Proceedings of Machine Learning Research vol xxx:1–22, 2021

MRAC-RL: A Framework for On-Line Policy Adaptation Under Parametric Model Uncertainty

Anubhav Guha ANGUHA@MIT.EDU
Anuradha Annaswamy AANNA@MIT.EDU
Massachusetts Institute of Technology, Cambridge, MA 02139

Abstract

Reinforcement learning (RL) algorithms have been successfully used to develop control policies for dynamical systems. For many such systems, these policies are trained in a simulated environment. Due to discrepancies between the simulated model and the true system dynamics, RL trained policies often fail to generalize and adapt appropriately when deployed in the real-world environment. Current research in bridging this "sim-to-real" gap has largely focused on improvements in simulation design and on the development of improved and specialized RL algorithms for robust control policy generation. In this paper we apply principles from adaptive control and system identification to develop the model-reference adaptive control & reinforcement learning (MRAC-RL) framework. We propose a set of novel MRAC algorithms applicable to a broad range of linear and nonlinear systems, and derive the associated control laws. The MRAC-RL framework utilizes an inner-loop adaptive controller that allows a simulation-trained outer-loop policy to adapt and operate effectively in a test environment, even when parametric model uncertainty exists. We demonstrate that the MRAC-RL approach improves upon state-of-the-art RL algorithms in developing control policies that can be applied to systems with modeling errors.

Keywords: Adaptive Control, Reinforcement Learning, System Identification, Sim-To-Real

1. Introduction

Reinforcement learning (RL) methods are quickly becoming popular in the development of control policies for complex systems and environments. Successful applications have been broad and varied - ranging from direct actuator-level control and state regulation to high-level planning and decision making (Mnih et al. 2015, Ng et al. 2006, Lillicrap et al. 2015, Kober et al. 2013, Silver et al. 2018). The effectiveness of reinforcement learning algorithms in overcoming constraints that typically limit classical control techniques has enabled RL's application to decision making and continuous control tasks (Recht 2019, Schulman et al. 2015).

Many RL algorithms are fundamentally data-driven methods. As a result, control polices are often learned largely in simulation. Training in simulation is a powerful technique, allowing for a near infinite number of agent-environment interactions - in comparison, training a policy on an actual plant could be expensive, time-consuming or dangerous. In practice, however, policies trained in simulation often exhibit degenerate performance when applied to real systems (Koos et al. 2010) due to modeling errors (Tan et al. 2018). As a result, many researchers have focused on methods to bridge the "sim-to-real" gap.

In this paper we introduce a framework that enables improved performance of RL-trained policies applied to systems with modeling errors. Termed Model-Reference Adaptive Control & Reinforcement Learning (MRAC-RL), the framework consists of adaptive control elements in the inner-

loop and RL elements in the outer-loop. This inner-outer loop architecture allows a trained policy to adapt control outputs on-line in order to account for model perturbations. The theoretical foundations of MRAC (Narendra and Annaswamy 1989) are leveraged to drive the "real-world" system states to match the simulated states. The central merit of this MRAC-RL framework is that it drives the true system to react to the learned control policy in the same way that the simulated system responded during training.

1.1. Related Work

1.1.1. REINFORCEMENT LEARNING

A number of reinforcement learning algorithms have been successfully used to solve continuous control tasks. We pay special attention to the class of deep reinforcement learning (DRL) algorithms which utilize deep neural networks for function approximation. Throughout this paper we specifically reference the Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC) and Deep Deterministic Policy Gradient (DDPG) algorithms (Schulman et al. 2017, Haarnoja et al. 2018, Lillicrap et al. 2015). These three DRL algorithms have been applied to a number of varied tasks, and are considered to be state-of-the-art (Langlois et al. 2019, Duan et al. 2016, Henderson et al. 2017). While RL/DRL algorithms show significant promise, it is often difficult to reliably predict the behavior of a learned policy - an issue that is exacerbated when the policy is applied to an environment different from the one seen during training (Fulton and Platzer 2018, Roy et al. 2017, Zhang et al. 2018, Rajeswaran et al. 2017, Packer et al. 2018).

Even the most powerful DRL algorithms may fail to generalize in the presence of modeling errors (Higgins et al. 2017, Nagabandi et al. 2018, Lake et al. 2017). The research community has largely tackled these challenges by developing specialized RL algorithms. For example, the model-based PILCO (Deisenroth and Rasmussen 2011) uses a learned probabilistic dynamics model to account for dynamic uncertainty, while DARLA (Higgins et al. 2017) improves sim-to-real transfer by learning robust features. Another popular approach is to directly modify the simulation & training protocols. In Rajeswaran et al. 2016 an ensemble of environments with varying dynamics were used to improve the robustness of learned policies, while Loquercio et al. 2019 used simulated domain randomization to bridge the sim-to-real gap on a drone racing task. Many approaches utilize a combination of these techniques. In Tan et al. 2018 a robust RL algorithm was used with a system identification technique to enable real-world quadruped control. In Nagabandi et al. 2018 meta-learning principles were used to modify the policy training process and subsequently adapt the policy to unmodeled errors at test-time.

As detailed above, the majority of the research in bridging the sim-to-real gap has focused on improved simulation techniques and improved RL algorithms (Pinto et al. 2017, Packer et al. 2018, Higgins et al. 2017, Deisenroth and Rasmussen 2011, Nagabandi et al. 2018, Rajeswaran et al. 2017, Berkenkamp et al. 2017). There has, however, been little attention paid to methods that may be used to inject additional robustness and adaptability into an already-trained policy.

1.1.2. ADAPTIVE CONTROL AND SYSTEM IDENTIFICATION

Adaptive control and system identification methods have long been used in the control of safety and performance sensitive systems (Leman et al. 2009, Dydek et al. 2012, Wise and Lavretsky 2011, Michini and How 2009, Wiese et al. 2013). Unlike many RL algorithms, adaptive control techniques excel in the "zero-shot" enforcement of control objectives - that is, in learning to accomplish a

task on-line (Recht 2019, Narendra and Annaswamy 1989). These adaptive techniques are able to accommodate, in real-time, constraints on the control input magnitude (Kárason and Annaswamy 1994, Lavretsky and Hovakimyan 2004) and rate (Gaudio et al. 2019). This ability to achieve control goals while accounting for parametric uncertainties in real-time is the strength of adaptive control. A weakness of adaptive methods is the general inability to integrate complex optimization objectives, as the underlying methods often focus on the minimization of tracking and regulation errors (Slotine et al. 1991). In contrast, RL-trained policies can handle a broad range of tasks & objectives (Sutton et al. 1992), but often fail to generalize appropriately in the presence of modeling errors (as discussed in Section 1.1.1). In this paper we propose a method to combine the strengths of RL and adaptive control, while minimizing the weaknesses. Specifically, we make prolific use of model-reference adaptive control (MRAC) techniques. In the MRAC paradigm, a known reference model (characterized by known model parameters and a known model form) defines the desired closed-loop behavior of the system. The "true" model is then driven to match the reference system by the MRAC algorithm. In classical application of MRAC, the form and structure of the reference model are treated as design parameters (Krstic et al. 1995). In this paper, however, we treat the reference model as the closed-loop system formed by the simulation model and the RL-derived control policy. The MRAC task is then to drive the "true" system (which is seen only at test-time, and not during training) to track this closed-loop reference model. By synthesizing such an MRAC-RL architecture, we use guidelines from RL to generate a policy for a specified reference/simulated model, and guidelines from adaptive control to adapt this policy in real-time in order to account for "sim-to-real" modeling discrepancies. The RL component may be viewed as an outer-loop block, while the MRAC component may be viewed as an inner-loop block.

The general problem of interest is posed in Section 2 with a motivating example. The MRAC foundation of the proposed MRAC-RL framework is laid in Section 3. An algorithm that implements this framework is provided in Section 4, and it is shown in Section 5 that MRAC-RL results in improved performance for an inverted pendulum task. Summary and conclusions are presented in Section 6.

2. Problem Statement

Consider a continuous-time (CT), deterministic dynamical system defined by the map $f : X \times U \to X$:

$$\dot{x} = f(x(t), u(t), \phi), \qquad x(0) = x_0, \qquad u(t) \in U \tag{1}$$

where ϕ corresponds to system parameters that may be subject to uncertainties. Associated with this system is some cost functional $c : X \times U \times \mathbb{N} \to \mathbb{R}$ so that the optimal (finite time-horizon) control problem is given by:

$$\begin{aligned} \min_{u(t)\in U \ \forall t\in[0,T]} \quad & \int_0^T c(x(t), u(t), t)dt \\ \text{subject to} \quad & \dot{x} = f(x(t), u(t), \phi) \quad \forall t \in [0, T] \\ & x(0) = x_0 \end{aligned} \tag{2}$$

Suppose reinforcement learning techniques are used to generate a control policy π such that $u(t) = \pi(x(t))$ produces approximately optimal solutions to the system in (2). If the system to be controlled is a physical system, we will likely train the policy largely in simulation. In developing this

simulation, we implicitly make a choice of an assumed state equation, henceforth referred to as the *reference model*: $\dot{x}_r = f_r(x_r(t), u_r(t), \phi_r)$, where ϕ_r denotes the nominal values of the system parameters ϕ. The subscript r denotes the fact that these quantities are simulated and their relationships are determined by the (known) reference model. Applying the reinforcement learning method of choice to the discrete-time (DT) variant of the optimization in (2) results in the approximate optimal control policy: $\pi(x)$. Note that most RL approaches will formulate (1)-(2) as a DT Markov decision process (MDP) (Sutton et al. 1998, Kaelbling et al. 1996). For the remainder of this paper, we utilize CT notation with the assumption that the policy-generated action is applied continuously over the MDP discrete time interval, and that the numerical integration frequencies are large enough to consider digital implementations of CT algorithms.

In the standard RL approach, the trained policy π is then applied to the *true model*: $\dot{x}(t) = f(x(t), \pi(x(t)), \phi)$ (Kober et al. 2013). Recall that π was trained entirely using the *reference model*. If the reference model is erroneous (e.g, system parameters were modeled imperfectly), then $f_r(x, u, \phi_r) \neq f(x, u, \phi)$ and the reference and true trajectories will likely diverge and performance may degrade. The goal of this paper is to determine the control policy u in (2) despite uncertainties in ϕ.

2.1. A Motivating Example

We introduce a variant of the canonical swing-up inverted pendulum task (Furuta et al. 1992), henceforth referred to as the set-point randomized inverted pendulum (SRIP). We will use this example to illustrate the advantages of the MRAC-RL architecture. In the classic swing-up problem, a rigid rod is fixed at one end by a joint. The goal is to apply torque at the joint so that the free end of the rod swings upright and subsequently holds the unstable equilibrium. The SRIP objective is to instead drive the pendulum angle to a random set-point. This random set-point is provided in an augmented state vector, and changes at a set rate. As a benchmark task for control under model uncertainty, SRIP is preferable to the swing-up task. In the swing-up task the goal/cost-minimizing state ($\theta = 0$, $\dot{\theta} = 0$) represents an equilibrium of the system. Even though the equilibrium is unstable, the ideal control magnitude tends to zero as the equilibrium is approached. In contrast, optimal control of the SRIP requires a non-zero steady-state control signal. Consider the following linear and nonlinear models of the inverted pendulum:

$$ml^2\ddot{\theta} = mgl\theta - b\dot{\theta} + u \quad \text{(linear)} \qquad ml^2\ddot{\theta} = mgl\sin\theta - b\dot{\theta} + u \quad \text{(nonlinear)} \tag{3}$$

where m, g, l, $k > 0$ are the mass, gravitational, length and viscous drag constants respectively. The goal of the task is to maintain a non-zero set-point $[\theta_0, 0]^T$. In order to hold this set-point, the required control effort is necessarily a function of the model parameters. As a result, the SRIP task is more punishing than the base swing-up problem when the true model parameters deviate from the simulated (reference) model parameters, and thus serves as a suitable benchmark for robust & adaptive reinforcement learning. Note that the linear & nonlinear variants of the SRIP task represent specific examples of the generic optimal control problem posed in (2). Here, the state $x = [\theta, \dot{\theta}]^T$ and the cost c is a function that penalizes deviation from the set-point (e.g, $c = q_1(\theta - \theta_0)^2 + q_2\dot{\theta}^2 + ru^2$ with $q_1,\ q_2,\ r > 0$).

One can use a reference model of the inverted pendulum to train a control policy π for the SRIP task via reinforcement learning (Lillicrap et al. 2015). Suppose that this policy π is then used to solve the SRIP task in a test environment in which the true dynamics model deviates from the

reference model. For example, the true mass m and length l of the test inverted pendulum may differ from the mass and length of the pendulum on which π was trained. In the MRAC-RL framework that we propose, the learned policy is never applied directly to the true system. Instead, we utilize an inner-outer loop architecture whereby the control policy is used to generate a closed-loop reference system. At runtime, adaptive control methods are used in the inner-loop to drive the true system to track the closed-loop reference system. This approach ensures that the reinforcement learning agent is only ever interacting with the environment in which it was trained, while the adaptive control loop independently handles the issue of parametric model uncertainty. Note that this is a strict departure from the standard RL paradigm, in which a policy trained in a simulated environment is directly used as a feedback controller in the true environment.

Central to the MRAC-RL framework is the ability to guarantee convergence of the true model to the closed-loop reference model. We hypothesize that the ability to track a simulated reference model will improve the performance and reliability of an RL-trained policy. In the next section we develop the MRAC algorithms necessary to construct the MRAC-RL framework.

3. Model Reference Adaptive Control

We now present the MRAC control approach for linear & nonlinear dynamic systems in the presence of parametric model uncertainties.

3.1. Linear Model

We develop an MRAC algorithm for a class of n-dimensional linear dynamic models of the form:

$$\begin{aligned} &\dot{x} = Ax + Bu \\ \text{with} \quad &x \coloneqq \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} A \coloneqq \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & a_3 & \dots & a_n \end{bmatrix} B \coloneqq \begin{bmatrix} 0 \\ \vdots \\ b \end{bmatrix} \end{aligned} \tag{4}$$

All $a_i,\ b$ are non-zero and have known signs but unknown values. This system is henceforth referred to as the true system, and the goal is to choose the control input u so as to accomplish a control objective. It is easy to see that the SRIP task with the linear model is a specific case of (4). A known and potentially inaccurate reference model of the system is given by $\dot{x}_r = A_r x + B_r u_r$. The subscript r denotes parameters and signals belonging to the reference model. (A_r, B_r) are in the same controllable form as (A, B) in (4). Let α_r refer to the last row of the matrix A_r, and b_r be the non-zero element of B_r. Note that α_r, b_r are known, and therefore one can adopt a host of control methods or an RL approach to determine u_r so that x_r behaves in a desired manner. The true system to be controlled (4) may be equivalently rewritten as $\dot{x} = Ax + \lambda B_r u$, where we have introduced an unknown scalar $\lambda > 0$. The MRAC goal is to determine the input $u(t)$ so that the tracking error converges to zero: $\lim_{t\to\infty} ||e(t)|| = 0$, with $e(t) \coloneqq x(t) - x_r(t)$.

As mentioned in Section 1.1, the goal of MRAC is to drive the current tracking errors to zero, rather than the global goal in (2) of optimizing a function over the entire trajectory. Because the MRAC solution is expected to occur in real-time, it is difficult to deliver globally optimal solutions while simultaneously learning about an uncertain system.

We pick a diagonal matrix $D \in \mathbb{R}^{nxn}$, with diagonal entries defined by:

$$D_{ii} = \begin{cases} \omega_i & \alpha_{i,r} > 0 \\ \psi_i & \alpha_{i,r} < 0 \end{cases}$$

$a_{i,r}$ refers to the ith component of the known vector α_r. ω_i, ψ_i are picked so that $\omega_i > 1$, $\psi_i \leq 0$ for $i = 1, 2, \ldots, n$. For convenience we define the vector $h \coloneqq [0, 0, \ldots, 1]^T \in \mathbb{R}^n$. Note then that the matrix $A_H \coloneqq A_r - h(D\alpha_r)^T$ is Hurwitz. We additionally define $\xi \coloneqq u_r - \frac{1}{b_r}(D\alpha_r)^T e$. We now introduce the MRAC control law and the associated adaptive parameter update laws:

$$u = \hat{K}_x^T x + \hat{k}_u \xi \tag{5}$$

$$\dot{\hat{K}}_x = -\Gamma_x x e^T P B_r, \quad \dot{\hat{k}}_u = -\gamma_u \xi e^T P B_r \tag{6}$$

where $\Gamma_x = \Gamma_x^T \succ 0, \gamma_u > 0$ and $P = P^T \succ 0$ solves the Lyapunov equation: $PA_H + A_H^T P = -Q$ with $Q = Q^T \succ 0$. $\hat{K}_x$ and $\hat{k}_u$ may be initialized arbitrarily - however for this work we propose setting $\hat{K}_x(0) = \mathbf{0}_{n\times 1}$, $\hat{k}_u(0) = 1$. If no model discrepancy exists, these initial parameter values immediately lead to perfect tracking of x_r by x. We now state the two main properties of MRAC that are relevant for our proposed MRAC-RL architecture (with parameter estimation errors $\tilde{K}_x, \tilde{k}_u$):

Theorem 1 *For the system (4), associated reference system $\dot{x}_r = A_r x_r + B_r u_r$, and adaptive laws (5)-(6), the function $V(e, \tilde{K}_x, \tilde{k}_u) = e^T P e + \lambda \operatorname{Tr}(\tilde{K}_x^T \Gamma_x^{-1} \tilde{K}_x) + \lambda \frac{\tilde{k}_u^2}{\gamma_u}$ is a valid Lyapunov function.*

Theorem 2 *Theorem 1 and (5)-(6) guarantee that $\lim_{t\to\infty} ||x(t) - x_r(t)|| = 0$ if $||x_r(t)|| < M_x$ and $||u_r(t)|| < M_u \ \forall t \in [0, T]$ for some $M_x, M_u > 0$*

3.2. Nonlinear Model

The MRAC approach can be extended to a class of n-dimensional nonlinear models in a straightforward manner. This class is given by:

$$\dot{x} = A\zeta(x) + \lambda B_r u \tag{7}$$

Where $\zeta : \mathbb{R}^n \to \mathbb{R}^n$ is a known nonlinear map of the form:

$$\zeta(x) = [\phi(x_1), x_2, x_3, \ldots, x_n]^T$$

With $\phi : \mathbb{R} \to \mathbb{R}$ a known nonlinear function. For notational simplicity, we use ζ to refer to $\zeta(x)$. The pair (A, B_r) are in the same form as (4). As in Section 3.1, $\lambda > 0$ is an unknown scalar, A is unknown and B_r is a known matrix in the desired reference model: $\dot{x}_r = A_r\zeta + B_r u_r$. As in Section 3.1 the goal is to determine a u such that the tracking error converges to zero: $\lim_{t\to\infty} ||e(t)|| = 0$, with $e(t) \coloneqq x(t) - x_r(t)$. We pick an n-dimensional vector β_r with strictly negative components. Additionally, let α_r be the known vector corresponding to the last row of the matrix A_r. For convenience, define the vector $h \coloneqq [0, 0, \ldots, 1]^T \in \mathbb{R}^n$. We then define the matrix $A_H \coloneqq A_r - h\alpha_r^T + h\beta_r^T$, which is Hurwitz by construction. Defining $\xi \coloneqq u_r - \frac{1}{b_r}\alpha_r^T(\zeta - \zeta_r) + \frac{1}{b_r}\beta_r^T e$, we introduce the MRAC adaptive laws:

$$u = \hat{K}_\zeta^T \zeta + \hat{k}_u \xi \tag{8}$$

$$\dot{\tilde{K}}_\zeta = -\Gamma_\zeta \zeta e^T P B_r, \quad \dot{\tilde{k}}_u = -\gamma_u \xi e^T P B_r \tag{9}$$

where $\Gamma_\zeta = \Gamma_\zeta^T \succ 0$, $\gamma_u > 0$ and $P = P^T \succ 0$ solves the Lyapunov equation: $PA_H + A_H^T P = -Q$ with $Q = Q^T \succ 0$.

Theorem 3 *For the MRAC system described in Section 3.2, the function* $V(e, \tilde{K}_\zeta, \tilde{k}_u) = e^T P e + \lambda \operatorname{Tr}(\tilde{K}_\zeta^T \Gamma_\zeta^{-1} \tilde{K}_\zeta) + \lambda \frac{\tilde{k}_u^2}{\gamma_u}$ *is a valid Lyapunov function.*

Theorem 4 *Theorem 3 and (8)-(9) guarantee that* $\lim_{t\to\infty} ||x(t) - x_r(t)|| = 0$ *if* $||\zeta(t)|| < M_\zeta$ *and* $||u_r(t)|| < M_u \; \forall t \in [0, T]$ *for some* $M_\zeta, M_u > 0$

4. MRAC-RL

We now present the MRAC-RL framework, as shown in Figure 1. The standard RL use case is shown in Figure 1a: A trained policy directly maps system states (x) to control actions (u) in order to control a physical system. The MRAC-RL approach is outlined in Figure 1b: The trained policy operates in a simulated reference system, mapping reference states (x_r) to reference actions (u_r). An inner loop adaptive controller modifies u_r to produce a control signal (u) that drives the true system to track the reference trajectory. As a result, the trained policy never interacts with the true system, instead relying on the adaptive control block to appropriately adjust and modify u. Theorems 1-4 are leveraged to guarantee satisfactory behavior in the presence of parametric modeling errors for dynamic systems in the form of (4) or (7).

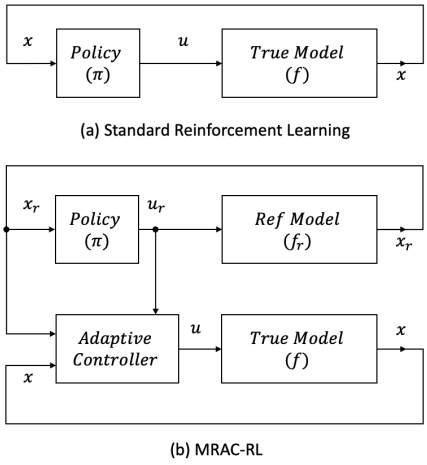

Figure 1: MRAC-RL & RL

Algorithm 1 MRAC-RL for the linear SRIP Task

Input: π; **Initialize:** $\hat{K}_x(0)$, $\hat{k}_u(0)$, $x(0)$
while not done **do**
  $u_r = \pi(x_r)$,
  for $i = 1, \ldots, F_1$ **do**
    Receive: x
    e_θ, $e_\omega = x - x_r$
    $u = \hat{K}_x^T x + \hat{k}_u u_r - \frac{2}{b_r} \hat{k}_u a_{1,r} e_\theta$
    $\hat{K}_x \leftarrow \hat{K}_x - \Delta_1 \Gamma_x x e^T P B_r$
    $\hat{k}_u \leftarrow \hat{k}_u - \Delta_1 \gamma_u (u_r - \frac{2}{b_r} a_{1,r} e_\theta) e^T P B_r$
    for $j = 1, \ldots, F_2$ **do**
      $x_r \leftarrow x_r + \Delta_2 (A_r x_r + B_r u_r)$
    end for
  end for
end while

As an example, the MRAC-RL framework as applied to the linear SRIP task is detailed in Algorithm 1. The state $x = [\theta_r, \dot{\theta}_r]^T$ and the matrices:

$$A_r = \begin{bmatrix} 0 & 1 \\ \frac{g}{l_r} & -\frac{b_r}{m_r l_r^2} \end{bmatrix} \quad B_r = \begin{bmatrix} 0 \\ \frac{1}{m_r l_r^2} \end{bmatrix}$$

define the reference model (with g, l_r, m_r, b_r known). Note, $a_{1,r} = g/l_r, b_r = 1/(m_r l_r^2)$. The true system is assumed to be well-modeled by dynamics of the same form, but with potential parameter differences. Additionally, F_1 represents the operating rate of the adaptive inner loop relative to the policy evaluation rate, F_2 represents the integration rate of the reference system relative to the adaptive loop rate,

and $\Delta_{1,2}$ are the corresponding intervals of numerical integration. To apply this algorithm to the nonlinear SRIP objective, we make the appropriate modifications in steps 7-9 of the algorithm, using (8) and (9) in Section 3.2.

We posit that the proposed combination of the MRAC and RL components as shown in Figure 1 is able to ensure that the true system behavior emulates the simulated behavior that the policy π was trained to control. In particular, our claim is that the MRAC-RL approach maintains the effectiveness of RL algorithms in generating control policies in the presence of modeling errors by combining the adaptive control components with an RL-trained policy. In the next section we validate this claim for the motivating example presented in Section 2.1.

5. Experimental Results

We test the MRAC-RL approach in solving the SRIP task for both the linear and nonlinear models of the inverted pendulum system given in (3) and evaluate the efficacy of the framework using three popular reinforcement learning algorithms: PPO, SAC and DDPG (Schulman et al. 2017, Haarnoja et al. 2018, Lillicrap et al. 2015). We utilize the Stable Baselines (Hill et al. 2018) implementations of these algorithms. Stable Baselines provides a number of high quality RL algorithms, and is based on the popular OpenAI Baselines implementations.

The RL algorithms were used to train control policies for the (linear and nonlinear) SRIP reference environment, with $m_r, l_r, b_r = 1$, $g = 10$. A quadratic cost functional was used for training: $c(\theta, \dot{\theta}, u) = q_1(\theta - \theta_0)^2 + q_2\dot{\theta}^2 + ru^2$, for $q_1, q_2, r > 0$. The policies were trained using an agent-environment interaction frequency of $10Hz$. Test environments were then generated using perturbed model parameters, picked from the following ranges: $l, m \in [.75, 1.25]$, $b \in [.001, 2.0]$. Further training and simulation details are provided in Appendix B. Four frameworks/algorithms were tested:

- ***100Hz RL:*** $\pi(x(t))$ is evaluated at 100Hz and the result is sent to the true model at 100Hz. This is a standard application of a trained policy. No adaptive control occurs at any level.
- ***10Hz RL; 100Hz MRAC:*** $u_r(t) = \pi(x_r(t))$ is evaluated at 10Hz. The MRAC inner loop converts $u_r \rightarrow u$ at 100Hz, which is sent to the true model. In the context of Algorithm 1, this corresponds to a **do** loop rate of 10Hz, with $F_1 = 10$, and $\Delta_1 = .01s$. F_2, Δ_2 are dependent on the numerical integration specifications
- ***10Hz RL:*** Similar to ***100Hz RL*** except actions are calculated and sent at 10Hz
- ***10Hz RL; 10Hz MRAC:*** Similar to ***10Hz RL; 100Hz MRAC*** except the inner loop occurs at 10Hz. That is, the MRAC loop operates in lock-step with the outer loop, providing only a single adaptive update per policy evaluation. In the context of Algorithm 1, this corresponds to a **do** loop rate of 10Hz, with $F_1 = 1$, and $\Delta_1 = .1s$.

For the linear SRIP task, we additionally test an LQR-based outer-loop control policy: $\pi(x) = -K_r x + u_{0,r}(x_0)$. The LQR feedback gain K_r is determined using the reference model parameters, x_0 represents the commanded set-point, and $u_{0,r}$ is the steady-state control required to hold x_0 (determined using the reference model). The use of three distinct RL algorithms along with an LQR-derived policy demonstrate the flexible and general nature of the MRAC-RL framework. We need only provide some map $\pi : X \rightarrow U$ at the outer loop, and the MRAC component will effectively

Algorithm		**Linear Model**		**Nonlinear Model**	
Outer Loop	**Inner Loop**	**Average Cost**	**Average** $\mathbf{e}_\theta^2$	**Average Cost**	**Average** $\mathbf{e}_\theta^2$
100Hz RL	$--$	**134**	78	317	1.3e3
10Hz RL	*100Hz MRAC*	140	**1.67**	**220**	**98**
10Hz RL	$--$	264	331	322	1.4e3
10Hz RL	*10Hz MRAC*	149	28	229	131
100Hz LQR	$--$	250	1.2e4	$--$	$--$
10Hz LQR	*100Hz MRAC*	228	14	$--$	$--$

Table 1: Results from the SRIP task with model discrepancy. A number of algorithms with varying inner/outer loop structures are tested. We opt to draw comparisons between algorithms that update the true control (u) at the same rate. For example, we compare *[100Hz Rl; $--$]* and *[10Hz Rl; 100Hz MRAC]*. Average cost is calculated as the average $c(\theta, \dot{\theta}, u)$ accumulated over all test sets. Average e_θ^2 (units are deg/s^2) is calculated as the average reference model θ tracking error (e.g, as $\overline{(\theta - \theta_r)^2}$). For both performance metrics, lower values are preferable.

account for the model discrepancies. Reinforcement learning algorithms are generally rated on their ability to maximize accumulated reward (or to minimize cost). Though this is certainly an important metric, we pay special attention to the reference tracking ability of a given algorithm in the presence of modeling errors. We claim that minimizing this divergence is important in the development of RL-based control algorithms that can effectively bridge the sim-to-real gap. Upon inspection of the **Average** $\mathbf{e}_\theta^2$ columns in Table 1, we see that the insertion of an MRAC inner loop improves reference tracking performance. Moreover, in *most* cases, the average cost $c(\theta, \dot{\theta}, u)$ incurred is substantially lowered by the use of an MRAC inner-loop. That is, the MRAC-RL framework demonstrates noticeably improved performance on the SRIP task, while significantly improving reference model tracking ability. Additionally, these results are *robust* over a broad range of perturbed model parameter values ($\pm 25\%$ error in l, m, and $\pm 100\%$ error in b), initial conditions, and reinforcement learning architectures (Appendix B).

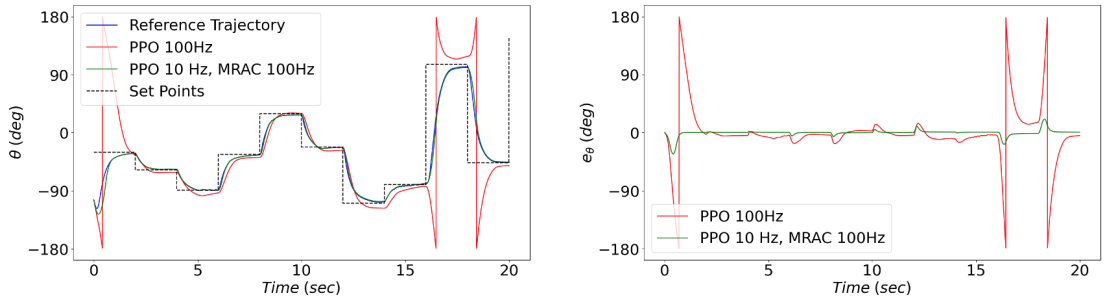

Figure 2: Reference model tracking performance. The reference trajectory is generated via application of a PPO-trained control policy to the reference model. A perturbed "true" model is produced, and we compare direct policy application (red), and the MRAC-RL inner-outer loop framework (blue). (a) depicts the θ trajectories and (b) shows the reference θ tracking error e_θ.

6. Summary & Conclusions

The overall goal of this effort is to solve optimal control problems in the form of (2), when system parameter (ϕ) modeling errors are present. We propose the MRAC-RL framework as a solution for specific classes of (1), in the forms of (4) and (7). We articulate the stability guarantees for these systems in Theorems 1-4, and demonstrate that, under mild conditions, the tracking objective $\lim_{t\to\infty} ||e(t)|| = 0$ is achieved in the presence of parametric uncertainties. The MRAC algorithms proposed in (5), (6), (8), and (9) are used to construct the inner-loop of the MRAC-RL framework. We then rely on extensive results and research in reinforcement learning (Mnih et al. 2015, Lillicrap et al. 2015, Haarnoja et al. 2018, Schulman et al. 2017) to produce a pseudo-optimal controller at the MRAC-RL outer-loop. We posit that this combined RL & adaptive control architecture enables predictable and performant solutions to (2). The MRAC algorithms proposed in (5) - (9) were used to construct and successfully apply the MRAC-RL solution to the linear and nonlinear variants of the motivating problem, with an example implementation given in Algorithm 1.

An inverted pendulum task was introduced and used to benchmark the MRAC-RL framework against a number of popular RL algorithms. We demonstrated that, on this task, the MRAC-RL approach augmented and improved upon three reinforcement learning algorithms: PPO, SAC and DDPG. The MRAC inner loop was able to confer enhanced adaptive properties upon RL-trained policies without requiring any domain randomization or retraining. This is in contrast with the majority of the methods discussed in Section 1.1.1, in which adaptive and robust properties are introduced via simulator & RL algorithm design. In theory, the MRAC-RL framework could operate in a modular manner with such methods and algorithms - for example, a robust RL algorithm such as PILCO (Deisenroth and Rasmussen 2011) could be used to train the outer loop control policy.

In this paper we have paid special attention to the minimization and convergence of *tracking* error, but did not address the convergence of *parameter* error. This is largely due to the inner-loop structure of the framework, which gives the MRAC algorithm no authority in determining the reference input. As a result, the persistent excitation (PE) condition, which is necessary & sufficient for parameter convergence, cannot be ensured. An interesting line of future research could be in investigating methods in which the policy is trained to promote the PE condition, so that an adaptive loop may effectively learn the model parameters. Such an approach was used for robust linear-quadratic regulation in Dean et al. 2019.

The next step in this line of research is to evaluate the MRAC-RL framework over a broader set of tasks. While the inverted pendulum is a good canonical control benchmark, the algorithms discussed in this paper can be extended to systems with more dimensions, greater degrees of non-linearity, and non-trivial dynamic interactions (e.g, contact forces).

Acknowledgments

This work was supported by the Boeing Strategic University Initiative

References

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.

Sarah Dean, Stephen Tu, Nikolai Matni, and Benjamin Recht. Safely learning to control the constrained linear quadratic regulator, 2019.

Marc Deisenroth and Carl E Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

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.

Nathan Fulton and André Platzer. Safe reinforcement learning via formal methods. In *AAAI Conference on Artificial Intelligence*, 2018.

Katsuhisa Furuta, M Yamakita, and S Kobayashi. Swing-up control of inverted pendulum using pseudo-state feedback. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 206(4):263–269, 1992.

Joseph E Gaudio, Anuradha M Annaswamy, Michael A Bolender, and Eugene Lavretsky. Adaptive flight control in the presence of limits on magnitude and rate. *arXiv preprint arXiv:1907.11913*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

Irina Higgins, Arka Pal, Andrei A Rusu, Loic Matthey, Christopher P Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. DARLA: Improving zero-shot transfer in reinforcement learning. *arXiv preprint arXiv:1707.08475*, 2017.

Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

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

S. P. Kárason and A. M. Annaswamy. Adaptive control in the presence of input constraints. *IEEE Transactions on Automatic Control*, 39(11):2325–2330, 1994. doi: 10.1109/9.333787.

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. doi: 10.1177/0278364913495721.

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.

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

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

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.

Tyler Leman, Enric Xargay, Geir Dullerud, Naira Hovakimyan, and Thomas Wendel. L1 adaptive control augmentation system for the x-48b aircraft. In *AIAA guidance, navigation, and control conference*, page 5619, 2009.

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.

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.

Buddy Michini and Jonathan How. L1 adaptive control for indoor autonomous vehicles: Design process and flight testing. In *AIAA Guidance, Navigation, and Control Conference*, page 5754, 2009.

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.

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.

K. S. Narendra and A. M. Annaswamy. *Stable Adaptive Systems*. Reprinted 2004, Dover Publications, 1989.

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.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. *arXiv preprint arXiv:1703.02702*, 2017.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

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.

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

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

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

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.

Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*, volume 199. Prentice hall Englewood Cliffs, NJ, 1991.

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

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

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. *arXiv preprint arXiv:1804.10332*, 2018.

Daniel P Wiese, Anuradha M Annaswamy, Jonathan A Muse, and Michael A Bolender. Adaptive control of a generic hypersonic vehicle. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, page 4514, 2013.

Kevin A Wise and Eugene Lavretsky. Robust and adaptive control of x-45a j-ucas: a design trade study. *IFAC Proceedings Volumes*, 44(1):6555–6560, 2011.

Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

Appendix

Appendix A. Convergence and Stability Proofs

A.1. Adaptive Controller: Linear System

Theorem 1 (Summary) *Let α_r be the vector corresponding to the last row of the known matrix A_r. Furthermore, b_r is the known non-zero element of B_r. Pick a diagonal matrix $D \in \mathbb{R}^{nxn}$, with components defined as:*

$$D_{ij} = \begin{cases} \omega_i & \alpha_{i,r} > 0 \quad \textit{and} \quad i = j \\ \psi_i & \alpha_{i,r} < 0 \quad \textit{and} \quad i = j \\ 0 & i \neq j \end{cases}$$

with $\omega_i > 1$, $\psi_i \leq 0$ for $i = 1, 2, \ldots, n$. We additionally define $\xi = u_r - \frac{1}{b_r}(D\alpha_r)^T e$. Using the following control and adaptive parameter update laws:

$$u = \hat{K}_x^T x + \hat{k}_u \xi \tag{10}$$

$$\dot{\hat{K}}_x = -\Gamma_x x e^T P B_r, \quad \dot{\hat{k}}_u = -\gamma_u \xi e^T P B_r \tag{11}$$

$V(e, \tilde{K}_x, \tilde{k}_u) = e^T P e + \lambda \operatorname{Tr}(\tilde{K}_x^T \Gamma_x^{-1} \tilde{K}_x) + \lambda \frac{\tilde{k}_u^2}{\gamma_u}$ *is a Lyapunov function*

Proof We consider dynamical systems given by the following linear model:

$$\dot{x} = Ax + Bu \tag{12}$$

with A, B in the controllable canonical form:

$$A = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ a_1 & a_2 & a_3 & \ldots & a_n \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ b \end{bmatrix} \tag{13}$$

Where the values $a_{1,2\ldots,n}, b \neq 0$ are unknown and the signs are known. We are given a known reference model in the same controllable canonical form:

$$\dot{x}_r = A_r x_r + B_r u_r \tag{14}$$

The subscript r denotes that the parameters and signals are known and belong to the reference model. For notational simplicity, we denote the last row of the matrix A_r by $\alpha_r = [a_{1,r}, a_{2,r}, \ldots, a_{n,r}]^T$ and the non-zero element of B_r by b_r. The goal is to track a system in the form of equation 13:

$$\dot{x} = Ax + \lambda B_r u \tag{15}$$

Where we have used knowledge of the form of B, B_r to slightly rewrite the equation. $\lambda > 0$ is an unknown scalar, A is an unknown matrix (in the same form as (A.1), with signs known) and B_r is known. We would then like to determine an input $u(t)$ to the system in equation 15 such that

$\lim_{t\to\infty} ||e(t)|| = 0$, where we have defined $e(t) \coloneqq x(t) - x_r(t)$. If the true system parameters are known, the following ideal control law provides perfect tracking of the reference system:

$$u^* = K_x^T x + k_u u_r \tag{16}$$

With $K_x,\ k_u$ satisfying the following matching conditions:

$$A + \lambda B_r K_x^T = A_r \quad \lambda k_u B_r = B_r \rightarrow \lambda k_u = 1 \tag{17}$$

We define the diagonal matrix $D \in \mathbb{R}^{nxn}$, with components defined as:

$$D_{ij} = \begin{cases} \omega_i & \alpha_{i,r} > 0 \quad \text{and} \quad i = j \\ \psi_i & \alpha_{i,r} < 0 \quad \text{and} \quad i = j \\ 0 & i \neq j \end{cases} \tag{18}$$

ω_i, ψ_i are user defined constant scalars, such that $\omega_i > 1$, $\psi_i \leq 0$ for $i = 1, 2, \ldots, n$. Note, then, that the vector defined by $v = \alpha_r - D\alpha_r$ contains entirely strictly negative values. Furthermore, we define the vector $h \in \mathbb{R}^n$ as $[0, 0, \ldots, 1]^T$. Then, the matrix A_H defined by:

$$A_H = A_r - h(D\alpha_r)^T \tag{19}$$

Is Hurwitz. Consider the following control law:

$$u = \hat{K}_x^T x + \hat{k}_u u_r - \frac{1}{b_r}\hat{k}_u (D\alpha_r)^T e \tag{20}$$

Where $\hat{K}_x$ and $\hat{k}_u$ represent adaptive estimates of K_x, k_u respectively. The error dynamics are then:

$$\begin{aligned}
\dot{e} &= \dot{x} - \dot{x}_r \\
\dot{e} &= Ax - A_r x_r - B_r u_r \\
&\quad + \lambda B_r (\hat{K}_x^T x + \hat{k}_u u_r - \frac{1}{b_r}\hat{k}_u (D\alpha_r)^T e)
\end{aligned}$$

Noting that $B_r = [0, 0, \ldots, b_r]^T \Rightarrow \frac{1}{b_r} B_r = h$, we then have:

$$\begin{aligned}
\dot{e} &= Ax - A_r x_r - B_r u_r \\
&\quad + \lambda B_r \hat{K}_x^T x + \lambda B_r \hat{k}_u u_r - \lambda \hat{k}_u h (D\alpha_r)^T e
\end{aligned}$$

Utilizing the matching conditions (17):

$$\begin{aligned}
\dot{e} &= (A_r - \lambda B_r K_x^T) x - A_r x_r - \lambda k_u B_r u_r \\
&\quad + \lambda B_r \hat{K}_x^T x + \lambda B_r \hat{k}_u u_r - \lambda \hat{k}_u h (D\alpha_r)^T e \\
\dot{e} &= A_r (x - x_r) - \lambda \hat{k}_u h (D\alpha_r)^T e \\
&\quad + \lambda B_r [(\hat{K}_x^T - K_x^T) x + (\hat{k}_u - k_u) u_r]
\end{aligned}$$

Again utilizing the matching condition $\lambda k_u = 1$, and defining the parameter estimation errors $\tilde{K}_x = \hat{K}_x - K_x$, $\tilde{k}_u = \hat{k}_u - k_u$:

$$\dot{e} = A_r e - h(D\alpha_r)^T e + \lambda k_u h(D\alpha_r)^T e - \lambda \hat{k}_u h(D\alpha_r)^T e + \lambda B_r[\tilde{K}_x^T x + \tilde{k}_u u_r]$$
$$\dot{e} = A_r e - h(D\alpha_r)^T e + \lambda B_r[\tilde{K}_x^T x + \tilde{k}_u(u_r - \frac{1}{b_r}(D\alpha_r)^T e)]$$

Applying (19) and defining an augmented reference input $\xi = u_r - \frac{1}{b_r}(D\alpha_r)^T e$, the error dynamics are then:

$$\dot{e} = A_H e + \lambda B_r[\tilde{K}_x^T x + \tilde{k}_u \xi] \tag{21}$$

Now consider the Lyapunov function candidate:

$$V(e, \tilde{K}_x, \tilde{k}_u) = e^T P e + \lambda \operatorname{Tr}(\tilde{K}_x^T \Gamma_x^{-1} \tilde{K}_x) + \lambda \frac{\tilde{k}_u^2}{\gamma_u} \tag{22}$$

With Γ_x positive definite and scalar $\gamma_u > 0$. We have also introduced a $P = P^T \succ 0$ that satisfies the Lyapunov equation:

$$P A_H + A_H^T P = -Q \quad \text{with} \quad Q = Q^T \succ 0$$

Because $P \succ 0$ the Lyapunov function V is positive definite. The time derivative is then calculated:

$$\dot{V} = \dot{e}^T P e + e^T P \dot{e} + 2\lambda \operatorname{Tr}(\tilde{K}_x^T \Gamma_x^{-1} \dot{\hat{K}}_x) + 2\lambda \frac{\tilde{k}_u \dot{\hat{k}}_u}{\gamma_u}$$
$$\dot{V} = (A_H e + \lambda B_r[\tilde{K}_x^T x + \tilde{k}_u \xi])^T P e + e^T P (A_H e + \lambda B_r[\tilde{K}_x^T x + \tilde{k}_u \xi])$$
$$+ 2\lambda \operatorname{Tr}(\tilde{K}_x^T \Gamma_x^{-1} \dot{\hat{K}}_x) + 2\lambda \frac{\tilde{k}_u \dot{\hat{k}}_u}{\gamma_u}$$
$$\dot{V} = e^T A_H^T P e + e^T P A_H e$$
$$+ 2\lambda[e^T P B_r \tilde{K}_x^T x + \operatorname{Tr}(\tilde{K}_x^T \Gamma_x^{-1} \dot{\hat{K}}_x)]$$
$$+ 2\lambda[e^T P B_r \tilde{k}_u \xi + \frac{\tilde{k}_u \dot{\hat{k}}_u}{\gamma_u}]$$
$$\dot{V} = -e^T Q e + 2\lambda[e^T P B_r \tilde{K}_x^T x + \operatorname{Tr}(\tilde{K}_x^T \Gamma_x^{-1} \dot{\hat{K}}_x)] + 2\lambda[e^T P B_r \tilde{k}_u \xi + \frac{\tilde{k}_u \dot{\hat{k}}_u}{\gamma_u}]$$

Note, for column vectors a, b, we have $a^T b = \operatorname{Tr}(b a^T)$. Then, $(e^T P B_r)(\tilde{K}_x^T x) = \operatorname{Tr}(\tilde{K}_x^T x e^T P B_r)$:

$$\dot{V} = -e^T Q e + 2\lambda \operatorname{Tr}(\tilde{K}_x^T [x e^T P B_r + \Gamma_x^{-1} \dot{\hat{K}}_x]) + 2\lambda \tilde{k}_u [e^T P B_r \xi + \frac{\dot{\hat{k}}_u}{\gamma_u}]$$

By defining the adaptive parameter update laws as:

$$\dot{\hat{K}}_x = -\Gamma_x x e^T P B_r \tag{23}$$

$$\dot{\hat{k}}_u = -\gamma_u \xi e^T P B_r \tag{24}$$

we find that the Lyapunov function time derivative is negative semi-definite:

$$\dot{V} = -e^T Q e \leq 0$$

Which implies that the tracking error vector $e(t)$ and the parameter estimation errors are bounded and that V (given by 22) is a Lyapunov function. ■

Theorem 2 *The MRAC system in Section 3.2 with control law 5 and parameter update laws 6 exhibits global uniform asymptotic tracking of the reference model dynamics 4, for any bounded reference input $u_r(t)$ that generates bounded signals in the reference model. That is:* $\lim_{t\to\infty} ||x(t) - x_r(t)|| = 0$ *if* $||x_r(t)|| < M_x$ *and* $||u_r(t)|| < M_u \; \forall t \in [0, T]$ *for* $M_x, M_u > 0$

Proof It is assumed that the reference input $u_r(t)$ is bounded and results in a reference system with bounded states. Note that this is a condition imposed on the trained reinforcement learning policy π - namely that the learned policy generates bounded responses in simulation. From Theorem 1, the tracking error $e(t)$ is uniformly bounded and stable, and the parameter estimates $\hat{K}_x(t)$ and $\hat{k}_r(t)$ are uniformly bounded. From the assumption, $x_r(t), \dot{x_r}(t)$ are bounded, and thus $x(t) = e(t) + x_r(t)$ is bounded. The boundedness of $\hat{K}_x, \hat{k}_r, x, u_r$ then implies boundedness of $u(t)$, which then implies the boundedness of $\dot{x} = Ax + \lambda B_r u$. Thus $\dot{e} = \dot{x} - \dot{x}_r$ is bounded. A direct result is that the second time derivative of V:

$$\ddot{V} = -2e^T Q e$$

is bounded. Thus, $\dot{V}$ is uniformly continuous. Because $V(t) \geq 0$ and $\dot{V}(t) \leq 0$ we have from Barbalat's Lemma that $\lim_{t\to\infty} \dot{V}(t) = 0$. Hence, $\lim_{t\to\infty} ||e(t)|| = 0$: the tracking error tends to the origin globally, uniformly and asymptotically. ■

A.2. Adaptive Controller: Nonlinear System

Theorem 3 (Summary) *Using the following control law and adaptive parameter update laws:*

$$u = \hat{K}_\zeta^T \zeta + \hat{k}_u u_r - \frac{1}{b_r} \hat{k}_u \alpha_r^T (\zeta - \zeta_r) + \frac{1}{b_r} \hat{k}_u \beta_r^T e \tag{8}$$

$$\dot{\hat{K}}_\zeta = -\Gamma_\zeta \zeta^T P B_r, \quad \dot{\hat{k}}_u = -\gamma_u \xi e^T P B_r \tag{9}$$

$V(e, \tilde{K}_x, \tilde{k}_u) = e^T P e + \lambda \operatorname{Tr}(\tilde{K}_x^T \Gamma_x^{-1} \tilde{K}_x) + \lambda \frac{\tilde{k}_u^2}{\gamma_u}$ *is a Lyapunov function*

Proof We proceed in a manner similar to the proof in A.1. Nonlinear dynamical systems of the following form are considered:

$$\dot{x} = A\zeta(x) + Bu \tag{25}$$

Where $\zeta : \mathbb{R}^n \to \mathbb{R}^n$ is a known nonlinear map of the form:

$$\zeta(x) = [\phi(x_1), x_2, x_3, \ldots, x_n]^T$$

With $\phi : \mathbb{R} \to \mathbb{R}$ a known nonlinear function. For notational simplicity, we use ζ to refer to $\zeta(x)$. In equation 25, matrix A is unknown (but the signs of the entries are known) and matrix B is unknown. Furthermore, the pair (A, B) are in a "pseudo"-controllable canonical form. That is, A and B are of the form:

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ a_1 & a_2 & a_3 & \dots & a_n \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ b \end{bmatrix} \tag{26}$$

Where the values $a_{1,2\dots,n}, b$ are unknown and the signs are known. We are given a known reference model in the same form as equation 25:

$$\dot{x}_r = A_r \zeta_r + B_r u_r \tag{27}$$

With corresponding reference signals x_r, ζ_r, u_r. (A_r, B_r) are in the same "pseudo"-controllable canonical form:

$$A_r = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{1,r} & a_{2,r} & a_{3,r} & \dots & a_{n,r} \end{bmatrix} \qquad B_r = \begin{bmatrix} 0 \\ \vdots \\ b_r \end{bmatrix} \tag{28}$$

Where the subscript r denotes that the parameters are known and belong to the reference model. For notational simplicity, we use the vector $\alpha_r = [a_{1,r}, a_{2,r}, \dots, a_{n,r}]^T$ to compactly represent the last row of A_r. The goal is to track a system with dynamics of the form:

$$\dot{x} = A\zeta + \lambda B_r u \tag{29}$$

Note, that this model form is equivalent to the form presented in equation 25. We have just used the known structures of B, B_r and the introduction of an unknown scalar $\lambda > 0$ to slightly rewrite the system. We would then like to choose an input to the system in equation 29 such that $\lim_{t\to\infty} ||e(t)|| = 0$, where we have defined $e(t) := x(t) - x_r(t)$ If the true system parameters are known, the following ideal control law provides perfect tracking of the reference system:

$$u^* = K_\zeta^T \zeta + k_u u_r \tag{30}$$

With $K_\zeta,\ k_u$ satisfying the matching conditions:

$$A + \lambda B_r K_\zeta^T = A_r \quad \lambda k_u B_r = B_r \to \lambda k_u = 1 \tag{31}$$

Consider the following adaptive control law:

$$u = \hat{K}_\zeta^T \zeta + \hat{k}_u u_r - \frac{1}{b_r}\hat{k}_u \alpha_r^T (\zeta - \zeta_r) + \frac{1}{b_r}\hat{k}_u \beta_r^T e \tag{32}$$

Where $\hat{K}_\zeta$ and $\hat{k}_u$ represent adaptive estimates of K_ζ, k_u respectively. The error dynamics are then:

$$\begin{aligned} \dot{e} &= \dot{x} - \dot{x}_r \\ \dot{e} &= A\zeta - A_r\zeta_r - B_r u_r \\ &\quad + \lambda B_r [\hat{K}_\zeta^T \zeta + \hat{k}_u u_r - \frac{1}{b_r}\hat{k}_u \alpha_r^T (\zeta - \zeta_r) + \frac{1}{b_r}\hat{k}_u \beta_r^T e] \end{aligned}$$

Defining the the vector $h \in \mathbb{R}^n$ as $[0, 0, \ldots, 1]^T$ and noting that $B_r = [0, 0, \ldots, b_r]^T \Rightarrow \frac{1}{b_r} B_r = h$, we then have:

$$\begin{aligned}\dot{e} &= A\zeta - A_r\zeta_r - B_r u_r \\ &+ \lambda B_r \hat{K}_\zeta^T \zeta + \lambda B_r \hat{k}_u u_r - \lambda \hat{k}_u h \alpha_r^T (\zeta - \zeta_r) + \lambda \hat{k}_u h \beta_r^T e\end{aligned}$$

We now utilize the matching conditions (31):

$$\begin{aligned}\dot{e} &= (A_r - \lambda B_r K_\zeta^T)\zeta - A_r\zeta_r - \lambda k_u B_r u_r \\ &+ \lambda B_r \hat{K}_\zeta^T \zeta + \lambda B_r \hat{k}_u u_r - \lambda \hat{k}_u h \alpha_r^T (\zeta - \zeta_r) + \lambda \hat{k}_u h \beta_r^T e \\ \dot{e} &= A_r(\zeta - \zeta_r) + \lambda B_r \zeta (\hat{K}_\zeta^T - K_\zeta^T) \\ &+ \lambda B_r u_r (\hat{k}_u - k_u) - \lambda \hat{k}_u h \alpha_r^T (\zeta - \zeta_r) + \lambda \hat{k}_u h \beta_r^T e\end{aligned}$$

Recall, $\zeta = [\phi(x_1), x_2, \ldots, x_n]^T$. Then, $\zeta - \zeta_r = [\phi(x_1) - \phi(x_{1_r}), e_2, e_3, \ldots e_n]^T$. Additionally, note that A_r may be rewritten in the following form:

$$A_r = \left[\begin{array}{c|c} \boldsymbol{O}_{(n-1)\times 1} & \boldsymbol{I}_{(n-1)\times(n-1)} \\ \hline \multicolumn{2}{c}{\alpha_r^T} \end{array}\right] = M + h\alpha_r^T \tag{33}$$

Where we have defined $M \coloneqq [\begin{array}{c|c} \boldsymbol{O}_{(n-1)\times 1} & \boldsymbol{I}_{(n-1)\times(n-1)} \\ \hline \multicolumn{2}{c}{\boldsymbol{O}_{1\times n}} \end{array}]$. We may then equivalently write:

$$A_r(\zeta - \zeta_r) = Me + h\alpha_r^T(\zeta - \zeta_r)$$

We utilize this substitution, along with the matching condition $\lambda k_u = 1$, to write the error dynamics as:

$$\begin{aligned}\dot{e} &= Me + \lambda k_u h \alpha_r^T (\zeta - \zeta_r) - \lambda \hat{k}_u h \alpha_r^T (\zeta - \zeta_r) \\ &+ h\beta_r^T e - \lambda k_u h \beta_r^T e + \lambda \hat{k}_u h \beta_r^T e \\ &+ \lambda B_r \zeta (\hat{K}_\zeta^T - K_\zeta^T) + \lambda B_r u_r (\hat{k}_u - k_u)\end{aligned}$$

Defining the parameter estimation errors, $\tilde{K}_\zeta = \hat{K}_\zeta - K_\zeta$ and $\tilde{k}_u = \hat{k}_u - k_u$, and noting that $M + h\beta_r^T = A_H$, we have:

$$\dot{e} = A_H e + \lambda B_r [\tilde{K}_\zeta^T \zeta + \tilde{k}_u (u_r - \frac{1}{b_r}\alpha_r^T(\zeta - \zeta_r) + \frac{1}{b_r}\beta_r^T e)]$$

Defining the augmented reference input $\xi(t) \coloneqq u_r(t) - \frac{1}{b_r}\alpha_r^T(\zeta(t) - \zeta_r(t)) + \frac{1}{b_r}\beta_r^T e(t)$, the error dynamics are then compactly represented as:

$$\dot{e} = A_H e + \lambda B_r [\tilde{K}_\zeta^T \zeta + \tilde{k}_u \xi] \tag{34}$$

Now we prove that $\lim_{t\to\infty} ||e(t)|| = 0$. Construct the following Lyapunov function:

$$V(e, \tilde{K}_\zeta, \tilde{k}_u) = e^T P e + \lambda \operatorname{Tr}(\tilde{K}_\zeta^T \Gamma_\zeta^{-1} \tilde{K}_\zeta) + \lambda \frac{\tilde{k}_u^2}{\gamma_u}$$

By the same exact procedure described in A.1, we find that $\dot{V} = -e^T Q e \leq 0$ when the following adaptive control laws are defined:

$$\dot{\hat{K}}_\zeta = -\Gamma_\zeta \zeta e^T P B_r$$

$$\dot{\hat{k}}_u = -\gamma_u \xi e^T P B_r$$

Thus V is a Lyapunov function and tracking error and parameter estimation errors are bounded. ■

Theorem 4

Proof The proof follows from the proof of Theorem 2 ■

Appendix B. Simulation and Training Details

The following equations are used to simulate the inverted pendulum:

$$ml^2\ddot{\theta} = mgl\theta - b\dot{\theta} + u \quad \text{(linear)} \qquad ml^2\ddot{\theta} = mgl\sin\theta - b\dot{\theta} + u \quad \text{(nonlinear)} \tag{35}$$

We use the following (unitless) nominal parameter values for the reference model: $m = 1,\ l = 1,\ b = 1,\ g = 10$. In order to simulate the reference model, we utilize Euler's method with a numerical integration frequency of $200Hz$.

Given the dynamics model, we then define the objective. For SRIP, a desired angular set-point, $\theta_0(t)$, is provided, and changes randomly every 5 seconds. The optimal control objective is then to minimize the following expression:

$$\sum_{k=0}^{T} q_1(\theta(k) - \theta_0)^2 + q_2\dot{\theta}(k)^2 + ru(k)^2 \tag{36}$$

which is a typical quadratic cost. We use an episode length of 20 seconds and an agent interaction frequency of $10Hz$. As a result, the number of episode steps is given by $T = 200$. In our implementation we set $q_1 = 1.0,\ q_2 = .1,\ r = .001$.

Algorithm	Linear Model		Nonlinear Model	
	Average Cost	**Hyperparameters**	**Average Cost**	**Hyperparameters**
PPO	103	[γ : .99, lr*: 8e$-$5, ent_coeff: 0.001, total_timesteps: 2e5]	177	[γ : .99, lr*: 2$-$e5, ent_coeff: 0.001, total_timesteps: 9e5]
DDPG	132	[γ : .99, lr*: 7e$-$4, Noise: OU**, total_timesteps: 7e5]	264	[γ : .99, lr*: 7e$-$4, Noise: OU**, total_timesteps: 8e6]
SAC	78	[γ : .99, lr*: 5e$-$4, total_timesteps: 8e4]	151	[γ : .99, lr*: 5e$-$4, total_timesteps: 9e5]

Table 2: Reinforcement learning training details. Average cost is measured as the average return over 100 episodes. For each algorithm, the most salient hyperparameter values are provided. Hyperparameters were chosen via simple grid search. The RL agents are trained using a 10Hz environment interaction frequency.
* Learning rates are for both actor and critic networks
** Ornstein-Uhlenbeck process with $\mu = 0, \sigma = 1.5$

We then utilize three popular reinforcement learning algorithms (PPO, SAC, DDPG) to train control policies for this environment. We utilize the Stable Baselines (Hill et al. 2018) implementations of these algorithms. Stable Baselines provides a number of high quality RL algorithms, and is based on the popular OpenAI Baselines implementations. Training details are provided in Table 2.

After training on the reference models, we create 1000 "test" environments for each of the linear and nonlinear pendulum models. For each test environment, model parameters are randomly sampled from the following ranges: $l \in [.75, 1.25]$, $m \in [.75, 1.25]$, $b \in [0.001, 2.0]$. Each test environment is also associated with a sequence of four angular set-points, sampled as: $\theta_0^i \in [-\pi, \pi]$, $i = 1, 2, 3, 4$. We then evaluate the performance of the various inner-outer loop algorithms on these test environments. The use of "RL" in Table 1 indicates the aggregation of results from using PPO, DDPG and SAC. For example, values provided for **[10Hz RL; 100Hz MRAC]** are calculated as the average of the values from **[10Hz PPO; 100Hz MRAC]**, **[10Hz DDPG; 100Hz MRAC]** and **[10Hz SAC; 100Hz MRAC]**