AC-RL: A Framework for Real-Time Control, Learning & Adaptation

by

Anubhav Guha

B.S., Massachusetts Institute of Technology (2018)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

Author . . . .
Department of Mechanical Engineering
August 19, 2022

Certified by . . . .
Anuradha Annaswamy
Senior Research Scientist
Thesis Supervisor

Accepted by . . . .
Nicolas G. Hadjiconstantinou
Chairman, Department Committee on Graduate Theses

AC-RL: A Framework for Real-Time Control, Learning & Adaptation

by

Anubhav Guha

Submitted to the Department of Mechanical Engineering
on August 19, 2022, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

This paper considers the problem of real-time control and learning in dynamic systems subjected to parametric uncertainties. A combination of Adaptive Control (AC) in the inner loop and a Reinforcement Learning (RL) based policy in the outer loop is proposed such that in real-time the inner-loop model reference adaptive controller contracts the closed-loop dynamics towards a reference system, while the RL in the outerloop directs the overall system towards approximately optimal performance. This AC-RL approach is developed for a class of control affine nonlinear dynamical systems, and employs extensions to systems with multiple equilibrium points, systems with input magnitude constraints, and systems in which a high-order tuner is required for adequate performance. In addition to establishing a stability guarantee with real-time control, the AC-RL controller is also shown to lead to parameter learning with persistent excitation. Numerical validations of all algorithms are carried out using a quadrotor landing task on a moving platform. These results point out the clear advantage of the proposed integrative AC-RL approach.

Thesis Supervisor: Anuradha Annaswamy
Title: Senior Research Scientist

Acknowledgments

I would first like to thank Dr. Anuradha Annaswamy for her unwavering support throughout the master's program. The work in this thesis would not have been possible without her guidance, mentorship, keen insight, and creative ideation. I would also like to express my thanks for her willingness to explore novel and exciting ideas, as well as her openness in accommodating my research interests. Adaptive control is truly a fascinating topic, and I've been lucky to learn it from one of the best in the field. I would also like to thank the members of the Active Adaptive Control Laboratory, both old and new. Specifically, I have been lucky enough to work with Yingnan Cui, Sunbochen Tang, José Moreu, and Joey Gaudio, and thank them for the help and input.

I would also like to acknowledge the support of the Boeing Strategic University Initiative, which has made my research and graduate career possible.

Lastly, I very warmly thank my fiancée Rianna Shah, to whom I am eternally grateful and appreciative of, and my parents, who first taught me to love the processes of learning and discovery.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This paper presents a hybrid adaptive control (AC) and reinforcement learning (RL) solution to the problem of real-time control in a dynamic system subject to parametric uncertainties. A number of methods for realizing these goals abound in the controls community, examples of which include adaptive control [55, 31, 72, 79, 40, 3, 27], robust control [97], model-predictive control [29, 68], and sliding-mode control [89]. More recently, reinforcement learning, a subfield of machine learning (ML), has been proposed for the development of control policies for complex systems and environments [69, 10, 83, 90, 91].

The field of adaptive control (AC) has always included as a central element in its design a parametric learning component. With the goal of controlling a dynamic system with parametric uncertainties, an adaptive control solution determines a control input that explicitly includes a parameter estimate. This estimate is recursively adjusted using all available data in real-time so as to converge to its true value. The overall goals of the adaptive system are to guarantee that the control input results in the closed-loop adaptive system having globally bounded solutions and to learn the true parameters. With adaptation to parametric uncertainties as the mechanism of the controller, robustness to nonparametric uncertainties such as disturbances and unmodeled dynamics have been ensured through the use of regularization and persistent excitation[54, 30]. The field of model-free RL, on the other hand, has as its goal the determination of a sequence of inputs that drives a dynamical system to minimize a suitable objective with minimal knowledge of the system model. The central structure of the RL solution revolves around a policy that is learned so as to maximize a desired reward [69, 90, 10, 91].

Both AC and RL-based control methods have addressed the problem of control in the presence of uncertainty, with the component of learning addressed explicitly in both. The two however have deployed entirely different approaches for accomplishing this objective. AC methods have been proven to be effective in a zero-shot" enforcement of objectives for specific classes of problems, such as control, learning and tracking, in real-time[69, 56, 5, 31]. These adaptive techniques are able to accommodate, in real-time, parametric uncertainties and constraints on the control input magnitude [33, 41] and rate [22]. Despite these abilities to accommodate the presence of modeling errors over short time-scales and meet tracking objectives, AC methods are unable to directly guarantee the realization of long-term optimality-based objectives. RL-trained policies, on the other hand, can handle a broad range of objectives [84], where the control policies are often learned in simulation. Training in simulation is a powerful technique, allowing for a near infinite number of agent-environment interactions to allow the policy to become near-optimal. In practice, however, offline policies trained in simulation often exhibit degenerate performance when used for real-time control due to modeling errors that can occur online [36, 86]. It may be difficult to reliably predict the behavior of a learned policy when it is applied to an environment different from the one seen during training [19, 71, 66]. The contribution of the paper is an integration of the AC and RL approaches so as to bridge this "sim-to-real" gap by realizing a combined set of advantages of both approaches.

Several RL approaches have been suggested to deal with the sim-to-real gap. In particular, the methods of domain randomization (DR-RL) [88, 63, 12, 62, 46], and meta-learning based RL (ME-RL) [2] should be noted. In domain randomization, the simulated training environment is perturbed throughout training. This leads to an RL-trained policy that is robust to perturbations that lie within the training distribution [88]. DR has seen broad success in applications to RL tasks [63], and can be used in conjunction with other methods - such as iteration-based system identification. One issue with DR is that the test environment is expected to fall within the test environment distribution. To relax this requirement, in [12], the distribution of training environments is tweaked and improved whenever real world data is collected. Robust RL has also been utilized to accommodate adversarial perturbations and disturbances [62, 46]. In ME-RL [2, 96], a meta-learning based adaptive algorithm allows for a refinement of the policy to occur online. This adaptation algorithm is often based on a neural-network, and thus raises more issues of stability [43] and generalizability [61]. In contrast, the approach we propose in this paper combines AC and RL and allows for provable guarantees on adaptation quality, rates, and bounds, at the cost of being applied to a more constrained class of environments.

The AC-RL approach that we propose in this paper consists of AC-based components in the inner-loop and RL-based components in the outer-loop. The role of RL is to train through simulation the optimal control needed to minimize a desired objective, where the simulation is assumed to have access to a reference system that is the best plant-model available. The role of AC is to accommodate the effect of parametric uncertainties through a suitably designed control input with nonlinear adaptive laws for adjusting its parameters in real-time. The proposed approach is a combination of these two methods such that in real-time the inner-loop AC contracts the closed-loop dynamics towards the reference system, and as the contraction takes hold, the RL in the outerloop directs the overall system towards optimal performance. These properties are guaranteed formally in the paper, and form one of two main contributions of the paper.

We consider a class of control-affine nonlinear dynamic systems are considered, both of which are control-affine. An AC-RL controller is proposed, and the resulting closed-loop system is proved to be stable when certain parametric uncertainties are present. In all cases, the states of the dynamic system are assumed to be accessible for measurement. The performance of AC-RL is guaranteed through closed-loop stability and a constant regret [67, 11], defined as the difference in an integral control performance between the controller employing a given algorithm and the best controller given full knowledge of the plant. Validations are carried out using a numerical experiment in which a quadrotor that is required to land on a moving platform, with medium-fidelity models that include realistic mechanisms of nonlinear kinematics, actuator nonlinearities, and measurement noise. The demonstration of real-time control of the AC-RL approach for these classes of dynamic systems through closed-loop stability and constant regret is the first contribution of this paper. Extension to a class of non-affine dynamic systems is also presented.

The central component of the AC algorithm is built on a high-order tuner which was first proposed by [52] in an effort to develop stable low-order adaptive controllers. Rather than utilizing a gradient-descent algorithm to directly generate a first-order tuner for determining the parameter estimates, the idea here is to use a higher-order tuner so as to allow the controller to implicitly generate a reference model that is strictly positive real. This idea was subsequently explored further in [60] for general adaptive control designs and in [16] to allow adaptive control of time-delay systems in a stable manner. Independently, higher-order tuners have also been sought after in the ML community, in an effort to obtain accelerated convergence of an underlying cost function and the associated accelerated learning of the minimizer of this function (see for example, [57, 58, 80, 93, 95]). The idea here is to include momentum-based updates so as to get a faster convergence of the performance error and have seen widespread applications in machine learning [37, 82]. Elements of the HT algorithms in [52] were utilized to develop several types of HTs in continuous-time [21] leading to stability, and in discrete-time [24] leading to stability with time-varying regressors and accelerated convergence with constant regressors as in [57]. We leverage these stability and accelerated convergence properties in the AC-RL control design in this paper by fully integrating the properties of RL into the controller.

Efforts to combine AC and RL approaches have been highlighted in several recent works, which include [25, 47, 92, 81]. Algorithms that combine AC and RL in continuous-time systems can be found in [28, 70, 13, 17]. [70] proposes the use of adaptive control for nonlinear systems in a data-driven manner, but requires offline trajectories from a target system and does not address accelerated learning or magnitude saturation. References [13, 17] study the linear-quadratic-regulator problem and its adaptive control variants from an optimization and machine learning perspective. In [92] a reinforcement learning approach is used to determine an adaptive controller for an unknown system, while in [81] principles from adaptive control and Lyapunov analysis are used to adjust and train a deep neural network. This thesis addresses a comprehensive treatment of an integrated AC-RL approach, including extensions with a high-order tuner, magnitude saturation, and nonlinear systems that may be linearized around multiple equilibrium points.

Chapter 2

Background

2.1 Reinforcement and Machine Learning

Reinforcement Learning (RL) methods are used to generate reward-maximizing policies over Markov Decision Processes (MDPs) [83]. Formally, an MDP is characterized by a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$. $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively, and may be either continuous or discrete (or hybrid) spaces. In this thesis we will largely consider continuous state/action spaces. P is a probabilistic transition function, where $P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$ is the probability of transitioning into state $s_{t+1} \in \mathcal{S}$ in timestep $t+1$ after taking action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$ at timestep t. $r(s_t, a_t, s_{t+1})$ is the reward function, which quantifies the immediate "goodness" of taking action a_t in state s_t resulting in state s_{t+1}. Often, the reward function may have a simpler form and is only a function of s_t, a_t. In the following we will assume that $r = r(s_t, a_t)$. $\gamma \in [0, 1)$ is the discount factor, which quantifies the degree to which immediate rewards are sought over distant rewards. The concepts of "goodness" and discounted rewards are formalized via the introduction of the value function.

Before defining the value function, we introduce the concept of a policy $\pi(a|s)$ which is a (often stochastic) function that maps a given state to a choice of (or distribution over) actions. The value function associated with a policy π, then, is a scalar function that maps $\mathcal{S} \to \mathbb{R}$:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s\right] \tag{2.1}$$

The infinite sequence of state-action pairs $(s_0, a_0, s_1, a_1, \dots)$ are subject to and generated by the probabilistic functions π and P. Note that the above deals with an infinite horizon MDP problem - throughout the thesis it will be assumed that all decision-making problems are infinite in horizon unless otherwise specified.

Dissecting (2.1), it can be seen that the value function provides a measure of the expected cumulative discounted reward achieved in the MDP when following a policy π. The goal of RL is to find an optimal policy π^* that solves the following:

$$\pi^* \in \arg\max_{\pi \in \Pi} V^{\pi}(s),\ \forall s \in \mathcal{S} \tag{2.2}$$

in which Π is the set of all admissible policies. For example, if π is to be parametrized by a deep neural network, Π represents the set of all functions that can be produced by the given neural network architecture [18].

The Q-function, closely related to the value function, is another "value-like" mapping that is crucial in the construction of a number of reinforcement learning algorithms:

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a, a_t = \pi(s_t)\ \forall t > 0\right] \tag{2.3}$$

Alternatively, the Q-function may be defined in terms of the value function:

$$Q^{\pi}(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(s'|s,a)}[V^{\pi}(s')] \tag{2.4}$$

That is, $Q^{\pi}(s, a)$ represents the expected discounted cumulative reward of taking action a in the current state s and proceeding to follow policy π for the remaining trajectory. Note also that $V^{\pi}(s) = Q^{\pi}(s, \pi(a))$. We additionally define the optimal value function as $V^* = \max_{\pi \in \Pi} V^{\pi}(s),\ \forall s \in \mathcal{S}$, and the optimal Q-function as $Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[V^*(s')]$, noting the convenient relation $V^*(s) = \max_{a \in \mathcal{A}} Q^*(s,a)$.

Most RL algorithms are concerned with learning one or more of the optimal functions V^*, Q^*, π^*. In the following, the dynamic programming (DP) principle will be briefly introduced. It will then be shown how DP paves the way for approximate dynamic programming and reinforcement learning when assumptions about the MDP are relaxed.

2.1.1 Dynamic Programming

Bellman's principle of optimality states that the optimal value function and policy satisfy the following [6]:

$$V^*(s) = \max_{a \in \mathcal{A}} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[V^*(s')] \; \forall s \in \mathcal{S} \tag{2.5}$$

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \left[r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[V^*(s')] \right] \; \forall s \in \mathcal{S} \tag{2.6}$$

Equation (2.5) is a non-linear system of equations and cannot be easily (in the general case) solved in close form. Instead, the Bellman operator T is introduced. Give an arbitrary initial function $W : \mathcal{S} \to \mathbb{R}$, we define T as:

$$TW(s) = \max_{a \in \mathcal{A}} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[W(s')] \; \forall s \in \mathcal{S} \tag{2.7}$$

This operator T can be used to define the value iteration algorithm, in which a sequence of value functions (indexed by i) are constructed via application of the Bellman operator: $V_{i+1}(s) = TV_i(s) \; \forall s \in \mathcal{S}$. It can be shown that the value iteration algorithm produces a sequence of value functions V_i that asymptotically approaches the optimal value function: $\lim_{i \to \infty} V_i = V^* \; \forall s \in \mathcal{S}$.

Value iteration is one algorithm for solving an MDP; other methods such as policy iteration employ similar dynamic programming principles and have relative merits and weaknesses when compared to value iteration [10, 9]. These methods have two weaknesses in common, however: 1) direct access to the model, P, is assumed (this dependency can be seen in the expectation in (2.7), which is taken with respect to the transition dynamics) and 2) the explicit evaluation of terms depending on states/actions scales poorly with high-dimensional or continuous state-action spaces. This can be seen in the Bellman operation (2.7) which scales as $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$.

2.1.2 Approximate Dynamic Programming

Relaxing the requirement that the transition function P is known is an attractive proposition. Many control systems of interest operate in environments where good analytic models may be difficult/impossible to obtain, such as an aircraft subject to turbulent flows and wind.

One approach is to construct Monte-Carlo estimates of the expectation in (2.7). In the following we utilize finite time-horizon notation for ease of exposition, however the developments are easily generalized to the infinite time-horizon case. Consider a set $\mathcal{D}$ of n trajectories, each of length T, generated by following a policy π: $\mathcal{D} = [(s_0^i, a_0^i), (s_1^i, a_1^i), \ldots, (s_{T-1}^i, a_{T-1}^i)]^{i=1:n}$ The cumulative discount return for each trajectory i is denoted as:

$$\hat{R}_i(s_0) = \sum_{t=0}^{T} \gamma^t r(s_t^i, a_t^i) \tag{2.8}$$

Each $\hat{R}_i(s_0)$ can be considered an unbiased of $V^\pi(s_0)$. Therefore, the value function may be approximated by the following unbiased estimator:

$$\hat{V}^\pi(s_0) = \frac{1}{n}\sum_{i=1}^{n} \hat{R}_i(s_0) \tag{2.9}$$

Such a Monte-Carlo approach can be used to generalize the value iteration and policy iteration algorithms to the approximate dynamic programming (ADP) case [8].

Another model-free iterative method that relaxes the need for direct dynamics access is Q-learning. While Q-learning is a powerful technique that does not employ Monte-Carlo rollouts, it is only applicable to MDPs with finite state-action spaces. Moreover, it exhibits an unfavorable scaling with respect to $|\mathcal{S}|, |\mathcal{A}|$, which limits its applicability to high-dimensional problems. In Q-learning iterates of the optimal

Q-function estimate, $\hat{Q}_i$, are updated using collected trajectories as:

$$\hat{Q}_{i+1}(s_t, a_t) = \hat{Q}_i(s_t, a_t) + \eta_t \left[r(s_t, a_t) + \gamma \max_{a' \in \mathcal{A}} [\hat{Q}_i(s_{t+1}, a') - \hat{Q}_i(s_t, a_t)] \right] \tag{2.10}$$

where η_t is a sequence of learning rates. If η_t satisfy the Robbins-Monro conditions:

$$\sum_{i=0}^{\infty} \eta_i = \infty \qquad \sum_{i=0}^{\infty} \eta_i^2 < \infty \tag{2.11}$$

and all state-action pairs are tried infinitely often, then asymptotic convergence of $\hat{Q}_i$ to Q^* is achieved [91].

The above methods, and many others, rely on iterating through state and/or action spaces. Therefore such methods are unusuable for MDPs with large or continuous state-action spaces.

2.1.3 Policy Gradient Methods

The methods in Section 2.1.2 may be considered iterative approaches to constructing optimal policies and value functions. An alternative approach is to instead *search* within a space of policies [85]. Let the policy π be parametrized by a d-dimensional vector θ so that it is given by π_θ. A policy optimization problem may then be defined:

$$\max_{\theta} V(\pi_\theta) \tag{2.12}$$

Therefore, if $\nabla_\theta V(\pi_\theta)$ can be computed, stochastic gradient ascent methods may be used to optimize (2.12). The policy gradient theorem states:

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}_{\tau \sim h(\cdot|\pi_\theta, MDP)} \left[R(\tau) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s_t, a_t) \right] \tag{2.13}$$

Here, $\tau = [(s_0, a_0), (s_1, a_1), \dots, (s_{T-1}, a_{T-1})]$ is a finite length trajectory emerging from application of policy π_θ to the MDP. As the transitions and π_θ may be stochastic, τ is treated as a random variable and therefore an expectation is performed with respect to τ and it's generating distribution. $R(\tau)$ is defined as the discounted sum of returns from the trajectory τ, analogous to (2.8). Note that the policy gradient theorem requires the differentiability of π_θ w.r.t it's parameters.

As the expression in (2.13) depends on an expectation taken with respect to MDP dynamics, it cannot be used as is in a model-free algorithm. However, Monte-Carlo approximation techniques similar to those used in Section 2.1.2 can be used, in conjunction with collected rollouts, to produced unbiased estimates of the required terms. Such an approach leads to Williams' well known REINFORCE algorithm [94].

While a Monte-Carlo approximate is indeed unbiased, the estimate is often plagued with a high variance [26]. This variance can cause vanilla policy gradient algorithms to exhibit slow convergence/high sample complexity. Since the original REINFORCE algorithm was introduced, many more modern RL algorithms have been proposed to reduce the variance of policy and value gradient estimates. A number of these methods will be explored in Section 3.1. Such methods often make sure of an actor-critic architecture, in which a parametric value function (or Q-function) estimate is introduced. This estimator, referred to as the "critic", is used, alongside a policy gradient based update, to update the parameters of the policy (the "actor"). A basic actor-critic algorithm is given by the following updates [35]:

$$\psi_{i+1} = \psi_i + \nu[r(s_t, a_t) + \gamma Q_{\psi_i}(s_{t+1}, a_{t+1}) - Q_{\psi_i}(s_t, a_t)]\nabla_{\psi_i} Q_{\psi_i}(s_t, a_t) \tag{2.14}$$

$$\theta_{i+1} = \theta_i + \eta \nabla_{\theta_i} \log \pi_{\theta_i}(a_t|s_t) Q_{\psi_i}(s_t, a_t) \tag{2.15}$$

Where η, ν are learning rates and ψ, θ are the parameters of the Q-function and policy estimates, respectively.

2.1.4 Model-Based Reinforcement Learning

The development of model-free RL/ADP methods from the model-based DP algorithms in motivated by 1) situations in which there exists no *a priori* specified transition or dynamics model, and 2) MDPs in which the state-action spaces are very large or continuous. While these model-free algorithms relax the analytic model condition, they come with a data requirement: all the model-free approaches surveyed in the previous sections rely on access to collected state-action pairs from application of actions and policies to the MDP. Furthermore, as many of the techniques rely on Monte-Carlo estimation, the sample requirement may be high for expansive or complex environments. The collection of these state-action pairs suggests another approach to solving the MDP - the data may be used to train a model of the environment dynamics, which can then be used to optimally determine actions (e.g, to generate a policy). Given a learned model, it may be possible to apply the aforementioned dynamic programming techniques of value or policy iteration. These approaches, however, will still face issues related to computational complexity [64]. In this section we will instead briefly survey methods that aim to use techniques and tools from optimization.

Model-based reinforcement learning begins conceptually with dynamics modeling [49]. Dynamics models can come in a number of forms, including inverse models (learning an a_t that causes the transition $s_t \to s_{t+1}$) and reverse models (learning which state-action pairs (s_{t-1}, a_{t-1}) may precede a state s_t. Here we consider forward models, which attempt to learn the mapping from a given state-action pair (s_t, a_t) to the next state s_{t+1}.

The following routine provides a rudimentary but straightforward model-based RL algorithm: Assume a given a dataset $\mathcal{D}$ of N state-action-state tuples:

$$\mathcal{D} = [(s_0, a_0, s_0'), (s_1, a_1, s_1'), \ldots, (s_N, a_N, s_N')]$$

where s_i' denotes the observed state after action a_i was applied in state s_i. A deterministic forward dynamics model, parametrized by θ, may be learned by solving:

$$\min_{\theta \in \Theta} \frac{1}{N} \sum_{i=0}^{N} ||s_i' - f_\theta(s_i, a_i)||^2 \tag{2.16}$$

If f_θ is represented as a neural network, solving this problem amounts to a standard supervised nonlinear regression neural network problem. Supposing an adequate forward dynamics model f_θ is learned, one may then apply this estimator to the problem of action generation. Given a reward function $r(s, a)$ and the current state s_t the following model-predictive control (MPC) optimization problem is solved [20]:

$$\begin{aligned} \max_{a_{t:t+T_h}} \quad & \sum_{\tau=t}^{t+T_h} \gamma^t r(s_t, a_t) \\ \text{s.t} \quad & s_{\tau+1} = f_\theta(s_\tau, a_\tau) \quad \tau = t, \ldots, t+T_h \\ & s_\tau \in \mathcal{S}, \quad \tau = t, \ldots, t+T_h \\ & a_\tau \in \mathcal{A}, \quad \tau = t, \ldots, t+T_h \end{aligned} \tag{2.17}$$

T_h denotes the length of the optimization horizon. In typical MPC fashion, the calculated control is obtained by accessing the value of the first decision variable, a_t, from the solution of the optimization problem. In the general case, (2.17) may be very difficult to solve exactly - owing to the potentially complex forms of the reward function r and the learned forward dynamics f_θ. As a result, a number of methods based on convex/quadratic approximations have been developed. Moreover, if the state-action constraints, learned dynamics and reward function have specific forms (such as convexity), efficient computational techniques may be used to arrive at globally optimal solutions.

2.2 Adaptive Control

A fundamental goal in adaptive control (AC) is to design an exogenous input $u(t) \in \mathbb{R}^m$for a dynamical system given by

$$\dot{x} \;=\; f(x, \theta, u, t) \tag{2.18}$$

where $x(t) \in \mathbb{R}^n$ represents the system state, $\theta \in \mathbb{R}^\ell$ represents system parameters that may be unknown, and $f(\cdot)$ denotes (potentially nonlinear) system dynamics that capture the underlying physics of the system. The function $f(\cdot)$ may vary with t, as disturbances and stochastic noise may affect the states and output. The goal is to choose $u(t)$ so that $x(t)$ tracks a desired command signal $x_c(t)$ at all t, and so that an underlying cost $J((x - x_c), x, u)$ is minimized. The challenge is to find this solution in real-time in the presence of uncertainties that are predominantly present in θ. In what follows, we will refer to the system that is being controlled as a plant.

As the description of the system as in (2.18) is based on a plant model, and as the goal is to determine the control input in real time, all control approaches make assumptions regarding what is known and unknown. The function f is often not fully known, as the plant is subject to various perturbations and modeling errors due to environmental changes, complexities in the underlying mechanisms, aging, and anomalies. The field of adaptive control takes a parametric approach to distinguish the known parts from the unknown. In particular, it is assumed that f is a known function, while the parameter θ is unknown. A real-time control input is then designed so as to ensure that the tracking goals are achieved by including an adaptive component that attempts to estimate the parameters online.

In the following subsections, we further breakdown the approach taken to address these problems, especially in the context of learning and optimization. While the description below is in the context of deterministic continuous-time systems, similar efforts have been carried out in stochastic and discrete-time dynamic systems as well.

2.2.1 An adapt-learn-optimize approach

The goal of the adaptive controller is to ensure that

$$\lim_{t \to \infty} e(t) = 0 \tag{2.19}$$

where $e(t) = ||x(t) - x_c(t)||$. As these decisions are required to be made in real time, the focus of the adaptive control approach is to lead to a closed-loop dynamic system that has bounded solutions at all time t and a desired asymptotic behavior as in (2.19) The central question is if this can be ensured even when there are parametric uncertainties in θ and several other non-parametric uncertainties that may due to unmodeled dynamics, disturbances, or other unknown effects. Once this is guaranteed, the question of learning, in the form of parameter convergence, is addressed. As a result, *control for learning* is a central question that is pursued in the class of problems addressed in adaptive control rather than *learning for control* [38]. Once the control and learning objectives are realized, one can then proceed to the optimization of a suitable cost J. This sequence of *adapt-learn-optimize* is an underpinning of much of adaptive control.

The above sequence can be reconciled with the well known certainty equivalence principle (CEP) which proceeds in the following manner: first, optimize under perfect foresight, then substitute optimal estimates for unknown values. This philosophy underlies all adaptive control solutions by first determining a controller structure that leads to an optimal solution when the parameters are known and then replace the parameters in the controller with their estimates. The difficulty in adopting this philosophy to its fullest stems from the dual nature of the adaptive controller, as it attempts to accomplish two tasks simultaneously, estimation and control. This simultaneous action introduces a strong nonlinearity into the picture and therefore renders a true deployment of the certainty equivalence principle intractable. Instead, an *adapt-learn-optimize* sequence is adopted, with the first step corresponding to an adaptive controller that leads to a *stable* solution. This is then followed by estimation of the unknown parameters, and optimization addressed at the final step.

A typical solution of the adaptive controller takes the form

$$u = C_1(\theta_c(t), \phi(t), t) \tag{2.20}$$

$$\dot{\theta}_c = C_2(\theta_c, \phi, t) \tag{2.21}$$

where $\theta_c(t)$ is an estimate of a control parameter that is intentionally varied as a function of time, $\phi(t)$ represents all available data at time t. The nonautonomous nature of C_1 and C_2 is due to the presence of exogenous signals such as set points and command signals. A stabilization task would render these functions autonomous. The functions $C_1(\cdot)$ and $C_2(\cdot)$ are deterministic constructions, and make the overall closed-loop system nonlinear and nonautonomous. The challenge is to suitably construct functions $C_1(t)$ and $C_2(t)$ so as to have $\theta_c(t)$ approach it's true value θ_c^*, and ensure that stability and asymptotic stability properties of the overall adaptive systems are ensured. These constructions have been delineated for deterministic systems in [55, 4, 31, 72, 39, 87] and other textbooks. The solutions in these books and several papers in premier control journals have laid the foundation for the construction of C_1 and C_2 for a large class of dynamic systems in (2.18)

2.2.2 Model Reference Adaptive Control

A tractable procedure for determining the structure of the functions C_1 and C_2, denoted as *Model Reference Adaptive Control* (MRAC), uses the notion of a reference model, and a two-step design consisting of an algebraic part for determining C_1 and an analytic part for finding C_2. A reference model provides a structure to the class of command signals $y_c(t)$ that the plant output x can follow. For a controller to exist for a given plant-model using which the closed-loop system can guarantee output following, the signal x_c needs to be constrained in some sense. A reference model is introduced to provide such a constraint. In particular, a model $\mathcal{M}$ and a reference input r is designed in such a way that the output $y_m(t)$ of $\mathcal{M}$ for an input $r(t)$ approximates the class of signals $x_c(t)$ that is desired to be followed. With a reference model in $\mathcal{M}$, the algebraic part of the MRAC corresponds to the choice of C_1 with a fixed parameter θ_c^* such that if $\theta_c(t) \equiv \theta_c^*$ in (2.20), then $\lim_{t\to\infty} ||x_p(t) - x_m(t)|| = 0$. The existence of such a θ^* is referred to as a *matching condition*. With such a C_1 determined, noting that θ_c^* could be unknown due to the parameteric uncertainty in the plant, the analytic part focuses on finding C_2 such that output following takes place with the closed-loop system remaining bounded. An alternative to the above direct approach of identifying the control parameters is an indirect one where the plant parameters are first estimated using which the control parameter $\theta_c(t)$ is determined at each t. In what follows, we describe the details of the MRAC approach for various classes of dynamic systems, ranging from simple and algebraic cases to nonlinear dynamic ones.

2.2.3 Linear Plants

Algebraic systems

Many problems in adaptive estimation and control may be expressed as

$$y(t) = \theta^{*T}\phi(t), \tag{2.22}$$

where $\theta^*, \phi(t) \in \mathbb{R}^N$ represent an unknown parameter and measurable regressor, respectively, and $y(t) \in \mathbb{R}$ represents an output that can be determined at each t. Given that θ^* is unknown, we formulate an estimator $\hat{y}(t) = \theta^T(t)\phi(t)$, where $\hat{y}(t) \in \mathbb{R}$ is the estimated output and the unknown parameter is estimated as $\theta(t) \in \mathbb{R}^N$. This in turn results two types of errors, a performance error $e_y(t)$ and a learning error $\widetilde{\theta}(t)$[1]

$$e_y = \hat{y} - y, \qquad \widetilde{\theta} = \theta - \theta^* \tag{2.23}$$

where the former can be measured but the latter is unknown though adjustable. From (2.22) and the estimator, it is easy to see that e_y and $\widetilde{\theta}$ are related using a simple regression relation

$$e_y(t) = \widetilde{\theta}^T\phi(t). \tag{2.24}$$

A common approach for adjusting the estimate $\theta(t)$ at each time t is to use a gradient rule and a suitable loss function. One example is a choice

$$L_1(\theta) = \frac{1}{2}e_y^2 \tag{2.25}$$

leading to the gradient rule

$$\dot{\theta}(t) = -\gamma\nabla_\theta L_1(\theta(t)), \qquad \gamma > 0 \tag{2.26}$$

That this leads to a stable estimation scheme can be shown using a Lyapunov function $V = \widetilde{\theta}^T\widetilde{\theta}$ as its time-derivative $\dot{V} = -e_y^2$.

[1]In what follows, we suppress the argument (t) unless needed for emphasis.

Dynamic Systems with States Accessible

The next class of problems that has been addressed in adaptive control corresponds to plants with all states accessible. We present the solution for the simple case for a scalar input:

$$\dot{x} = A_p x + b_p u \tag{2.27}$$

where A_p and b_p are unknown, u is the control input and is a scalar, and x is the state and is accessible for measurement. As mentioned in Section 2.2.1, the first step is find a reference model $\mathcal{M}$, which takes the form

$$\dot{x}_m = A_h x_m + br \tag{2.28}$$

and is such that the state $x_m(t)$ encapsulates the desired solution expected from the controlled plant. This can be accomplished by choosing a reference input r, A_h to be a Hurwitz matrix, (A_h, b) is controllable so that together they produce an $x_m(t)$ that approximates the signal that the plant is required to track.

With the reference model chosen as above, the next step pertains to Matching Condition [55], stated as follows:

Assumption 1 *A vector θ^* and a scalar k^* exist that satisfy*

$$\begin{aligned} A_p + b_p\theta^{*T} &= A_h \qquad (2.29)\\ b_p k^* &= b \qquad (2.30)\end{aligned}$$

This implies that a fixed control exists of the form

$$u(t) = \theta^{*T} x(t) + k^* r(t) \tag{2.31}$$

that matches the closed-loop system to the reference model. This corresponds to the Algebraic Part of the problem described in Section 2.2.1.

The final step is the analytic part, the rule for estimating the unknown parameters θ^* and k^* and the corresponding adaptive control input that replaces the input choice in (2.31). These solutions are given by

$$\begin{aligned} u &= \theta^T(t)x + k(t)r && (2.32)\\ \dot{\theta} &= -sign(k^*)\Gamma_\theta(e^T P b_m)x && (2.33)\\ \dot{k} &= -sign(k^*)\gamma_k(e^T P b_m)r && (2.34) \end{aligned}$$

where $\Gamma_\theta > 0$ is a positive definite matrix, $\gamma_k > 0$ is a positive constant, $e = x - x_m$, and $P = P^T \in \mathbb{R}^{n\times n}$ is a positive definite matrix that solves the Lyapunov equation

$$A_m^T P + P A_m = -Q \tag{2.35}$$

with a positive definite matrix $Q = Q^T \in \mathbb{R}^{n\times n}$. It can be shown that

$$V = e^T P e + |k^*| \left[(\theta - \theta^*)^T \Gamma^{-1}(\theta - \theta^*) + (1/\gamma_k)(k - k^*)^2\right] \tag{2.36}$$

is a Lyapunov function with $\dot{V} = -e^T Q e$ and that $\lim_{t\to\infty} e(t) = 0$. The reader is referred to Chapter 3 in [55] for further details. In summary, the adaptive controller that is proposed here can be viewed as an *action-response-correct* sequence where the action is the control input given by (2.32), the response is the resulting state error e, and the correction is the parameter adaptive laws in (2.33)-(2.34)

It should be noted that the adaptation rules in (2.33)-(2.34) can also be expressed as the gradient of a loss function [24]

$$L_2(\bar{\theta}) = \frac{d}{dt}\left\{\frac{e^T P e}{2}\right\} + \frac{e^T Q e}{2}, \tag{2.37}$$

where $\bar{\theta} = [\theta^T, k]^T$, and it is assumed that $k^* > 0$ for ease of exposition. It is noted that this loss function L_2 differs from that in (2.25), and includes an additional component that reflects the dynamics in the system. It is easy to see that

$$\dot{\bar{\theta}}(t) = -\Gamma \nabla_{\bar{\theta}} L_2(\theta(t)), \qquad \Gamma > 0, \tag{2.38}$$

and is implementable as $\nabla_\theta L_2(\theta) = \phi e^T P b_m$, can be computed at each time t, where $\phi = [x_p^T, r]^T$. This implies that a real-time control solution that is stable depends critically on choosing an appropriate loss function.

The matching condition (2.30) is akin to the controllability condition, albeit somewhat stronger, as it requires the existence of a θ^* for a known Hurwitz matrix A_m [42, 55]. The other requirement is that the sign of k^* needs to be known, which is required to ensure that V is a Lyapunov function.

Chapter 3

Methods

3.1 Deep Reinforcement Learning

As highlighted in Section 2.1.3, parametric representations of Q-functions, value functions and/or policies can be useful in the development of model-free reinforcement learning algorithms. Although RL is a principled mathematical approach to solving decision-making problems, applications to high-dimensional systems and complex problems were historically stymied by issues of computational complexity, sample efficiency and memory limits. The advent of deep learning techniques, however, has led to an explosion of successful reinforcement learning algorithms and applications, many of which leverage the power of deep neural networks [78, 48, 59, 44, 34]. As mentioned in Section 2.1.3, neural networks may be utilized as the workhorse function approximators underlying the parametric estimates in model-free ADP/RL algorithms. Although deep neural networks prove to be powerful technical tools, the core RL issues of variance-reduction, sample complexity, and non-local optimization remain at play. Furthermore, the inclusion of neural networks raises additional questions of robustness and generalization. In the following, two popular and effective deep reinforcement learning algorithms are presented in brief. One of these algorithms, Proximal Policy Optimization (PPO) [75] is used as an outer-loop RL candidate for the proposed AC-RL approach.

3.1.1 Trust Region & Proximal Policy Optimization

In order to introduce the Trust Region Policy Optimization (TRPO) and Proximal Policy Optimization (PPO) algorithms, we first define a few required quantitites.

First, the advantage function A is a value-like function defined in terms of the value and Q-functions:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \tag{3.1}$$

The scalar value $A^{\pi}(s,a)$ can be thought of as relative change in accumulated value when action a is taken rather than the action dictated by policy π. A positive advantage value indicates that the action a is superior to the action (or distribution over actions) dictated by the policy π.

We additionally define the Kullback-Leibler (KL) divergence D_{KL}, which measures the difference between two probabilitiy distributions P, Q. For discrete probability distributions P, Q defined over the same probability space $\mathcal{X}$:

$$D_{KL}(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)} \tag{3.2}$$

Trust Region Policy Optimization

TRPO is an actor-critic method which constructs a parametric model of a policy π_θ as well as a parametric model of a value function V_ψ [73]. In deep reinforcement learning applications, both θ and ψ represent the parameters of different neural networks. In order to determine the policy parameters θ, TRPO formulates and solves the following constrained optimization:

$$\begin{aligned} \theta_{i+1} = \arg\max_{\theta_{i+1}} \quad & \mathbb{E}_{s \sim \rho_{\pi_{\theta_i}}, a \sim \pi_{\theta_i}} \left[\frac{\pi_{\theta_{i+1}}(a|s)}{\pi_{\theta_i}(a|s)} A_\psi(s,a) \right] \\ \text{s.t} \quad & \mathbb{E}_{s \sim \rho_{\pi_{\theta_i}}} [D_{KL}(\pi_{\theta_i}(\cdot|s) || \pi_{\theta_{i+1}}(\cdot|s))] \le \delta \end{aligned} \tag{3.3}$$

Here, δ is a hyperparameter specifying the maximimum desired change in KL-divergence between the old policy and new policy, at each timestep. $\rho_\pi(s) = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_2 = s)$ is the discounted visitation frequency when following policy π. The optimization objective can be neatly understood - actions associated with a positive advantage will be made more likely under the new probability distribution $\pi_{\theta_{i+1}}$. The constraint prevents the policy from changing too rapidly at each iteration of the algorithm, which can help stabilize training. The value/advantage function parameters ψ may be updated using a standard method based on temporal differencing. Furthermore, Monte-Carlo sampling techniques are used, as previously discussed, to approximate the expectations as required. As (3.3) represents a constrained optimization problem, the policy neural network paramters θ cannot be simply updated via stochastic gradient descent. Instead the conjugate gradient method is used, which incurs computational complexity and overhead.

Proximal Policy Optimization

PPO builds off the TRPO algorithm. The main difference is that in PPO the constrained optimization in TRPO is turned into an unconstrained maximization. The objective in PPO is instead [75]:

$$\theta_{i+1} = \arg\max_{\theta_{i+1}} \quad \mathbb{E}_{s\sim\rho_{\pi_{\theta_i}}, a\sim\pi_{\theta_i}}\left[\frac{\pi_{\theta_{i+1}}(a|s)}{\pi_{\theta_i}(a|s)}A_{\psi}(s,a)\right] - \beta_i \mathbb{E}_{s\sim\rho_{\pi_{\theta_i}}}[D_{KL}(\pi_{\theta_i}(\cdot|s)||\pi_{\theta_{i+1}}(\cdot|s))] \tag{3.4}$$

Where the hard KL-divergence constraint in TRPO has been replaced soft penalty in the objective function. The scalar β_i represents a hyperparameter weighting this penalty. A number of variants of PPO exist, in which heuristic approaches are used to more effectively incorporate the KL constraint into the maximization objective. However, all PPO algorithms are similar in that they utilize an unconstrained approximation to the TRPO objective. This enables the application of straightforward deep neural network training techniques, such as stochastic gradient descent and it's ilk.

3.2 Domain Randomization

Consider first the discrete-time dynamics of a system are given by

$$s_{t+1} \sim h(s'|s_t, a_t, \bar{\omega}) \tag{3.5}$$

where $\bar{\omega}$ represents some fixed parameter vector that influences the dynamical system. For example, in a cart-pole system $\bar{\omega}$ may represent the mass and inertia of the constituent rigid bodies. In a practical application of reinforcement learning to a physical system, it is often the case that a simulator exists for the physical environment. As many reinforcement learning algorithms are sample inefficient it can be desirable to train the policy using simulation. However, the "true" parameter vector $\bar{\omega}$ associated with the physical system may be unknown or uncertain. In this case, domain randomization is often used to train the policy in simulation. Domain randomization is a straightforward technique in which each instantiation of the simulated environment utilizes a random parameter vector ω drawn from some underlying distribution [88]:

$$s_{t+1} \sim h(s'|s_t, a_t, \omega), \quad \omega \sim \mathcal{P} \tag{3.6}$$

This process, however, simply induces a new dynamic system:

$$s_{t+1} \sim h'(s'|s_t, a_t) \tag{3.7}$$

In which the stochastic effect of sampling a random parameter vector has been subsumed into a new transition probability function, h'. Therefore, standard model-free reinforcement learning algorithms can be cleanly applied to the domain randomized simulation. As a result, the RL algorithm does not need to be altered; only the underlying simulation needs to be augmented.

3.3 Meta-Learning

A general meta-learning RL (ME-RL) policy can be split into two sub-policies: a base learner and an adaptive learner [2, 96, 53]. The base learner constructs a policy π_θ that generalizes robustly over the distribution of environments. A separate adaptive learner, parametrized as π_ϕ is then trained to update the outputs of π_θ in an online fashion. Colloquially, the adaptive meta-learner is "learning to adapt". The base learner may first be trained to find θ^*:

$$\begin{aligned} \theta^* = \arg\max_{\theta} \quad & \mathbb{E}\left[\sum_{t=0}^{T} r(s_t, a_k)\right] \\ \text{s.t} \quad & s_{t+1} \sim h(s_t, a_t, \bar{\omega}), \quad a_t = \pi_\theta(s_t) \end{aligned}$$

A typical adaptive meta learner is then trained to use state-action histories to determine an adaptive policy:

$$\begin{aligned} \phi^* = \arg\max_{\phi} \quad & \mathbb{E}\left[\sum_{t=0}^{T} r(s_t, a_t)\right] \\ \text{s.t} \quad & s_{t+1} \sim h(s_t, a_t, \omega), \quad \omega \sim \mathcal{P} \\ & \hat{a}_t = \pi_{\theta^*}(s_t) \\ & a_t = \pi_\phi(\hat{a}_t, s_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) \end{aligned}$$

Such a method is similar to classical adaptive control techniques, with the main distinction that that a deep network parameterizes the adaptive law.

It should be noted that though RL has led to compelling successes, analytical guarantees of convergence to optimal policies have been shown only in simple settings [91, 14, 7, 45]. No analytical guarantees are provided when general deep networks are used, except for papers such as [43] which use additional correction terms to prove stability.

3.4 High-Order Tuners

The core of the adaptive components proposed in this paper is based on high-order tuners (HT) [52, 60, 16]. The idea of HT is to use a high-order filter in order to generate the parameter estimate rather than the first-order gradient method employed in the classical adaptive controller. The motivation for its use in adaptive control has come from the need to develop robust adaptive controllers that are low-order and can accommodate large lags in the system dynamics. These controllers require parameter estimates that are differentiable to an arbitrary order. The contribution in [52] lies in the development of such a HT that guarantees that the closed-loop adaptive system is globally stable. A completely independent direction of research that has also led to HT is due to a particular body of work in ML which has focused on accelerated convergence of an underlying cost function [57, 58, 80, 93, 95]. The idea behind HT is summarized below.

Parameter identification in a linear regression problem of the form $y^*(t) = \theta^{*T}\phi(t)$ where $\theta^* \in \mathbb{R}^n$ represents an unknown parameter, and $\phi(t) \in \mathbb{R}^n$ is an underlying regressor that can be measured at each t can be formulated as a minimization problem $L_t(\theta) = e^2(t)$ where $e = y - y^*$ and $y(t) = \theta^T\phi(t)$. A first-order tuner that can be used to solve the minimization problem is of the form $\dot{\theta} = -\gamma\nabla_\theta L_t(\theta)$. In [21] [24], a HT of the form

$$\ddot{\theta} + \beta\dot{\theta} = -\frac{\gamma\beta}{\mathcal{N}_t}\nabla_\theta L_t(\theta) \tag{3.8}$$

was proposed and shown to correspond to the Lagrangian

$$\mathcal{L}(\theta, \dot{\theta}, t) = e^{\beta(t-t_0)}\left(\frac{1}{2}\|\dot{\theta}\|^2 - \frac{\gamma\beta}{\mathcal{N}_t}L_t(\theta)\right). \tag{3.9}$$

The benefits of the HT in (3.8) are that (a) it can be guaranteed to be stable even with time-varying regressors for a dynamic error model with a scalar control input [21], and (b) a particular discretization was shown in [24] to lead to an accelerated algorithm which reaches an ϵ sub-optimal point in $\tilde{\mathcal{O}}(1/\sqrt{\epsilon})$ iterations for a linear regression-type convex loss function with constant regressors, as compared to the

$\mathcal{O}(1/\epsilon)$ guaranteed rate for the standard gradient descent algorithm. As our goal here is to achieve fast real-time control, we employ elements of HT proposed in [21] in the AC part of the control design.

Chapter 4

Adaptive Control and Reinforcement Learning Algorithm (AC-RL)

4.1 Problem Statement

Consider a continuous-time, deterministic nonlinear system described by the following dynamics [1]:

$$\dot{X}(t) = F(X(t), U(t)) \tag{4.1}$$

where $X(t) \in \mathbb{R}^n$, and $U(t) \in \mathbb{R}^m$. We define $x = X - X_0$ and $u = U - U_0$, where (X_0, U_0) is an equilibrium point[2], i.e., $F(X_0, U_0) = 0$.Using a Taylor series expansion on (4.1) yields

$$\dot{x} = Ax + Bu + f(x, u) \tag{4.2}$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, and (A, B) is controllable. It should be noted that (4.1) can always be written in the form of (4.2) for any analytic function F. The following assumption is made about f:

Assumption 2 *The higher order effects represented by the nonlinearity $f(x, u)$ in*

[1]In previous sections detailing reinforcement learning methods, states and actions were denoted as s and a respectively. In order to keep with classical control conventions, the following notation will be used in the subsequent sections: x will refer to the state of a dynamical system and u will refer to the control input (analagous to an action in RL).

[2]In most of what follows, we suppress the argument t for ease of exposition.

(4.2) *(a) lies in the span of B and (b) are solely a function of the state x, i.e. the system in* (4.2) *is control affine and $g(x)$ exists such that $Bg(x) = f(x, u)$:*

$$\dot{x} = Ax + B[u + g(x)] \tag{4.3}$$

The goal then is the determination of the control input $u(t)$ in real-time, when parametric uncertainties are present in the system dynamics in (4.3), so as to minimize the cost function

$$\min_{u(t)\in\mathcal{U},\ \forall t\in[0,T]} \quad \int_0^T c(x(t), u(t), t)dt \tag{4.4}$$

subject to the dynamics in (4.3), where $\mathcal{U}$ represents the action space and c is a bounded cost function [91].

4.2 Underlying Tools

4.2.1 An Offline Approach Based on Reinforcement Learning

Control policies are determined using RL through offline training in simulation, which implies that the system in (4.2) needs to be fully known and accessible by the simulator during the offline training. We rewrite (4.2) as

$$\dot{x}_r = Ax_r + Bu_r + f_r(x_r, u_r) \tag{4.5}$$

where the subscript r is used to denote signals and parameters belonging to the reference system. We briefly describe the offline training procedure: First, it is assumed that the continuous-time dynamics in (4.5) are sampled with sufficient accuracy, resulting in the discrete time dynamics:

$$x_{r,k+1} = h(x_{r,k}, u_{r,k}) \tag{4.6}$$

An appropriate numerical integration scheme ensures that this discrete-time formulation closely approximates the dynamics. When training in simulation the RL process begins with the construction of a simulation environment defined by (4.6). The reference system in (4.6) is used to collect data with which RL learns a feedback policy $u_{r,k} = \pi(x_{r,k})$ with repeated training of $\pi(\cdot)$ so as to achieve the control objective of (4.4) [32] as follows.

At each timestep, an observation $x_{r,k}$ is received, a control $u_{r,k}$ is chosen, and the resulting cost $c_k = c(x_{r,k}, u_{r,k})$ is received. Repeating this process, a set of input-state-cost tuples $\mathcal{D} = [(x_{r,1}, u_{r,1}, c_1), \dots, (x_{r,N}, u_{r,N}, c_N)]$ is formed. This data is used to train and update the policy π [32]. In many RL algorithms the policy is parametric, often using neural networks, so that $u_k = \pi_\theta(x_k)$. The learning algorithm then seeks to adjust θ so that the expected accumulated cost is minimized [74, 90], i.e

$$\min_\theta \quad J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{k=0}^{T} c_k\right] \tag{4.7}$$

In this work, we use the Proximal Policy Optimization (PPO) and Deep Deterministic Policy Gradient RL algorithms [75] to train the policy π which constitutes the RL portion of the proposed AC-RL controller.

4.2.2 The Classical Online Approach Based on Adaptive Methods

We start with an error model description of the form

$$\dot{e}(t) = A_m e(t) + B\Lambda\left[\widetilde{\Theta}(t)\Phi(t)\right] \tag{4.8}$$

where $e(t) \in \mathbb{R}^n$ is a performance error that is required to be brought to zero. Tracking error, identification error, or state estimation error are a few examples. $\widetilde{\Theta}(t) \in \mathbb{R}^{m\times l}$ is a matrix of parameter errors that quantify the learning error. If the true parameter in the system dynamics is $\Theta^* \in \mathbb{R}^{m\times l}$, and it is estimated as $\widehat{\Theta}(t)$ at time t, then $\widetilde{\Theta} = \widehat{\Theta} - \Theta^*$. A_m is a Hurwitz matrix, A_m and B are known with (A_m, B) controllable, and $\Lambda \in \mathbb{R}^{m\times m}$ is an unknown parameter matrix that is positive definite. Finally $\Phi(t) \in \mathbb{R}^l$ is a vector of regressors that correspond to all real-time information measured or computed from the system dynamics and controllers at time t. Such an error model is ubiquitous in adaptive control of a large class of nonlinear dynamic systems with parametric uncertainties [56] including that in (4.3). The following theorem summarizes the standard adaptive control result in the literature:

Theorem 1 *Let $\Gamma \in \mathbb{R}^{m\times m}$ and Q be symmetric positive-definite matrices with P corresponding to the solution of the Lyapunov equation*

$$A_m^T P + PA_m = -Q. \tag{4.9}$$

An adaptive law that adjusts the parameter error as

$$\dot{\widetilde{\Theta}} = -\Gamma B^T P e \Phi^T \tag{4.10}$$

guarantees that $e(t)$ and $\widetilde{\Theta}(t)$ are bounded for any initial conditions $e(0)$ and $\widetilde{\Theta}(0)$. If in addition $\Phi(t)$ is bounded for all t, then $\lim_{t\to\infty} e(t) = 0$.

In addition to closed-loop stability and tracking performance, we also use Regret to quantify the performance of the proposed controllers, defined as in [25], as

$$\mathcal{R} = \int_0^T e^T(\tau) Q e(\tau) d\tau \tag{4.11}$$

where Q is a positive definite matrix. It is clear that for the classical AC approach, $\mathcal{R} = O(1)$, i.e., independent of T. Decaying exponential signals can also be included in (4.11) [25].

4.2.3 Persistent Excitation and Parameter Learning

Definition 1 (Persistent Excitation) *[56] A bounded function $\Phi : [t_0, \infty) \to \mathbb{R}^N$ is persistently exciting (PE) if there exists $T > 0$ and $\alpha > 0$ such that*

$$\int_t^{t+T} \Phi(\tau)\Phi^T(\tau) d\tau \geq \alpha I, \quad \forall t \geq t_0. \tag{4.12}$$

The definition of PE in (4.12) is equivalent to the following:

$$\frac{1}{T}\int_t^{t+T} \left|\Phi(\tau)^T w\right| d\tau \geq \epsilon_0, \forall \text{ unit vectors } w \in \mathbb{R}^N, \forall t \geq t_0 \tag{4.13}$$

for some $\epsilon_0 > 0$. In what follows, we will utilize an alternate definition, which is equivalent to (4.12) and (4.13) if $\|\dot{\Phi}(t)\|$ is bounded for all t [50]:

Definition 2 Φ *is PE if there exists an* $\epsilon > 0$ *a* t_2 *and a sub-interval* $[t_2, t_2 + \delta_0] \subset [t, t+T]$ *with*

$$\frac{1}{T}\left|\int_{t_2}^{t_2+\delta_0} \Phi(\tau)^T w d\tau\right| \geq \epsilon_0, \forall \textit{ unit vectors } w \in \mathbb{R}^N, \forall t \geq t_0. \tag{4.14}$$

The goal of this paper is to find solutions not only for real-time control but also for learning the unknown parameters. Adaptive control systems enable parameter learning by imposing properties of persistent excitation defined in Section 4.2.3. We briefly summarize the classical result related to this topic, which was first established for continuous-time systems in 1977 in [50, 51]. The starting point is the same error model as in (4.10), which contains two errors, $e(t) \in \mathbb{R}^n$ is a performance error that can be measured, and $\widetilde{\theta}^T(t) = \widetilde{\Theta}(t) \in \mathbb{R}^{1\times l}$ is a parameter learning error. As before we assume that A_m is known and Hurwitz, and B is known. The following theorem summarizes this result:

Theorem 2 ([51]) *The solutions of the error dynamics in* (4.8) *together with the adaptive law in* (4.10) *lead to* $\lim_{t\to\infty} \widetilde{\theta}(t) = 0$ *if the regressor* Φ *satisfies the PE condition in Definition 2.*

It should be noted that the PE condition in Definition 2 is stronger than that required in Definition 1 which is for error models that have an algebraic relation between the performance error and the parameter error, and is equivalent if the regressor $\Phi(t)$ is smooth.

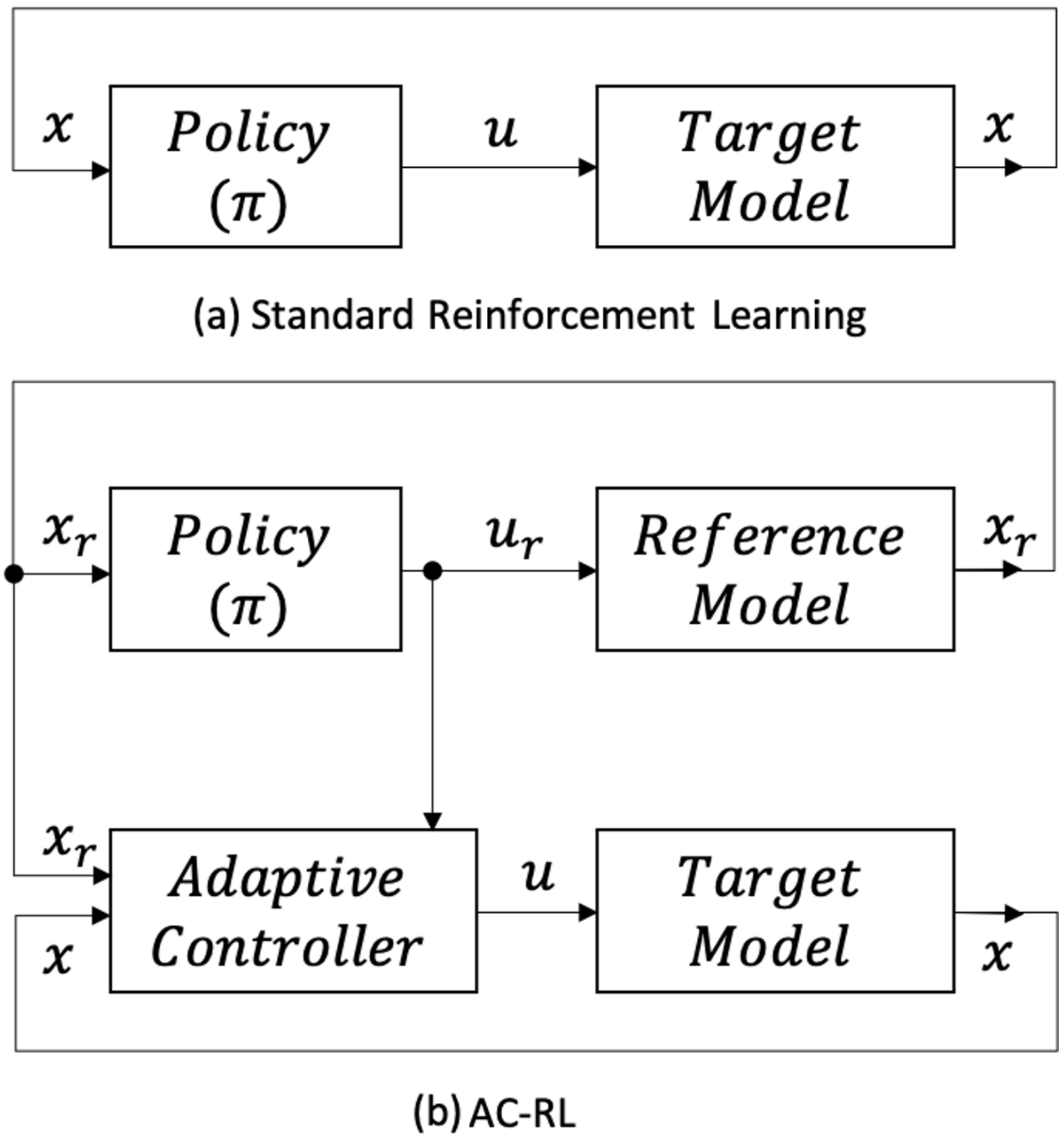

Figure 4-1: RL vs. AC-RL. (a) represents a standard application of a trained policy, in which the trained policy is inserted directly into the target system: $u = \pi(x)$. (b) shows how AC-RL is used. The policy is inserted into the reference system, producing $u_r = \pi(x_r)$. The MRAC update laws are then used to calculate u.

4.3 AC-RL Algorithm

4.3.1 Model Reference Adaptive Control

As the problem we will address in this paper considers the effect of parametric uncertainties in (4.3), we shall denote (4.3) when there are no parametric uncertainties as the reference system, and express it as

$$\dot{x}_r = Ax_r + B[u_r + g(x_r)] \tag{4.15}$$

We assume that u_r is designed using RL:

$$u_r = \pi(x_r, t) \tag{4.16}$$

so that for all bounded exogenous inputs, u_r ensures that x_r is bounded, and is such that (4.4) is accomplished. It should be noted that the dependence of the policy π on t represents the effect of all exogenous inputs needed to accomplish the control objective (e.g, a reference trajectory). It should also be noted that $\pi(x_r, t)$ in (4.16) does not necessarily cancel $g(x_r)$, but accommodates it so that the closed-loop system behaves in a satisfactory manner for the requisite control task. This is quantified in Assumption 3, a desirable property of RL controllers.

Assumption 3 *RL is used to train a feedback policy $\pi : x \to u$ such that for a given positive constant R_2, a positive constant R_1 exists such that $||x_r(0)|| \leq R_1$ implies $||x_r(t)|| \leq R_2\ \forall t$.*

Parametric uncertainties

We now introduce the following assumption to address parametric uncertainties.

Assumption 4 *The higher order term in* (4.3) *is parameterized linearly, i.e., $g(x) = \Theta_{n,r}\Phi_n(x)$ where $\Theta_{n,r} \in \mathbb{R}^{m\times l}$ and $\Phi_n(x) \in \mathbb{R}^l$.*

Assumption 4 implies that the nonlinearities in (4.3) can be approximated using l basis functions. In what follows, we consider two dominant sources of uncertainties, one in the form of control effectiveness, i.e., u gets perturbed as Λu, and the second as a perturbation in $g(x)$ from $\Theta_{n,r}\Phi_n(x)$ to $\Theta_n^{'}\Phi_n(x)$. The plant equation in (4.3) then becomes

$$\dot{x} = Ax + B\Lambda\left[u + \Lambda^{-1}\Theta_n^{'}\Phi_n(x)\right] \tag{4.17}$$

where $B \in \mathbb{R}^{n\times m}, \Lambda \in \mathbb{R}^{m\times m}$, and $\Theta_n' \in \mathbb{R}^{m\times l}$. Loss of control effectiveness is ubiquitous in practical problems, due to unforeseen anomalies that may occur in real-time, such as accidents or aging in system components, especially in actuators. Parametric uncertainties in the nonlinearity $g(x)$ may be due to modeling errors. We note that the problem is the control of (4.17) where A, B, and $\Phi_n(x)$ are known, but Λ and Θ_n' are unknown parameters.

4.3.2 The AC-RL controller

Suppose that an AC-RL control input is designed as

$$u = u_r + u_{ad} \tag{4.18}$$

whose goal is to ensure that in the presence of the aforementioned perturbations in Λ and Θ_n' the true system state x converges to the reference system state x_r. Subtracting (4.15) from (4.17), we obtain the error dynamics:

$$\begin{aligned} \dot{e} \;=\; & A_H e + B\Lambda[u_{ad} + (I - \Lambda^{-1})u_r + \Lambda^{-1}\Theta_n^{'}\Phi_n(x) \\ & - \Lambda^{-1}(g(x_r) + \Theta_{l,r}e)] \end{aligned} \tag{4.19}$$

where $e \coloneqq x - x_r$ and $\Theta_{l,r}$ is such that $A_H \coloneqq A + B\Theta_{l,r}$ is Hurwitz. The bracketed terms in (4.19) can be rearranged into:

$$u_{ad} + u_r - \Lambda^{-1}\left(u_r + g(x_r) + \Theta_{l,r}e\right) + \Lambda^{-1}\Theta_n^{'}\Phi_n(x)$$

leading to the compact error dynamics

$$\dot{e} = A_H e + B\Lambda[u - \Theta\Phi] \tag{4.20}$$

where

$$\Theta := \left[\Lambda^{-1},\ -\Lambda^{-1}\Theta_n'\right] \qquad \Phi := \begin{bmatrix} \Phi_r(u_r, x_r, x) \\ \Phi_n(x) \end{bmatrix} \tag{4.21}$$

and

$$\Phi_r(u_r, x_r, x) = u_r + g(x_r) + \Theta_{l,r} e \tag{4.22}$$

In (4.21), $\Theta \in \mathbb{R}^{m\times(m+l)}$ corresponds to an unknown parameter matrix and $\Phi(t) \in \mathbb{R}^{m+l}$ is the regressor vector used for adaptation. The error equation (4.20) is central to the development of the AC-RL algorithms derived in the following subsections.

The AC-RL approach is exemplified by several key features:

(i) Two different regressors are employed in the adaptive control input: Φ_r in (4.22) and Φ_n, which are utilized to address the two different sources of parametric uncertainties Λ and Θ_n. The first regressor component, Φ_r, comes predominantly from the closed RL system: u_r and $g(x_r)$ come from the RL policy and reference environment, respectively. $B[u_r + g(x_r)]$ in (4.15) can be viewed as a variable determined by an oracle; this could potentially be accomplished by monitoring the entire vector field $\dot{x}_r$ and subtracting the linear part Ax. The second regressor accommodates the uncertainty Θ_n' in the nonlinear component $g(x)$ (and employs assumption 4).

(ii) The additional feedback from e in (4.22) is essential in guaranteeing global stability.

(iii) In contrast to the standard AC formulation, the AC-RL controller allows the use of a nonlinear reference model as in (4.15) and a nonlinear controller based on RL as in (4.16).

We now propose the AC-RL controller.

$$u = \widehat{\Theta}(t)\Phi(t) \tag{4.23}$$

$$\dot{\Xi} = -\gamma B^T P e \Phi^T \tag{4.24}$$

$$\dot{\widehat{\Theta}} = -\beta(\widehat{\Theta} - \Xi)\mathcal{N}_t, \tag{4.25}$$

where

$$\mathcal{N}_t = 1 + \mu \Phi^T \Phi \tag{4.26}$$

$$\mu \geq \frac{2\gamma}{\beta}\|PB\|_F^2 \tag{4.27}$$

$\|\cdot\|_F$ in (4.27) denotes the Frobenius matrix norm, and $P = P^T \in \mathbb{R}^{n\times n}$ is a positive definite matrix that solves the equation $A_H^T P + PA_H = -Q$, where Q is a positive-definite matrix and $Q \geq 2I$. It should be noted that (4.24) - (4.25) is a second-order tuner, and an extension of earlier results in [21] to the multivariable case. We also note that our choice of RL in the outer-loop is due to its abilities to be agnostic to the model structure, incorporation of general optimization costs, and superior computational efficiency, in contrast to methods such as MPC. In comparison to a purely AC approach the AC-RL controller can be viewed as one where the RL plays the role of a reference model that is nonlinear with a controller that is nonlinear as well and trained so as to elicit desirable properties from the reference system. The following theorem presents the stability property of the AC-RL, which requires the following additional assumption:

Assumption 5 *Λ is symmetric and positive definite, with $\|\Lambda\| \leq 1$.*

Theorem 3 *Under Assumptions 2-5, the closed-loop adaptive system specified by the plant in* (4.17)*, the reference system in* (4.15)*, and the adaptive controller in* (4.23)-(4.25) *leads to globally bounded solutions with* $\lim_{t\to\infty} e(t) = 0$ *with* $\mathcal{R} = O(1)$*.*

It is clear from (4.23) - (4.25) that if there are no parametric uncertainties, and if the initial conditions of (4.17) are identical to those of (4.15), then the choices of

$\widehat{\Theta}(0) = [I, \Theta_{n,r}^T]^T$ and $\Xi(0) = 0$ ensures that the AC-RL control $u(t)$ coincides with the RL-input, thereby accomplishing (4.4). When there are parametric uncertainties, the control input needs to be modified from (4.16) as in (4.23). Theorem 1 guarantees that with this input, the closed-loop system state x converges to x_r satisfying (4.4) in the limit. Additional smoothness conditions on Φ have to be imposed to show that $\widehat{\Theta}$ goes to a constant. Denoting this constant as Θ_{ss}, it follows that the parametric uncertainties are accommodated by the online AC-RL policy by replacing the offline policy u_r, which may be in the form of $\Theta_{n,r}\Phi(x)$, by $u_r + u_{ad}$ as in (4.23), which is of the form $\Theta_{ss}\Phi(x)$.

4.3.3 Validity of the AC-RL controller

The RL and AC components of the AC-RL controller imposed distinctly different requirements on the system (4.1). The AC controller in (4.23)-(4.25) required that the system be expanded as in (4.2), and imposed Assumptions 1 and 4 to bring (4.2) into the form (4.17). The RL controller in (4.16) required none of these assumptions, but that the reference system be amenable to a simulation experiment offline, with u_r determined through training, as quantified in Assumption 3. We explore the benefits of combining these two methods by evaluating the implications of these assumptions and requirements on the original system (4.2).

We note that the effect of the approximations due to Assumptions 1 and 4 can be expressed by considering (4.2) in the presence of uncertainties Λ and Θ_n' given by

$$\dot{x} = Ax + B\Lambda u + f'(x, u) \tag{4.28}$$

and expressing the nonlinearity $f'(x, u)$ as:

$$f'(x, u) = B\Theta_n'\Phi(x) + \epsilon(x) \tag{4.29}$$

where $\epsilon(x)$ represents the approximation error due to assumptions 1 and 2.

We assume that for a given non-negative bound M, basis functions $\Phi(x)$ can be chosen such that

$$||\epsilon(x)|| \leq M \qquad \forall x \in S(M) \tag{4.30}$$

where $S(M)$ is a compact set in $\mathbb{R}^n$. It can be shown that a compact set S_0 exists such that the closed-loop system with the AC-RL controller guarantees that for all $x(0) \in S_0$, $x(t) \in S(M)$. This is established via the following: Let

$$||e(0)|| \leq X_0 + R_1 \tag{4.31}$$

where $X_0 = \max_{x \in S_0} ||x||$ with R_1, R_2 given by Assumption 3. From the Lyapunov function, we have

$$||e(t)|| \leq (k_0 + X_0 + R_1)\frac{\sigma_{max}(P)}{\sigma_{min}(P)} \tag{4.32}$$

where k_0 is the maximum bound on the parameter estimates. Finally, denoting $E_0 = (k_0 + X_0 + R_1)\frac{\sigma_{max}(P)}{\sigma_{min}(P)}$, we have

$$||x(t)|| \leq E_0 + R_2 \; \forall t \geq t_0 \tag{4.33}$$

The bound k_0 can be shown to exist with a projection operator added to the HT laws using tools in [23, 24]. Let $R_M = \max_{x \in S(M)} ||x||$. If $E_0 + R_2 \leq R_M$, there exists a compact set S_0 such that the closed-loop AC-RL system solutions $x(t)$ for the original system in (4.2) are guaranteed to lie inside $S(M)$. Although Assumption 3 guarantees that $||x_r(t)|| \leq R_2$ it provides no such statement about the states $x(t)$ in (4.29). Therefore, if RL is applied without AC one cannot make any claim on the boundedness of $x(t)$. In comparison, AC-RL guarantees that $||x(t)|| \leq E_0 + R_2$.

We note that the condition that $E_0 + R_2 \leq R_M$ trivially holds when there is no approximation error, i.e $||\epsilon(x)|| = 0$ which in turn implies that $R_M = \infty$. Therefore AC-RL guarantees the boundedness of $x(t)$ $\forall t \geq t_0$. In comparison, direct application of the RL policy π to the target system cannot provide any such guarantee.

In summary, the actual solutions of the closed-loop system with the AC-RL can have a bound that is as large as $E_0 + R_2$, whereas the RL-based controller leads to a solution that is less than R_2. Therefore it is clear that the addition of the AC in the inner-loop allows an online policy that accommodates a larger compact set.

4.3.4 AC-RL with Magnitude Saturation

As control inputs are often subject to magnitude saturation, we propose another AC-RL controller: Magnitude Saturation AC-RL (MSAC-RL), which builds on the ideas introduced in [33]. The saturated control input into the true plant is calculated as:

$$u_i(t) = u_{i,\max}\text{sat}\left(\frac{u_{i,c}(t)}{u_{i,\max}}\right) \tag{4.34}$$

where $u_c(t)$ denotes the output of the controller and $u_{i,\max}$ is the allowable magnitude limit on u_i. This induces a saturation-triggered disturbance Δu vector defined by

$$\Delta u(t) = u_c(t) - u(t) \tag{4.35}$$

It is easy to see that $\Delta u(t) = 0$ when the desired control $u_c(t)$ does not saturate. The output of the controller, u_c, is given by:

$$u_c = \widehat{\Theta}(t)\Phi(t) \tag{4.36}$$

The presence of the disturbance Δu causes the error equation to vary from (4.20) to

$$\dot{e} = A_H e + B\Lambda[u_c - \Delta u - \Theta\Phi] \tag{4.37}$$

We introduce a new performance error e_a in order to accommodate the disturbance Δu as follows:

$$\dot{e}_a = A_H e_a + BK_a(t)\Delta u \tag{4.38}$$

which leads to a new augmented error $e_u = e - e_a$:

$$\dot{e}_u = A_H e_u + B\Lambda\widetilde{\Theta}(t)\Phi(t) + B(\Lambda - K_a^T)\Delta u \tag{4.39}$$

This suggests a different set of adaptive laws,

$$\begin{aligned} \dot{\Xi} &= -\gamma B^T P e_u \overline{\Phi}^T && (4.40) \\ \dot{\overline{\Theta}} &= -\beta(\overline{\Theta} - \Xi)\mathcal{N}_t, && (4.41) \end{aligned}$$

where $\overline{\Theta} = [\widehat{\Theta}^T, -K_a^T]^T$ and $\overline{\Phi} = [\Phi^T, \Delta u^T]^T$, γ and β are positive constants, and $\mathcal{N}_t$ defined as in (4.26) with Φ replaced by $\overline{\Phi}$ and μ as in (4.27). The following theorem provides the analytical guarantees for this MSAC-RL controller.

Theorem 4 *Under Assumptions 2-5, the closed-loop adaptive system specified by the plant in* (4.17)*, the reference system in* (4.15)*, the magnitude constraint in* (4.34) *and the MSAC-RL controller given by* (4.36)*, (4.40) and* (4.41) *leads to*

(i) globally bounded solutions, with $\lim_{t\to\infty} \|e_u(t)\| = 0$ *if the target system in* (4.17) *is open-loop stable.*

(ii) bounded solutions for all initial conditions $x(0)$*,* $\Xi(0)$*,* $\widehat{\Theta}(0)$ *and* $K_a(0)$ *in a bounded domain, with* $\|e(t)\| = O[\int_0^t \|\Delta u(\tau)\| d\tau]$ *if the target system in* (4.15) *is not open-loop stable.*

It is clear that the performance guarantees of the closed-loop system in Theorems 3-4 approximate the online control goal stated in (4.4) for the case when the cost function does not depend on the control input, as $x(t)$ approaches $x_r(t)$ and the choice of $u_r(t)$ optimizes the behavior of the reference system. Any dependence on the control input is implicitly addressed in MSAC-RL by accommodating magnitude saturation. More remains to be done in bridging this optimization gap, a topic for future research.

4.3.5 Extension to multiple equilibrium points

Suppose the system in (4.1) has p equilibrum points $(X_1, U_1), \ldots, (X_p, U_p)$, so that $F(X_i, U_i) = 0$ for $i = 1, \ldots, p$. Define $x_i = x - X_i$, $u_i = u - U_i$. Denoting the state and action spaces as $\mathbb{X}$ and $\mathbb{U}$, respectively, partition the composite domain $\mathbb{D} = \mathbb{X} \times \mathbb{U}$ into p disjoint subsets $S_1, \ldots, S_p$, such that:

$$\mathbb{D} = S_1 \cup S_2 \cup \ldots S_p$$
$$S_i \cap S_j = \emptyset, i \neq j$$

One can then express (4.1) in the presence of parametric uncertainties as

$$\dot{x} = \begin{cases} A_1 x_1 + B_1[\Lambda u_1 + g(x_1)], & x_1, u_1 \in S_1 \\ & \vdots \\ A_p x_p + B_p[\Lambda u_p + g(x_p)], & x_p, u_p \in S_p. \end{cases} \tag{4.42}$$

If we now consider the effect of parametric uncertainties in u_i and $g(x_i)$ as in Section III-A, it is easy to have a switching set of controllers as in (4.23)-(4.25) that are invoked when the trajectories enter the set S_i. A corresponding stability result to Theorem 4 can be derived, provided the command signal is such that the dwell time in each set S_i exceeds a certain threshold, which is not discussed here. We summarize the overall AC-RL controller in Algorithm 1, which specifies the discrete time implementation of the overall AC-RL Controller. Δt denotes the integration timestep and the AdaptiveControl function corresponds to the adaptive control input; in the single equilibrium case, this function corresponds to equations (4.23)-(4.27). F and F_r represent the velocity vector fields of the target and reference dynamical systems, respectively. That is, F and F_r are the right hand sides of equations (4.2) and (4.5), respectively.

4.3.6 Extensions to a class of nonaffine systems

We once again start with (4.1), expand the dynamics around (X_0, U_0), which together with Assumption 1 leads to a class of nonlinear systems

$$\dot{x} = Ax + B[u + h(x, u)] \tag{4.43}$$

Algorithm 1 Multiple Equilibrum AC-RL

```
Require: $F, F_r, \pi, x_r, x$
  while running do
    $u_r = \pi(x_r)$
    $x_i \leftarrow x - X_i$
    $u_i \leftarrow u - U_i$
    $e \leftarrow x_i - (x_r - X_i)$
    $\Phi \leftarrow [x_i, u_i]$
    $u \leftarrow$AdaptiveControl$(\Phi, e, \hat{\Theta}_i, P_i, B_i) + U_i$
    $x_r \leftarrow x_r + F_r(x_r, u_r)\Delta t$
    $x \leftarrow x + F(x, u)\Delta t$
    $i \leftarrow j : x, u \in S_j$
  end while
```

Noting that u is required to minimize the cost functional subject to the constraints specified in (4.4), we assume that u can be expressed as an analytic function of the state x, which in turn leads to the assumption that

$$h(x, u) = g(x, t) \tag{4.44}$$

The nonautonomous nature in (4.44) is due to the cost functional dependent on time which in turn may be due to an objective that may vary with t. The goal is to determine u in real time when parametric uncertainties are present in (4.43)-(4.44) so that (4.4) is accomplished for initial condition x_0. Denoting the case when there are no parametric uncertainties as the reference system, we obtain its dynamics to be

$$\dot{x}_r = Ax_r + B[u_r + g(x_r, t)] \tag{4.45}$$

We proceed to make a similar assumption as in Section 4.3.1 regarding the higher order term:

Assumption 6 *$g(x, t)$ is parameterized linearly, i.e., $g(x, t) = \Theta_{nl,r}\Phi_{nl}(x, t)$ where $\Phi_{nl}(x, t) \in \mathbb{R}^l$.*

Similar to Assumption 4, Assumption 6 implies that the nonlinearities of the affine nonlinear system (4.3) are confined to the first l higher-order terms, with additional nonautonomous components due to the nonaffine nature of the dynamics.

As in Section 4.3.1, we assume that the plant dynamics in (4.43)-(4.44) are subject to two parametric uncertainties, with u perturbed as Λu, and $g(x,t)$ from $\Theta_{nl,r}\Phi_{nl}(x,t)$ to $\Theta_{nl}^{'}\Phi_{nl}(x,t)$, which causes the plant equation to become

$$\dot{x} = Ax + B\Lambda[u + \Lambda^{-1}\Theta_{nl}^{'}\Phi_{nl}(x)] \tag{4.46}$$

As the structure of the plant dynamics in (4.46) is almost identical to that in (4.17), the development of the AC-RL controller and its control solution is identical to the descriptions above and are summarized in the following subsection.

4.3.7 Learning in AC-RL controllers with persistent excitation

Sections IV-A through IV-E focused on the control solution, i.e. the solution to the problem in (4.4). The AC-RL controller in (4.23)-(4.25) guaranteed that the resulting solution $x(t)$ of the closed loop system converged to the nominal reference solution $x_r(t)$ with the AC-RL policy $u(t)$ converging to the RL policy $u_r(t)$. In this section, we return to the learning problem, i.e. the conditions under which the parameter estimate $\widehat{\Theta}$ converges to the true parameter Θ with the AC-RL algorithm. We limit our discussion to the case when $u(t)$ is a scalar. A few remarks follow regarding its extension to the multi-input case, which falls outside the scope of this paper.

The starting point is the error equation in (4.20) and the AC-RL control input in (4.23). We note that with a scalar input, $m = 1$, which leads to $\Lambda \in \mathbb{R}^+$ (using Assumption 5), $\Theta \in \mathbb{R}^{1\times(l+1)}$, $\Phi_r(t) \in \mathbb{R}$, and $\Phi_n(t) \in \mathbb{R}^l$ in (4.21). Similar to Assumption 5, we assume that $0 < \Lambda < 1$. This in turn leads to the error equation

$$\dot{e} = A_H e + B\Lambda[(\widehat{\Theta} - \Theta)\Phi] \tag{4.47}$$

and the adaptive laws (4.24)-(4.25). The following theorem establishes the conditions under which $\lim_{t\to\infty} \widetilde{\Theta}(t) = 0$.

Defining $\tilde{\theta} = (\widehat{\Theta} - \Theta)^T$, $\tilde{\vartheta} = (\Xi - \Theta)^T$ and $B_0 = B\Lambda$, we rewrite (4.47), (4.24), and (4.25) as

$$\dot{e} = A_H e + B_0 \tilde{\theta}^T \Phi \tag{4.48}$$

and

$$\dot{\tilde{\vartheta}} = -\gamma \Phi e^T P B \tag{4.49}$$

$$\dot{\tilde{\theta}} = -\beta(\tilde{\theta} - \tilde{\vartheta})\mathcal{N}_t \tag{4.50}$$

where $\mathcal{N}_t = 1+\mu\Phi^T\Phi$, $\mu \geq 2\gamma\|PB\|^2/\beta$. As before, the matrix P solves $A_H^T P + P A_H = -Q$ and $Q \geq 2I$ is a symmetric, positive-definite matrix. The constants γ and β are positive. Let $x_1(t) = [e(t)^T, (\tilde{\theta}(t) - \tilde{\vartheta}(t))^T]^T$ and $z(t) = [x_1(t)^T, \tilde{\vartheta}(t)^T]^T$.

We note that the stability result related to the AC-RL controller stated in Theorem 4 guarantees that $z(t)$ is uniformly bounded, which follows from a Lyapunov function of the form

$$V = \frac{\Lambda}{\gamma}\tilde{\vartheta}^T\tilde{\vartheta} + \frac{\Lambda}{\gamma}\left[(\tilde{\vartheta} - \tilde{\theta})^T(\tilde{\vartheta} - \tilde{\theta})\right] + e^T P e \tag{4.51}$$

whose derivative is bounded by

$$\dot{V} \leq -\frac{2\beta\Lambda}{\gamma}\|\tilde{\theta} - \tilde{\vartheta}\|^2 - \Lambda\|e\|^2 \tag{4.52}$$

Noting that our goal is to show that $\lim_{t\to\infty}\tilde{\theta}(t) = 0$, it suffices to show that $V(t) \to 0$ as $t \to \infty$. As the time-derivative $\dot{V}$ in (4.52) is negative-definite only in x_1, the goal cannot be accomplished in a straight forward manner. We show below that this is indeed possible under conditions of persistent excitation, and corresponds to the second contribution of this paper, which is parameter learning using the AC-RL controller.

The following lemmas are useful in proving the main result, which is stated in Theorem 5. We note that Theorem 4 guarantees that $z(t)$ and $\Phi(x, e, u_r)$ is bounded, making the properties of persistent excitation applicable for what follows.

Lemma 1 *Let $\epsilon_1 > \epsilon_2 > 0$, then there is an $n = n(\epsilon_1, \epsilon_2)$ such that if $z(t) =$*

$\left[x_1(t)^T, \tilde{\vartheta}(t)^T\right]^T$ *is a solution with* $\|z(t_1)\| \leq \epsilon_1$ *and* $S = \{t \in [t_1, \infty) | \|x_1(t)\| > \epsilon_2\}$, *then* $\mu(S) \leq n$ *where* μ *denotes Lebesgue measure.*

Lemma 2 *Let* $\delta > 0$ *and* $\epsilon_1 > 0$ *be given. Then there exist positive numbers* ϵ *and* T *such that if* $z(t)$ *is a solution with* $\|z(t_1)\| \leq \epsilon_1$ *and if* $\left\|\tilde{\vartheta}(t)\right\| \geq \delta$ *for* $t \in [t_1, t_1 + T]$, *then there exists a* $t_2 \in [t_1, t_1 + T]$ *such that* $\|x_1(t_2)\| \geq \epsilon$.

Lemma 3 *Let* ϵ_1 *and* δ *be given positive numbers. Then there is a* $T = T(\epsilon_1, \delta)$ *such that if* $z(t)$ *is a solution and* $\|z(t_1)\| \leq \epsilon_1$, *then there exist some* $t_2 \in [t_1, t_1 + T]$ *such that* $\left\|\tilde{\vartheta}(t_2)\right\| \leq \delta$.

Theorem 5 *If* $\Phi(t)$ *satisfies the persistent excitation property in* (4.14), *then the origin* $(e = 0, \tilde{\vartheta} = 0, \tilde{\theta} = 0)$ *in* (4.48)-(4.50) *is uniformly asymptotically stable.*

The proof of Theorem 5 stems from the three lemmas listed above, which represent the three main steps. Lemmas 1 and 2 establish that $x_1(t)$ cannot remain small over the entire period of persistent excitation. Lemma 3 then leverages this fact to show that this leads to the parameter error $\tilde{\vartheta}(t)$ to decrease. Together this allows the conclusion of u.a.s. of the origin (4.48)-(4.50) and therefore that $\lim_{t\to\infty} \tilde{\theta}(t) = 0$. As is apparent from the details of the proof provided in the Appendix, first principles based arguments had to be employed in order to derive this result. No standard observability properties or time-scale transformations as in [60] have been employed; these are inadequate as the error model structure in (4.48) includes system dynamics and no filtering techniques are used to convert the error model to a static linear regression model.

Chapter 5

Numerical Validation

We validate the AC-RL controller with the MSAC-RL algorithm using a simulated task. The task requires a quadrotor moving in 3-D to land on a moving platform, assuming full-state feedback. A sparse reward function is chosen in which positive reward (negative cost) is attained only when the quadrotor enters a relatively small compact set in the state-space. Negative reward (positive cost) is attained if the quadrotor altitude falls below the platform altitude, or if a termination time of 15 seconds is reached.

5.1 Quadrotor Dynamics

We first describe the quadrotor dynamical model. The squared angular velocities of each propeller are used as input, that is: $u = [\omega_1^2, \omega_2^2, \omega_3^2, \omega_4^2]^T$. The thrust produced by each propeller is calculated by multiplying the squared angular speed by a propeller specific constant κ. Letting $K = diag(\kappa_1, \kappa_2, \kappa_3, \kappa_4)$, the vector of thrusts produced by each propeller is then given by Ku. Finally, denoting the (body-frame) vertical force, roll moment, pitch moment, and yaw moment by $f_z, \tau_\phi, \tau_\theta, \tau_\psi$ respectively, we have:

$$\begin{bmatrix} f_z \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ L & 0 & -L & 0 \\ 0 & L & 0 & -L \\ \mu & -\mu & \mu & -\mu \end{bmatrix} Ku \tag{5.1}$$

where L is the distance from each propeller to the quadrotor center of mass, and μ is a rotational drag constant. Assuming low-speeds, we may then construct a simple rigid-body model for the quadrotor dynamics:

$$\begin{aligned} \ddot{x} &= (\cos\phi\cos\theta\cos\psi + \sin\phi\sin\psi)\frac{f_z}{m} \\ \ddot{y} &= (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)\frac{f_z}{m} \\ \ddot{z} &= \cos\phi\cos\theta\frac{f_z}{m} - g \\ \ddot{\phi} &= \dot{\theta}\dot{\psi}(\frac{I_y - I_z}{I_x}) + \frac{L}{I_x}\tau_\phi \\ \ddot{\theta} &= \dot{\phi}\dot{\psi}(\frac{I_z - I_x}{I_y}) + \frac{L}{I_y}\tau_\theta \\ \ddot{\psi} &= \dot{\phi}\dot{\theta}(\frac{I_x - I_y}{I_z}) + \frac{1}{I_z}\tau_\psi \end{aligned} \tag{5.2}$$

where $x,\ y,\ z$ represent the center of mass position in an inertial frame and ϕ, θ, ψ are the roll, pitch, and yaw angles of the quadrotor body frame, respectively, in the inertial frame [15]. m is the mass of the quadrotor; I_x, I_y, I_z are the moments of inertia. A linearized model of (5.2) around the hover equilibrium point is given by:

$$\begin{aligned} \ddot{x} &= g\theta \quad \ddot{\theta} = \frac{L}{I_y}\tau_\theta \\ \ddot{y} &= -g\phi \quad \ddot{\phi} = \frac{L}{I_x}\tau_\phi \\ \ddot{z} &= \frac{\Delta f_z}{m} \quad \ddot{\psi} = \frac{1}{I_z}\tau_\psi \end{aligned} \tag{5.3}$$

where $\Delta f_z = f_z - mg$. We utilize (5.3) as the design model for the AC-RL, and (5.2) as the underlying simulation model for the numerical experiment. (5.3) and (5.1) form a linear & controllable system. Therefore, the adaptive control algorithms developed in the previous sections may be applied (when the linearized system is used as the design model).

The model in (5.2) is implemented using an RK4 integration scheme with a step of 1 millisecond. The following parameters are used to construct the reference environment: $I_x = I_y = .22kg \cdot m^2,\ I_z = .44kg \cdot m^2,\ m = 1.2kg,\ L = .30m.$ The control input is updated every 50 milliseconds to emulate latencies arising from measurement, communication, actuation and computation.

5.2 Quadrotor Landing Task

Due to the strength of RL algorithms in solving "unconstrained" control problems, we are free to define the landing task in an elegant and concise manner (instead of having to formulate a convex/simplified objective to enable tractable solutions via LQR or MPC). We assume that a platform of known inertial position is moving with known inertial velocity. The goal is to utilize full state feedback to determine a control policy that enables a quadrotor to landing on the moving platform from a wide array of initial conditions. Let Δz and Δxy be the inertial vertical distance and lateral distance, respectively, from the quadrotor to the platform. Furthermore, let $v_{xy} = \sqrt{\dot{x}^2 + \dot{y}^2}$ be the quadrotor's lateral velocity. We define the boolean variable `box` to be True if ALL of the following simultaneously hold: $|\Delta z| \leq z_{max}$; $|\Delta xy| \leq l_{max}$; $|\phi| \leq \phi_{max}$; $|\theta| \leq \theta_{max}$; $|v_{xy}| \leq v_{l,max}$; $|v_z| \leq v_{z,max}$, where $z_{max}, l_{max}, \phi_{max}, \theta_{max}, v_{l,max}, v_{z,max}$ are user provided parameters that determine if the quadcopter has successfully landed.

A ternary cost function then quite naturally defines the control objective:

$$c(\vec{x}, t) = \begin{cases} -1 & \texttt{box} \\ 1 & \neg\texttt{box} \wedge (\Delta z \leq 0 \vee t \geq T_{max}) \\ 0 & \texttt{else} \end{cases} \tag{5.4}$$

where $\vec{x}$ captures the whole quadrotor state. The first case in (5.4) defines success, the second case represents failure either due to a crash or a timeout, and the third is a neutral case and therefore set to zero. This cost function is a natural formulation for the problem at hand, as the goal is to have the quadrotor land as quickly and accurately as possible. Because the function is complex and non-quadratic, standard optimal control methods become inadequate. RL is a good alternative as it allows the determination of a policy with such a cost function.

5.3 Reinforcement Learning Outer Loop

PPO [76] is used to learn an appropriate feedback policy π in order to solve the optimization problem in (4.7). The actor and critic networks each have two hidden layers; each layer contains 64 neurons and uses tanh activation functions. A learning rate of 1e−4, a discount factor of .99, a clipping range of .2, and a generalized advantage estimator discount of .95 are used as hyperparameters. Applying the PPO algorithm to the reference environment with the cost function in (5.4) results in the successful training of a feedback policy π, which accomplishes the task when no parametric uncertainties or loss of propeller effectiveness are present.

Three popular reinforcement learning algorithms (PPO, SAC, DDPG) were used to train control policies for this environment. However, the policies trained by PPO consistently outperformed those generated via SAC or DDPG on the quadrotor task, so the PPO policies were chosen for the outer loop of AC-RL. We utilize the Stable Baselines [65] implementations of these algorithms. Stable Baselines provides a number of high quality RL algorithms, and is based on the popular OpenAI Baselines implementations.

5.4 Results (no noise)

We test two types of parametric uncertainties: 1) the mass, length and inertia properties of the quadrotor are varied between $\pm 25\%$ of their nominal (reference) values (Table 5.1) and 2) an abrupt loss of effectiveness (LOE) in the fourth propeller occurs (Table 5.2). The LOE diminishes the total thrust producible by the quadrotor, and induces an additional moment on the quadrotor if the LOE is not accounted for. Such a LOE may occur if the propeller blades are broken midflight, as demonstrated in [15]. Both types of parametric uncertainties correspond to the target-system structure as in (4.17), with the symmetric part of Λ being positive definite. Two performance metrics, success rate (SR) and success time (ST), are measured. Success time is measured as the mean time required to complete a task, averaged over all successful tests.

The PPO reinforcement learning algorithm [75] is used to train a control policy π using (4.15). The policy is then applied to a target environment which contains parametric uncertainties (see [1] for details). We compare four control methods: 1) RL, which just uses the trained π to dictate control, 2) AC-RL which utilizes both π and the adaptive laws in 4.3.4, 3)DR-RL which trains a policy using PPO on a domain-randomized simulation environment and 4) ME-RL which uses different trained policies π_{θ^*} and π_{ϕ^*}, trained using meta-learning. For further details on the quadrotor model and adaptive control implementation, refer to [15, 28, 1]. The following observations follow:

Algorithm	**Results**	
	SR	**ST**
RL	48%	7.5s
AC-RL	82%	3.5s
DR-RL	75%	7.1s
ME-RL	88%	3.4s

Table 5.1: ±25% parametric uncertainty results

A. From Table 5.1, AC-RL performs favorably when compared to pure RL or DR-RL.

B. We note that ME-RL outperforms AC-RL on both metrics. This comes with two qualifiers: 1) the meta-learner in ME-RL is trained using the same distributional shift on which it was tested (the ±25% parameter perturbations) and 2) the meta-learner is a DNN - hence the ME-RL policy does not provide the guarantees of convergence within a compact set that are afforded by AC-RL.

C. The relevance of point 1) becomes apparent from the ME-RL results in Table 5.2, where it can be see that ME-RL greatly underperforms AC-RL. This is because the specific type of perturbation (asymmetric LOE) studed in Table 5.2 was not incorporated into the meta-learner's training regimen.

AC-RL SR	**ME-RL SR**	**RL SR**	**LOE**
97%	98%	97%	0%
90%	78%	74%	10%
72%	41%	34%	25%
50%	8%	14%	50%
9%	0%	0%	75%
AC-RL ST	**ME-RL ST**	**RL ST**	**LOE**
$2.6s$	$2.8s$	$2.6s$	0%
$2.7s$	$3.9s$	$3.1s$	10%
$2.6s$	$5.1s$	$4.7s$	25%
$3.0s$	$7.5s$	$5.4s$	50%
$3.1s$	$--$	$--$	75%

Table 5.2: Results from the simulated quadrotor experiments. The LOE column represents the degree of propeller thrust lost (with 0% being no loss). For a 75% LOE there is no data on the RL or ME-RL success time because there were no successful tests. No measurement noise was introduced in these experiments. See [1] for results with noise.

5.5 Results (with noise)

We further studied (experimentally) the efficacy of AC-RL when noise is introduced into the system. A set of new reference environments are constructed, in which aleatoric measurement noise is introduced. Two reference environments are used - one with positional measurement noise only, and one with positional and orientation measurement noise. In each case the noise is taken to be zero mean and normally distributed - this may correspond, for example, to posterior position/orientation estimates that result from a Kalman filtering algorithm. RL is used to train a policy for each of these environments, and the AC-RL and RL approaches are tested on the quadrotor task with LOE and the appropriate measurement noise. The results are reported in Tables 5.3 and 5.4, which clearly indicate the large improvement of AC-RL over RL.

AC-RL SR	**ME-RL SR**	**RL SR**	**LOE**
93%	89%	93%	0%
68%	77%	42%	25%
31%	35%	4%	50%

Table 5.3: Success rates for LOE with position measurement noise. Measurement of the Cartesian positions x, y, z contain additive aleatoric uncertainty given by random variables drawn from $\mathcal{N}(0, 0.05)$

AC-RL SR	**ME-RL SR**	**RL SR**	**LOE**
84%	79%	84%	0%
51%	33%	24%	25%
23%	12%	0%	50%

Table 5.4: Success rates for LOE with position and orientation measurement noise. Measurement of the Cartesian positions x, y, z contain additive aleatoric uncertainty given by random variables drawn from $\mathcal{N}(0, 0.05)$ and ϕ, θ, ψ are perturbed by random variables drawn from $\mathcal{N}(0, 5\frac{\pi}{365})$.

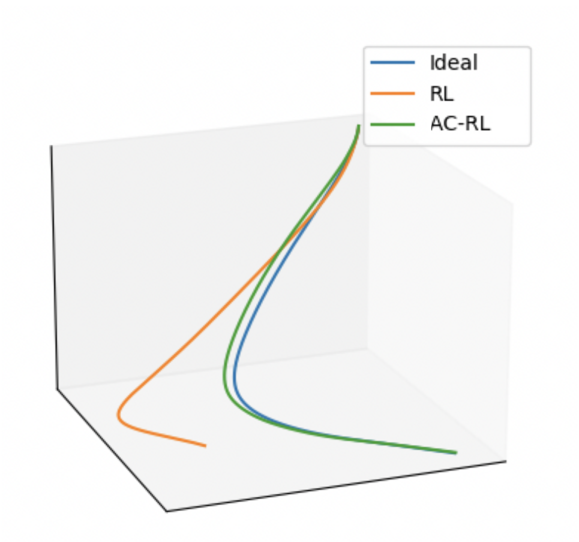

Figure 5-1: AC-RL (green) and pure RL (orange) trajectory rollouts when a single propeller loses effectiveness. The baseline "ideal" trajectory (blue) is shown, when there is no loss of effectiveness. Note that the AC-RL trajectory more closely tracks this baseline than the RL rollout.

Chapter 6

Conclusions and Future Work

This thesis proposes solutions for real time control and learning in dynamic systems using a combination of adaptive control and reinforcement learning. The AC-RL algorithm was developed for and applied to a class of control affine nonlinear dynamic systems. This class includes nonlinear systems which may be adequately linearized around equilibrium points, as well as a large subset of linear dynamical systems. The AC-RL controller is shown to produce online policies that guarantee stability, leverage accelerated convergence and lead to parameter convergence if the appropriate excitation conditions are satisfied. This thesis takes a step towards real-time control using adaptive control & machine/reinforcement learning with provable guarantees. This is accomplished by drawing upon key insights, tools, and approaches developed in the two disparate and powerful methodologies of adaptive control and reinforcement learning.

A realistic quadrotor control task was introduced. Using this task as a benchmark we demonstrated that AC-RL produced improved control policies when compared to straightforward RL, domain randomized RL and meta-learning RL. The latter two approaches are considered to be state-of-the-art methodologies in the control of uncertain/domain-shifted environments; this improvement is therefore a promising indication of the efficacy of AC-RL. In order to fully cement the performant nature of AC-RL, future work will apply the algorithm to a broader class of nonlinear dynamic systems with more complex reward/cost functions.

Another future line of work involves an exploration of the nonlinear regressors used in the adaptive control algorithms. While structured systems such as rotorcraft and pendulums admit low-dimensional representations that can be used for AC-RL, a learning-based or data-driven approach may be required to formulate these nonlinear regressors for higher-dimensional or more complex systems. Adequate modeling of such regressors may enable application of AC-RL and linear control techniques to these more complicated dynamical systems.

Appendix A

Proofs

Key steps in the proofs of the non-trivial theorems above are given below. Further details can be found in [1].

A.0.1 Proof of Theorem 3:

The closed-loop system dynamics with the AC-RL controller is represented by the error equation (4.20), the adaptive control input (4.23), and the adaptive laws in (4.24)-(4.27). A Lyapunov function candidate

$$V = \frac{1}{\gamma}\mathrm{Tr}\left[(\Xi-\Theta)^T\Lambda^T(\Xi-\Theta)\right] + \frac{1}{\gamma}\mathrm{Tr}\left[(\widehat{\Theta}-\Xi)^T\Lambda^T(\widehat{\Theta}-\Xi)\right] + e^TPe \tag{A.1}$$

yields a time-derivative

$$\begin{aligned}
\dot{V} &= -\mathrm{Tr}\left[(\Xi-\Theta)^T(\Lambda+\Lambda^T)B^TPe\Phi^T\right] \\
&\quad - \frac{\beta}{\gamma}\mathrm{Tr}\left[(\widehat{\Theta}-\Xi)^T(\Lambda+\Lambda^T)(\widehat{\Theta}-\Xi)\right]\mathcal{N}_t \\
&\quad + \mathrm{Tr}\left[(\widehat{\Theta}-\Xi)^T(\Lambda+\Lambda^T)B^TPe\Phi^T\right] \\
&\quad + e^T\left(A_H^TP+PA_H\right)e + 2e^TPB\Lambda\widetilde{\Theta}\Phi \\
&= -\frac{\beta}{\gamma}\mathrm{Tr}\left[(\widehat{\Theta}-\Xi)^T(\Lambda+\Lambda^T)(\widehat{\Theta}-\Xi)\right] \\
&\quad - \frac{\mu\beta}{\gamma}\mathrm{Tr}\left[(\widehat{\Theta}-\Xi)^T(\Lambda+\Lambda^T)(\widehat{\Theta}-\Xi)\right]\|\Phi\|^2
\end{aligned}$$

$$- e^T Q e + 4e^T PB\Lambda(\widehat{\Theta} - \Xi)\Phi$$

Through algebraic manipulations, it can be shown that

$$\begin{aligned}
\dot{V} &\leq -\frac{2\beta}{\gamma}\mathrm{Tr}\left[(\widehat{\Theta} - \Xi)^T\Omega^T\Omega(\widehat{\Theta} - \Xi)\right] - 2\|e\|^2 \\
&\quad - 4\left\|PB\right\|_2^2 \mathrm{Tr}\left[(\widehat{\Theta} - \Xi)^T\Omega^T\Omega(\widehat{\Theta} - \Xi)\right]\|\Phi\|^2 \\
&\quad + 4\|e\|\|PB\Lambda(\widehat{\Theta} - \Xi)\|_F\|\Phi\| \\
&\leq -\|e\|^2 - \frac{2\beta}{\gamma}\left\|(\widehat{\Theta} - \Xi)^T\Omega\right\|_F^2 \\
&\quad - \left[\|e\| - 2\|PB\|_2\|(\widehat{\Theta} - \Xi)^T\Omega\|_F\|\Phi\|\right]^2 \leq 0
\end{aligned}$$

where we have expressed the positive definite matrix Λ using Ω with $2\Omega^T\Omega = \Lambda + \Lambda^T$, $\|\cdot\|_F$ is the Frobenius matrix norm and $\|\cdot\|_2$ is the matrix 2-norm. In the above derivation we have used (4.26), (4.27), Assumption 5, and the inequality $\|AB\|_F \leq \|A\|_2\|B\|_F$. It can be concluded that $e, \widehat{\Theta}, \Xi \in \mathcal{L}_\infty$. Using Barbalat's lemma, as before, we conclude that $\lim_{t\to\infty} e(t) = 0$ and that $\mathcal{R} = O(1)$.

A.0.2 Proof of Theorem 4

As in Theorem 4, we consider a candidate

$$V = \frac{1}{\gamma}\mathrm{Tr}\left[(\Xi - \Theta)^T\Lambda^T(\Xi - \Theta)\right] + \frac{1}{\gamma}\mathrm{Tr}\left[(\overline{\Theta} - \Xi)^T\Lambda^T(\overline{\Theta} - \Xi)\right] + e_u^T P e_u \qquad \text{(A.2)}$$

Using (4.39) and (4.40), we obtain that $\dot{V} \leq -e_u^T e_u$ since $Q \geq 2I$. This in turn allows us to conclude that $e_u, \Xi, \overline{\Theta} \in \mathcal{L}_\infty$. For case (i), since all parameters are bounded, together with magnitude saturation, we have that the input u is bounded. For this case, it implies that the state x is bounded. Therefore $e_u \in \mathcal{L}_\infty$. Proceeding in a similar fashion to the proof of Theorem 1, Barbalat's lemma is applied to find that $\lim_{t\to\infty} \|e_u(t)\| = 0$. It is easy to show then that $\|e(t)\| = O[\int_0^t \|\Delta u(\tau)\|d\tau]$.

For case (ii), as the structure of the error model in (4.39) is identical to that considered in Theorem 1 in [77], the same arguments can be used to establish boundedness of the state.

A.0.3 Proof of Lemma 1

Note: In the proofs of Lemma 1 - 3 and Theorem 5, we assume that $\|B_0\Phi(t)^T\| \leq c_1$. Such a c_1 exists as Φ is bounded following Theorem 3.

Consider the following candidate Lyapunov function

$$V = \frac{\Lambda}{\gamma}\|\tilde{\vartheta}\|^2 + \frac{\Lambda}{\gamma}\|\tilde{\theta} - \tilde{\vartheta}\|^2 + e^T P e$$

Similar to what has been shown in the proof of Theorem 3, the time derivative of V may be bounded by

$$\begin{aligned}\dot{V} &\leq -\frac{2\beta\Lambda}{\gamma}\|\tilde{\theta} - \tilde{\vartheta}\|^2 - \Lambda\|e\|^2 - \Lambda\left[\|e\| - 2\|PB\|\|\tilde{\theta} - \tilde{\vartheta}\|\|\Phi\|\right]^2 \\ &\leq -c_2\|x_1\|^2\end{aligned}$$

where $c_2 = \Lambda \min\left\{1, \frac{2\beta}{\gamma}\right\}$. Noting that P is a positive definite matrix, it follows that for any vector $v \in \mathbb{R}^n$, $\alpha, \rho > 0$ exist such that

$$\alpha v^T v \leq v^T P v \leq \rho v^T v \tag{A.3}$$

We prove Lemma 1 by using contradiction. Assume $\mu(S) > n$ and denote $\bar{S} = \{t \in [t_1, \infty)\}$. Integrating $\dot{V}$, we have

$$\begin{aligned}\int_{t_1}^{\infty} \dot{V}(\tau)d\tau &= \int_S \dot{V}(\tau)d\tau + \int_{\bar{S}-S} \dot{V}(\tau)d\tau \\ &\leq \int_S -c_2\|x_1(\tau)\|^2 d\tau + \int_{\bar{S}-S} \dot{V}(\tau)d\tau \\ &\leq -c_2 n\epsilon_2^2\end{aligned}$$

Choose $n(\epsilon_1, \epsilon_2) = c_3\epsilon_1^2/(c_2\epsilon_2^2)$, then we have a contradiction since $\|V(t_1)\| \leq c_3\epsilon_1^2$, where $c_3 = \max\left\{\frac{\Lambda}{\gamma}, \rho\right\}$.

A.0.4 Proof of Lemma 2

Let $z(t)$ be a solution with initial condition $\|z(t_1)\| \leq \epsilon_1$. Suppose that $\left\|\tilde{\vartheta}(t)\right\| \geq \delta$ for all $t \in [t_1, t_1 + T]$, where $T = T_0 + \delta_0$.

From the error model in (4.48), for any $t \geq t_1$,

$$e(t + \delta_0) = e(t) + \int_t^{t+\delta_0} A_H e(\tau) + B_0 \Phi(\tau)^T \tilde{\theta}(\tau) d\tau \tag{A.4}$$

from which we have

$$\|e(t + \delta_0)\| \geq \left\| \int_t^{t+\delta_0} B_0 \Phi(\tau)^T \tilde{\theta}(\tau) d\tau \right\| - \left\| e(t) + \int_t^{t+\delta_0} A_H e(\tau) d\tau \right\| \tag{A.5}$$

Given $\epsilon' = \epsilon_0 \delta / (2 c_2 \delta_0)$ and $T = T_0 + \delta_0$, from the adaptive law in (4.50), there is an $\epsilon_2 > 0$ such that if $z(\tau)$ is a solution to (4.49)-(4.50) with $\|x_1(\tau)\| \leq \epsilon_2$ for all $\tau \in [t_1, t_1 + T]$, then $\|\tilde{\theta}(\tau) - \tilde{\theta}(t_1)\| \leq \epsilon'$. Define $\epsilon = \min\left\{\frac{\delta\epsilon_0}{8}, \frac{\delta\epsilon_0}{8c_1\delta_0}, \epsilon_2\right\}$. Now we show the lemma holds for this choice of T and ϵ.

If $\|x_1(t_2)\| \geq \epsilon$ for some $t_2 \in [t_1, t_1 + T]$, then we are done. Assume $\|x_1(t)\| \leq \epsilon$ for all $t \in [t_1, t_1 + T]$, then

$$\left\| e(t) + \int_t^{t+\delta_0} A_H e(\tau) d\tau \right\| \leq \epsilon + c_1 \epsilon \delta_0 \leq \frac{\epsilon_0 \delta}{4}$$

for all $t \in [t_1, t_1 + T]$. By hypothesis, there exist a $t' \in [t_1, t_1 + T_0]$ such that

$$\left\| \int_{t'}^{t'+\delta_0} B_0 \Phi(\tau)^T w d\tau \right\| \geq \epsilon_0$$

where $w = \tilde{\theta}(t_1) / \left\|\tilde{\theta}(t_1)\right\|$ is a unit vector.

We have

$$\begin{aligned} \left\| \int_{t'}^{t'+\delta_0} B_0 \Phi(\tau)^T \left[w \left\|\tilde{\theta}(t_1)\right\| - \tilde{\theta}(\tau) \right] d\tau \right\| &\leq c_2 \int_{t'}^{t'+\delta_0} \left\| \tilde{\theta}(t_1) - \tilde{\theta}(\tau) \right\| d\tau \\ &\leq c_2 \delta_0 \epsilon' = \frac{\epsilon_0 \delta}{2} \end{aligned}$$

Since $\|e(\tau)\| \leq \|x_1(\tau)\| \leq \epsilon \leq \epsilon_2$ for $\tau \in [t_1, t_1 + T]$,

$$\left\|\tilde{\theta}(t_1)\right\| \left\|\int_{t'}^{t'+\delta_0} B_0 \Phi(\tau)^T w d\tau\right\| - \left\|\int_{t'}^{t'+\delta_0} B_0 \Phi(\tau)^T \tilde{\theta}(\tau) d\tau\right\| \leq \frac{\epsilon_0 \delta}{2}$$

which implies

$$\left\|\int_{t'}^{t'+\delta_0} B_0 \Phi(\tau)^T \tilde{\theta}(\tau) d\tau\right\| \geq \epsilon_0 \delta - \frac{\epsilon_0 \delta}{2} = \frac{\epsilon_0 \delta}{2}$$

Thus

$$\|x_1(t' + \delta_0)\| \geq \|e(t' + \delta_0)\| \geq \frac{\epsilon_0 \delta}{2} - \frac{\epsilon_0 \delta}{4} = \frac{\epsilon_0 \delta}{4} > \epsilon$$

which is a contradiction.

A.0.5 Proof of Lemma 3

By Lemma 2, the assumption that $\|z(t_1)\| \leq \epsilon_1$ and $\left\|\tilde{\vartheta}(t_2)\right\| \geq \delta$ implies that there exist an ϵ such that $\|x_1(t)\|$ is periodically both less than $\epsilon/2$ and greater than ϵ. This leads to a contradiction with Lemma 1 if we choose $\epsilon_1 = \|z(t_1)\|$ and $\epsilon_2 = \epsilon/2$. We thus conclude that $\left\|\tilde{\vartheta}(t_2)\right\| \leq \delta$.

A.0.6 Proof of Theorem 5

Consider the candidate

$$V = \frac{\Lambda}{\gamma}\|\tilde{\vartheta}\|^2 + \frac{\Lambda}{\gamma}\|\tilde{\theta} - \tilde{\vartheta}\|^2 + e^T P e \tag{A.6}$$

With $\mu \geq 2\gamma\|PB\|^2/\beta$ and $Q \geq 2I$ which solves $A_H^T P + P A_H = -Q$, the time derivative of (A.6) may be bounded by

$$\dot{V} \leq -\frac{2\beta\Lambda}{\gamma}\|\tilde{\theta} - \tilde{\vartheta}\|^2 - \Lambda\|e\|^2 - \Lambda[\|e\| - 2\|PB\|\|\tilde{\theta} - \tilde{\vartheta}\|\|\Phi\|]^2 \leq 0 \tag{A.7}$$

Since P is positive-definite, (A.3) holds for some $\alpha, \rho > 0$. Now we show that given $\epsilon_1 > \epsilon_2 > 0$, there is a η with $0 < \eta < 1$ and $\Delta T_1 > 0$ such that if $z(t)$ is a solution with

$$\epsilon_2 \leq V(t) \leq \epsilon_1, \quad \text{for } t \in [t_1, t_1 + \Delta T_1],$$

then there is a $t_2 \in [t_1, t_1 + \Delta T_1]$ such that $V(t_2) \leq \eta V(t_1)$. Choose $0 < \nu < 1$, $\nu \leq \sigma < 1$ and $\Delta T_2 > 0$ so that $\rho\sqrt{1-\sigma} - \Delta T_2 \left(\frac{c_1}{\sqrt{\alpha}} + 2c_2\sqrt{\gamma}\right) > 0$, $\sqrt{\gamma(1-\nu)} - \Delta T_2 \left(\beta\sqrt{\gamma} + \frac{c_2\gamma\rho}{\sqrt{\alpha}}\right) > 0$, $0 < \Delta T_2 \left[\rho\sqrt{1-\sigma} - \Delta T_2 \left(\frac{c_1}{\sqrt{\alpha}} + 2c_2\sqrt{\gamma}\right)\right]^2 < 1$ and $0 < \frac{2\beta\Delta T_2}{\gamma}\left[\sqrt{\gamma(1-\nu)} - \Delta T_2 \left(\beta\sqrt{\gamma} + \frac{c_2\gamma\rho}{\sqrt{\alpha}}\right)\right]^2 < 1$. From Lemma 3, we can obtain a T when $\epsilon = \epsilon_1$ and $\delta = \sqrt{\epsilon_2\nu}$. Define $\eta = 1 - \min\left\{\Delta T_2\left[\rho\sqrt{1-\sigma} - \Delta T_2\left(\frac{c_1}{\sqrt{\alpha}} + 2c_2\sqrt{\gamma}\right)\right]^2,\right.$ $\left.\frac{2\beta\Delta T_2}{\gamma}\left[\sqrt{\gamma(1-\nu)} - \Delta T_2\left(\beta\sqrt{\gamma} + \frac{c_2\gamma\rho}{\sqrt{\alpha}}\right)\right]^2\right\}$ and $\Delta T_1 = T + \Delta T_2$. Next we show that for this η and ΔT_1 the results hold.

Let $t_2' \in [t_1, t_1 + T]$ be such that $\left\|\tilde{\vartheta}(t_2')\right\| \leq \delta\sqrt{\gamma}$. If $V(t_2') \leq \epsilon_2$, we are done. If $V(t_2') \geq \epsilon_2$, then

$$V(t_2') = \frac{\Lambda}{\gamma}\left\|\tilde{\vartheta}(t_2')\right\|^2 + \frac{\Lambda}{\gamma}\left\|\tilde{\theta}(t_2') - \tilde{\vartheta}(t_2')\right\|^2 + e(t_2')^T P e(t_2') \tag{A.8}$$

implies

$$\begin{aligned}(1-\nu)V(t_2') &\leq V(t_2') - \delta^2 &\text{(A.9)}\\ &\leq \frac{\Lambda}{\gamma}\left\|\tilde{\theta}(t_2') - \tilde{\vartheta}(t_2')\right\|^2 + e(t_2')^T P e(t_2') &\text{(A.10)}\\ &\leq \frac{\Lambda}{\gamma}\left\|\tilde{\theta}(t_2') - \tilde{\vartheta}(t_2')\right\|^2 + \rho e(t_2')^2 &\text{(A.11)}\end{aligned}$$

Case 1: $\frac{\Lambda}{\gamma}\|\tilde{\theta}(t_2') - \tilde{\vartheta}(t_2')\|^2 < (1-\nu)V(t_2')$. From (A.9),

$$e(t_2')^2 \geq \frac{1}{\rho}(1-\sigma)V(t_2'), \tag{A.12}$$

where $0 < \nu \leq \sigma < 1$. From (4.48), for any $t \geq t_2'$,

$$\begin{aligned}\|e(t_2')\| - \|e(t)\| &\leq \int_{t_2'}^{t}\left\|A_H e(\tau) + B_0\Phi(\tau)^T\tilde{\theta}(\tau)\right\| d\tau\\ &\leq \left(\frac{c_1}{\sqrt{\alpha}} + 2c_2\sqrt{\gamma}\right)(t - t_2')\sqrt{V(t_2')}\end{aligned}$$

where the last inequality is due to the assumption that $\|A_H\| \leq c_1$ and $\left\|B_0\Phi(\tau)^T\right\| \leq c_2$ for all τ. If $t_2 = t_2' + \Delta T_2$, we obtain

$$\begin{aligned}
\|e(t)\| &\geq \|e(t_2')\| - \left(\frac{c_1}{\sqrt{\alpha}} + 2c_2\sqrt{\gamma}\right)(t_2 - t_2')\|z(t_2')\| \\
&\geq \rho\sqrt{1-\sigma}\sqrt{V(t_2')} - \left(\frac{c_1}{\sqrt{\alpha}} + 2c_2\sqrt{\gamma}\right)\Delta T_2\sqrt{V(t_2')} \\
&= \left[\rho\sqrt{1-\sigma} - \Delta T_2\left(\frac{c_1}{\sqrt{\alpha}} + 2c_2\sqrt{\gamma}\right)\right]\sqrt{V(t_2')}
\end{aligned}$$

Integrating the derivative of the Lyapunov candidate function, we have

$$\begin{aligned}
V(t_2') - V(t_2) &= \int_{t_2'}^{t_2} -\dot{V}(\tau)d\tau \\
&\geq \int_{t_2'}^{t_2}\left(\|e(\tau)\|^2 + \frac{2\beta}{\gamma}\left\|\tilde{\theta}(\tau) - \tilde{\vartheta}(\tau)\right\|^2\right)d\tau \\
&\geq \Delta T_2\left[\rho\sqrt{1-\sigma} - \Delta T_2\left(\frac{c_1}{\sqrt{\alpha}} + 2c_2\sqrt{\gamma}\right)\right]^2 V(t_2')
\end{aligned}$$

Therefore, $V(t_2) \leq \eta V(t_2')$ and uniform asymptotic stability holds.

Case 2: $\frac{1}{\gamma}\left\|\tilde{\theta}(t_2') - \tilde{\vartheta}(t_2')\right\|^2 \geq (1-\nu)V(t_2')$. For any $t \geq t_2'$, following the process in case 1, we can show that

$$\begin{aligned}
\left\|\tilde{\theta}(t_2') - \tilde{\vartheta}(t_2')\right\| - \left\|\tilde{\theta}(t) - \tilde{\vartheta}(t)\right\| &\leq \left\|\tilde{\theta}(t) - \tilde{\vartheta}(t) - \tilde{\theta}(t_2') + \tilde{\vartheta}(t_2')\right\| \\
&\leq \int_{t_2'}^{t}\left\|\dot{\tilde{\theta}}(\tau) - \dot{\tilde{\vartheta}}(\tau)\right\|d\tau \\
&= \int_{t_2'}^{t}\left\|-\beta\left[\tilde{\theta}(\tau) - \tilde{\vartheta}(\tau)\right] + \frac{\gamma}{\mathcal{N}_t}\Phi e^T PB\right\|d\tau \\
&\leq (t - t_2')\left(\beta\sqrt{\gamma} + \frac{c_2\gamma\rho}{\sqrt{\alpha}}\right)\sqrt{V(t_2')}
\end{aligned}$$

If we let $t_2 = t_2' + \Delta T_2$, then

$$\begin{aligned}
\left\|\tilde{\theta}(t) - \tilde{\vartheta}(t)\right\| &\geq \left\|\tilde{\theta}(t_2') - \tilde{\vartheta}(t_2')\right\| - (t_2 - t_2')\left(\beta\sqrt{\gamma} + \frac{c_2\gamma\rho}{\sqrt{\alpha}}\right)\sqrt{V(t_2')} \\
&\geq \sqrt{\gamma(1-\nu)}\sqrt{V(t_2')} - \Delta T_2\left(\beta\sqrt{\gamma} + \frac{c_2\gamma\rho}{\sqrt{\alpha}}\right)\sqrt{V(t_2')}
\end{aligned}$$

$$\geq \left[\sqrt{\gamma(1-\nu)} - \Delta T_2 \left(\beta\sqrt{\gamma} + \frac{c_2\gamma\rho}{\sqrt{\alpha}}\right)\right] \sqrt{V(t_2')}$$

Integrating $\dot{V}$, we have

$$\begin{aligned}
V(t_2') - V(t_2) &= \int_{t_2'}^{t_2} -\dot{V}(\tau)d\tau \\
&\geq \int_{t_2'}^{t_2} \left(\|e(\tau)\|^2 + \frac{2\beta}{\gamma}\left\|\tilde{\theta}(\tau) - \tilde{\vartheta}(\tau)\right\|^2\right) d\tau \\
&\geq \frac{2\beta\Delta T_2}{\gamma}\left[\sqrt{\gamma(1-\nu)} - \Delta T_2 \left(\beta\sqrt{\gamma} + \frac{c_2\gamma\rho}{\sqrt{\alpha}}\right)\right]^2 V(t_2')
\end{aligned}$$

Therefore, $V(t_2) \leq \eta V(t_2')$ which proves the theorem.

Bibliography

[1] Anuradha M. Annaswamy, Anubhav Guha, Yingnan Cui, Sunbochen Tang, and Joseph E. Gaudio. Online algorithms and policies using adaptive and machine learning approaches. *arXiv preprint arXiv:2105.06577*, 2021.

[2] Karol Arndt, Murtaza Hazara, Ali Ghadirzadeh, and Ville Kyrki. Meta reinforcement learning for sim-to-real domain adaptation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2725–2731. IEEE, 2020.

[3] Alessandro Astolfi, Dimitrios Karagiannis, and Romeo Ortega. *Nonlinear and adaptive control with applications.* Springer Science & Business Media, 2007.

[4] Karl J. Åström and Björn Wittenmark. *Adaptive Control: Second Edition.* Addison-Wesley Publishing Company, 1995.

[5] Karl J Åström and Björn Wittenmark. *Adaptive control.* Courier Corporation, 2013.

[6] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[7] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems*, pages 908–918, 2017.

[8] Dimitri P Bertsekas. *Approximate dynamic programming.* Citeseer, 2008.

[9] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 2017.

[10] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming.* Athena Scientific, 1996.

[11] Nicholas M Boffi and Jean-Jacques E Slotine. Implicit regularization and momentum algorithms in nonlinearly parameterized adaptive control and prediction. *Neural Computation*, 33(3):590–673, 2021.

[12] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[13] Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. Regret bounds for robust adaptive control of the linear quadratic regulator. In *Advances in Neural Information Processing Systems 31*, pages 4192–4201. Curran Associates, Inc., 2018.

[14] Adithya M Devraj and Sean P Meyn. Zap q-learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2232–2241, 2017.

[15] Zachary T Dydek, Anuradha M Annaswamy, and Eugene Lavretsky. Adaptive control of quadrotor UAVs: a design trade study with flight evaluations. *IEEE Transactions on control systems technology*, 21(4):1400–1406, 2012.

[16] S. Evesque, A M. Annaswamy, S. Niculescu, and A P. Dowling. Adaptive control of a class of time-delay systems. *Journal of Dynamic Systems, Measurement, and Control*, 125(2):186, 2003.

[17] Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *International Conference on Machine Learning*, pages 1467–1476. PMLR, 2018.

[18] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.

[19] Nathan Fulton and André Platzer. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[20] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[21] Joseph E. Gaudio, Anuradha M. Annaswamy, Michael A. Bolender, Eugene Lavretsky, and Travis E. Gibson. A class of high order tuners for adaptive systems. *IEEE Control Systems Letters*, 5(2):391–396, apr 2021.

[22] Joseph E Gaudio, Anuradha M Annaswamy, and Eugene Lavretsky. Adaptive control of hypersonic vehicles in the presence of rate limits. In *2018 AIAA Guidance, Navigation, and Control Conference*, page 0846, 2018.

[23] Joseph E. Gaudio, Anuradha M. Annaswamy, Eugene Lavretsky, and Michael A. Bolender. Parameter estimation in adaptive control of time-varying systems under a range of excitation conditions. *IEEE Transactions on Automatic Control (to appear)*, 2022.

[24] Joseph E Gaudio, Anuradha M Annaswamy, José M Moreu, Michael A Bolender, and Travis E Gibson. Accelerated learning with robustness to adversarial regressors. *Proceedings of the 3rd Conference on Learning for Dynamics and Control, PMLR 144:636-650*, 2020.

[25] Joseph E. Gaudio, Travis E. Gibson, Anuradha M. Annaswamy, Michael A. Bolender, and Eugene Lavretsky. Connections between adaptive control and optimization in machine learning. *58th IEEE Conference on Decision and Control (CDC)*, 2019.

[26] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(9), 2004.

[27] Martin Guay and Tao Zhang. Adaptive extremum seeking control of nonlinear dynamic systems with parametric uncertainties. *Automatica*, 39(7):1283–1293, 2003.

[28] Anubhav Guha and Anuradha Annaswamy. Online policies for real-time control using mrac-rl. In *60th IEEE Conference on Decision and Control (CDC)*, pages 1806–1811. IEEE, 2021.

[29] Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[30] Heather S. Hussain, Yildiray Yildiz, Megumi Matsutani, Anuradha M. Annaswamy, and Eugene Lavretsky. Computable delay margins for adaptive systems with state variables accessible. *IEEE Transactions on Automatic Control*, 62(10):5039–5054, 2017.

[31] Petros A Ioannou and Jing Sun. *Robust Adaptive Control.* PTR Prentice-Hall, 1996.

[32] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[33] S. P. Karason and A. M. Annaswamy. Adaptive control in the presence of input constraints. *IEEE Transactions on Automatic Control*, 39(11):2325–2330, 1994.

[34] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[35] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[36] Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. Crossing the reality gap in evolutionary robotics by promoting transferable controllers. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 119–126, 2010.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[38] Miroslav Krstic. Control has met learning: Aspirational lessons from adaptive control theory. Online Event: Control Meets Learning Seminar, 2021.

[39] Miroslav Krstić, Ioannis Kanellakopoulos, and Petar Kokotović. *Nonlinear and Adaptive Control Design*. Wiley, 1995.

[40] Miroslav Krstic, Petar V Kokotovic, and Ioannis Kanellakopoulos. *Nonlinear and adaptive control design*. John Wiley & Sons, Inc., 1995.

[41] Eugene Lavretsky and Naira Hovakimyan. Positive/spl mu/-modification for stable adaptation in the presence of input constraints. In *Proceedings of the 2004 American Control Conference*, volume 3, pages 2545–2550. IEEE, 2004.

[42] Eugene Lavretsky and Kevin A. Wise. *Robust and Adaptive Control with Aerospace Applications*. Springer London, 2013.

[43] Frank L Lewis, Aydin Yesildirek, and Kai Liu. Multilayer neural-net robot controller with guaranteed tracking performance. *IEEE Transactions on neural networks*, 7(2):388–399, 1996.

[44] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[45] Antonio Loquercio, Elia Kaufmann, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: From simulation to reality with domain randomization. *IEEE Transactions on Robotics*, 36(1):1–14, 2019.

[46] Daniel J Mankowitz, Nir Levine, Rae Jeong, Yuanyuan Shi, Jackie Kay, Abbas Abdolmaleki, Jost Tobias Springenberg, Timothy Mann, Todd Hester, and Martin Riedmiller. Robust reinforcement learning for continuous control with model misspecification. *International Conference on Learning Representations*, 2020.

[47] Nikolai Matni, Alexandre Proutiere, Anders Rantzer, and Stephen Tu. From self-tuning regulators to reinforcement learning and back again. In *IEEE 58th Conference on Decision and Control (CDC)*, 2019.

[48] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[49] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

[50] A. P. Morgan and K. S. Narendra. On the uniform asymptotic stability of certain linear nonautonomous differential equations. *SIAM Journal on Control and Optimization*, 15(1):5–24, 1977.

[51] AP Morgan and KS Narendra. On the stability of nonautonomous differential equations $\dot{x} = [A + B(t)]x$, with skew symmetric matrix $B(t)$. *SIAM Journal on Control and Optimization*, 15(1):163–176, 1977.

[52] A. S. Morse. High-order parameter tuners for the adaptive control of linear and nonlinear systems. In *Systems, Models and Feedback: Theory and Applications*, pages 339–364. Birkhauser Boston, 1992.

[53] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

[54] K. Narendra and A. Annaswamy. Robust adaptive control in the presence of bounded disturbances. *IEEE Transactions on Automatic Control*, 31(4):306–315, 1986.

[55] Kumpati S. Narendra and Anuradha M. Annaswamy. *Stable Adaptive Systems.* Dover Publications, NJ, 2005. (original publication by Prentice-Hall Inc., 1989).

[56] Kumpati S. Narendra and Anuradha M. Annaswamy. *Stable Adaptive Systems.* Dover, 2005.

[57] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27:372–376, 1983.

[58] Yurii Nesterov. *Lectures on Convex Optimization.* Springer International Publishing, 2018.

[59] Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental robotics IX*, pages 363–372. Springer, 2006.

[60] Romeo Ortega. On morse's new adaptive controller: parameter convergence and transient performance. *IEEE transactions on Automatic Control*, 38(8):1191–1202, 1993.

[61] Abhishek Patkar and Anuradha M Annaswamy. An adaptive controller for a class of nonlinear plants based on neural networks and convex parameterization. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 126–131. IEEE, 2020.

[62] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[63] Riccardo Polvara, Massimiliano Patacchiola, Marc Hanheide, and Gerhard Neumann. Sim-to-real quadrotor landing via sequential deep q-networks and domain randomization. *Robotics*, 9(1):8, 2020.

[64] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[65] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[66] Aravind Rajeswaran, Kendall Lowrey, Emanuel V Todorov, and Sham M Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*, pages 6550–6561, 2017.

[67] Anders Rantzer. Concentration bounds for single parameter adaptive control. In *2018 Annual American Control Conference (ACC)*, pages 1862–1866. IEEE, 2018.

[68] James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017.

[69] Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:253–279, 2019.

[70] Spencer M. Richards, Navid Azizan, Jean-Jacques Slotine, and Marco Pavone. Adaptive-Control-Oriented Meta-Learning for Nonlinear Systems. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[71] Aurko Roy, Huan Xu, and Sebastian Pokutta. Reinforcement learning under model mismatch. In *Advances in neural information processing systems*, pages 3043–3052, 2017.

[72] Shankar Sastry and Marc Bodson. *Adaptive Control: Stability, Convergence and Robustness*. Prentice-Hall, 1989.

[73] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[74] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[75] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[76] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[77] Mac Schwager and Anuradha M Annaswamy. Direct adaptive control of multi-input plants with magnitude saturation constraints. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 783–788. IEEE, 2005.

[78] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[79] Jean-Jacques E Slotine and Weiping Li. *Applied nonlinear control*. Prentice hall Englewood Cliffs, NJ, 1991.

[80] Weijie Su, Stephen Boyd, and Emmanuel J. Candès. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. *Journal of Machine Learning Research*, 17(153):1–43, 2016.

[81] Runhan Sun, Max L Greene, Duc M Le, Zachary I Bell, Girish Chowdhary, and Warren E Dixon. Lyapunov-based real-time and iterative adjustment of deep neural networks. *IEEE Control Systems Letters*, 2021.

[82] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1139–1147. PMLR, 2013.

[83] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[84] Richard S. Sutton, Andrew G. Barto, and Ronald J. Williams. Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems*, 12(2):19–22, 1992.

[85] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[86] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[87] Gang Tao. *Adaptive Control Design and Analysis*, volume 37. John Wiley & Sons, 2003.

[88] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 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)*, pages 23–30. IEEE, 2017.

[89] Vadim I Utkin. *Sliding modes in control and optimization.* Springer Science & Business Media, 2013.

[90] M Vidyasagar. Recent advances in reinforcement learning. In *2020 American Control Conference (ACC)*, pages 4751–4756. IEEE, 2020.

[91] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[92] Tyler Westenbroek, Eric Mazumdar, David Fridovich-Keil, Valmik Prabhu, Claire J Tomlin, and S Shankar Sastry. Adaptive control for linearizable systems using on-policy reinforcement learning. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 118–125. IEEE, 2020.

[93] Andre Wibisono, Ashia C. Wilson, and Michael I. Jordan. A variational perspective on accelerated methods in optimization. *Proceedings of the National Academy of Sciences*, 113(47):E7351–E7358, 2016.

[94] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[95] Jingzhao Zhang, Aryan Mokhtari, Suvrit Sra, and Ali Jadbabaie. Direct runge-kutta discretization achieves acceleration. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[96] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[97] Kemin Zhou, John Comstock Doyle, Keith Glover, et al. *Robust and optimal control*, volume 40. Prentice hall New Jersey, 1996.