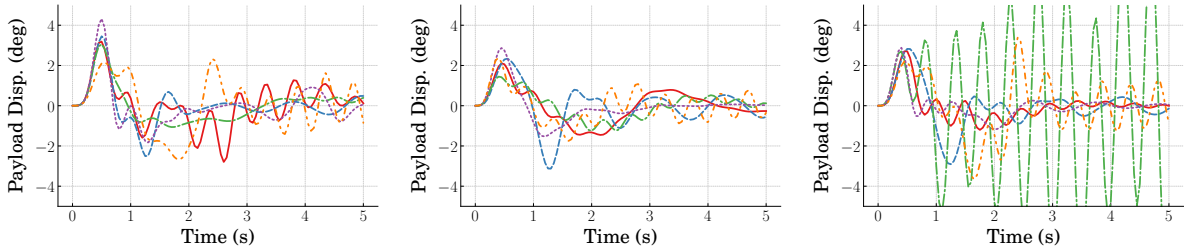
(a) Agent response **(b)** 1 local approximation **(c)** 5 local approximations

Figure 73. Pure RL curve fit responses for crane payload for $x(0) = 0.185$m

approximation in Figure 73b have a lower peak magnitude than the responses with five local approximations in Figure 73c. However, the higher peak amplitude with five local approximations more closely matches that of the agent response in Figure 73a.

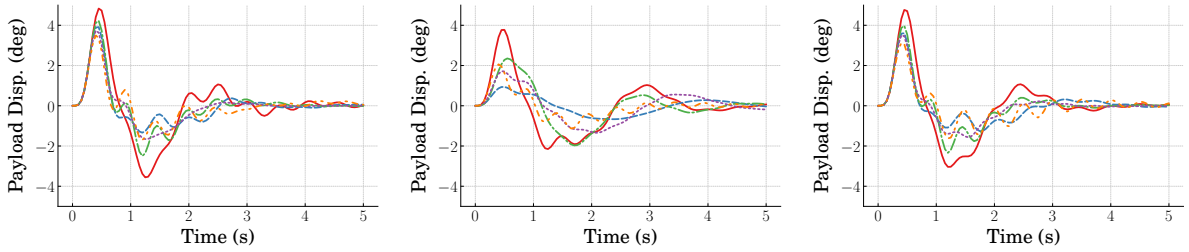
The time responses for the RL-PD switching controllers are in Figure 74. Most of the responses have low oscillation amplitude and reasonably match the agent responses in Figure 74a; however, the response shown in green for five local approximations in Figure 74c has significantly higher oscillation amplitude. This corresponds to the trolley response in Figure 69c that oscillated about the switching boundary.

The payload responses of the switched controllers for RL-LA are shown in Figure 75. The cases with one local approximation in Figure 75b and five local approximations in Figure 75c both satisfactorily match the agent responses by mitigating oscillation. However, the case with five local controllers more closely



(a) Agent response **(b)** 1 local approximation **(c)** 5 local approximation

Figure 74. RL-PD curve fit responses for crane payload for $x(0) = 0.185$m



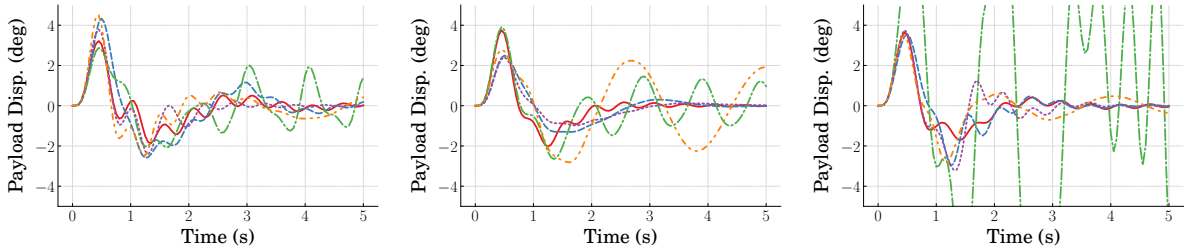
(a) Agent response **(b)** 1 local approximation **(c)** 5 local approximations

Figure 75. RL-LA curve fit responses for crane payload for $x(0) = 0.185$m

matches the agent responses.

Figure 76 shows the time responses of the switching controllers approximating the RL-PD-LA agents. The approximations tend to satisfactorily match the responses of the agents. However, the response shown in orange for the case with one local approximation in Figure 76b has higher oscillation amplitude than the corresponding agent responses. Also, one of the responses with five local approximations in Figure 76c has significantly more payload oscillation amplitude than the agent responses. The other responses in this case quickly mitigate the residual oscillation.

In general, the switching controller approximations for the double-pendulum crane tend to be able to accurately model the behavior of the agents. Using a larger number of local approximations in the switching controller tended to improve the modeling accuracy between the approximations and the agents. However, the increase in the number of local approximations also increased the probability for some switching controller responses to be unstable. This instability is common in switching controllers,



(a) Agent response **(b)** 1 local approximation **(c)** 5 local approximations

Figure 76. RL-PD-LA curve fit responses for crane payload for $x(0) = 0.185$m

often caused by discontinuity at the boundary between the local controllers [58, 59]. This was particularly common for the approximations of controllers that had an agent-driven gain scheduling component. Despite this, the approximations can be used to improve interpretability of the agents by aiding in the development of intuitive understanding of the agent.

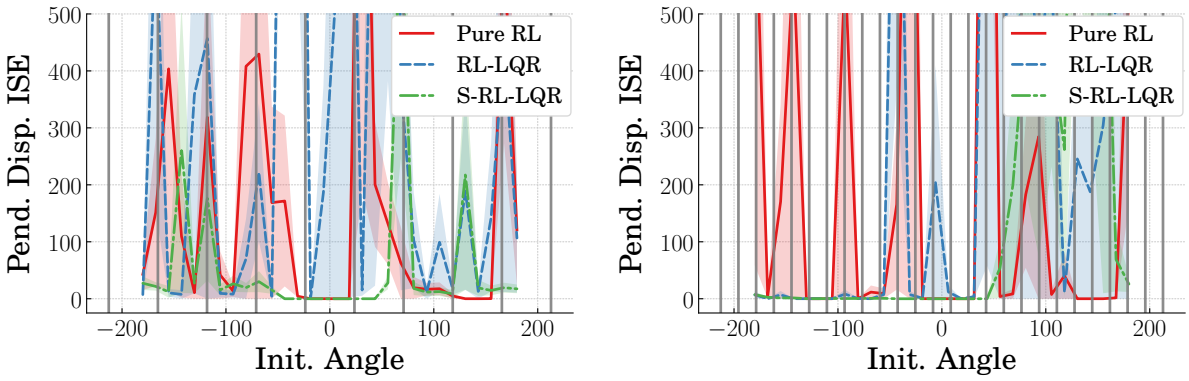
5.4.2 *Inverted Pendulum Agent Approximation*

The same process used to curve fit switching controllers for the double-pendulum crane was repeated for the inverted pendulum controllers:

$$\begin{aligned}
&\text{Pure RL:} && u = a_t \\
&\text{RL-LQR:} && u = -k_1\theta - k_2\dot{\theta} + a_t \\
&\text{S-RL-LQR:} && u = \begin{cases} -k_1\theta - k_2\dot{\theta}, & \forall(\theta,\dot{\theta}) \in S_{st} \\ a_t, & \forall(\theta,\dot{\theta}) \notin S_{st} \end{cases}
\end{aligned}$$

where Pure RL is the controller that only uses the agent action as the output, RL-LQR uses LQR as a fixed-gain component as well as a single lumped action, and S-RL-LQR switches between the agent and LQR when the system is near the upright equilibrium. Polynomials were used to generate local approximations as before, where each controller has the form:

$$\hat{a}_l = \sum_{i=1}^{p} a_i x^i + b_i \dot{x}^i + c_i \theta^i + d_i \dot{\theta}^i \tag{56}$$



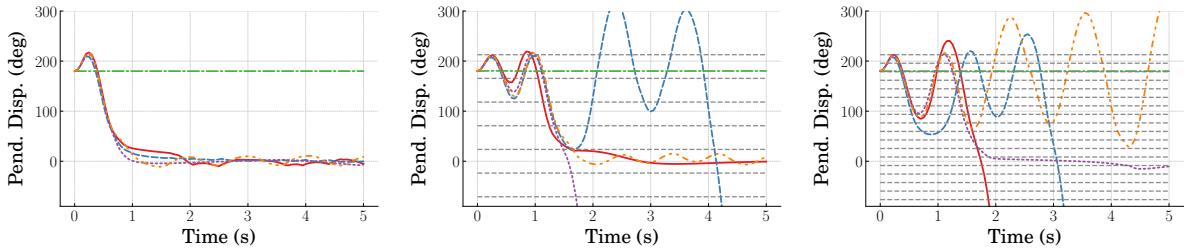
(a) 9 local approximations **(b)** 25 local approximations

Figure 77. ISE of inverted pendulum local controllers for $\theta(0) = 180°$

where x and $\dot{x}$ are the displacement and velocity of the cart, θ and $\dot{\theta}$ are angular displacement and angular velocity, and $p = 3$ is the degree of the polynomial fit. Angular displacement was used as the scheduling variable.

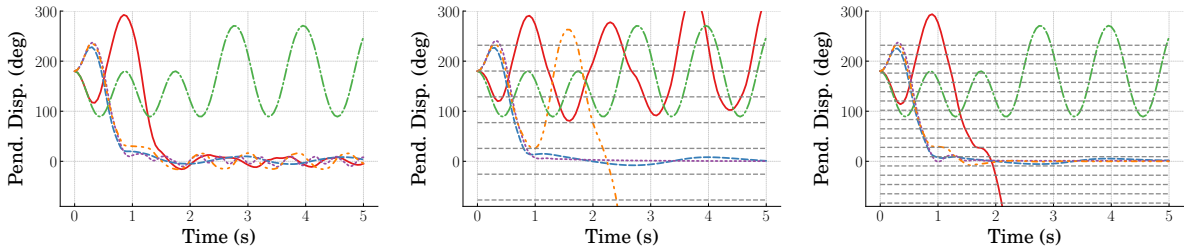
5.4.2.1 Approximation for Swing-Up Phase. Figure 77 shows the mean ISE of angular displacement between the responses of each agent for each controller type and the corresponding switching controller response. The shaded region shows the standard deviation for each controller type. The ISE was evaluated for a range of initial angular displacements from $\theta = -180°$ to $\theta = 180°$. Figure 77a shows the mean ISE when the agent is divided into nine local controllers, and Figure 77b shows the mean ISE for twenty-five local controllers. The boundaries for the local controllers are indicated by the solid vertical gray lines. Although there are some areas where the mean ISE is low for both the switching controllers with nine local approximations and 25 local approximations, there are many areas with high mean and high standard deviation. The widest ranges for which ISE is low for Pure RL and S-RL-LQR are found for initial angular displacements near $\theta = 0$. The response error for S-RL-LQR is zero for initial angular displacements $-45° < \theta < 45°$ due to the agent output being unused in that range.

Figure 78 shows time responses for the switching controllers of Pure RL from an



(a) Agent response **(b)** Curve fit with 9 bins **(c)** Curve fit with 25 bins

Figure 78. Pure RL curve fit responses for $\theta(0) = 180°$

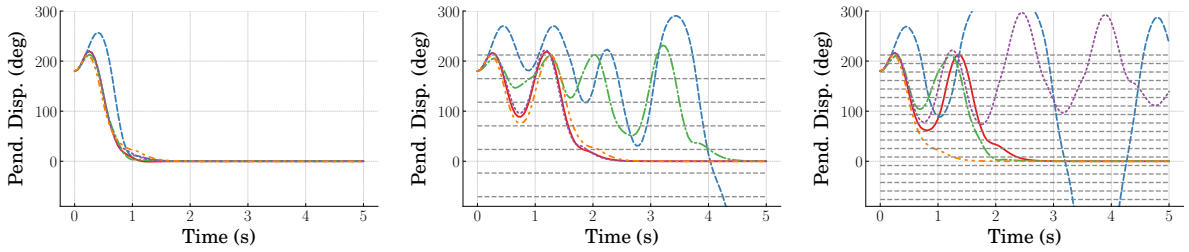


(a) Agent response **(b)** Curve fit with 9 bins **(c)** Curve fit with 25 bins

Figure 79. RL-LQR curve fit responses for $\theta(0) = 180°$

initial displacement of $\theta = 180°$. Figure 78a shows the time responses from the neural network controller for reference. The Pure RL switching controller using nine local controllers is shown in Figure 78b. Two of the switching controllers successfully stabilize the system while two other controllers diverge from the equilibrium, contributing to the high ISE. The response shown in blue moves beyond the upper switching boundary for which the local approximations were generated. The switching controller continues using the same local approximation beyond this boundary. Figure 78c shows the performance of the switching controller with nine local controllers. One of the controllers stabilizes the inverted pendulum while the others are unstable.

Figure 79 shows the time responses from the switching controllers of RL-LQR from an initial displacement of $\theta = 180°$. The controllers with nine local approximations in Figure 79b have two responses that are stable while the others are unstable. However, none of the responses diverge far from the equilibrium as they do for the nine local controller case for Pure RL. This corresponds to the lower ISE of



(a) Agent response **(b)** Curve fit with 9 bins **(c)** Curve fit with 25 bins

Figure 80. S-RL-LQR curve fit responses for $\theta(0) = 180°$

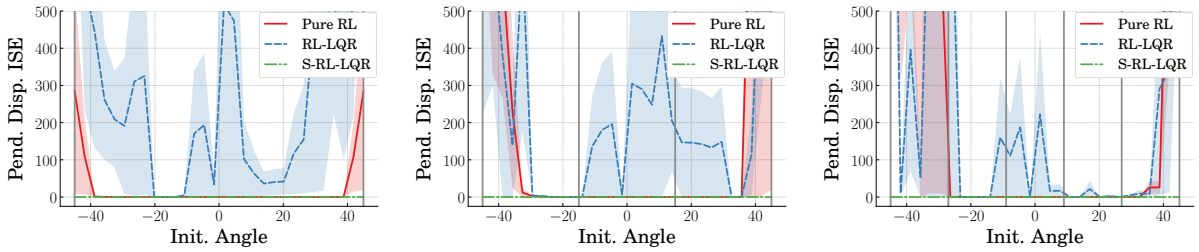
RL-LQR at $\theta = 180°$ in Figure 77a. The responses for the switching controller with twenty-five local approximations of RL-LQR are shown in Figure 79c. There are three stabilizing controllers and two unstable responses. Even though there are more stable responses for the RL-LQR case with twenty-five local controllers than for the case with nine local controllers, the ISE for twenty-five local controllers is higher due to one of the responses diverging far from the equilibrium. With the exception of the response that diverges from the equilibrium, the other responses for the switching controller with twenty-five local approximations more closely match the responses from the RL-LQR agent controller compared to the switching controller approximations for Pure RL.

Figure 80 shows the time responses with the approximations of S-RL-LQR. The approximate controllers use the same switching criteria that was established for switching between the agent action and LQR in the neural network controller. For the switching controller with nine local approximations in Figure 80b, there are four stable responses and one unstable response. Similarly, there are four stable responses for the case with twenty-five local approximations in Figure 80c. Compared to the approximations for Pure RL, the approximations for the S-RL-LQR case were more capable to stabilize the inverted pendulum within the time frame. However, the stabilized responses have higher rise times than the corresponding agent responses.

In summary, the performance of the switching controllers for RL-LQR most closely matched the corresponding agent controllers from $\theta(0) = 180°$ despite the high

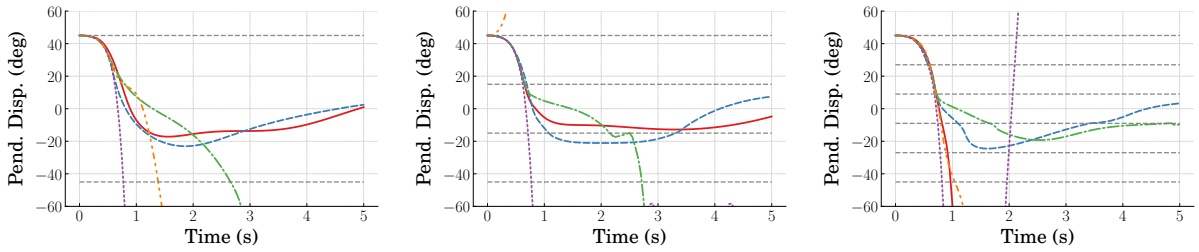
ISE. The switching controller approximations for S-RL-LQR were less accurate than RL-LQR, but were more accurate than Pure RL. While a few of the responses with the switching controllers were adequately accurate to match the agent responses, many were not adequately accurate. This inconsistency caused the high standard deviation and spikes in the mean ISE seen in Figure 77. Additionally, some switching controllers that were stabilizing for one set of local approximations were not stabilizing for corresponding switching controllers with a different number of local approximations. The inconsistency of fit accuracy among different initial conditions and different sizes of the switching controllers indicates the fit is very sensitive to these factors. There may also be better candidates for local approximation functions to use for curve fits.

5.4.2.2 Approximation Near the Equilibrium. The widest regions for which Mean ISE is low for the inverted pendulum switching controllers are for initial conditions near the equilibrium, $\theta = 0$. It is therefore beneficial to split the analysis of the curve fits into categories of near equilibrium and not near equilibrium. Figure 81 shows the mean ISE for $-45^\circ \leq \theta \leq 45^\circ$. There is no error between the approximations and the agent controllers for S-RL-LQR since this is the region for which the S-RL-LQR controller uses only LQR. Figure 81a shows the ISE when only one approximated controller is used for each agent. The Pure RL approximations have low error for most of the range except near the extremes. The case with only one approximate control law for the entire range has the widest region of low mean ISE, with the increase in the number of approximations used reducing the range for which the ISE is low. As was common with the other approximations, the switching controllers for RL-LQR have high mean error and variance in this region and do not show a clear trend in performance of the approximations. However, in contrast to the Pure RL approximations, the mean ISE for the RL-LQR approximations is lowest for the case with five local approximations and is higher for decreases in the number of



(a) 1 approximation **(b)** 3 approximations **(c)** 5 approximations

Figure 81. ISE of inverted pendulum local controllers for $\theta(0) = 45°$



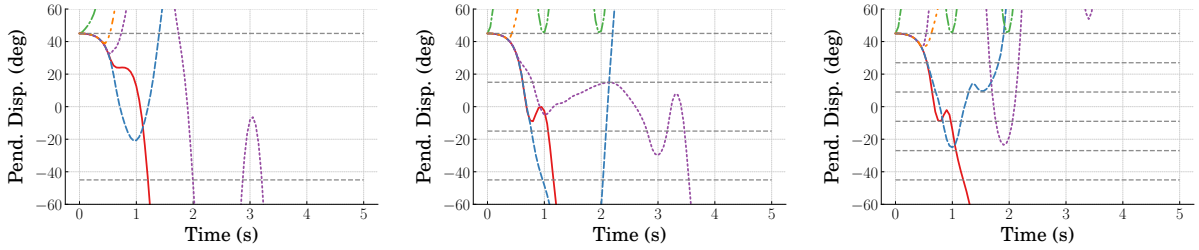
(a) 1 approximation **(b)** 3 approximations **(c)** 5 approximations

Figure 82. Switching controller responses for Pure RL for $\theta(0) = 45°$

local controller approximations.

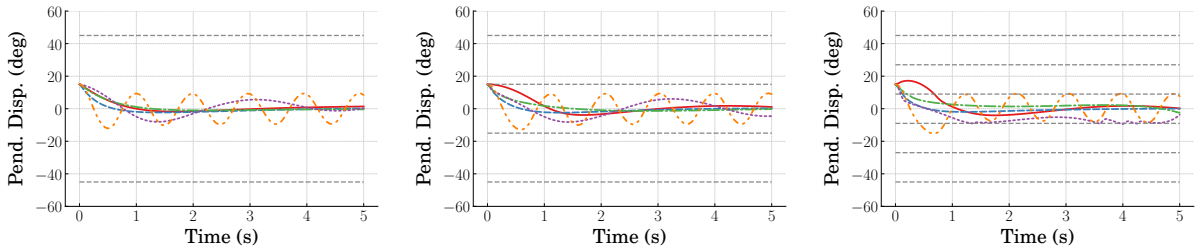
Time responses of the switching controllers for Pure RL are shown in Figure 82. These responses start from rest with initial angular displacement of $\theta = 45°$. For the case with only one local approximation in Figure 82a, only two responses remain near the equilibrium. For the switching controller with three local approximations in Figure 82b, only two responses remain near the equilibrium, similar to the case with one local approximation. For the case with five local approximations in Figure 82c, two of the responses remain near the equilibrium. However, the response in red that previously remained near the equilibrium for the other cases diverges for this case. This shows that the conditions necessary to generate good approximations are not identical for the different agents. Figure 83 shows the time responses with initial displacements of $\theta(0) = 45°$ for the switching controller approximation for RL-LQR. For all cases shown, none of the approximated controllers are stabilizing.

Based on the plots for mean ISE near equilibrium, the response error tends to be



(a) 1 approximation **(b)** 3 approximations **(c)** 5 approximations

Figure 83. Switching controller response for RL-LQR for $\theta(0) = 45°$



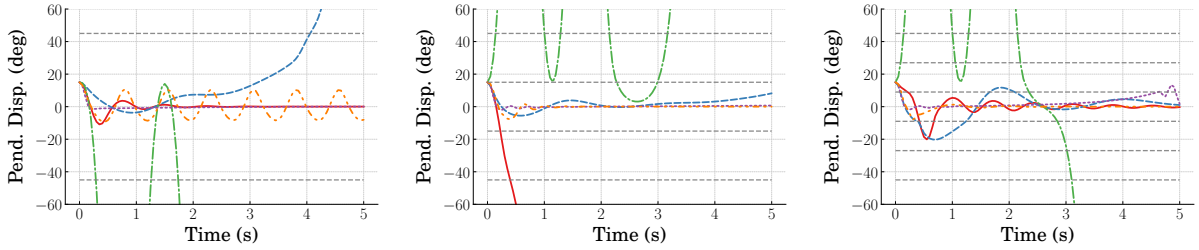
(a) 1 approximation **(b)** 3 approximations **(c)** 5 approximations

Figure 84. Switching controller response for Pure RL for $\theta(0) = 15°$

lower for initial angular displacements closer to $\theta = 0$. Figure 84 shows the time responses from Pure RL curve fits for an initial condition of $\theta = 15°$. All responses from this initial condition stay near the equilibrium as was expected from the Mean ISE in Figure 81, and there is more consistency in the behaviors of the approximations with different numbers of local controllers.

The responses of the switching controllers for RL-LQR from an initial angular displacement of $\theta = 15°$ are shown in Figure 85. Two of the responses diverge from the equilibrium in Figure 85a. However, two settle at the equilibrium and another oscillates about the equilibrium. In Figure 85b, three of the responses remain near the equilibrium in the time range shown. However, two of the responses are unstable and the response in blue begins to diverge away from the equilibrium. In Figure 85c, only four of the five responses remain near the equilibrium during the time range shown.

Although the approximate controllers for the crane were generally able to provide accurate responses that closely matched the behavior of the agents, the



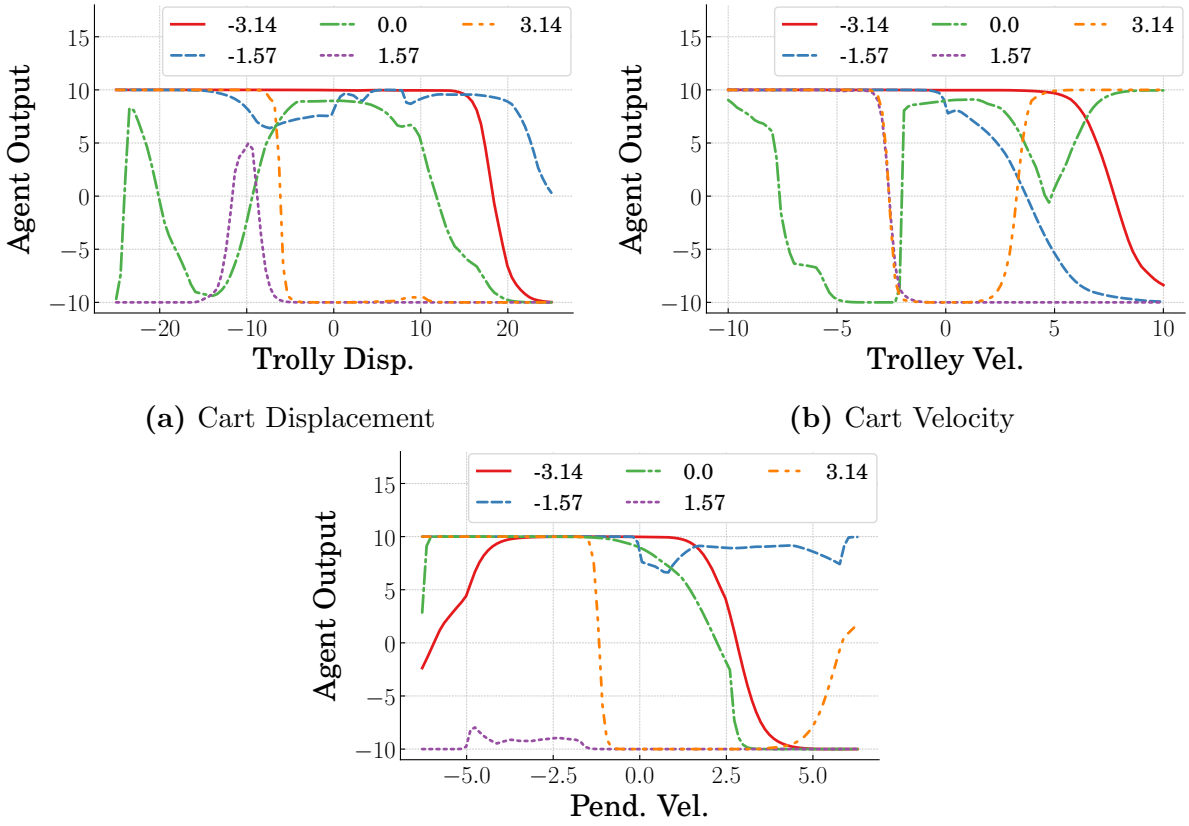
(a) 1 approximation **(b)** 3 approximations **(c)** 5 approximations

Figure 85. Switching controller response for RL-LQR for $\theta(0) = 15°$

approximate controllers for the inverted pendulum tended to have less accuracy. In addition to this, behavior was often inconsistent, where approximations were stable for one switching controller and unstable for another switching controller with a different number of local approximations. This suggests that the polynomial fit function was a poor choice to approximate the agents for the inverted pendulum controllers.

5.5 Agent Output Contributions

The switching controllers in this chapter were all generated by modeling the contributions from all states using the same fit function. Using a generalized fit function to approximate the contribution from the switching variable is appropriate since only a small range of the variable is used for each local approximation. However, there is an indefinite range for the other states within each subset of the operating space. Generic fit functions may not always be appropriate for potentially wide ranges of states, especially as the controller switches between different local approximations. The unsuitability of generic fit functions can be illustrated in Figure 86, which plots state contributions to the output of an agent for the inverted pendulum. The range of the state in question is shown on the horizontal axis. Each line shows the agent output for a fixed value of the switching variable with all other states equal to zero. Although a third-order polynomial was used to approximate all state contributions, this generic function would be unable to accurately approximate all of the agent outputs. This suggests that some initial interpretation of the behavior of the agent is necessary in



(a) Cart Displacement

(b) Cart Velocity

(c) Pendulum Velocity

Figure 86. Inverted pendulum agent contributions for different states

order to manually generate appropriate curvefit functions for the agents.

5.6 Conclusion

This chapter presented a method to generate interpretable models of black-box agents by curve fitting the agent output to local controller approximations. These local approximations were then combined into switching controllers that approximate the behavior of the agent in the entire operating space. The switching controllers for the double-pendulum crane produced results that tended to accurately approximate the behavior of the agents, suggesting that those models can be used to help interpret the neural network policies. However, the switching controllers for the inverted pendulum tended to poorly approximate the agent controllers. This is most likely due to third-degree polynomials being poor candidates for the local approximations. This

suggests that some agents may require manual interpretation to generate appropriate fit functions before using numerical curve fitting.

VI Commercialization

This work presents the initial development phases of methods using domain knowledge for model-based control and RL and the analysis of learned controllers using established control methods. These methods have the potential to provide significant economic impact to various markets that utilize robotics or automation. For instance, the use of automation in industrial robotics has been shown to increase productivity while lowering output prices [60]. The proposed methods can further this trend since they can improve performance of existing systems with existing controllers. These methods also have the potential to accelerate the control design for new systems and applications. This chapter describes some challenges and future work necessary to bring the process from this initial stage of development to a marketable process.

6.1 Envisioned Market

The methods proposed in this work to combine domain knowledge and RL can be introduced to a wide range of markets. Applications include any system where there is existing domain knowledge that enables model-based controllers to be developed, but where the controllers are not optimal. This may include systems that are difficult to model or applications where operating conditions are changing, in which case adaptive controllers might normally be employed. This could for instance have applications in the field robotics market, where robotic systems must be able to interact with a dynamic and unstructured environment. RL can be employed as an adaptive mechanism for various subsystems such as for manipulators interacting with payloads of unknown mass and awkward shapes or legged locomotion adapting to rough terrain [61]. This work may also be used for applications in which controllers are necessary to account for processes that are difficult to model. For instance in the UAV market, controllers are needed to reject disturbances from wind. In these cases, a conventional model-based controller can be used to control the UAV under nominal conditions while an RL

Table 14. Technology Readiness Levels

TRL 1	TRL 2	TRL 3	TRL 4	TRL 5
Basic Research	Technology Formulation	Proof of Concept	Laboratory Testing	Field Testing

TRL 6	TRL 7	TRL 8	TRL 9
Viability Demo	Commercial Transition	Technology Demo	Commercial Deployment

component can augment the model-based controller in the presence of disturbances [62].

6.2 Market Viability

The market readiness for this work can be quantified by Technology Readiness Level (TRL) [63, 64]. TRL was initially developed by NASA in the 1970s to describe the level of development of different systems used for space exploration. Systems had to go through rigorous testing before being deemed mission ready. The stage of development of the system can fall on a scale from 1 to 9 and characterizes systems from initial basic research principle to deployable product, as illustrated in Table 14. The TRL method has since been adopted and modified by many other entities including the US Department of Defense and the ISO. Using this method, the TRL of the entire body of work in this dissertation is evaluated at approximately TRL 2 or TRL 3. Because of this, this work is not ready to be introduced to market. However, some parts of this work may be given a higher TRL evaluation if considered in isolation. For instance, although the methods proposed to interpret the learned controllers have low TRL, the combined control methods may be categorized with a higher TRL if it was intended as an augmentation of existing products. Since the

existing product and its controllers will have already gone through rigorous development and testing before being deployed, the addition of RL to the existing controller will require less development than a completely new controller.

6.3 Increasing Market Viability

More work is necessary to develop the concepts and methods proposed in this dissertation to increase its TRL and make it ready for market. First, this dissertation proposed multiple controller architectures to combine domain-knowledge controllers with RL and established some initial design guidelines. These different controller architectures will need to undergo more testing on various systems so that the combined controller design guidelines can be formalized. In addition to general controller design, more work is needed to develop the foundational principles describing the effect of controller design and reward function design on stability. This additional development of the methods proposed in this dissertation will make the process easier for the customer to develop reliable and safe controllers.

In addition to further research needed to increase TRL, other commercial considerations must be made. For instance, the product or service must be made user-friendly or easy to learn in order to maximize sales and profit. Also, in order for a product to be profitable, it must be divided into individual units of sale. This is challenging since the design of the combined controller using the proposed methods will need to be customized for the specific application. Due to this, the proposed methods can be sold as a software package for rapid design and verification of multiple combined controller architectures.

6.4 Market Challenges

Although RL has already been introduced in many applications including AI-assisted design and optimization of various scheduling, allocation, and logistics problems [65, 66], many potential customers may still be hesitant to adopt RL for

applications in which it is used as an adaptive controller. The performance of fixed-gain controllers can be verified for a variety of situations prior to deployment, and conventional adaptive controllers are designed to adapt in limited and predictable ways. The use of RL and neural networks for adaptive controllers allows for a high degree of customizability to the baseline controller. However, these changes cannot be verified before deployment if these customizations are made online during operation of the system. This may prevent the adoption of the proposed work in safety critical or high risk applications.

6.5 Impact to the Engineering Community

RL alone has already been used to solve optimal control problems that cannot be solved analytically. The use of RL in this way has a very simple implementation since no domain knowledge or manual controller design is required. Although using combined controllers introduces some design complexity compared to RL alone, benefits are gained such as reduced training time and safety of the policies. Additionally, with combined controllers, model-based components can provide a foundation based on domain knowledge while the RL components account for parts of the system for which conventional optimal control design is not possible. This has the potential to reduce mechanical design constraints that are imposed by limits in conventional controller design practices such that the design of new systems may be more innovative.

6.6 Economic Evaluation of Process

The economic impact of the proposed methods can be estimated by analyzing the current growth of the global robotics market. The global robotics market in 2023 is estimated to be between approximately $82 billion and $114 billion with a compound annual growth rate (CAGR) between 14.7% and 17.64% [67, 68]. The largest subset of the global robotics market is industrial robots. Collaborative robots, or cobots, are also a rapidly growing segment of the industrial robotics market. Cobots are required to

assist humans with various tasks which will require them to quickly adapt to new situations or different humans. The methods proposed in this work can improve the efficiency of this collaboration, further increasing productivity and economic impact. In addition to industrial robots, service robots are a subset of the robotics market that is expected to see significant growth in the coming years [67]. Service robots are often utilized for inspection and maintenance, professional cleaning, and domestic tasks. Robots used for these applications will need adaptive control that can continuously optimize their behavior in unstructured and dynamic environments. The growth of this market is an indication that there is a demand for further innovation in learning controllers such that this work will have economic impact beyond the current projections.

6.7 Potential Competitors

With the demand to increasingly automate difficult tasks, there is a need for controllers that can provide optimal solutions for complex systems in dynamic and unstructured environments. Data-driven control methods such as RL are good candidates to provide the needed controllers for these applications. Due to this, there are several existing companies utilizing RL and ML to provide solutions for automation problems. These companies are all working to develop reliable learning controllers, making them potential competitors to the methods proposed in this dissertation.

VII Conclusion

7.1 Dissertation Contribution

This work proposed and analyzed methods to design controllers that combine conventional control methods with Reinforcement Learning (RL). Although RL can be used to learn agents without needing domain knowledge of dynamics and control theory, RL often requires long training times, has poor initial performance, and lacks performance guarantees. Conventional model-based control does not suffer from the same problems as RL, but most control problems are not amenable to analytical solutions. Combining RL with conventional control can take advantage of the strengths of both methods while limiting the drawbacks.

Several combined controller architectures were proposed and evaluated for three benchmark systems. The combined controller architectures tested included those that contain model-based fixed-gain components as well as those with agent-driven gain scheduling. The combined controllers often had better initial performance compared to controllers using RL alone and reduced the required training time to achieve desirable performance from the controllers. Although not every combined controller outperformed the RL-alone controllers after training, there was always at least one combined controller with superior performance. This indicates that the combined controller architecture that results in the best performance depends on the dynamics of the system. The combined controllers were also evaluated for performance guarantees in stability and robustness. A method to improve interpretability of the policies by approximating the agents with piecewise switching controllers was also proposed. The contributions made by this work are:

- **Reduced Training Time** – It was shown in Chapter 2 that combined controllers that were appropriately designed for the system improved initial performance and reduced required training time compared to controllers using RL alone. This alleviates the main challenge preventing widespread adoption of RL

for robotics beyond research settings.

- **Controller Design Guidelines** – The baseline results from Chapter 2 were used to generate guidelines for designing combined controllers. These guidelines facilitate the process for designing combined controllers and reduce barriers to adopting RL.

- **Improved Stability** – Chapter 3 presented stability evaluations for the controllers showing that appropriately designed controllers tended to have improved BIBO and Lyapunov stability compared to the RL-alone controllers.

- **Analysis of Robustness** – Chapter 4 presented an analysis of the combined controllers for robustness to modeling error. These evaluations showed that combined controllers and RL-alone controllers tended to have comparable insensitivity to modeling error.

- **Agent Interpretation Method** – A method was proposed in Chapter 5 to improve interpretability of agents by approximating them as piecewise switching controllers. This provides more concise representations of the agent that facilitate intuitively understanding its behavior.

7.2 Future Work

Combining conventional model-based control with RL can provide better initial performance and require shorter training time than when using RL alone. This enables future work to investigate the use of combined controllers for sim-to-real. Ordinarily, an agent is trained in simulation before being transferred to a physical system. Combined controllers may be able to accelerate this process since less training is required. Additionally, the initial policy with the combined controller may be safe to enable training on the physical system without pretraining in simulation.

In this work, training was done in a fully-known and static environment. Future work may investigate the design of combined controllers in unknown or dynamic environments. Combined controllers could be designed to be robust to the changes in the environment, or they could be applied as adaptive controllers where the RL component trains continuously. Appropriate design of the combined controller would depend on the system dynamics and the expected variability in the environment.

The method to improve the interpretability of the agents by modeling them as switching controllers resulted in accurate approximations of the double pendulum controllers. However, the switching controller approximations for the inverted pendulum had low accuracy when using generic curve fit functions. Future work may investigate methods to manually interpret the agent in order to generate models that accurately approximate the contributions of the states to the agent output. These models can then be combined to provide a more comprehensive interpretation of the behavior of the agent.

Bibliography

[1] Dogan, A. and Venkataramanan, S., "Nonlinear control for reconfiguration of unmanned-aerial-vehicle formation," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 4, pp. 667–678, 2005.

[2] Khan, M. A., Abid, M., Ahmed, N., Wadood, A., and Park, H., "Nonlinear control design of a half-car model using feedback linearization and an lqr controller," *Applied Sciences*, vol. 10, no. 9, 2020.

[3] Sutton, R. S. and Barto, A. G., *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 2018.

[4] Li, Y., "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[5] Cheng, R., Orosz, G., Murray, R. M., and Burdick, J. W., "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3387–3395, Jul. 2019.

[6] Hart, P., Rychly, L., and Knoll, A., "Lane-merging using policy-based reinforcement learning and post-optimization," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 3176–3181, 2019.

[7] Tambon, F., Laberge, G., An, L., Nikanjam, A., Mindom, P. S. N., Pequignot, Y., Khomh, F., Antoniol, G., Merlo, E., and Laviolette, F., "How to certify machine learning based safety-critical systems? a systematic literature review," *Automated Software Engineering*, vol. 29, no. 2, p. 38, 2022.

[8] song WANG, X., hu CHENG, Y., and SUN, W., "A proposal of adaptive pid controller based on reinforcement learning," *Journal of China University of Mining and Technology*, vol. 17, no. 1, pp. 40–44, 2007.

[9] Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Ojea, J. A., Solowjow, E., and Levine, S., "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029, 2019.

[10] Silver, T., Allen, K., Tenenbaum, J., and Kaelbling, L., "Residual policy learning," *arXiv preprint arXiv:1812.06298*, 2018.

[11] Lewis, F. L., Vrabie, D., and Syrmos, V. L., *Optimal control*. John Wiley & Sons, 2012.

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

[13] Beard, R. W., Saridis, G. N., and Wen, J. T., "Galerkin approximations of the generalized hamilton-jacobi-bellman equation," *Automatica*, vol. 33, no. 12, pp. 2159–2177, 1997.

[14] Beard, R. W., Saridis, G. N., and Wen, J. T., "Approximate solutions to the time-invariant hamilton–jacobi–bellman equation," *Journal of Optimization Theory and Applications*, vol. 96, no. 3, pp. 589–626, 1998.

[15] Boulbrachene, M. and Haiour, M., "The finite element approximation of hamilton-jacobi-bellman equations," *Computers & Mathematics with Applications*, vol. 41, no. 7, pp. 993–1007, 2001.

[16] Saberi Nik, H., Effati, S., and Shirazian, M., "An approximate-analytical solution for the hamilton–jacobi–bellman equation via homotopy perturbation method," *Applied Mathematical Modelling*, vol. 36, no. 11, pp. 5614–5623, 2012.

[17] Schwenzer, M., Ay, M., Bergs, T., and Abel, D., "Review on model predictive control: an engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, pp. 1327–1349, 2021.

[18] Chen, J., Zhan, W., and Tomizuka, M., "Autonomous driving motion planning with constrained iterative lqr," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 2, pp. 244–254, 2019.

[19] Watkins, C. J. C. H., *Learning from delayed rewards*. PhD thesis, 1989.

[20] Watkins, C. J. C. H. and Dayan, P., "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[21] Clifton, J. and Laber, E., "Q-learning: Theory and applications," *Annual Review of Statistics and Its Application*, vol. 7, pp. 279–301, 2020.

[22] Jang, B., Kim, M., Harerimana, G., and Kim, J. W., "Q-learning algorithms: A comprehensive classification and applications," *IEEE Access*, vol. 7, pp. 133653–133667, 2019.

[23] Baird, L., "Residual algorithms: Reinforcement learning with function approximation," in *Machine Learning Proceedings 1995* (Prieditis, A. and Russell, S., eds.), pp. 30–37, San Francisco (CA): Morgan Kaufmann, 1995.

[24] Xu, X., Zuo, L., and Huang, Z., "Reinforcement learning algorithms with function approximation: Recent advances and applications," *Information Sciences*, vol. 261, pp. 1–31, 2014.

[25] Alloghani, M., Al-Jumeily, D., Mustafina, J., Hussain, A., and Aljaaf, A. J., *A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science*, pp. 3–21. Cham: Springer International Publishing, 2020.

[26] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[27] Shao, K., Tang, Z., Zhu, Y., Li, N., and Zhao, D., "A survey of deep reinforcement learning in video games," *arXiv preprint arXiv:1912.10944*, 2019.

[28] Bengio, Y., "Learning deep architectures for ai," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[29] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M., "Playing Atari with Deep Reinforcement Learning," *arXiv e-prints*, p. arXiv:1312.5602, Dec. 2013.

[30] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations 2016*, 2016.

[31] Fujimoto, S., van Hoof, H., and Meger, D., "Addressing Function Approximation Error in Actor-Critic Methods," *arXiv e-prints*, p. arXiv:1802.09477, Feb. 2018.

[32] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M., "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning* (Xing, E. P. and Jebara, T., eds.), vol. 32 of *Proceedings of Machine Learning Research*, (Bejing, China), pp. 387–395, PMLR, 22–24 Jun 2014.

[33] Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P., "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3803–3810, IEEE, 2018.

[34] Zhao, W., Queralta, J. P., and Westerlund, T., "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 737–744, 2020.

[35] Perkins, T. J. and Barto, A. G., "Lyapunov design for safe reinforcement learning," *Journal of Machine Learning Research*, vol. 3, no. Dec, pp. 803–832, 2002.

[36] Berkenkamp, F., Turchetta, M., Schoellig, A., and Krause, A., "Safe model-based reinforcement learning with stability guarantees," *Advances in neural information processing systems*, vol. 30, 2017.

[37] Alharin, A., Doan, T.-N., and Sartipi, M., "Reinforcement learning interpretation methods: A survey," *IEEE Access*, vol. 8, pp. 171058–171077, 2020.

[38] Amir, O., Doshi-Velez, F., and Sarne, D., "Summarizing agent strategies," *Autonomous Agents and Multi-Agent Systems*, vol. 33, pp. 628–644, 2019.

[39] Lage, I., Lifschitz, D., Doshi-Velez, F., and Amir, O., "Exploring computational user models for agent policy summarization," in *IJCAI: proceedings of the conference*, vol. 28, p. 1401, NIH Public Access, 2019.

[40] Bastani, O., Pu, Y., and Solar-Lezama, A., "Verifiable reinforcement learning via policy extraction," *Advances in neural information processing systems*, vol. 31, 2018.

[41] Eaglin, G. and Vaughan, J., "Leveraging conventional control to improve performance of systems using reinforcement learning," in *Proceedings of the ASME 2020 Dynamic Systems and Control Conference*, vol. Volume 2, Oct 2020.

[42] Eaglin, G., Poche, T., and Vaughan, J., "Controlling a double-pendulum crane by combining reinforcement learning and conventional control," in *2023 American Control Conference (ACC)*, pp. 788–793, IEEE, 2023.

[43] Åström, K. J. and Furuta, K., "Swinging up a pendulum by energy control," *Automatica*, vol. 36, no. 2, pp. 287–295, 2000.

[44] Park, M.-S. and Chwa, D., "Swing-up and stabilization control of inverted-pendulum systems via coupled sliding-mode control method," *IEEE transactions on industrial electronics*, vol. 56, no. 9, pp. 3541–3555, 2009.

[45] Riachy, S., Orlov, Y., Floquet, T., Santiesteban, R., and Richard, J.-P., "Second-order sliding mode control of underactuated mechanical systems i: Local stabilization with application to an inverted pendulum," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 18, no. 4-5, pp. 529–543, 2008.

[46] Ibanez, C. A., Frias, O. G., and Castanon, M. S., "Lyapunov-based controller for the inverted pendulum cart system," *Nonlinear Dynamics*, vol. 40, pp. 367–374, 2005.

[47] Muskinja, N. and Tovornik, B., "Swinging up and stabilization of a real inverted pendulum," *IEEE Transactions on Industrial Electronics*, vol. 53, no. 2, pp. 631–639, 2006.

[48] Dong, Y., Zhang, S., Liu, X., Zhang, Y., and Shen, T., "Variance aware reward smoothing for deep reinforcement learning," *Neurocomputing*, vol. 458, pp. 327–335, 2021.

[49] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P., "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

[50] Al-Nima, R. R. O., Han, T., Al-Sumaidaee, S. A. M., Chen, T., and Woo, W. L., "Robustness and performance of deep reinforcement learning," *Applied Soft Computing*, vol. 105, p. 107295, 2021.

[51] Blevins, R., *Formulas for Natural Frequency and Mode Shape*. Van Nostrand Reinhold Co, 1979.

[52] Lu, W., Magg, S., Zhao, X., Gromniak, M., and Wermter, S., "A closer look at reward decomposition for high-level robotic explanations," *arXiv preprint arXiv:2304.12958*, 2023.

[53] Hyde, R. and Glover, K., "The application of scheduled $\mathcal{H}_\infty$ controllers to a vstol aircraft," *IEEE Transactions on Automatic Control*, vol. 38, no. 7, pp. 1021–1039, 1993.

[54] Leith, D. J. and Leithead, W. E., "Survey of gain-scheduling analysis and design," *International Journal of Control*, vol. 73, no. 11, pp. 1001–1025, 2000.

[55] Rugh, W. J. and Shamma, J. S., "Research on gain scheduling," *Automatica*, vol. 36, no. 10, pp. 1401–1425, 2000.

[56] Hoffmann, C. and Werner, H., "A survey of linear parameter-varying control applications validated by experiments or high-fidelity simulations," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 416–433, 2015.

[57] Shamma, J. and Athans, M., "Analysis of gain scheduled control for nonlinear plants," *IEEE Transactions on Automatic Control*, vol. 35, no. 8, pp. 898–907, 1990.

[58] Liberzon, D. and Morse, A., "Basic problems in stability and design of switched systems," *IEEE Control Systems Magazine*, vol. 19, no. 5, pp. 59–70, 1999.

[59] Liberzon, D., *Switching in systems and control*, vol. 190. Springer, 2003.

[60] Graetz, G. and Michaels, G., "Robots at work," *Review of Economics and Statistics*, vol. 100, no. 5, pp. 753–768, 2018.

[61] Gangapurwala, S., Geisert, M., Orsolino, R., Fallon, M., and Havoutis, I., "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2908–2927, 2022.

[62] Ishihara, Y., Hazama, Y., Suzuki, K., Yokono, J. J., Sabe, K., and Kawamoto, K., "Improving wind resistance performance of cascaded pid controlled quadcopters using residual reinforcement learning," *arXiv e-prints*, 2023.

[63] Mankins, J. C., "Technology readiness assessments: A retrospective," *Acta Astronautica*, vol. 65, no. 9, pp. 1216–1223, 2009.

[64] Kimmel, W. M., Beauchamp, P. M., Frerking, M. A., Kline, T. R., Vassigh, K. K., Willard, D. E., Johnson, M. A., and Trenkle, T. G., "Technology readiness assessment best practices guide," 2020.

[65] Kuhnle, A., Kaiser, J.-P., Theiss, F., Stricker, N., and Lanza, G., "Designing an adaptive production control system using reinforcement learning," *Journal of Intelligent Manufacturing*, vol. 32, no. 3, pp. 855–876, 2021.

[66] Panzer, M. and Bender, B., "Deep reinforcement learning in production systems: a systematic literature review," *International Journal of Production Research*, vol. 60, pp. 4316–4341, 07 2022.

[67] "Robotics industry size & share analysis - growth trends & forecasts (2023 - 2028)," tech. rep., Mordor Intelligence, 2023.

[68] "Robotics technology market," tech. rep., Precedence Research, 2023.

Appendix A: Environment Configuration

All agents were trained using Twin Delay Deep Deterministic Policy Gradient (TD3). Training was implemented using Python 3.10.10, Stable Baselines3 1.5.0, and NumPy 1.24.3. The same set of five seeds for the random number generator were used to train the five agents for each controller type: [699, 227, 788, 316, 204]. The standard hyperparameters from the original TD3 paper [31] and the defaults from the Stable Baselines3 implementation were used to train the agents for the duffing oscillator and the double-pendulum crane. The hyperparameters used to train the inverted pendulum agents were tuned using RL Baselines3 Zoo.

Environment Parameters

Duffing Oscillator

Parameter	Variable	Value
Mass	m	1.0kg
Linear Stiffness	α	$4\pi^2$N/m
Nonlinear Stiffness	β	$\frac{1}{6}\alpha$N/m^3
Damping Ratio	ζ	0.01
Damping Coefficient	c	$2\zeta\sqrt{\frac{\alpha}{m}}m$
Time Step	Δt	0.02s
Desired Displacement Limits	x_d	$[0, 1]$m
Force Input Limits	u	$[-50, 50]$N

Double Pendulum Crane

Parameter	Variable	Value
Hook Mass	m_h	0.0864kg
Payload Mass	m_p	0.0809kg
Hoist Length	l_1	0.73m
Rigging Length	l_2	0.15m
Time Step	Δt	0.05s
Trolley Displacement Limits	x	$[-0.37, 0.37]$m
Trolley Velocity Limits	$\dot{x}$	$[-0.45, 0.45]$m/s
Acceleration Input Limits	u	$[-1.8, 1.8]$m/s^2
Damping Ratio	ζ	$\frac{\sqrt{2}}{2}$
Proportional Gain	k_p	$u_{\max}/x_{\max}$
Derivative Gain	k_d	$2\zeta\sqrt{k_p}$
Reward Weight for x	ω_x	$\frac{1}{0.37^2}$
Reward Weight for θ_1	ω_{θ_1}	$(\frac{12}{\pi})^2$
Reward Weight for θ_2	ω_{θ_2}	$(\frac{12}{\pi})^2$

Inverted Pendulum

Parameter	Variable	Value
Natural Frequency	ω_n	2π rad/s
Pendulum Length	l	g/ω_n^2m
Time Step	Δt	0.05s
Cart Displacement Limits	x	$[-25, 25]$m
Cart Velocity Limits	$\dot{x}$	$[-31.6, 31.6]$m/s
Acceleration Input Limits	u	$[-10, 10]$m/s^2
Reward Weight	ω	$\frac{1}{4\pi^2}$

Hyperparameters for Inverted Pendulum

Hyperparameters used to train the inverted pendulum agents were tuned using RL Baselines3 Zoo.

	Pure RL	RL-LQR	S-RL-LQR
Learning Rate	0.003	0.0018845	0.003
Target Update Rate	0.001	0.001	0.001
Buffer Size	10000	10000	10000
Batch Size	1024	64	1024
Discount Rate	0.99	0.9999	0.99
Action Noise SD	0.7	0.55	0.7

Reward Function Design

The reward functions for this work were all quadratic, which is a common choice of cost functions for optimal control. The terms of each cost function were multiplied by a weighting factor, ω. Each weighting factor was chosen according to Bryson's rule, which is a heuristic for choosing weights based on maximum tolerable values of the

states: $\omega = \frac{1}{(\text{maximum tolerable value})^2}$. This normalizes each term of a multi-objective reward function.

Biographical Sketch

Gerald Eaglin began pursuing an undergraduate degree in mechanical engineering at the University of Louisiana at Lafayette in the Fall of 2012 and completed his bachelor's degree in the Spring of 2016. He began pursuing a master's degree in mechanical engineering in the Fall of 2016 and researched path planning and vibration control under the guidance of Dr. Joshua Vaughan. He then received his master's degree in the Summer of 2018. In the Fall of 2018, he began pursuing his PhD in systems engineering with a concentration in mechanical engineering and graduated in the Fall of 2023. His research interests include reinforcement learning and data-driven control, vibration control, and path planning.