ASEN 6519 Final Project REPORT

Mark Stephenson

TOTAL POINTS

(not graded)

QUESTION 1

1 Final Project Report

- 0 pts Correct

Practical Generalizable Methods for Learning Asymptotically Stabilizing Controllers

Mark A. Stephenson

***Abstract*—While control of dynamic systems with reinforcement learning is a commonly-studied problem, control for asymptotic convergence is not often considered outside of the context of formal guarantees. This paper presents a set of methods for learning asymptotically stabilizing controllers that are practical for real-world applications. Transformations to the states, observations, and actions in the Markov decision process are implemented to encourage asymptotical stabilization while avoiding issues with numerical conditioning. The methods are demonstrated on a simple wheel control problem, and the resulting controller is compared to a Lyapunov-based controller. The controller is then tested on an uncertain version of the environment to demonstrate advantages of the reinforcement learning-based control over classical solutions.**

I. Introduction

LOW-LEVEL control of dynamic systems is a common application for reinforcement learning (RL). Broadly, two classes of low-level control problems with RL are well-studied in the literature:

1) Many papers studying deep reinforcement learning (DRL) for control are applied to classic continuous state and action space environment such as half-cheeta, hopper, and cart-pole [1]. These environments tend to optimize for a high-level objective such as maximizing distance traveled or maintaining the system near an unstable equilibrium. While these environments are useful for benchmarking DRL algorithms, the threshold for success is in some ways low: if the solution looks good, it is good. This idea extends to physical robotics applications, where the threshold for success is often whether the robot can perform a task.
2) The other class of papers looks at control from a formal perspective: What guarantees — such as stability and asymptotic convergence — can be made for a neural controller? Much of the work in this domain uses a falsification based approach to learning a Lyapunov function to prove stability: This consists of a cycle of training a candidate function, then finding counterexamples where the candidate is not Lyapunov [2]–[4]. Some approaches leverage network architecture to guarantee certain properties of the Lyapunov function regardless of weights and biases [5]. Control functions are learned in conjunction with the Lyapunov function in some methods [3], [4].

This leaves a poorly-studied gap in the literature: Systems where provable stability is overkill, but a greater degree of precision is needed than "good enough" control. These applications include systems where many control occurs over many orders of magnitude to asymptotic-like convergence, such as spacecraft attitude regulation, rendezvous and proximity operations, and precision robotic manipulation.

This paper presents a set of methods for learning asymptotically stabilizing controllers that are practical for real-world applications. They consist of a set of transformations to the Markov decision process (MDP) representation of the problem that encourage learning across orders of magnitude by numerically conditioning the state, action, and reward to avoid issues of vanishing gradients and overstepping in learning. The methods are demonstrated on a simple wheel control problem, and the resulting controller is compared to a Lyapunov-based controller. The controller is then tested on an uncertain version of the environment to demonstrate advantages of the RL-based control over classical solutions.

II. Motivation

Dynamic systems are defined by

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{1}$$

where $\boldsymbol{x}$ is the state, $\boldsymbol{u}$ is the control input, and $\boldsymbol{f}$ describes the system dynamics. The goal of control is to find a control input $\boldsymbol{u}$ that drives the system state $\boldsymbol{x}$ to a desired state $\boldsymbol{x}_r$, minimizing the error

$$\delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_r(t) \tag{2}$$

between the state and a reference trajectory $\boldsymbol{x}_r(t)$. In regulation problems, $\dot{\boldsymbol{x}}_r = 0 \ \forall \ t$, while in tracking problems the reference is time-varying. In many applications, such as spacecraft attitude control and precision tasks with robotic manipulators, it is desireable to asymptotically drive the error to zero.

A number of classical techniques exist to find a control law $u(\boldsymbol{x}, \delta\boldsymbol{x})$ for a known $\boldsymbol{f}$. These include results from linear and nonlinear control theory as well as optimization-based methods like model predictive control.

A. Control as an MDP

The control problem can be posed as an MDP consisting of the tuple $(\mathcal{S}, \mathcal{A}, R, T, \gamma)$. The state space $\mathcal{S}$ consists of all

This work utilized the Alpine high performance computing resource at the University of Colorado Boulder. Alpine is jointly funded by the University of Colorado Boulder, the University of Colorado Anschutz, Colorado State University, and the National Science Foundation (award 2201538).

M. A. Stephenson is with the Autonomous Vehicle Systems Lab at the University of Colorado, Boulder, CO 80303 USA (e-mail: Mark.A.Stephenson@colorado.edu).

possible states $\boldsymbol{x}$ and state errors $\delta\boldsymbol{x}$. The action space $\mathcal{A}$ consists of all possible control inputs $\boldsymbol{u}$. The reward function

$$R(s,a) = R_\delta(\delta x) + R_u(u) \tag{3}$$

where R_δ is a decreasing function that rewards smaller errors, and R_u is optionally a decreasing function that penalizes larger control inputs. The generative transition function $T(s,a,s')$ deterministically steps the state using $\boldsymbol{f}$ and reference forward in time by Δt.

For continuous MDPs, DRL algorithms like proximal policy optimization (PPO) can be used to find a control law policy $a = \pi(s)$ represented by a neural network [6]. DRL is notoriously sensitive to the order of magnitude of actions, observations, and rewards because of how stochastic gradient descent functions [7]. Asymptotic control problems take place over a wide range of magnitudes: large controls are needed when errors are large, and controls get asymptotically smaller with the error. As a result, standard techniques for numerically conditioning the observation, action, and reward through normalization do not work well; when state errors are small, relatively small policy updates have a large effect on the control.

The MDP representation of the problem does have some advantages over classical techniques. Only a generative model of the system is needed, which could be a physical system with no explicitly known $\boldsymbol{f}$. Random failures, disturbances, and underactuated systems that are difficult to represent in classical control approaches can be easily accounted for in an RL-based solution.

III. Methods

Three methods are applied to an MDP representation of the dynamic system for use in training.

1) Rewards are defined as a logarithmic function of the error of a minimal set of state error elements.
2) Actions produce a control scaled by the current magnitude of state error.
3) Observations of state error $\delta\boldsymbol{x}$ are scaled logarithmically.

Combined, this produces an MDP which appears to remain on the same order of magnitude with respect to states, actions, and observations as the dynamic system converges to lower orders of magnitude.

The MDP is considered for a K-step horizon, and it is assumed that states in $\boldsymbol{x}$ are normalized to be in the domain $[-1,1]$, though this normalization is not strictly required.

A. Minimal Logarithmic Rewards

Two techniques are used to create a reward function: 1) minimizing the number of error terms that contribute to reward and 2) logarithmically scaling the error to encourage learning across orders of magnitude.

It is observed that reference trajectories are often overspecified; for example, if $\boldsymbol{x} = \{x, \dot{x}\}$, defining $x_r(t)$ implicitly defines $\dot{x}_r(t)$. Let the minimal set of elements needed to define the trajectory $\boldsymbol{x}_r$ be $\boldsymbol{x}_r^*$.

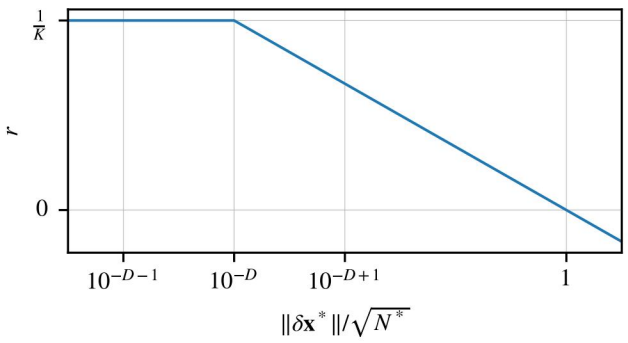

Fig. 1: Reward function (Equation 6) with logarithmically scaled x-axis.

Definition 1: The set of elements $\boldsymbol{x}^* \subset \boldsymbol{x}$ is representative for the system $\mathbf{0} = \boldsymbol{f}(\boldsymbol{x})$ if it satisfies a constructability condition

$$\exists\, G \quad \text{s.t.} \quad x_i = G(\boldsymbol{x}^*)\ \forall\ x_i \notin \boldsymbol{x}^* \tag{4}$$

and a nonredundancy condition

$$\nexists\, G \quad \text{s.t.} \quad x_i = G(\boldsymbol{x}^* \backslash x_i)\ \forall\ x_i \in \boldsymbol{x}^*. \tag{5}$$

All $x_i \in \boldsymbol{x}$ can be reconstructed from $\boldsymbol{x}^*$ by definition: if $x_i \notin \boldsymbol{x}^*$, there exists an operator G such that $x_i = G(\boldsymbol{x}^*)$; if $x_i \in \boldsymbol{x}^*$, then x_i is trivially known.

While $\boldsymbol{x}_r^*$ is not necessarily uniquely defined, the number of elements $N^* = |\boldsymbol{x}_r^*|$ is a local minimum (and is conjectured to be a unique minimum): Proof 1 shows that no element can be removed from a representative sets while maintaining representativeness.

The reward function is set to a logarithmic function of the representative set of elements, resolving up to D decimal digits of error, as shown in Figure 1:

$$r = \frac{1}{K} \min\left(-\frac{1}{D} \log_{10} \frac{\|\delta\boldsymbol{x}^*\|}{\sqrt{N^*}}, 1 \right). \tag{6}$$

Clamping rewards for accuracy beyond D digits is important in relation to the observation scaling function, Equation 8; the reward function should not be able to reward accuracy beyond what can be observed.

This reward function displays a number of other practical properties. While normalization is preferred, its is sensibly defined for unnormalized states, i.e. $\|\delta\boldsymbol{x}^*\|/\sqrt{N^*} > 1$. The reward is scaled such that the MDP has a maximum episode reward of 1 if $\|\delta\boldsymbol{x}^*\|/\sqrt{N^*} \leq 10^{-D}$ at all K steps, and a minimum episode reward ≤ 0 if $\|\delta\boldsymbol{x}^*\|/\sqrt{N^*} \geq 1$ at all steps.

The intuition for using a representative subset of error terms is to avoid diluting the reward signal. For example, consider a system with $\boldsymbol{x}$ consisting of x and the first $N-1$ derivatives of x. With a large N, a policy could track the derivative terms perfectly while maintaining a large error in x and receive a comparable — if not higher — reward than one that tracks x after an initial transient to reduce the state error. Of course, if $\boldsymbol{f}$ is a black box, no reduction can be made.

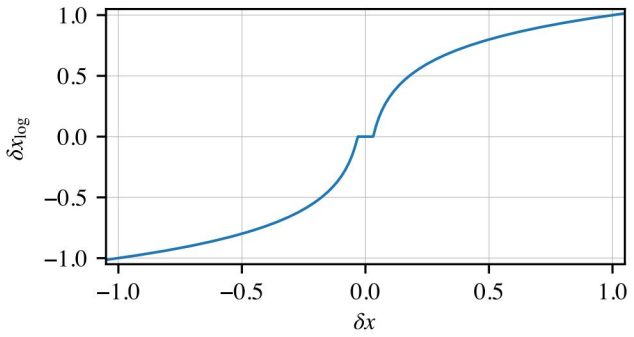

Fig. 2: Logarithmic observation transformation (Equation 8) with $D = 1.5$.

B. Logarithmically-Scaled Observations

For the control problem of tracking $\boldsymbol{x}_r(t)$ with $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$, the maximum observable information is

$$\boldsymbol{o} = \{\boldsymbol{x}, \delta\boldsymbol{x}, \dot{\boldsymbol{x}}_r\}, \tag{7}$$

noting that $\boldsymbol{x}_r$ is excluded because it is trivially computed from other terms.

If $\boldsymbol{f}$ is known, only $\delta\boldsymbol{x}$ and terms necessary to compute the dynamics of $\delta\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) - \dot{\boldsymbol{x}}_r(t)$ are relevant to the observation; call this minimal state $\boldsymbol{o}^*$. As with the reward function, if $\boldsymbol{f}$ is a black-box this type of reduction if not possible.

Observations of error terms can be scaled logarithmically to encourage learning across orders of magnitude. This transformation, plotted in Figure 2, is defined as

$$\delta x_{\log,i} = \operatorname{sgn}(\delta x_i) \max\left(\frac{\log_{10}|\delta x_i| + D}{D}, 0\right). \tag{8}$$

Equation 8 resolves up to D decimal digits of error ($\|\delta x\| \geq 10^{-D}$) and is sign-preserving in the observable domain.

Using the these methods, the observation passed to the learning agent is

$$\boldsymbol{o}^*_{\log} = \{\delta\boldsymbol{x}, \delta\boldsymbol{x}_{\log}, (\boldsymbol{x}, \dot{\boldsymbol{x}}_r)^*\}. \tag{9}$$

C. Error-Scaled Actions

The final transformation made to the MDP is to scale the control action by the magnitude of the state error.[1] This is motivated by the assumption that the magnitude of the control input is typically proportional to the magnitude of the error. Assuming each actuator has some maximum control input $u_{\max,i}$, Equation 10 maps $a_i \in [-1, 1] \to [-u_{\max,i}, u_{\max,i}]$ when the error is one, and $a_i \in [-1/\epsilon, 1/\epsilon] \to [-u_{\max,i}, u_{\max,i}]$ when the error is zero:

$$u_i = u_{\max,i}\left(\sqrt{\frac{\|\delta\boldsymbol{x}^*\|^2 + \|\delta\dot{\boldsymbol{x}}^*\|^2}{2N^*}} + \epsilon\right) a_i + u_{ff}(\boldsymbol{x}) \tag{10}$$

The representative subset of the error state and its normalized derivative are used to scale the action, with the latter present to account for the need to exert control when $\delta\dot{\boldsymbol{x}}^* \neq 0$. A small $0 < \epsilon \ll 1$ is added to allow for control even when perfectly tracking the reference trajectory to account for disturbances or nonzero higher derivatives in the reference. If the system has a known feedforward control, it can be accounted for by $u_{ff}(\boldsymbol{x})$, making the assumption that the magnitude of error and control are proportional more reasonable.

[1]Exponential scaling of the action using the inverse of Equation 8 was considered but found to be ineffective.

IV. Results

A simple dynamics problem is described to demonstrate the proposed methods. The effect of each transformation is compared to the combined effect of all methods. The resulting RL-based controller is compared to a Lyapunov-based controller. Then, an uncertain version of the environment is used to demonstrate advantages of the RL-based control over classical solutions.

A. Wheel Control Problem

A one degree of freedom problem is considered, consisting of a disk of inertia $I = 10\ \mathrm{kg{\cdot}m^2}$ being controlled by torque $|u| \leq 1\ \mathrm{N{\cdot}m}$, with a control frequency of 2 Hz. The system state is

$$\boldsymbol{x} = \{\theta, \dot{\theta}\} \tag{11}$$

with equations of motion

$$\dot{\boldsymbol{x}} = \begin{Bmatrix}\dot{\theta} \\ \ddot{\theta}\end{Bmatrix} = \begin{Bmatrix}\dot{\theta} \\ u/I\end{Bmatrix} = \boldsymbol{f}(\boldsymbol{x}) \tag{12}$$

The objective of the control problem is for a marker on the disk to either regulate to a fixed angle

$$\theta_r(t) = k \tag{13}$$

or track some trajectory, in this case a random sinusoid with maximum $\dot{\theta} < 0.1\ \mathrm{rad/s}$

$$\theta_r(t) = A * \cos(ft + \phi). \tag{14}$$

1) Lyapunov Controller: Using Lyapunov theory, an asymptotically stabilizing control law

$$u = I(-P\delta\dot{\theta} - \delta\theta + \ddot{\theta}_r) \tag{15}$$

can be found (Proof 2).

2) MDP Expression: Using the reward, observation, and action transformations previously described, the MDP uses the following elements: The reward function (Equation 6) and action function (Equation 10) use the representative set of error terms

$$\delta\boldsymbol{x}^* = \{\delta\theta\}. \tag{16}$$

Parameters $D = 7$ and $\epsilon = 10^{-9}$ are used, with 300 second ($K = 600$ step) episodes. The observation spaces consists of

$$\delta\boldsymbol{o}^*_{\log} = \{\delta\theta, \delta\dot{\theta}, \delta\theta_{\log}, \delta\dot{\theta}_{\log}, \ddot{\theta}_r\} \tag{17}$$

which can be used to express the dynamics of $\delta\boldsymbol{x}$ and are the same terms that appear in the Lyapunov control law. No feedforward control is used.

B. Training

For each combination of reward, observation, and action transformation, a policy was trained for 3M steps using the Stable Baselines 3 implementation of PPO over both regulation and tracking tasks [8]. The discount factor γ was 0.99. Separate 2×64 neural networks were used for the policy and value function, with a batch size of 50. A learning rate of 0.0001 was used to mitigate instability seen in some trials. Other parameters used the Stable Baselines 3 defaults.

Table I summarizes the performance of each combination of transformations with by reporting the average and standard error of the mean of 3 training runs. Since each trial's results are collected from an average of final training episodes, they are for a policy with exploration and may have slightly depressed performance from the actual policy. In addition to episode final states, the percent of maximum control effort Σu (where 100% indicates that the controller always returns $\pm u_{\max}$) and the percent of chatter (where 100% indicates that the controller switches sign every step) are reported.

Individually, each transformation provides only a modest improvement over the baseline, which uses untransformed actions and observations and the reward function $r = 1 - \delta\theta$. Reward and action transformations combined demonstrate an order-of-magnitude improvement across the error metrics. However, all three transformations are needed to maximize performance. For regulation tasks, the all-transformations controller reaches the minimum error observable with $D = 7$; for tracking tasks, the controller performs at least two orders of magintude better than the baseline. The all-transformations controller also is the only one to maintain a low control effort and not produce a bang-bang type controller; while other controllers that were trained with the action transformation initially showed low control effort as training converged, control effort eventually grew over time without improving performance.

C. Performance

The policy trained with all transformations is compared to the Lyapunov controller for a regulation and tracking case. In the regulation case, Figure 3, a high initial angular velocity is given ($\dot{\theta}_0 = \pi/2$ rad/s) which well outside the training bounds of $|\dot{\theta}_0| < 0.1$ rad/s. The RL controller is able to despin and drive the error to the level resolveable by the observation transformation. The Lyapunov controller is able to drive the error to numerical precision, but with a somewhat longer settling time; while this could be tuned to be more aggressive, the RL solution directly optimizes for minimizing error over all time.

In the tracking case, Figure 4, a reference function not seen in training is given to the system; the reference gets more challenging over time due to an increasing frequency oscillation. The RL controller is inferior to the Lyapunov controller, but still performs well. Noteably, the RL-based policy finds a solution that is very similar to the Lyapunov-based law's output.

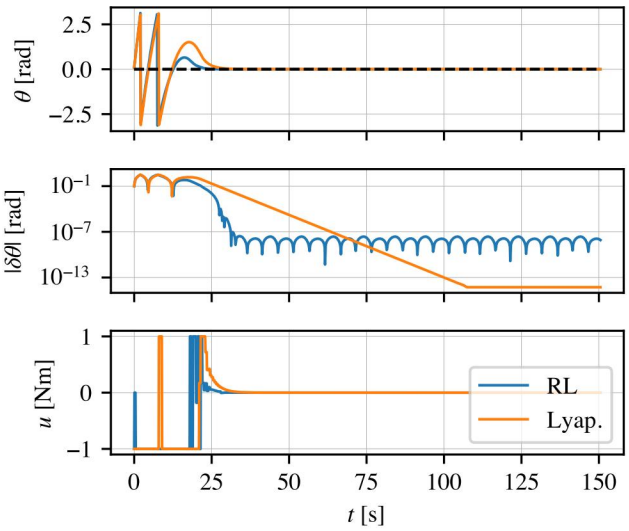

Fig. 3: Controller performance on a regulation case with high initial velocity.

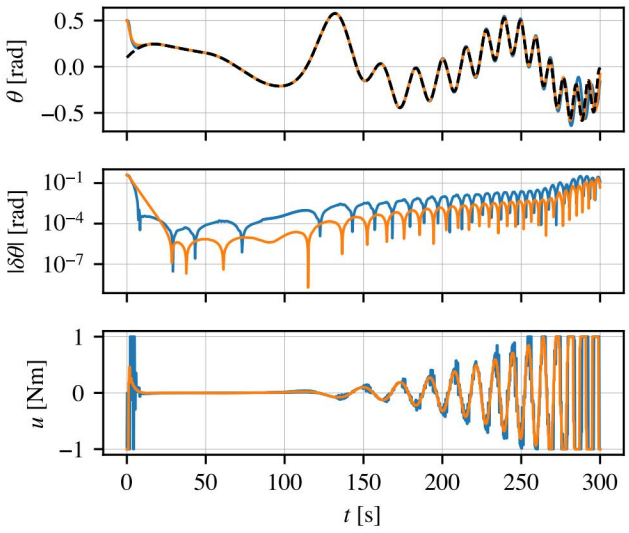

Fig. 4: Controller performance on a tracking case with increasing frequency.

D. Performance Under Uncertainty

A non-ideal case is considered with a nonlinear stick-slip friction model. The Stribeck model is used, which accounts for Coulumb friction L_C, breakaway friction L_{brk}, and viscous friction $f\dot{\theta}$ [9]:

$$L_f = \sqrt{2e}(L_{brk} - L_C) \exp\left(-\left(\frac{v}{\dot{\theta}_{brk}\sqrt{2}}\right)^2\right) \frac{\dot{\theta}}{\dot{\theta}_{brk}\sqrt{2}} + L_C \tanh\frac{10\dot{\theta}}{\dot{\theta}_{brk}} + f\dot{\theta} \quad (18)$$

The environment uses random values for the friction parameters in the ranges given in Table II, representing uncertainty

TABLE I: Performance of different combinations of reward, action, and observation transformations. † excludes a run that demonstrated no learning. †† includes some runs that converged to local minima; best runs are not superior to other configurations.

Transformed			**Regulation**		**Tracking**			
r	a	o	$\delta\theta$ [rad]	$\delta\dot{\theta}$ [rad/s]	$\delta\theta$ [rad]	$\delta\dot{\theta}$ [rad/s]	Σu [%]	Chatter [%]
			$(5.5 \pm 0.0) \times 10^{-2}$	$(4.2 \pm 0.0) \times 10^{-2}$	$(5.7 \pm 0.2) \times 10^{-2}$	$(4.1 \pm 0.1) \times 10^{-2}$	98.5 ± 0.1	55.3 ± 0.6
✓			$(1.3 \pm 0.0) \times 10^{-2}$	$(2.8 \pm 0.0) \times 10^{-2}$	$(1.3 \pm 0.0) \times 10^{-2}$	$(2.9 \pm 0.1) \times 10^{-2}$	99.1 ± 0.1	69.7 ± 0.3
	✓		$(2.9 \pm 0.3) \times 10^{-2}$	$(2.0 \pm 0.2) \times 10^{-2}$	$(2.8 \pm 0.2) \times 10^{-2}$	$(1.9 \pm 0.2) \times 10^{-2}$	45.3 ± 5.0	55.5 ± 1.2
		✓	$(2.1 \pm 0.7) \times 10^{-1}$	$(4.6 \pm 0.3) \times 10^{-2}$	$(2.0 \pm 0.6) \times 10^{-1}$	$(4.5 \pm 0.4) \times 10^{-2}$	98.3 ± 0.7	$64.1 \pm 0.8^{\dagger\dagger}$
✓	✓		$(2.6 \pm 0.0) \times 10^{-3}$	$(7.4 \pm 0.5) \times 10^{-3}$	$(2.9 \pm 0.0) \times 10^{-3}$	$(8.6 \pm 0.6) \times 10^{-3}$	30.9 ± 0.9	$63.3 \pm 0.2^{\dagger}$
✓		✓	$(1.3 \pm 0.0) \times 10^{-2}$	$(2.9 \pm 0.0) \times 10^{-2}$	$(1.3 \pm 0.0) \times 10^{-2}$	$(2.8 \pm 0.0) \times 10^{-2}$	98.9 ± 0.1	66.7 ± 0.1
	✓	✓	$(9.4 \pm 4.4) \times 10^{-2}$	$(2.9 \pm 0.5) \times 10^{-2}$	$(9.9 \pm 4.4) \times 10^{-2}$	$(3.0 \pm 0.5) \times 10^{-2}$	41.6 ± 15.9	$54.6 \pm 6.9^{\dagger\dagger}$
✓	✓	✓	$\mathbf{(1.7 \pm 0.9) \times 10^{-6}}$	$\mathbf{(9.5 \pm 4.5) \times 10^{-7}}$	$\mathbf{(4.6 \pm 0.4) \times 10^{-4}}$	$\mathbf{(9.3 \pm 1.6) \times 10^{-4}}$	$\mathbf{5.3 \pm 0.3}$	$\mathbf{45.4 \pm 3.6}$

TABLE II: Values used for the friction model in Equation 18.

Value	**Low**	**Nominal**	**High**
$\dot{\theta}_{brk}$ [rad/s]	1×10^{-4}	1×10^{-3}	2×10^{-3}
L_{brk} [N·m]	0.1	0.2	0.3
L_C [N·m]	0.005	0.01	0.015
f [-]	1.0	1.25	1.5

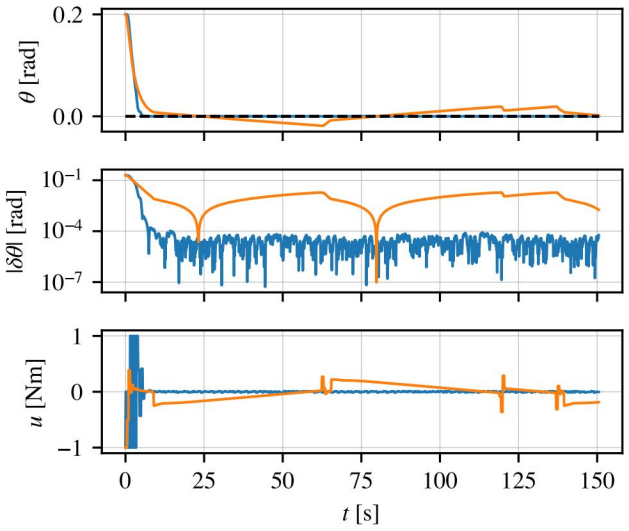

Fig. 5: Friction-retrained controller performance on a regulation case with randomized nonlinear friction shows RL outperforming a friction compensated Lyapunov controller.

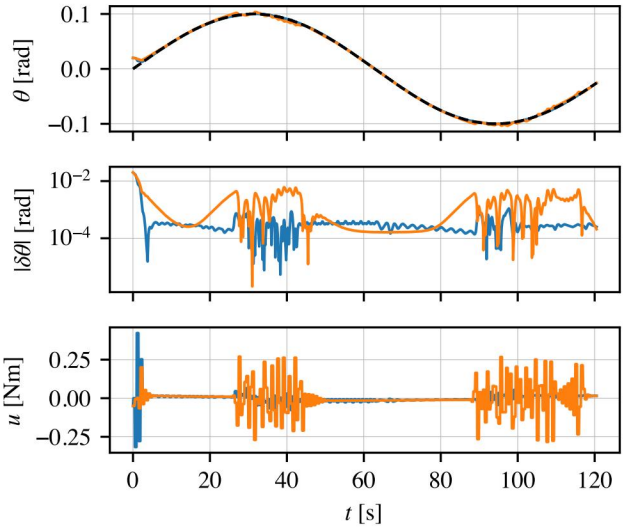

Fig. 6: Friction-retrained controller performance on a tracking case with randomized nonlinear friction.

due to temperature, lubrication, motor wear, and other factors. The RL controller is retrained over varied friction parameters, while the Lyapunov controller is given a feedforward term using the nominal friction parameters; due to a lack of certainty equivalence, it is expected that the RL controller will outperform the Lyapunov controller.

When regulating the uncertain system, the RL controller is vastly superior as shown in Figure 5. Due to the large nonlinearities near zero velocity, the feedforward term prevents convergence due to poor estimates of the friction force compensation.

A similar behavior is observed at low-velocity points in tracking problems, as in Figure 6. While the Lyapunov controller chatters near the low-velocity sticking points due to mismodeling of feedforward terms, the RL controller maintains a calmer and more accurate trajectory, having been trained over a range of friction models. Compensating stick-slip friction is also difficult for the Lyapunov controller because the stick-slip dynamics are faster than the control frequency.

V. Conclusion

The three MDP transformations proposed in this paper — logarithmic scaling of the state and observation spaces and error scaling of the action spaces — are shown to be effective in numerically conditioning problems for RL algorithms. While unable to outperform classical control methods in ideal cases, performance is still orders of magnitude better than the naïve approach. In cases with uncertainy, the RL methods outperform classical control methods since they are able to learn a policy that is robust to the uncertainty.

Future[2] work includes applying these methods to more complex control problems with a higher dimensionality and/or problems that do not have classical solutions such as underactuated control.

[2]Current work for my ASEN 6010 — Advanced Spacecraft Dynamics project.

ACKNOWLEDGMENT

Thanks to John Martin and Giovanni Fereoli for entertaining discussion and giving advice on this research.

CONTRIBUTIONS AND RELEASE

Mark was the sole author of this work. The author grants permission for this report to be posted publicly.

APPENDIX

Proof 1: Given a representative set $\boldsymbol{x}^* \subset \boldsymbol{x}$ (Definition 1), no x_k can be removed from $\boldsymbol{x}^*$ while maintaining representativeness.

By contradiction, if $\boldsymbol{x}^* \backslash \{x_k\}$ is representative, it has the constructability property

$$\exists\ G \quad \text{s.t.} \quad x_i = G(\boldsymbol{x}^* \backslash \{x_k\})\ \forall\ x_i \notin \boldsymbol{x}^* \backslash \{x_k\}. \tag{19}$$

However, directly contradicts the nonredundancy property of $\boldsymbol{x}^*$ (Equation 5) for $x_i = x_k$. Therefore, either $\boldsymbol{x}^*$ or $\boldsymbol{x}^* \backslash \{x_k\}$ is not representative and contradiction is demonstrated.

Proof 2: For the equations of motion given in Equation 12, the control law Equation 15 is asymptotically stabilizing. Consider the positive definite candidate Lyapunov function

$$V(\delta\theta, \delta\dot{\theta}) = \frac{1}{2}\delta\theta^2 + \frac{1}{2}\delta\dot{\theta}^2. \tag{20}$$

The closed-loop system is Lyapunov because $\dot{V}$ is negative semi-definite:

$$\dot{V}(\delta\theta, \delta\dot{\theta}) = \delta\theta\delta\dot{\theta} + \delta\dot{\theta}\left(\frac{u}{I} - \ddot{\theta}_r\right) = -P\delta\dot{\theta}^2 \tag{21}$$

The closed-loop system is asymptotic because on $\dot{V} = 0$, the odd derivative $\dddot{V}$ is negative definite [10]:

$$\dddot{V}(\delta\theta = 0, \delta\dot{\theta}) = -4P^3\delta\dot{\theta}^2 \tag{22}$$

REFERENCES

[1] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. de Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. J. S. Tan, and O. G. Younis, "Gymnasium," Oct. 2023, original-date: 2022-09-08T01:58:05Z. [Online]. Available: https://github.com/Farama-Foundation/Gymnasium

[2] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, "Formal Synthesis of Lyapunov Neural Networks," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 773–778, Jul. 2021, arXiv:2003.08910 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2003.08910

[3] G. Manfredi, L. De Cicco, and S. Mascolo, "On Asymptotic Stability of Nonlinear Systems with Deep Reinforcement Learning Controllers," in *2022 30th Mediterranean Conference on Control and Automation (MED)*. Vouliagmeni, Greece: IEEE, Jun. 2022, pp. 306–311. [Online]. Available: https://ieeexplore.ieee.org/document/9837155/

[4] Y.-C. Chang, N. Roohi, and S. Gao, "Neural Lyapunov Control," Sep. 2022, arXiv:2005.00611 [cs, eess, stat]. [Online]. Available: http://arxiv.org/abs/2005.00611

[5] N. Gaby, F. Zhang, and X. Ye, "Lyapunov-Net: A Deep Neural Network Architecture for Lyapunov Function Approximation," Aug. 2022, arXiv:2109.13359 [cs, math]. [Online]. Available: http://arxiv.org/abs/2109.13359

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017, arXiv:1707.06347 [cs]. [Online]. Available: http://arxiv.org/abs/1707.06347

[7] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep Reinforcement Learning That Matters," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11694

[8] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations."

[9] B. Armstrong and C. C. de Wit, "Friction Modeling and Compensation," in *The Control Handbook*. CRC Press, 1995.

[10] R. Mukherjee and D. Chen, "Asymptotic Stability Theorem for Autonomous Systems," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 5, Sep. 1993.

1 Final Project Report

- 0 pts Correct