Proceedings of Machine Learning Research vol xxx:1–16, 2021

”@ |

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

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

Abstract

Reinforcement learning has demonstrated impressive results in complex environments
Keywords: Reinforcement Learning, Model-Reference Adaptive Control, Hybrid Control

1. Introduction

Reinforcement learning (RL) is quickly becoming a popular and viable approach in the control of complex systems and environments. Successful applications have been broad and varied - ranging from direct actuator control and regulation to high-level planning and decision making. While the use of reinforcement learning for systems and tasks that are well characterized by the theory of optimal control (e.g, the linear-quadratic regulator) is questionable, RL has undeniably been effective in solving the types of problems that classical control theory has long struggled with. Notably, with enough data it is often tractable to determine approximate optimal control polices for high-dimensional systems with complex dynamics and sparse, non-convex reward/cost functionals. The explosion of research in deep learning and deep reinforcement learning, coupled with advancements in processing power and GPU technology, has led to a number of RL algorithms that have demonstrated previously unthinkable performance across a number of platforms. The fact that reinforcement learning can often overcome constraints that typically limit classical control techniques (e.g, state dimensionality, nonlinearity, cost-functional forms) has enabled the application of RL to decision making tasks. For example, consider robotic pick-and-place problem, where we would like the robotic arm to intelligently sort items in a cluttered bin. While formulating the task as a regulation or adaptive control problem is difficult, a deep reinforcement learning algorithm may be able to learn an effective policy that is able to ”reason” and solve the optimal control problem over relatively long time-horizons. Crucially, this example task is still fundamentally dynamically constrained, which makes it susceptible to physical perturbation. If, for example, the inertial properties of the robotic arm changed significantly, or if the joint motor constants shifted, it is likely that the previously effective trained policy would exhibit degenerate performance. Because deep reinforcement learning algorithms are ultimately data-driven methods, control policies are often learned largely in simulation. Ideally, these simulations accurately model the true physical system upon which the learned algorithms are to be deployed. What happens when the simulated model and the true model differ substantially? The inability to answer this question is a current weakness of RL that is especially pronounced in deep reinforcement learning. Unlike many control methods, it is difficult to guarantee stability, robustness and other properties that are important in the development of algorithms for safety-critical and high-performance systems. That is not to say that we shouldn't use simulation to train. Simulation is a powerful tool, especially when interactions with the true system are expensive - ideally we can find ways to train in simulation while also ensuring adequate generalization and transfer to the real-world.

The majority of the research in this area (often referred to as transfer learning, or bridging the ”sim-to-real” gap) has been focused on the development of new learning algorithms and simulation methods that allow learned policies to more robustly generalize in the presence of unmodeled dy-

1.1.2. ADAPTIVE CONTROL

1.1.3. SYSTEM IDENTIFICATION

1.2. Contributions

In this work we:

- Develop a general framework that improves the performance and robustness properties of arbitrary reinforcement learning algorithms in uncertain environments
- Propose a variant on the inverted pendulum swing-up task as a benchmark task for robust control
- Develop (and prove stability of) model-reference adaptive control algorithms for a class of linear and nonlinear models

1.3. Organization

In Section 2 we briefly describe the generic optimal control problem and the challenges associated with developing control laws for models with unknown or varying parameters. A motivating example is then introduced and justified. Section 3 develops the model-reference adaptive control algorithm for the motivating example, providing justification and proof when necessary. Section 4 presents the MRAC-RL framework and provides two example algorithms for the adaptive control of the inverted pendulum. Section 5 presents the experimental results of the MRAC-RL algorithm. Finally, in Section 6 we provide our concluding remarks, as well as a discussion of future work

2. Problem Statement

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

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

Associated with this system is some cost functional $l : X \times U \times \mathbb{N} \to \mathbb{R}$ so that the optimal (finite time-horizon) control problem is given as:

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

With no further constraints on the form of l, f the generic optimal control problem can be quite difficult to solve.

2.1. Linear Systems

When the the system to be controlled is linear, that is f is of the form $f(x(t), u(t)) = A(t)x(t) + B(t)u(t)$, the optimization in (2) is often more tractable. Numerical methods such as shooting and collocation methods can often provide locally (or globally, depending on the exact problem) optimal solutions, and efficient software implementations and packages have been succesfully tested on a broad range of applications. If f is linear and l is of the form $l(x, u) = x^T Qx + u^T Ru + 2x^T Nu$, the problem is said to be linear-quadratic and the solution is provided by the Linear-Quadratic Regulator (LQR). The LQR problem and it's variants (including control under model uncertainty) are some of the most well studied problems in the theory of optimal control, and analytic solutions exist for a large class of these problems.

2.2. Nonlinear Systems

In the general case, the state equation f can be an arbitrary nonlinear map. Unlike the LQR, there are no simple closed-form solutions to this general optimization problem. Further, the lack of restrictions on l introduces additional complexity, even for state-of-the-art solvers. For example, if l is zero over "most" of the set $X \times U$ and only non-zero on a small subset, the cost is said to be sparse and lacks well-behaved gradients. This poses a challenge for numerical methods that is especially pronounced in high-dimensional systems.

2.3. Reinforcement Learning for Control

In sections 1 and 1.1 we discussed RL-based methods that have been used as viable alternatives to classical control techniques. Generally speaking, RL algorithms attempt to solve a discrete-time variant of the optimal control problem:

$$\begin{aligned} \min_{\theta \in \Theta} \quad & \sum_{k=0}^{K} l(x(k), u(k), k) \\ \text{subject to} \quad & x(k+1) = f(x(k), \pi(x(k)|\theta) \quad k = 0, \ldots, K-1 \\ & x(0) = x_0 \end{aligned} \tag{3}$$

For a thorough discussion on continuous-time vs. discrete-time representations in reinforcement learning, see. In this paper, we assume that a reinforcement learning algorithm has been trained to solve the (potentially discrete-time) optimal control problem, with the intention of applying the trained control policy onto a necessarily continuous-time physically realistic system (e.g, a robotic manipulator). As such, we present the later formulations of the inner-loop adaptive controller in continuous time, with the assumption that the adaptive interaction frequency is high enough to approximate a continuous time system.

2.4. Model Perturbation

Suppose we would like to use reinforcement learning techniques to generate a control policy π that produces approximately optimal solutions to the optimization in (2). If the system to be controlled is a physical system, we will likely train the policy largely in simulation. In developing simulation, we implicitly make a choice of an assumed state equation, henceforth referred to as the *reference*

model:

$$\dot{x}_r = f_r(x_r(t), u_r(t)) \tag{4}$$

The subscript r denotes the fact that these quantities are simulated and their relationships are determined by the (known) reference model. The control task is then characterized by the tuple $(f_r(x, u), l^*(x, u))$ where l^* defines the objective. In practice, the l^* that is initially provided may define an overly challenging learning problem. Often, some auxiliary relaxed task (f_r, l) is solved, where l is determined so that the optimal solutions to both tasks are (hopefully) equivalent.

Applying the reinforcement learning method of choice to the optimization in (3) results in the approximate optimal control policy: $\pi(\cdot|\theta^*)$ where θ^* is the minimizing argument of (3). This produces the closed-loop reference state equation:

$$\dot{x}_r = f_r(x(t), \pi(x_r(t)|\theta^*)) \tag{5}$$

Note that, relevant to the discussion in section 2.3, the trained policy may have been developed for a discrete-time system. In practice this is rarely an issue: the policy output u will just be applied to the CT system over the time interval $[t, t + \frac{1}{f}]$, where f is the RL agent's environment interaction frequency. The trained policy is then applied to the *true model*:

$$\dot{x}(t) = f(x(t), \pi(x(t)|\theta^*)) \tag{6}$$

If the reference model is perfectly representative of the true dynamics, then $x_r(t) = x(t)\ \forall t \in [0, T]$ if $x_r(0) = x(0)$. If the reference model is erroneous, $f_r(x, u) \neq f(x, u)$, the reference and true trajectories will diverge. Fundamentally, the learned policy was trained to solve a different optimization problem than the one it is being asked to solve at runtime. Though the typically opaque nature of common RL methodologies makes robustness analysis difficult (when compared to classical control techniques), we hypothesize that the divergence of the expected state transition function from the actual process will degrade the control performance. More severely, certain types of model perturbations could lead to catastrophic and expensive failures. For example, consider a linear system where the state matrices A_r, A are Hurwitz. If $||A||_2$ is much larger than $||A_r||_2$, a control policy that was viable in the reference system may result in extremely high state derivative magnitudes that could potentially damage the plant or cause harm to an operator. We posit that a trained policy will perform most effectively and safely when the true model dynamics can be artificially made to match the reference model.

2.5. A Motivating Example

We introduce a variant of the canonical swing-up inverted pendulum task - referred to as the set-point randomized inverted pendulum task (SRIPT). In the classic inverted pendulum 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 SRIPT 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. Note, if the random set-point is 0 degrees, where angle is measured counter-clockwise from the vertical, then SRIPT reduces to the standard swing-up task.

As a benchmark task for control under model uncertainty, SRIPT 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 optimal control magnitude (measured, for example, by a quadratic objective) goes to zero as the equilibrium is approached, regardless of the model parameters. In contrast, optimal control of the SRIPT requires a generally non-zero asymptotic control signal. Consider the following linear and nonlinear models of the inverted pendulum:

$$\text{linear} \quad ml^2\ddot{\theta} = mgl\theta - b\dot{\theta} + u \tag{7a}$$

$$\text{nonlinear} \quad ml^2\ddot{\theta} = mgl\sin\theta - b\dot{\theta} + u. \tag{7b}$$

Where $m, g, l, k > 0$ are the mass, gravitational, length and viscous drag constants respectively. In order to maintain the state $[\theta, \dot{\theta}]^T = [\theta_0, 0]^T$, where θ_0 is the randomly chosen set-point, the control is chosen as $u = -mgl\theta_0$ in the linear case, and $u = -mgl\sin\theta_0$ in the nonlinear case. In both cases, the stabilizing control is necessarily a function of the model parameters. As a result, SRIPT is more punishing than the base swing-up problem when the true model parameters (in this case m and l) deviate significantly from the simulated (reference) model parameters, and thus serves as a suitable benchmark for robust reinforcement learning.

The goal is then to use a reference model of the inverted pendulum system (e.g, $m_r l_r^2\ddot{\theta}_r = m_r g l_r\theta - b_r\dot{\theta} + u_r$ in the linear case) to train a control policy π that is ultimately used to control the true system (e.g, $ml^2\ddot{\theta} = m_g l\theta - b\dot{\theta} + u$).

$$\text{Reference Model} \quad \dot{x}_r = f_r(x_r, \pi(x_r)) \tag{8a}$$

$$\text{Adaptive Control Operator} \quad u = g(x, x_r, \pi(x_r)|\phi) \tag{8b}$$

$$\text{True Model} \quad \dot{x} = f(x, u) \tag{8c}$$

In the MRAC-RL framework 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 that is run in the background. At runtime, an adaptive control method is used to drive the true closed-loop system to track the closed-loop reference system. Equations 8 provide an outline of this approach for a general nonlinear plant. The function g represents the adaptive inner loop, with controller parameters ϕ. The details of 8b will be discussed in the next section.

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, whereby a policy trained in a simulated environment is directly used as a feedback controller in the true environment. A powerful and direct consequence of this framework, is that the safety and stability verification of a trained policy can be *completely separated into two independent problems*:

i. Safety verification of the closed-loop reference model

ii. Convergence of the closed-loop true model to the closed-loop reference model

Because the reference model corresponds to the simulation, i can often be accomplished by exhaustive simulation. To achieve ii we can leverage the extensive theory of adaptive control.

3. Direct Model Reference Adaptive Control

Here we present the direct model-reference adaptive control (MRAC) algorithms for the linear and nonlinear inverted pendulum models. While the nonlinear system is a more accurate representation of the inverted pendulum, we first discuss the linear model. Clearly this model is inaccurate over much of the state-space, but it is useful to introduce a number of important concepts and theorems.

3.1. Linear Model

The reference model for the linearized system is given by:

$$\begin{aligned} \ddot{\theta}_r &= a_r\theta_r + b_r\dot{\theta}_r + c_r u_r \\ \text{with} \quad & a_r > 0, \quad b_r < 0 \end{aligned} \tag{9}$$

In control canonical form, this is written as:

$$\begin{aligned} \dot{x}_r &= A_r x_r + B_r u_r \\ \text{with} \quad x_r &\coloneqq \begin{pmatrix} \theta_r \\ \dot{\theta}_r \end{pmatrix} A_r \coloneqq \begin{pmatrix} 0 & 1 \\ a_r & b_r \end{pmatrix} B_r \coloneqq \begin{pmatrix} 0 \\ c_r \end{pmatrix} \end{aligned} \tag{10}$$

Similarly, the true model is given by:

$$\begin{aligned} \dot{x} &= Ax + \lambda B_r u \\ \text{with} \quad x &\coloneqq \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} A \coloneqq \begin{pmatrix} 0 & 1 \\ a & b \end{pmatrix} \end{aligned} \tag{11}$$

Where $a > 0$, $b < 0$ are unknown true model parameters that correspond to a_r, b_r respectively, and $\lambda > 0$ is an unknown scalar quantity that represents uncertainty in control effectiveness. If A and λ were known, we could apply an ideal control law $u^* = k_\theta\theta + k_\omega\dot{\theta} + k_u u_r$, with k_θ, k_ω, k_u satisfying the so-called matching conditions

$$a + \lambda c_r k_\theta = a_r, \quad b + \lambda c_r k_\omega = b_r, \quad \lambda k_u = 1 \tag{12}$$

This leads to the closed-loop system, which is the desired reference model:

$$\ddot{\theta} = (a + \lambda c_r k_\theta)\theta + (b + \lambda c_r k_\omega)\dot{\theta} + (\lambda c_r k_u)u_r = a_r\theta + b_r\dot{\theta} + c_r u_r$$

We define the Hurwitz matrix $A_H \coloneqq \begin{pmatrix} 0 & 1 \\ -a_r & b_r \end{pmatrix}$ and the error vector $e \coloneqq [e_\theta, e_\omega]$ with $e_\theta \coloneqq \theta - \theta_r$ and $e_\omega \coloneqq \dot{\theta} - \dot{\theta}_r$. We also introduce $\hat{k}_\theta,\ \hat{k}_\omega,\ \hat{k}_u$ as adaptive estimates of $k_\theta,\ k_\omega,\ k_u$ respectively.

Theorem 1 *For the systems described by 9-12, $V(e, \tilde{K}_x, \tilde{k}_u) = e^T P e + \lambda \operatorname{Tr}(\tilde{K}_x^T \Gamma_x^{-1} \tilde{K}_x) + \lambda \frac{\tilde{k}_u^2}{\gamma_u}$ is a valid Lyapunov function if the control law*

$$u = \hat{k}_\theta\theta + \hat{k}_\omega\dot{\theta} + \hat{k}_u u_r - \frac{2}{c_r}\hat{k}_u a_r e_\theta \tag{13}$$

is used along with the parameter update equations:

$$\dot{\hat{K}}_x = -\Gamma_x x e^T P B_r, \quad \dot{\hat{k}}_u = -\gamma_u \xi e^T P B_r \tag{14}$$

Where Γ_x is positive definite, $\gamma_u > 0$, and $\xi := u_r - \frac{2}{c_r} a_r e_\theta$. $P > 0$ solves the Lypaunov equation: $PA_H + A_H^T P = -Q$ with Q positive definite. $\hat{K}_x := [\hat{k}_\theta, \hat{k}_\omega]^T$ and $\tilde{k}_u := \hat{k}_u - k_u$, $\tilde{K}_x = \hat{K}_x - K_x$ with K_x similarly defined.

Theorem 2 *The MRAC system given by 11 with control law 13 and parameter update laws 14 exhibits global uniform asymptotic tracking of the reference model dynamics 10, 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 $M_x, M_u > 0$*

3.2. Nonlinear Model

Next we consider the nonlinear model of an inverted pendulum. Unless otherwise redefined, variables are defined in the same manner as for the above linear system analysis. The reference model is given by:

$$\begin{aligned} \ddot{\theta}_r &= a_r \sin\theta_r + b_r \dot{\theta}_r + c_r u_r \\ \text{with} \quad a_r &> 0, \quad b_r < 0 \end{aligned} \tag{15}$$

$\ddot{\theta}_r = a_r \sin\theta_r + b_r \dot{\theta}_r + c_r u_r$. The true model is:

$$\begin{aligned} \ddot{\theta} &= a \sin\theta + b\dot{\theta} + \lambda c_r u + c_r u_r \\ \text{with} \quad a &> 0, \quad b < 0 \end{aligned} \tag{16}$$

As above, the goal is to have the true system track the reference model.

Theorem 3 *For the systems described by 15, 16, $V(e, \tilde{K}_x, \tilde{k}_u) = e^T P e + \lambda \operatorname{Tr}(\tilde{K}_x^T \Gamma_x^{-1} \tilde{K}_x) + \lambda \frac{\tilde{k}_u^2}{\gamma_u}$ is a valid Lyapunov function if the control law*

$$u = \hat{k}_\theta \sin\theta + \hat{k}_\omega \dot{\theta} + \hat{k}_u u_r - \frac{1}{c_r} \hat{k}_u a_r [(\sin\theta - \sin\theta_r) + e_\theta] \tag{17}$$

is used along with the parameter update equations:

$$\begin{aligned} \dot{\hat{K}}_x &= -\Gamma_x \zeta e^T P B_r, \quad \dot{\hat{k}}_u = -\gamma_u \xi e^T P B_r \\ \textit{with} \quad \zeta &:= [\sin\theta, \ \dot{\theta}]^T \quad \textit{and} \quad \xi(t) := u_r(t) - \frac{a_r}{c_r}[(\sin\theta - \sin\theta_r) + e_\theta(t)] \end{aligned} \tag{18}$$

All other terms are defined in the same manner as in Theorem 1

Theorem 4 *The MRAC system described in 15, 16 and Theorem 3 exhibits global uniform asymptotic tracking of the reference model dynamics under the same conditions as in Theorem 2.*

4. MRAC-RL Framework

In this section, we present the main contribution of this paper - a general and flexible framework for combining model-reference adaptive control with reinforcement learning. The MRAC-RL approach maintains the effectiveness of recent deep reinforcement learning algorithms in generating control policies, while leveraging adaptive control theory to stabilize the system to a known dynamics model. The algorithm as applied to the linearized SRIPT objective is presented below. F_1

Algorithm 1 MRAC-RL for on-line adaptive control of the linearized SRIPT

1: **Input:** π
2: **Initialize:** $\hat{K}_x(0)$, $\hat{k}_u(0)$, $x(0)$
3: $x_r(0) \leftarrow x(0)$
4: **while not** done **do**
5: $u_r = \pi(x_r)$,
6: **for** $i = 1, \ldots, F_1$ **do**
7: **Receive:** x
8: $e_\theta, e_\omega = x - x_r$
9: $u = \hat{K}_x^T x + \hat{k}_u u_r - \frac{2}{c_r}\hat{k}_u a_r e_\theta$
10: $\hat{K}_x \leftarrow \hat{K}_x - \Delta_1 \Gamma_x x e^T P B_r$
11: Let $\xi = u_r - \frac{2}{c_r} a_r e_\theta$
$\hat{k}_u \leftarrow \hat{k}_u - \Delta_1 \gamma_u \xi e^T P B_r$
12: **for** $j = 1, \ldots, F_2$ **do**
13: $x_r \leftarrow x_r + \Delta_2(A_r x_r + B_r u_r)$
14: **end for**
15: **end for**
16: **end for**

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 time deltas. In practice, a more sophisticated integration method may be used.

To apply this algorithm to the nonlinear SRIPT objective, we make the appropriate modifications in steps 8-13 of the algorithm, using the results from 3.2

While this paper presents experimental results on the inverted pendulum task, the approach can be applied to the broad range of dynamical systems that MRAC techniques are amenable to. Furthermore, the development of the MRAC algorithm is largely uncoupled from the development and training of the reinforcement learning algorithm. Indeed, the only requirement is that a fixed control policy π is provided prior to operation. In the next section we provide experimental results for a number of different reinforcement learning algorithms, as well as an LQR-optimal control policy.

5. Experimental Results

5.1. Experimental Setup

We test the MRAC-RL approach in solving the SRIPT objective for both the linear and nonlinear models of the inverted pendulum system. We evaluate the efficacy of the framework using three popular reinforcement learning algorithms: PPO, SAC and DDPG. Each algorithm is written largely using the Stable Baselines library (link).

The algorithms are used to train control policies for the SRIPT task, with model parameters $l = 1,\ m = 1,\ g = 10,\ b = 1$, where $l,\ m,\ g,\ b$ represent the length, mass, gravitational and viscous drag properties of the system. The algorithms are trained using a $10Hz$ agent interaction

frequency. The cost/reward function used is a simple quadratic function of state error, measured relative to the desired set point. We then test these policies on inverted pendulum models with perturbed dynamics. Specifically, the perturbed model parameters are picked from the ranges: $l \in [.75, 1.25], m \in [.75, 1.25], b \in [0, 2]$. Over a test set of 1000 environment conditions (which determine the initial pendulum state and the set-point locations) we evaluate four frameworks:

- ***100Hz RL:*** The trained policy π is evaluated at x and the resulting control value is sent to the true model at 100Hz
- ***10Hz RL; 100Hz MRAC:*** π is evaluated at x_r at 10Hz to produce u_r. The MRAC inner loop converts $u_r \rightarrow u$ at 100Hz, which is sent to the true model
- ***10Hz RL*** Similar to ***100Hz RL:*** except the loop occurs 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.

For the linear inverted pendulum model, we additionally evaluate an LQR feedback controller, where the LQR gain was determined using reference model parameters.

Reinforcement learning algorithms are generally rated on their ability to maximize accumulated reward (equivalently; to minimize cost) on benchmark tasks. This is certainly an important metric - we would not consider an extremely safe and stable learning algorithm useful if it consistently achieved poor rewards on basic tasks. However, in this section we draw attention special attention to the reference tracking ability of a given algorithm: if we apply a control policy to both the reference (simulated) model and the true model, how greatly do these trajectories differ? We claim that minimizing this divergence is a crucial step in the development of robust and generalizable RL-based control algorithms. Further, if tracking of the true model to the reference model can be guaranteed, then we may only need to guarantee the safety of a trained controller in simulation. Thus, we pay special attention to the second and third column of experimental values in the following tables. We compare methods that operate on the plant at the same frequency (for example, ***100Hz RL*** and ***10Hz RL*****; 100Hz MRAC** are compared with each other). We see that the insertion of an MRAC inner loop greatly improves tracking performance, as measured by a quadratic performance index. Moreover, in *most* cases, the average cost achieved is substantially lowered with the incorporation of the MRAC algorithm. Furthermore, these results are *robust* over a broad range of perturbed model parameter values, initial conditions, and reinforcement learning architectures (refer to the Appendix for a detailed discussion on the reinforcement learning algorithms and implementations used), in that the MRAC-RL framework consistently outperforms the standard reinforcement learning approach on these metrics (average cost and average tracking error).

6. Discussion & Conclusions

In this paper, we propose the MRAC-RL framework, and demonstrate it's efficacy in adapting trained control policies to plants and environments that exhibit model parameter uncertainty and perturbation. We have shown that, under mild conditions, that global asymptotic tracking of an *a priori* unseen plant model to a reference model may be guaranteed. This is analogous to enforcing the simulated dynamics onto the real-world dynamics. We claim that this property of the

	Linear Model		Nonlinear Model	
Algorithm	**Average Cost**	**Average** e_θ^2 (deg^2)	**Average Cost**	**Average** e_θ^2 (deg^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	$--$	$--$

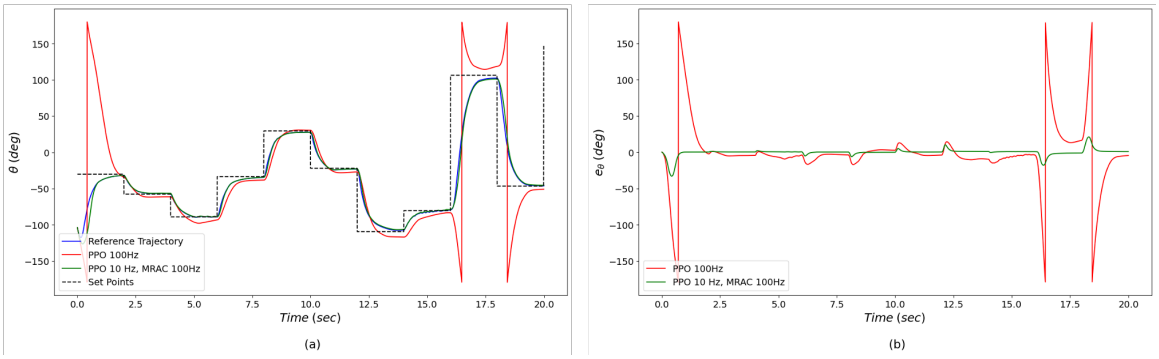

Figure 1: Comparison of reference (blue trace) tracking performance. The red trajectory is generated by direct application of a controlled policy trained using PPO. The green trajectory is generated by the MRAC-RL algorithm using the same trained policy at the outer loop. The black dotted line represents the desired set points throughout the episode.

MRAC-RL framework greatly eases the burden of safety verification, effectively reducing the problem of verification in the real world to verifying safety in simulation. We additionally introduced a novel control task for benchmarking the adaptability and robustness properties of reinforcement learning algorithms. Further, we demonstrated that on this task, MRAC-RL was able to improve model error tolerances and performances of a number of popular algorithms. This paper represents a novel attempt at "robustifying" a policy *after* it is has been trained. There has, however, been significant investigation into methods to add robustness into the learning process - such as injecting model error and noise into the simulations, or even developing reinforcement learning algorithms that are designed to specifically generated robust policies. In theory, the MRAC-RL style of post-processing robustness and adaptability could operate in a synergistic manner with such methods and algorithms. This certainly represents an exciting direction for future work. In this paper, we have paid special attention to the minimization and convergence of tracking error. In the MRAC-RL approach, it is generally impossible to claim the convergence of parameter errors. This is largely due to the inner-loop structure of the framework, which gives the MRAC no power in determining the reference input. As a result, we cannot ensure persistent excitation of the input or system signals

- this is a property that the trained policy must ensure. An interesting line of future work could be in investigating methods whereby the policy is trained to promote persistency of excitation, so that an adaptive loop may effectively learn the parameters. Lastly, a kernel of future work that forms a direct throughline to the results presented in this paper is the application of the MRAC-RL framework to more complicated systems - such as plants with higher degrees of freedom, or environments with difficult-to-model interactions (such as contact forces).

Acknowledgments

We thank a bunch of people.

Appendix

Appendix A. Convergence and Stability Proofs

A.1. Adaptive Controller: Linear System

Theorem 1 (Summary) *Using the following control and adaptive parameter update laws:*

$$u = \hat{k}_\theta \theta + \hat{k}_\omega \dot{\theta} + \hat{k}_u u_r - \frac{2}{c_r} \hat{k}_u a_r e_\theta \tag{13}$$

$$\dot{\hat{K}}_x = -\Gamma_x x e^T P B_r, \quad \dot{\hat{k}}_u = -\gamma_u \xi e^T P B_r \tag{14}$$

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

Proof The proof continues from Equations 9 - 12. Recall that we defined the Hurwitz matrix $A_H \coloneqq \begin{pmatrix} 0 & 1 \\ -a_r & b_r \end{pmatrix}$, the error vector $e \coloneqq [e_\theta, e_\omega]$ and introduced $\hat{k}_\theta,\ \hat{k}_\omega,\ \hat{k}_u$ as adaptive estimates of $k_\theta,\ k_\omega,\ k_u$ respectively. The goal is to design a control law that promotes global uniformly asymptotic tracking of the system state x to the state x_r of the reference model. That is:

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

If the true model parameters are known, then we can use an ideal control law:

$$u^* = k_\theta \theta + k_\omega \dot{\theta} + k_u u_r \tag{19}$$

with k_θ, k_ω, k_u satisfying the matching conditions

$$a + \lambda c_r k_\theta = a_r, \quad b + \lambda c_r k_\omega = b_r, \quad \lambda k_u = 1 \tag{20}$$

We propose the following control law:

$$u = \hat{k}_\theta \theta + \hat{k}_\omega \dot{\theta} + \hat{k}_u u_r - \frac{\alpha}{c_r} \hat{k}_u a_r e_\theta + \frac{\beta}{c_r} \hat{k}_u b_r e_\omega \tag{21}$$

Where $\alpha > 1$, $\beta \geq 0$. Note that when the estimated gains achieve the ideal values, $e_\theta = 0$ and this modified control law matches the ideal law 19

The error dynamics are then:

$$\begin{aligned}
\ddot{\theta} - \ddot{\theta}_r &= (a_r - \lambda c_r k_\theta)\theta - a_r \theta_r + (b_r - \lambda c_r k_\omega)\dot{\theta} - b_r \dot{\theta}_r \\
&\quad + c_r \lambda \hat{k}_\theta \theta + c_r \lambda \hat{k}_\omega \dot{\theta} - \alpha \lambda \hat{k}_u a_r e_\theta + \beta \lambda \hat{k}_u b_r e_\omega + c_r \lambda \hat{k}_u u_r - c_r \lambda k_u u_r \\
\ddot{\theta} - \ddot{\theta}_r &= a_r(\theta - \theta_r) - \alpha \lambda \hat{k}_u a_r e_\theta + b_r(\dot{\theta} - \dot{\theta}_r) + \beta \lambda \hat{k}_u b_r e_\omega \\
&\quad + c_r \lambda \theta (\hat{k}_\theta - k_\theta) + c_r \lambda \dot{\theta} (\hat{k}_\omega - k_\omega) + c_r \lambda u_r (\hat{k}_u - k_u)
\end{aligned}$$

We define the parameter estimation errors: $\tilde{k}_\theta = \hat{k}_\theta - k_\theta,\ \tilde{k}_\omega = \hat{k}_\omega - k_\omega,\ \tilde{k}_u = \hat{k}_u - k_u$

Utilizing the matching condition $\lambda k_u = 1$:

$$\begin{aligned}
\ddot{\theta} - \ddot{\theta}_r &= [\alpha\lambda k_u - (\alpha-1)]a_r e_\theta - \alpha\lambda\hat{k}_u a_r e_\theta + [1+\beta-\beta\lambda k_u]b_r e_\omega + \beta\lambda\hat{k}_u b_r e_\omega \\
&\quad + c_r\lambda(\tilde{k}_\theta\theta + \tilde{k}_\omega\dot{\theta} + \tilde{k}_u u_r) \\
&= -(\alpha-1)a_r e_\theta + \alpha\lambda a_r e_\theta(k_u - \hat{k}_u) + (1+\beta)b_r e_\omega + \beta\lambda b_r e_\omega(\hat{k}_u - k_u) \\
&\quad + c_r\lambda(\tilde{k}_\theta\theta + \tilde{k}_\omega\dot{\theta} + \tilde{k}_u u_r) \\
\ddot{\theta} - \ddot{\theta}_r &= -(\alpha-1)a_r e_\theta + (1+\beta)b_r e_\omega + c_r\lambda[\tilde{k}_\theta\theta + \tilde{k}_\omega\dot{\theta} + \tilde{k}_u(u_r - \frac{\alpha}{c_r}a_r e_\theta + \frac{\beta}{c_r}b_r e_\omega)]
\end{aligned}$$

Recall that $a_r > 0,\ b_r < 0,\ \alpha > 1,\ \beta \geq 0$. Then we define the Hurwitz matrix:

$$A_H \coloneqq \begin{pmatrix} 0 & 1 \\ -(\alpha-1)a_r & (1+\beta)b_r \end{pmatrix}$$

We also define the parameter error vector $\tilde{K}_x = \begin{pmatrix} \tilde{k}_\theta \\ \tilde{k}_\omega \end{pmatrix}$ and the augmented input $\xi(t) \coloneqq u_r(t) - \frac{\alpha}{c_r}a_r e_\theta(t) + \frac{\beta}{c_r}b_r e_\omega(t)$. The error dynamics are then compactly represented as:

$$\dot{e} = A_H e + \lambda B_r[\tilde{K}_x^T x + \tilde{k}_u\xi] \tag{22}$$

Now consider the Lyapunov function candidate:

$$V(e, \tilde{K}_x, \tilde{k}_u) = e^T P e + \lambda\,\mathrm{Tr}(\tilde{K}_x^T\Gamma_x^{-1}\tilde{K}_x) + \lambda\frac{\tilde{k}_u^2}{\gamma_u} \tag{23}$$

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:

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

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

$$\begin{aligned}
\dot{V} &= \dot{e}^T P e + e^T P\dot{e} + 2\lambda\,\mathrm{Tr}(\tilde{K}_x^T\Gamma_x^{-1}\dot{\tilde{K}}_x) + 2\lambda\frac{\tilde{k}_u\dot{\tilde{k}}_u}{\gamma_u} \\
\dot{V} &= (A_H e + \lambda B_r[\tilde{K}_x^T x + \tilde{k}_u\xi])^T P e + e^T P(A_H e + \lambda B_r[\tilde{K}_x^T x + \tilde{k}_u\xi]) \\
&\quad + 2\lambda\,\mathrm{Tr}(\tilde{K}_x^T\Gamma_x^{-1}\dot{\tilde{K}}_x) + 2\lambda\frac{\tilde{k}_u\dot{\tilde{k}}_u}{\gamma_u} \\
\dot{V} &= e^T A_H^T P e + e^T P A_H e \\
&\quad + 2\lambda[e^T P B_r\tilde{K}_x^T x + \mathrm{Tr}(\tilde{K}_x^T\Gamma_x^{-1}\dot{\tilde{K}}_x)] \\
&\quad + 2\lambda[e^T P B_r\tilde{k}_u\xi + \frac{\tilde{k}_u\dot{\tilde{k}}_u}{\gamma_u}]
\end{aligned}$$

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

Note, column vectors a, b, we have $a^T b = \text{Tr}(ba^T)$. Then, $(e^T PB_r)(\tilde{K}_x^T x) = \text{Tr}(\tilde{K}_x^T x e^T PB_r)$:

$$\dot{V} = -e^T Qe + 2\lambda \,\text{Tr}(\tilde{K}_x^T [xe^T PB_r + \Gamma_x^{-1} \dot{\hat{K}}_x]) + 2\lambda \tilde{k}_u [e^T PB_r \xi + \frac{\dot{\hat{k}}_u}{\gamma_u}]$$

By defining the adaptive parameter update laws as:

$$\dot{\hat{K}}_x = -\Gamma_x x e^T PB_r \tag{24}$$

$$\dot{\hat{k}}_u = -\gamma_u \xi e^T PB_r \tag{25}$$

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

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

Which implies that the tracking error vector $e(t)$ and the parameter estimation errors are bounded and that V (given by 23) is a Lyapunov function. For the purposes of this paper, we take $\alpha = 2 > 1$ and $\beta = 0$. However, it is possible to pick α, β to define a desired error state transition matrix A_H. This will effect the Lyapunov function level sets and contracting rates, but will maintain the stability and convergence properties demonstrated above. ■

Theorem 2 *The MRAC system given by 11 with control law 13 and parameter update laws 14 exhibits global uniform asymptotic tracking of the reference model dynamics 10, 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 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 policies 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:

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

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.

$\dot{x} = Ax + \lambda B_r$ ■

A.2. Adaptive Controller: Nonlinear System

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

$$u = \hat{k}_\theta \sin\theta + \hat{k}_\omega \dot{\theta} + \hat{k}_u u_r - \frac{1}{c_r}\hat{k}_u a_r[(\sin\theta - \sin\theta_r) + e_\theta] \tag{13}$$

$$\dot{\hat{K}}_x = -\Gamma_x \zeta^T P B_r, \quad \dot{\hat{k}}_u = -\gamma_u \xi e^T P B_r \tag{14}$$

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

Proof We proceed in a manner similar to the proof in A.1. The ideal control law provides perfect tracking of the reference system:

$$u^* = k_\theta \sin\theta + k_\omega \dot{\theta} + k_u u_r \tag{26}$$

With $k_\theta,\ _\omega,\ k_u$ satisfying the matching conditions:

$$a + \lambda c_r k_\theta = a_r, \quad b + \lambda c_r k_\omega = b_r, \quad \lambda k_u = 1 \tag{27}$$

Consider the following adaptive control law:

$$u = \hat{k}_\theta \theta + \hat{k}_\omega \dot{\theta} + \hat{k}_u u_r - \frac{1}{c_r}\hat{k}_u a_r(\sin\theta - \sin\theta_r) - \frac{\alpha}{c_r}\hat{k}_u a_r e_\theta + \frac{\beta}{c_r}\hat{k}_u b_r e_\omega \tag{28}$$

With $\alpha > 0,\ \beta \geq 0$. Unless otherwise redefined, variables are defined in the same manner as in A.1. Applying the matching conditions, we then determine the error dynamics:

$$\begin{aligned}
\ddot{\theta} - \ddot{\theta}_r &= (a_r - \lambda c_r k_\theta)\sin\theta - a_r \sin\theta_r + (b_r - \lambda c_r k_\omega)\dot{\theta} - b_r \dot{\theta}_r + c_r \lambda \hat{k}_\theta \theta + c_r \lambda \hat{k}_\omega \dot{\theta} \\
&\quad - \lambda \hat{k}_u a_r(\sin\theta - \sin\theta_r) - \alpha \lambda \hat{k}_u a_r e_\theta + \beta \lambda \hat{k}_u b_r e_\omega + c_r \lambda \hat{k}_u u_r - c_r \lambda k_u u_r \\
\ddot{\theta} - \ddot{\theta}_r &= -\alpha a_r e_\theta + (1+\beta) b_r e_\omega \\
&\quad + \lambda c_r [\tilde{k}_\theta \sin\theta + \tilde{k}_\omega \dot{\theta} + \tilde{k}_u (u_r - \frac{1}{c_r} a_r(\sin\theta - \sin\theta_r) - \frac{\alpha}{c_r} a_r e_\theta + \frac{\beta}{c_r} b_r e_\omega)]
\end{aligned}$$

Recall that $a_r > 0,\ b_r < 0,\ \alpha > 0,\ \beta \geq 0$. Then we define the Hurwitz matrix:

$$A_H \coloneqq \begin{pmatrix} 0 & 1 \\ -\alpha a_r & (1+\beta) b_r \end{pmatrix}$$

We also define the parameter error vector $\tilde{K}_x = \begin{pmatrix} \tilde{k}_\theta \\ \tilde{k}_\omega \end{pmatrix}$, the augmented input $\xi(t) \coloneqq u_r(t) - \frac{1}{c_r} a_r(\sin\theta(t) - \sin\theta_r(t)) - \frac{\alpha}{c_r} a_r e_\theta(t) + \frac{\beta}{c_r} b_r e_\omega(t)$ and the augmented state $\zeta \coloneqq [\sin\theta,\ \dot{\theta}]^T$ The error dynamics are then compactly represented as:

$$\dot{e} = A_H e + \lambda B_r [\tilde{K}_x^T \zeta + \tilde{k}_u \xi] \tag{29}$$

Construct the following Lyapunov function:

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

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:

$$\dot{\hat{K}}_x = -\Gamma_x^T P B_r$$

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

Thus V is a Lyapunov function and tracking error and parameter estimation errors are bounded. For the purposes of this paper, we take $\alpha = 1, \beta = 0$ ■

Theorem 4

Proof The proof follows from the proof of Theorem 2 ■