Feedback Linearization for Unknown Systems via Reinforcement Learning

Tyler Westenbroek*, David Fridovich-Keil*, Eric Mazumdar*, Shreyas Arora, Valmik Prabhu,
S. Shankar Sastry, and Claire J. Tomlin

***Abstract*— We present a novel approach to control design for nonlinear systems, which leverages reinforcement learning techniques to learn a linearizing controller for a physical plant with unknown dynamics. Feedback linearization is a technique from nonlinear control which renders the input-output dynamics of a nonlinear plant *linear* under application of an appropriate feedback controller. Once a linearizing controller has been constructed, desired output trajectories for the nonlinear plant can be tracked using a variety of linear control techniques. A single learned policy then serves to track arbitrary desired reference signals provided by a higher-level planner. We present theoretical results which provide conditions under which the learning problem has a unique solution which exactly linearizes the plant. We demonstrate the performance of our approach on two simulated problems and a physical robotic platform. For the simulated environments, we observe that the learned feedback linearizing policies can achieve arbitrary tracking of reference trajectories for a fully actuated double pendulum and a 14 dimensional quadrotor. In hardware, we demonstrate that our approach significantly improves tracking performance on a 7-DOF Baxter robot after less than two hours of training.**

I. Introduction

Recent progress in the reinforcement learning (RL) community [1–5] has renewed a debate on the utility and role of models in controlling uncertain robotic systems. In this paper, we present a unifying viewpoint in which RL algorithms provide a mechanism for computing a reference tracking controller. This controller may then be used modularly in a variety of hierarchical control and planning schemes.

Specifically, this paper focuses on tracking desired output trajectories for a special class of nonlinear systems using a technique from geometric control theory known as *feedback linearization*. Feedback linearization renders the input-output behavior of a nonlinear system *linear* via application of an appropriately chosen control law. Desired output trajectories for the plant can then be generated using a linear reference model and tracked using well-established techniques from linear systems theory, such as LQR [6] or linear MPC [7].

However, the primary drawback of feedback linearization is that it requires accurate knowledge of the plant's dynamics. Many real-world robotic systems display dynamics with parameters that may be difficult to identify and nonlinearities which may be impractical to incorporate into a system dynamics model. While there have been extensive efforts to develop robust forms of feedback linearization using combinations of feedback and adaptation [8–15], current methods in the literature make strong structural assumptions about the plant's nonlinearities. This is highlighted in the case of multiple-input multiple-output nonlinear systems, for which the above references either assume that there is no coupling between the system inputs, or that a highly structured parametric representation of this coupling is available.

EECS, UC Berkeley. westenbroekt@berkeley.edu.
* indicates equal contribution.

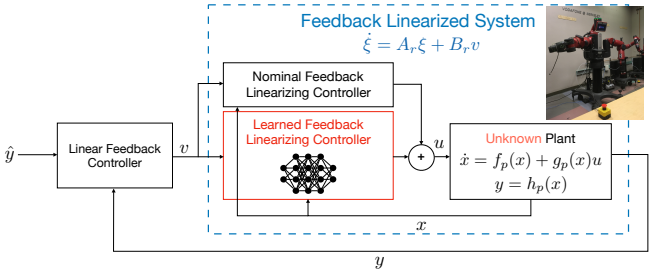

Fig. 1: Schematic diagram of our framework. By learning an appropriate feedback linearizing controller, we render an initially unknown nonlinear system (with state x, output y, and input u) *linear* in an auxiliary input, v. In order to track desired output trajectories $\hat{y}$, v may follow a linear feedback control design, e.g., LQR. We demonstrate our approach in a variety of systems, including a Baxter robot arm (pictured).

In sharp contrast to these methods, we propose a framework for constructing a linearizing controller for a plant with unknown dynamics using policy optimization algorithms from reinforcement learning. Our approach requires no *a priori* information about the structure of the coupling between the inputs and outputs of the plant. While our approach can naturally incorporate information from a nominal dynamics model into the learning process, it can also be applied when nothing but the structure of the linear reference model is known. Specifically, our approach begins by constructing a linearizing controller for the nominal dynamics model (if available). Then, it augments this nominal controller with an *arbitrarily structured* parametric component. The parameters of this learned component are trained using a reinforcement signal which encourages actions which better match the desired input-output behaviour described by the liner reference model. We demonstrate that for linearly parameterized controllers, the resulting optimization problem is convex, meaning that globally optimal solutions can be found reliably. Additionally, we present conditions which guarantee that an exact linearizing control law can be recovered.

We evaluate our framework in simulation, where it successfully learns to control a double pendulum (4 dimensional state) and a quadrotor (14 dimensional state) along arbitrary reference trajectories. We also demonstrate our method on a Baxter robot arm. In each case, a single learned linearizing control law can accomplish multiple tasks and track multiple reference signals. We report significant improvements in tracking performance within one hour of training time.

II. Related Work

Most approaches for constructing linearizing control laws for plants with *a priori* unknown dynamics are based on linear adaptive control theory [8]. The earliest approaches employing *indirect adaptive control* generally assume that a parameterized model of the plant's true dynamics is available [8–12]. The model parameters are then updated online deterministically using data collected from the plant, and the refined dynamics model yields an improved linearizing control law. When accompanied by an appropriate (exponentially stabilizing) feedback law, such methods can be shown to track desired output signals asymptotically on the plant. A large body of subsequent work [16–18] has extended these results to more general classes of function approximators (i.e., neural networks) to approximate the systems dynamics and improve the linearizing control law. Recent efforts have also investigated the use of nonparametric methods for estimating the plant dynamics [13–15].

Frameworks employing *direct adaptive control* [19, 20] directly parameterize the linearizing controller for the system. These methods also propose deterministic online update laws and feedback control architecture which ensure asymptotic tracking of desired reference signals. As discussed above, each of these methods makes strong assumptions about the coupling of the input-output dynamics of the system. A notable exception to the above literature is [21], where a temporal differencing scheme is used to learn a linearizing controller for single-input single-output nonlinear systems. We build on this contribution by developing a framework for learning linearizing controllers for multiple-input multiple-output systems and by providing theoretical conditions under which an exact linearizing control law can be constructed.

III. Feedback Linearization

This section outlines how to compute input-output linearizing controllers for a known dynamics model. We refer the reader to [22], [23] for a more thorough introduction. In this paper, we consider square control affine systems of the form

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x), \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^q$ is the input and $y \in \mathbb{R}^q$ is the output. The mappings $f\colon \mathbb{R}^n \to \mathbb{R}^n$, $g\colon \mathbb{R}^n \to \mathbb{R}^{n\times q}$ and $h\colon \mathbb{R}^n \to \mathbb{R}^q$ are each assumed to be smooth. We restrict our attention to a compact subset $D \subset \mathbb{R}^n$ of the state space containing the origin.

A. Single-input single-output systems

We begin by introducing feedback linearization for single-input, single-output (SISO) systems (i.e., $q = 1$). In order to construct this control law, we take time derivatives of $y = h(x)$ until the input u appears, and then invert the relationship to enforce linear input-output behavior. We begin by examining the first time derivative of the output:

$$\dot{y} = \frac{dh}{dx}(x) \cdot \Big(f(x) + g(x)u\Big) \tag{2}$$

$$= \underbrace{\frac{dh}{dx}(x) \cdot f(x)}_{L_f h(x)} + \underbrace{\frac{dh}{dx}(x) \cdot g(x)}_{L_g h(x)} u \tag{3}$$

Here the terms $L_f h(x)$ and $L_g h(x)$ are known as *Lie derivatives* [22], and capture the rate of change of $y = h(x)$ along the vector fields f and g, respectively. In the case that $L_g h(x) \neq 0$ for each $x \in D$, we can exactly control $y = h(x)$ on D. In particular, consider the control

$$u(x, v) = \frac{1}{L_g h(x)}(-L_f h(x) + v) \ , \tag{4}$$

which when applied to the system exactly 'cancels out' the nonlinear portion of the differential equation and enforces the linear relationship $\dot{y} = v$. However, it may be the case that $L_g h(x) \equiv 0$ (that is, the input does not directly affect the first derivative of the output), in which case the control law (4) will be undefined. In general, we can differentiate y multiple times, until the input shows up in one of its higher order derivatives. Assuming that the input does not appear the first $\gamma - 1$ times we differentiate the output, the γ-th time derivative of y will be of the form

$$y^{(\gamma)} = L_f^\gamma h(x) + L_g L_f^{\gamma-1} h(x) u \tag{5}$$

Here, L_f^γ and $L_g L_f^{\gamma-1} h(x)$ are higher order Lie derivatives. More information on how to compute these nonlinear functions can be found in [22, Chapter 9]. If $L_g L_f^{\gamma-1} h(x) \neq 0$ for each $x \in D$ then the control law

$$u(x, v) = \frac{1}{L_g L_f^{\gamma-1} h(x)}\big(- L_f^\gamma h(x) + v\big) \tag{6}$$

enforces the relationship $y^{(\gamma)} = v$. γ is referred to as the *relative degree* of the nonlinear system.

B. Multiple-input multiple-output systems

Next, we consider (square) multiple-input, multiple-output (MIMO) systems, i.e., $q > 1$. Due to space constraints, we leave a full development of this case to [22, Chapter 9], but outline the main ideas here. As in the SISO case, we differentiate each of the output channels until at least one input appears. Let γ_j be the number of times we need to differentiate y_j (the j-th entry of y) for an input to appear. We then obtain an input-output relationship of the form :

$$[y_1^{(\gamma_1)}, \ldots, y_p^{(\gamma_p)}]^T = b(x) + A(x)u \tag{7}$$

The square matrix $A(x)$ is referred to as the *decoupling matrix* and $b(x)$ is know as the *drift term*. If $A(x)$ is nonsingular on D then we observe that the control law

$$u(x, v) = A^{-1}(x)(-b(x) + v) \tag{8}$$

where $v \in \mathbb{R}^q$ yields the decoupled linear system

$$[y_1^{\gamma_1}, y_2^{\gamma_2}, \ldots, y_q^{\gamma_q}]^T = [v_1, v_2, \ldots, v_q]^T, \tag{9}$$

where v_k is the k-th entry of v. We refer to $(\gamma_1, \gamma_2, \ldots, \gamma_q)$ as the *vector relative degree* of the system. The decoupled dynamics (9) can be compactly represented with the LTI system

$$\dot{\xi}_r = A_r \xi_r + B_r v_r \tag{10}$$

which we will hereafter refer to as the *reference model*. Here, we have collected the states $\xi_r = (y_1, \dot{y}_1, \ldots, \ldots, y_1^{\gamma_1}, \ldots, y_q, \ldots, y_q^{\gamma_q})$ and constructed A_r and B_r to represent the dynamics of (10).

IV. Directly Learning a Linearizing Controller

In this work, we will examine how to construct a linearizing controller for a physical plant of the form (1) with unknown dynamics

$$\begin{aligned} \dot{x} &= f_p(x) + g_p(x)u \\ y &= h_p(x) \end{aligned} \tag{11}$$

starting from the linearizing controller for the model system

$$\begin{aligned} \dot{x} &= f_m(x) + g_m(x)u \\ y &= h_m(x) \end{aligned} \tag{12}$$

which represents our "best guess" for the true dynamics of the plant. We make the following standard assumption:

Assumption 1: The model system (12) and plant (11) both have the same vector relative degree $\gamma_1^m, \gamma_2^m, \ldots, \gamma_q^m$ on some compact set $D \subset \mathbb{R}^n$.

With this assumption in place, we know that there exists linearizing controllers of the form

$$u_m(x, v) = \beta_m(x) + \alpha_m(x)v \tag{13}$$

$$u_p(x, v) = \beta_p(x) + \alpha_p(x)v \tag{14}$$

for the physical model and plant, respectively. We can construct u_m using the techniques discussed above, but the terms in u_p are unknown. However, we do know that

$$\beta_p(x) = \beta_m(x) + \Delta\beta(x) \tag{15}$$

$$\alpha_p(x) = \alpha_m(x) + \Delta\alpha(x) \tag{16}$$

for some continuous functions $\Delta\beta$ and $\Delta\alpha$. We construct parameterized estimates for these functions:

$$\Delta\beta(x) \approx \beta_{\theta_1}(x), \ \Delta\alpha(x) \approx \alpha_{\theta_2}(x) \tag{17}$$

Here, $\theta_1 \in \Theta_1 \subset \mathbb{R}_1^{K_1}$ and $\theta_2 \in \Theta_2 \subset \mathbb{R}^{K_2}$ are parameters to be trained by running experiments on the plant. We will assume that Θ_1 and Θ_2 are convex compact sets, and we will frequently abbreviate $\theta = (\theta_1, \theta_2) \in \Theta_1 \times \Theta_2 := \Theta$. We assume that β_{θ_1} and (α_{θ_2}) are continuous in x and continuously differentiable in θ_1 and θ_2, respectively.

Altogether, for a given $\theta = (\theta_1, \theta_2) \in \Theta$ our estimate for the controller which exactly linearizes the plant is given by

$$\hat{u}_\theta(x, v) = [\beta_m(x) + \beta_{\theta_1}(x)] + [\alpha_m(x) + \alpha_{\theta_2}(x)]v \tag{18}$$

In the case where no prior information about the dynamics of the plant is available (other than its vector relative degree), we simply remove u_m from $\hat{u}_\theta$ the above expression. Next we define a conceptual optimization problem which selects the parameters for the learned controller which, in a sense we will make precise shortly, best linearize the plant. We then describe a practical variant of this problem which is more amenable to real-world implementation.

A. Conceptual problem

From Section III we know that the input-output dynamics of the plant are of the form

$$y^\gamma = b_p(x) + A_p(x)u \tag{19}$$

where the terms b_p and A_p are unknown to us, and we have written the highest order derivatives as $y^\gamma = (y_1^{\gamma_1}, \ldots, y_2^{\gamma_1}, \ldots, y_q^{\gamma_q})^T$ to simplify notation. Under application of $\hat{u}_\theta$ the dynamics given by:

$$y^\gamma = b_p(x) + A_p(x)\Big([\beta_m(x) + \beta_{\theta_1}(x)] + [\alpha_m(x) + \alpha_{\theta_2}(x)]v \Big) \tag{20}$$

Letting $W_\theta(x, v)$ equal the right-hand side of the above expression, we would ideally like to find $\theta^* \in \Theta$ such that for each $x \in D$ and $v \in V$, $W_{\theta^*}(x, v) \approx v$ for each $x \in D$ and $v \in \mathbb{R}^m$. That is, we would ideally like our feedback linearizing controller to accurately control the highest order derivatives of our output. However, since the dynamics of the plant are unknown to us, we do not know the terms in $\pi_\theta(x, v)$, and thus we cannot directly solve for θ^*. Instead, we define the pointwise loss $\ell \colon \mathbb{R}^n \times \mathbb{R}^q \times \mathbb{R}^{K_1+K_2}$

$$l(x, v, \theta) = \|v - W_\theta(x, v)\|_2^2, \tag{21}$$

which provides a measure of how well the learned controller $\hat{u}_\theta$ linearizes the plant at the state x when the virtual input v is applied. We then specify a probability distribution X over $\mathbb{R}^n$ with support D, which we use to model our preference for having an accurate linearizing controller at different points in the state space. We let V be the uniform distribution over the set $\{v \in \mathbb{R}^q \colon \|v\| \leq 1\}$ and then define the weighted loss

$$L(\theta) = \mathbb{E}_{x \sim X, v \sim V}\, \ell(x, v, \theta) \tag{22}$$

and then define our optimal choice of the parameters for the learned controller by solving the following optimization problem:

$$\min_{\theta \in \Theta} L(\theta) \tag{23}$$

Although we do not know the terms in L, we can query this function by applying $u_\theta(x, v)$ at various points in the statespace and recording the resulting value of y^γ. Thus zeroth order optimization methods can be used to solve (23). In the following section, we formulate an approximation to this problem which is more directly amenable to policy gradient reinforcement learning algorithms.

Our insistence that X is supported on D and that V uniformly excites all directions in $\mathbb{R}^q$ is analogous to the *persistence of excitation* conditions commonly found in the adaptive control literature [8], and is crucial for the following parameter convergence results.

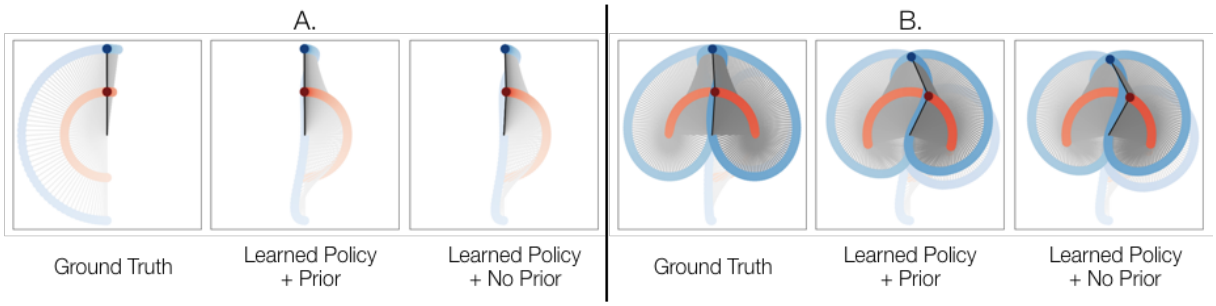

Fig. 2: Figures A) and B) display the trajectories from two distinct tracking tasks. In each plot 'Ground Truth' is the trajectory resulting from the feedback linearization of the true dynamics. The task in A) is a simple swing-up task with a constant reference signal. The task in B) is to follow a sinusoidal desired sinusoidal joint angles. The desired signal for the blue trajectory is given by $\pi cos(t)$, and the desired signal red joint is $-\pi sin(t)$.

Lemma 1: Suppose that there exists $\theta^* \in \Theta$ such that $\hat{u}_{\theta^*}(x, v) = u_p(x, v)$ for each $x \in D$ and $v \in V$. Then θ^* is a globally optimal solution to (23).

Proof: Note that if $u_{\theta^*}(x, v) = u_p(x, v)$ for each $x \in X$ and $v \in V$ then $L(\theta) = 0$. Moreover, we clearly have $L(\theta) \geq 0$ for each $\theta \in \Theta$. Thus, θ^* must be a global minimizer of the optimization (23). ■

However, (23) is generally a nonconvex optimization problem which means we cannot reliably find its globally optimal solution. Thus, we seek conditions on β_{θ_1} and α_{θ_2} which ensure that (23) is actually a convex optimization problem. In particular, we now consider the case where β_θ and α_θ take the form

$$\beta_{\theta_1}(x) = \sum_{k=1}^{K_1} \theta_k^1 \beta_k(x) \quad \text{and} \quad \alpha_{\theta_2}(x) = \sum_{k=1}^{K_2} \theta_k^2 \alpha_k(x) \quad (24)$$

where $\{\beta_k\}_{k=1}^{K_1}$ and $\{\alpha_k\}_{k=1}^{K_2}$ are nonlinear features, which are each assumed to be continuous functions. The proof of the following result can be found in the Appendix.

Lemma 2: Assume that β_{θ_1} and α_{θ_2} are of the form (24), and that the sets $\{\beta_k\}_{k=1}^{K_1}$ and $\{\alpha_k\}_{k=1}^{K_2}$ are each linearly independent. Then (23) is strongly convex.

Taken together, the above Lemmas immediately imply the following result, which provides conditions under which we can reliably recover the true linearizing controller for the plant by solving (23).

Theorem 1: Suppose that for some $\theta^* \in \Theta$ we have $u_p(x, v) = \hat{u}_{\theta^*}(x, v)$ for each $x \in D$ and $v \in \mathbb{R}^m$, and assume that the hypothesis of Lemma 2 holds. Then θ^* is the unique global (and local) minimizer of (23).

B. Reinforcement learning for practical implementation

To be able to solve (23) efficiently in practical settings we now cast it as a canonical reinforcement learning problem [24]. This allows us to leverage off-the-shelf implementations of on-policy reinforcement learning algorithms to efficiently learn feedback linearizing policies.

Indeed, if we take $\pi_\theta(x, v)$ to be our policy which takes in both the current state x and an auxiliary input v and returns the control action $\hat{u}_\theta$, and take the reward for a given state to be $R(x, v, u_\theta) = \ell(x, v, \theta)$, the above problem can be written as:

$$\min_{\theta \in \Theta} \quad \mathbb{E}_{x_0 \sim X, v \sim V, w \sim \mathcal{N}(0,\sigma^2)} \left[\int_0^T R(x(\tau), v(\tau), u_\theta(\tau)) d\tau \right]$$
$$\text{subject to:} \quad \dot{x} = f(x) + g(x)(\pi_\theta(x, v) + w_t)$$

Where X_0 is the initial state distribution, V is a distribution over auxiliary inputs, $T > 0$ is the time horizon of the problem, and w is additive zero-mean noise term to make the effect of the policy random.

A discretized version of this problem can be solved with on-policy reinforcement learning algorithms. Indeed, for a given fixed value of θ, we can sample N rollouts of length T, and use sequences of the state, output of the policy, and rewards to construct estimates of the gradient of J with respect to θ. This can be done with any method including, but not restricted to REINFORCE [24] with baseline, Deep Deterministic Policy Gradients [25], Proximal Policy Optimization [26], or Trust Region Policy Optimization [27].

V. Examples

We now use our approach to learn feedback linearizing policies for three different systems to highlight its versatility. The first two examples are trained *in silico* while the third is in hardware. In all cases, the input to the parameterized policy replaces all angles with their sine and cosine, and does not include Cartesian positions.

A. Simulations

1) Double pendulum with polynomial policies: We first test our approach on a fully actuated double pendulum with state $x = [\theta_1, \theta_2, \omega_1, \omega_2]^T$, output $y = [\theta_1, \theta_2]^T$, where θ_1 and θ_2 represent the angles of the two joints, with angular rates ω_1 and ω_2 respectively. The system has two inputs u_1 and u_2 that control the torque at both joints. Although the system is relatively low dimensional and fully actuated, it is highly nonlinear and can produce chaotic trajectories [28].

We train linearizing controllers for the double pendulum in two cases where very poor prior information of the model is available. In the first case, we assume that our estimates for the mass and length of the pendulum arms are only $\frac{1}{3}$ of their true values. In the second case, we assume that no prior information on the system's dynamics is available, so that our trained controller has $\beta_m = 0$ and $\alpha_m = 0$. In both settings we parameterize the learned portion of our

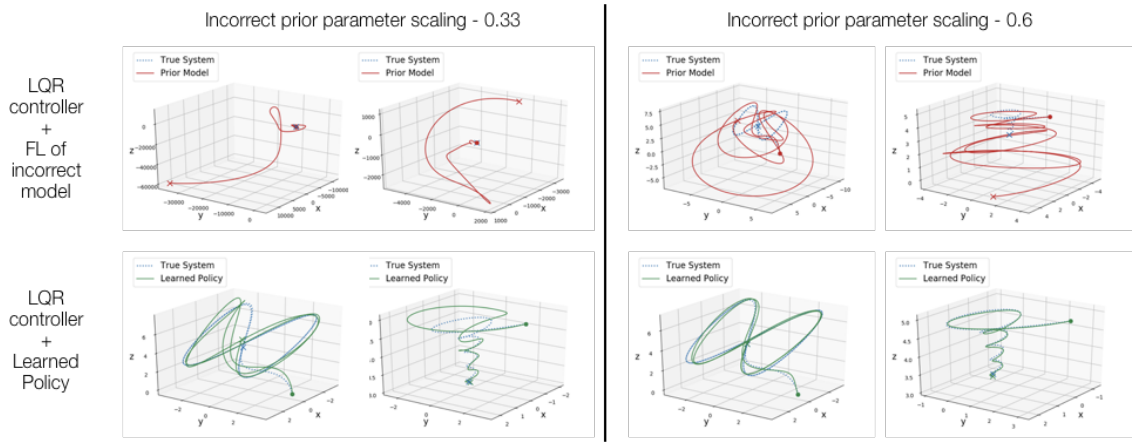

Fig. 3: 14D quadrotor examples for two learned policies. In the first row, we show (in red) the performance of a LQR controller calculated from the desired linear system in conjunction with a feedback linearizing controller based on an the initial (incorrect) model of the dynamics. The second row depicts the tracking performance after the learning process. The first task is a figure-eight, and the second is a corkscrew maneuver. In both maneuvers the quadrotor also tracks an oscillating reference in the yaw angle. Our method successfully corrects the initial misspecification of the linearizing controller.

policies by linear combinations of second order polynomials such that the reinforcement learning problem conforms to the assumptions of Theorem 1. We use the REINFORCE algorithm [24], and baseline state-value estimates with the average reward over all states. At each iteration (or epoch) we collect 50 rollouts of 0.25 seconds each, and we train for 2000 epochs. Figure 2 presents the resulting trajectories for each learned controller. We do not plot the trajectories generated by the nominal model based controllers, since in both cases the initial controllers are unable to move the pendulum arms more than a few degrees from the downwards position. For each controller we observe improvement in tracking ability even though a low order polynomial policy is employed. In order to track the desired reference signals we apply a linear feedback gain on the reference model which is found by solving an infinite horizon LQR problem We used a state penalty matrix of $30 \cdot \mathrm{diag}(1, 1, 0, 0)$ and control penalty matrix of $\mathrm{diag}(1, 1)$ to generate the feedback gain.

2) 14D quadrotor with neural network policies: Our second simulation environment uses the quadrotor model and feedback linearization controller proposed in [29], which makes use of dynamic extension [22]. In particular, the states for the model are $(x, y, z, \psi, \theta, \varphi, \dot{x}, \dot{y}, \dot{z}, p, q, r, \xi, \zeta)$ where x, y and z are the Cartesian coordinates of the quadrotor, and ψ, θ and φ represent the roll, pitch and yaw of the quadrotor, respectively. The next six states represent the time derivatives of these state: $\frac{d}{dt}(x, y, z, \psi, \theta, \varphi) = (\dot{x}, \dot{y}, \dot{z}, p, q, r)$. Finally, ξ and ζ are the extra states obtained from the dynamic extension procedure. The outputs for the model are the x, y, z and ψ coordinates.

In Figure 3 we show the performance of two learned feedback linearizing policies on two different high-performance reference tracking tasks. For the first learned policy, we initialized the training with an incorrect prior model where all the parameters of the model (mass and moments of inertia) were scaled by a factor of $1/3$. For the second learned policy the parameters of the incorrect prior model were scaled by $3/5$. The policies were feed-forward neural networks with $\tanh$ activations with 2 hidden layers of

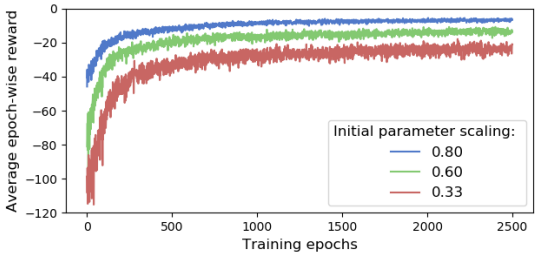

Fig. 4: Average rewards over the training epoch for three learned policies initialized with prior models with different scalings on the parameters.

width 64. For each training epoch, 50 rollouts of length 25 were collected and the parameters were updated using PPO. We trained both policies for 2500 epochs. As shown in Figure 3, both prior models were unable to successfully track the desired references for both tasks, leading to highly unstable dynamics. The learned policies, on the other hand, were able to achieve high quality tracking of both references. For all trajectories a linear feedback gain was applied to the reference model, by solving an LQR problem where deviations in the position and yaw were penalized 20 times more than the norm of the control.

Figure 3 also highlights how better prior models leads to better performance of the learned policy. Figure 4 highlights this trend through the learning curves of three policies initialized with prior models of decreasing quality. We observe that worse initial models result in worse policy performance, given the same network architecture and training time.

B. Robotic experiment: 7-DOF manipulator arm

We also evaluate our approach in hardware, on a 7-DOF Baxter robot arm. The dynamics of this 14-dimensional system are extremely coupled and nonlinear. Taking the 7 joint angles as output y, however, the system is input-output linearizable with relative degree two. We use the system measurements (i.e., masses, link lengths, etc.) provided with Baxter's pre-calibrated URDF [30] and the OROCOS Kinematics and Dynamics Library (KDL) [31] to compute a nominal feedback linearizing control law.

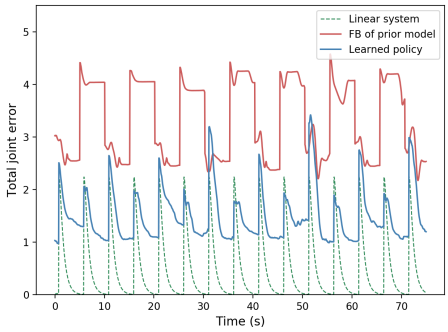

Fig. 5: Total ℓ_2 error on a set-point tracking task after 104 minutes of training for the desired linear system is (dotted, green), the nominal feedback linearizing controller (red), and our learned controller (blue).

This nominal controller suffers from several inaccuracies. First, Baxter's actuators are series-elastic, meaning that each joint contains a torsion spring [32] which is unmodeled, and the URDF itself may not be perfectly accurate. Second, the OROCOS solver is numerical, which can lead to errors in computing the decoupling matrix and drift term. Finally, our control architecture is implemented in the Robot Operating System [33], which can lead to minor timing inconsistency.

We use the PPO algorithm to tune the parameters of a 128×2 neural network with $\tanh$ activations. The neural network maps from the (sine-cosine augmented) 21 states to 56 outputs (7×7 inverse decoupling matrix, 7×1 drift term). For each training epoch, 1250 rollouts of one timestep (0.05 s) each were collected. We trained for 100 epochs with learning rate 3×10^{-4}, which took 104 minutes. Figure 5 summarizes typical results on tracking a square wave reference trajectory for each joint angle with period 5 s. As shown, the ideal linear system displays an exponential step response and rapidly converges to each new setpoint. The nominal feedback linearized model from OROCOS has significant steady-state error. Our learned approach significantly reduces, but does not eliminate, this error. We conjecture that this remaining error is a sign that the (relatively small) neural network may not be sufficiently expressive.

VI. Conclusion

In this paper, we introduced a framework for learning a linearizing control law for a plant with *a priori* unknown dynamics and no assumptions on the coupling between the nonlinear components of the system. We provided theoretical guarantees for conditions under which it is possible to learn the exact linearizing controller. In more general settings, we cast the learning of a feedback linearizing controller as an on-policy reinforcement learning problem.

We validated our proposed approach on three problems. We first showed that it was possible to learn a feedback linearizing controller with *no prior model* to control a highly nonlinear fully actuated double pendulum. Second, we demonstrated that neural network-based feedback linearizing policies could efficiently track arbitrary trajectories of a high dimensional problem. We also empirically observed the advantage of incorporating prior knowledge into the control design. Finally we tested our approach in hardware on a Baxter robot, where we observed that after 104 minutes of training we saw a significant improvement in the tracking error over the baseline. Together, these empirical results confirm the effectiveness of our approach as a general method for designing high quality model reference controllers for high-dimensional systems with unknown dynamics.

Appendix

A. Proof of Lemma 2

First, we rearrange (19) into the form

$$y_p^\gamma = b_p(x) + A_p(x)\beta_m(x) + A_p(x)\alpha_m(x)v + \sum_{i=1}^{K_1} \theta_k^1 A_p(x)\beta_k(x) + \sum_{k=1}^{K_2} \theta_k^2 A_p(x)\alpha_k(x)v \quad (25)$$

to separate out the portions that depend on θ. This can be further condensed by putting $y^\gamma = \bar{W}(x,v) + \hat{W}(x,v)\theta$ where we set

$$\bar{W}(x,v) = b_p(x) + A_p(x)\beta_m(x) + A_p(x)\alpha_m(x)v$$

$$\hat{W}(x,v) = A_p(x)[\beta_1(x), \ldots, \beta_{K_1}(x), A_1(x)v, \ldots, A_{K_2}(x)v]$$

Letting $c(x,v) = (v - \bar{W}(x,v))$, we can rewrite

$$\begin{aligned}\ell(x,v,\theta) &= \left(c(x,v) - \hat{W}(x,v)\theta\right)^T \left(c(x,v) - \hat{W}(x,v)\theta\right) \\ &= \theta^T \hat{W}^T(x,v)\hat{W}(x,v)\theta - 2\theta\hat{W}(x,v)c(x,v) \\ &\quad + c(x,v)^T c(x,v)\end{aligned}$$

From here we observe that $L(\theta) = \theta^T W\theta + \theta^T F + d$ where $W = \mathbb{E}_{x\sim X, v\sim V}\, \hat{W}(x,v)^T\hat{W}(x,v)$ is a positive semi-definite matrix, $F = \mathbb{E}_{x\sim X, v\sim V}\, \hat{W}(x,v)c(x,v)$ and $d = \mathbb{E}_{x\sim X, v\sim V}\, c(x,v)^T c(x,v)$. Thus, recalling that Θ is assumed to be a convex set, we see that (23) is a convex optimization problem which will be strictly convex if, and only if, W is positive definite. Letting $w_k^1 \colon (x,v) \to A_p(x)\beta_k(x)$ and $w_k^2 \colon (x,v) \to A_p(x)\alpha_k(x)v$, we see that W is nothing but the Grammian of the set $\Omega = \{w_1^1, \ldots, w_{K_1}^1, \omega_1^2, \ldots, w_{K_2}^2\}$ on $C(D \times B_v, \mathbb{R}^q)$ with respect to an inner product which is weighted by the distributions X and V. Thus, W will be positive definite if and only if Ω is linearly independent on $C(D \times B_v, \mathbb{R}^q)$. For the sake of contradiction assume that Ω is not linearly independent. Then there exists scalars $\{c_k^1\}_{k=1}^2$ and $\{c_k^2\}_{k=1}^{K_2}$ such that for each $x \in D$ and $v \in V$

$$\sum_{k=1}^{K_1} c_k^1 \omega_k^1(x,v) + \sum_{k=1}^{K_2} c_k^2 \omega_k^2(x,v) = 0 \quad (26)$$

Since we know that $A_p(x)$ is invertible for each $x \in D$, this statement is equivalent to

$$\sum_{k=1}^{K_1} c_k^1 \beta_k(x) + \sum_{k=1}^{K_2} c_k^2 \alpha_k(x)v = 0. \quad (27)$$

holding for each $x \in D$ and $v \in V$. However, it is not difficult to see that this condition is ruled out in the case that $\{\beta_k\}_{k=1}^{K_1}$ and $\{\alpha_k\}_{k=1}^{K_2}$ are linearly independent sets.

References

[1] T. P. Lillicrap et al. "Continuous control with deep reinforcement learning". *arXiv preprint arXiv:1509.02971* (2015).

[2] V. Mnih et al. "Asynchronous methods for deep reinforcement learning". *International conference on machine learning*. 2016.

[3] S. Levine et al. "End-to-end training of deep visuomotor policies". *The Journal of Machine Learning Research* 17.1 (2016).

[4] A. Nagabandi et al. "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning". *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018.

[5] OpenAI et al. "Learning Dexterous In-Hand Manipulation". *CoRR* (2018). URL: http://arxiv.org/abs/1808.00177.

[6] R. E. Kalman et al. "Contributions to the theory of optimal control". *Bol. soc. mat. mexicana* 5.2 (1960).

[7] F. Borrelli, A. Bemporad, and M. Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[8] S. Sastry and M. Bodson. *Adaptive control: stability, convergence and robustness*. Courier Corporation, 1989.

[9] J. J. Craig, P. Hsu, and S. S. Sastry. "Adaptive control of mechanical manipulators". *The International Journal of Robotics Research* 6.2 (1987).

[10] S. S. Sastry and A. Isidori. "Adaptive control of linearizable systems". *IEEE Transactions on Automatic Control* 34.11 (1989).

[11] K. Nam and A. Araposthathis. "A model reference adaptive control scheme for pure-feedback nonlinear systems". *IEEE Transactions on Automatic Control* 33.9 (1988).

[12] I. Kanellakopoulos, P. V. Kokotovic, and A. S. Morse. "Systematic design of adaptive controllers for feedback linearizable systems". *1991 American Control Conference*. IEEE. 1991.

[13] J. Umlauft et al. "Feedback linearization using Gaussian processes". *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017.

[14] G. Chowdhary et al. "Bayesian nonparametric adaptive control using gaussian processes". *IEEE Transactions on Neural Networks and Learning Systems* 26.3 (2014).

[15] G. Chowdhary et al. "Bayesian nonparametric adaptive control of time-varying systems using Gaussian processes". *2013 American Control Conference*. IEEE. 2013.

[16] J. T. Spooner and K. M. Passino. "Stable adaptive control using fuzzy systems and neural networks". *IEEE Transactions on Fuzzy Systems* 4.3 (1996).

[17] F.-C. Chen and H. K. Khalil. "Adaptive control of a class of nonlinear discrete-time systems using neural networks". *IEEE Transactions on Automatic Control* 40.5 (1995).

[18] C. P. Bechlioulis and G. A. Rovithakis. "Robust adaptive control of feedback linearizable MIMO nonlinear systems with prescribed performance". *IEEE Transactions on Automatic Control* 53.9 (2008).

[19] R. M. Sanner and J.-J. Slotine. "Gaussian networks for direct adaptive control". *IEEE Transactions on Neural Networks* 3.6 (1992).

[20] L.-X. Wang. "Stable adaptive fuzzy control of nonlinear systems". *IEEE Transactions on Fuzzy Systems* 1.2 (1993).

[21] K.-S. Hwang, S.-W. Tan, and M.-C. Tsai. "Reinforcement learning to adaptive control of nonlinear systems". *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 33.3 (2003).

[22] S. Sastry. *Nonlinear systems: analysis, stability, and control*. Vol. 10. Springer Science & Business Media, 1999.

[23] A. Isidori. *Nonlinear control systems*. Springer Science & Business Media, 2013.

[24] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. 1st. Cambridge, MA, USA: MIT Press, 1998.

[25] D. Silver et al. "Deterministic Policy Gradient Algorithms". *Proceedings of the 31st International Conference on Machine Learning*. Proceedings of Machine Learning Research. 2014.

[26] J. Schulman et al. "Proximal Policy Optimization Algorithms". *CoRR* ().

[27] J. Schulman et al. "Trust region policy optimization". *International Conference on Machine Learning*. 2015.

[28] T. Shinbrot et al. "Chaos in a double pendulum". *American Journal of Physics* 60.6 (1992).

[29] S. A. Al-Hiddabi. "Quadrotor control using feedback linearization with dynamic extension". *2009 6th International Symposium on Mechatronics and its Applications*. IEEE. 2009.

[30] R. Robotics. "Baxter". *Retrieved Jan* 10 (2013).

[31] H. Bruyninckx. "Open robot control software: the OROCOS project". *Proceedings 2001 ICRA. IEEE international conference on robotics and automation (Cat. No. 01CH37164)*. Vol. 3. IEEE. 2001.

[32] R. L. Williams II. "Baxter Humanoid Robot Kinematics© 2017 Dr. Bob Productions Robert L. Williams II, Ph. D., williar4@ ohio. edu Mechanical Engineering, Ohio University, April 2017" (2017).

[33] M. Quigley et al. "ROS: an Open-Source Robot Operating System". *ICRA Workshop on Open Source Software*. 2009.