Robust Data-Driven Passivity-Based Control of Underactuated Systems via Neural Approximators and Bayesian Inference

Wankun Sirichotiyakul[1], Nardos Ayele Ashenafi[1], and Aykut C. Satici[2]

***Abstract*— We synthesize controllers for underactuated robotic systems using data-driven approaches. Inspired by techniques from classical passivity theory, the control law is parametrized by the gradient of an energy-like (Lyapunov) function, which is represented by a neural network. With the control task encoded as the objective of the optimization, we systematically identify the optimal neural net parameters using gradient-based techniques. The proposed method is validated on the cart-pole swing-up task, both in simulation and on a real system. Additionally, we address questions about controller's robustness against model uncertainties and measurement noise, using a Bayesian approach to infer a probability distribution over the parameters of the controller. The proposed robustness improvement technique is demonstrated on the simple pendulum system.**

I. INTRODUCTION

Policy gradient techniques in deep reinforcement learning have been the dominant approach used by researchers to tackle learning-based control problems. In this setting, the main task is to seek a direct mapping from the system states to the control input that optimizes some notion of reward or loss. The control law is parametrized by a deep neural network, which is trained through repeated interactions with an unknown environment. While these approaches have been successful in robot locomotion and manipulation tasks [1, 2], as well as underactuated control problems [3], they suffer from poor sample complexity, requiring prohibitive amount of training data and computational resources [4].

Recent development in machine learning has indicated that incorporating prior knowledge about the system into the learning problem is, unsurprisingly, highly advantageous in terms of performance and data efficiency. In [5], a framework for discovering the Hamiltonian dynamics from time derivatives of the observed system coordinates is proposed. This approach inherently obeys the conservation of energy, and hence yields a more faithful representation of the true dynamics than directly parametrizing the equations of motion by a neural network. Some extensions of this work include [6], where the authors have taken the Lagrangian point of view to dynamics, and [7], where the Hamiltonian is further parametrized into a form that is faithful to simple mechanical systems [8]. In the latter, the learned quantities lend themselves well to developing control algorithms using familiar results from the literature, e.g. energy-shaping techniques and passivity-based control.

Passivity-based control (PBC) [9, 10], which may be viewed as a generalization of the energy-shaping approach, is a prominent choice for solving underactuated control problems due to its applicability to a wide class of systems. The core objective of PBC is to use the control authority to modify the energy of the system such that the closed-loop system is rendered passive, a property from which stability can be inferred. A common challenge in many PBC approaches is the need to analytically solve nonlinear partial differential equations (PDE). In general, control methodologies that involve solving nonlinear PDEs, such as optimal control (i.e. the Hamilton-Jacobi-Bellman equation) and feedback linearization, are computationally challenging, making them intractable for high-dimensional systems with complicated dynamics.

The motivation of this work is to combine passivity-based control and the expressive power of neural networks to design controllers for underactuated robots. Instead of taking a black-box approach to learn a control policy, we leverage the passivity structure and parametrize the controller by the gradients of an energy-like (Lyapunov) function of the closed-loop system. Our aim is to learn this function that minimizes a user-defined performance objective, such as bringing the states to the desired equilibrium. To this end, we present a data-driven PBC framework, which includes the formulation of a physics-based neural network optimization problem, along with the algorithm to obtain a solution via stochastic gradient descent. Furthermore, this work provides an attempt to rigorously quantify and improve the robustness property of the control laws emanated from the data-driven PBC method, using Bayesian inference theory. We leverage the Markov Chain Monte Carlo (MCMC) technique [11] to find a posterior distribution over the weight vector of the controller, where the prior is constructed from the data-driven PBC.

The contributions of this paper are summarized as follows: 1) data-efficient learning framework for underactuated control problems based on passivity theory; 2) an algorithm for learning a statistical model that represents the parameters of a given control system, with the aim of improving performance when subjected to model uncertainties; 3) validation of the proposed learning methods through simulation and experiments.

[1]Electrical and Computer Engineering Department
[2]Mechanical and Biomedical Engineering Department
[1,2]College of Engineering, Boise State University, Boise, ID, USA

II. BACKGROUND

In this paper, we consider the control design problem for a class of robotic systems whose motion can be faithfully described by Hamiltonian dynamics [8]. We begin by summarizing the technical background that is used in the remainder of the paper.

A. Passivity-Based Control

Let $x \in \mathcal{X} \subset \mathbb{R}^{2n}$ denote the state of the robot. The state x represents the generalized positions and momenta $x = (q, p)$. With M denoting the symmetric, positive-definite mass matrix, the Hamiltonian H of the robot is expressed as

$$H(q,p) = \frac{1}{2} p^\top M^{-1}(q) p + V(q), \tag{1}$$

where $V(q)$ represents the potential energy. The system's equations of motion can then be expressed as

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix} \begin{bmatrix} \nabla_q H \\ \nabla_p H \end{bmatrix} + \begin{bmatrix} 0 \\ G(q) \end{bmatrix} u, \tag{2}$$

where $G(q) \in \mathbb{R}^{n \times m}$ is the input matrix, I_n denotes the $n \times n$ identity matrix, and $u \in \mathbb{R}^m$ is the control input. The system (2) is *underactuated* if $\text{rank}(G) < n$.

The general idea of passivity-based control is to design the control input u that transforms the system into a passive one. In the particular case of Interconnection and Damping Assignment (IDA-PBC) [9], passivity is achieved by imposing the input-state-model port-Hamiltonian structure [10] on the closed-loop dynamics. The control policy consists of the energy-shaping term and the damping-injection term, i.e.

$$u(x) = u_{es}(q, \dot{q}) + u_{di}(q, \dot{q}), \tag{3}$$

where

$$u_{es}(q, \dot{q}) = -G^\dagger \frac{\partial H_d}{\partial q} = -(G^\top G)^{-1} G^\top \frac{\partial H_d}{\partial q}, \tag{4}$$

$$u_{di}(q, \dot{q}) = -K_d G^\top \frac{\partial H_d}{\partial p}. \tag{5}$$

with $K_d \succ 0$ a user-defined damping gain matrix, and H_d the desired closed-loop energy function:

$$H_d(q,p) = \frac{1}{2} p^\top M_d^{-1}(q) p + V_d(q). \tag{6}$$

Here, $M_d \succ 0$ and V_d represent the closed-loop mass matrix and potential energy function, respectively. Appropriate choices for M_d and V_d are found by solving a set of nonlinear partial differential equations.

In this work, we do not constrain the desired Hamiltonian to be of the form (6). Rather, we exploit the universal approximation capability of neural networks to formulate a learning framework that finds H_d automatically. The controller is then derived from trained neural net according to equations (3)-(5). The formulation of the learning framework is elaborated in Section III-A.

B. Bayesian Estimation

In Section III, we discuss how a faithful approximation of a passivity-based controller, such as the one given in (3), may be discovered using a neural network. The parameters of this neural network may be viewed as deterministic or stochastic quantities. While in Section III-A we take the deterministic point of view, in Section III-E, we shift our attention to the stochastic framework, where we employ ideas from Bayesian estimation. Our goal is to learn a probability distribution, $p(\theta)$, over the neural network weights θ, based on the performance of the closed-loop system, which is measured by the metric elaborated in Section III-B.

It is impossible to compute $p(\theta)$ in closed-form as it depends on the time-evolution of a nominal dynamical model. We can still make progress by invoking MCMC methods [11] to numerically estimate $p(\theta)$. Several methods have been proposed to define the transition probabilities of the Markov Chain, from which samples of $p(\theta)$ may be generated. In this work, we invoke the No-U-Turn Sampler (NUTS), recently developed as a variant of the well-known Hamilton Monte Carlo (HMC) method [12], which is an efficient method to sample from probability distributions whose normalization constant may be unknown.

III. ROBUST DATA-DRIVEN DESIGN OF PASSIVITY-BASED CONTROL

The objective of this paper is to formulate a data-driven framework for systematically designing controllers for underactuated mechanical systems. We leverage the known dynamics of the system and combine universal function approximation capability of neural networks with the inherent stabilization properties of passivity-based control. In particular, we tackle the task of determining an energy-like function H_d, from which the controller is derived, such that the closed-loop trajectories minimize a certain objective function. Control synthesis is performed with nominal dynamics in order to use the available data efficiently. Any potential robustness concerns against model uncertainties and measurement noise are addressed by letting the controller parameters be random variables on a probability space, whose posterior distribution is estimated using Bayesian inference.

The organization of this section is as follows. Section III-A outlines the main optimization framework that produces a set of deterministic controller parameters. Section III-B describes the loss function in which the control objective is encoded. Sections III-C and III-D explain, respectively, the methods for sampling the state space and solving the deterministic optimization problem. The Bayesian framework to improve controller's robustness is presented in Section III-E, followed by sampling strategies for selecting controller parameters from the learned distribution in Section III-F.

A. Deterministic Learning Problem

Let $x^\star = (q^\star, 0)$ be the desired equilibrium of the system. The passivity-based control design can be formulated as the search for an energy-like function H_d that solves the following feasibility problem:

$$\begin{aligned} \underset{H_d:\mathcal{X}\to\mathbb{R}}{\text{minimize}} \quad & 0, \\ \text{subject to} \quad & \dot{x} = \begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ G(q) \end{bmatrix} u, \\ & u = -G^\dagger \nabla_q H_d - K_d G^\top \nabla_p H_d, \\ & \dot{H}_d \leq 0, \\ & x^\star = \arg\min_x H_d. \end{aligned} \tag{7}$$

This is an infinite-dimensional, nonlinear optimization problem that is, in general, intractable to solve in closed-form. We therefore seek to reduce the problem to a finite-dimensional one by leveraging the universal approximation property of neural networks.

Let $H_d^\theta : \mathcal{X} \to \mathbb{R}$ be a neural network whose parameters are denoted by $\theta \in \mathbb{R}^s$. Let $\phi = \phi(t; x_0, u)$ denote the flow of the equations of motion (2) with the initial condition $x_0 \in \mathcal{X}$. We approximate the solution to (7) by solving the following minimization problem:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \int_0^T \ell\left(\phi, u^\theta(\phi)\right) \, \mathrm{d}t, \\ \text{subject to} \quad & \dot{x} = \begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ G(q) \end{bmatrix} u^\theta, \\ & u^\theta = -G^\dagger \nabla_q H_d^\theta - K_d G^\top \nabla_p H_d^\theta, \end{aligned} \tag{8}$$

where the time horizon $T \in (0, \infty)$ is a hyperparameter and $\ell : \mathbb{R}^{2n} \times \mathbb{R}^m \to \mathbb{R}$ is a running cost function, whose design we elaborate in the following subsection. Henceforth, we shall refer to each trajectory $\gamma : t \to \phi(t; x_0, u^\theta)$ as a prediction from the initial condition x_0 with the current control law.

B. Loss Function

The running cost function $\ell(x, u)$ that yields the objective function of (8) consists of three main parts: 1) transversal distance, $\ell_\perp$, to a preferred orbit, $\gamma^\star$, 2) set distance, ℓ_{set}, between the current prediction and the goal set, and 3) the regularization term, ℓ_{reg}, to avoid overtraining the neural network parameters. The running cost is the sum of these components:

$$\ell(\theta) := \ell_\perp(x, u) + \ell_{\text{set}}(x) + \ell_{\text{reg}}(\theta), \tag{9}$$

where the first two terms depend on the neural net parameters θ through the dependence of the system trajectory on θ.

1) Distance to a preferred orbit: The first term that makes up the running loss is the distance to a preferred orbit, $\gamma^\star$. The preferred orbit may be generated by using any motion planning algorithm, such as RRT [13], A* [14], virtual holonomic constraints [15], etc. In some cases, there are natural choices for these preferred orbits; for instance, the homoclinic orbit of the pendulum [16].

The loss $\ell_\perp$ is computed as the distance from the current trajectory γ to $\gamma^\star$, defined in terms of the transverse coordinates $x_\perp$ along the preferred orbit, using the ideas outlined in [17, 18]. In this setting, the prediction converges to $\gamma^\star$ if and only if $x_\perp \xrightarrow{t\to\infty} 0$. In order to encourage this behavior, it makes sense to set the first term in our running cost as a quadratic loss on $x_\perp$. Since we also want to use reasonable control effort while having γ converge to $\gamma^\star$, we append a cost on control effort:

$$\ell_\perp(x, u) = \frac{1}{2N} x_\perp^\top Q x_\perp + \frac{1}{2} u^\top R u, \quad Q \succ 0, \ R \succeq 0. \tag{10}$$

2) Set Distance Loss: The ultimate goal of the framework is to bring the states of a robotic system to some desired equilibrium point $x^\star$. We encourage the controller to move the system states to a set around $x^\star$ by penalizing whenever the prediction trajectory spends time away from this set.

The penalty, ℓ_{set} is constructed by defining a convex open neighborhood S containing $x^\star$ and computing the set distance of $\gamma(t)$ to S

$$\ell_{\text{set}}(x) = \inf\{\|x - y\| : x \in \gamma(t), \ t \in (0, T), \ y \in S\}.$$

The set S can be chosen as, for instance, a ball around $x^\star$ of radius $r > 0$ in the standard norm topology. In this case, r becomes a hyperparameter to be selected by the user before the training. With this choice, if any point along the prediction γ gets closer than r to the point $x^\star$, no additional loss is incurred.

C. Sampling the State Space

The objective function of (8) depends on the initial state through the prediction $\gamma : t \to \phi(t; x_0, u)$. If we view $x_0 \in \mathcal{X}$ as a random variable over the state space, then this objective function may be expressed as the expectation of the loss over the distribution of x_0

$$J(\theta) = \mathbb{E}_{x_0 \sim p(\mathbf{x}_0)} \Big[\ell(\phi(t; x_0), u) \Big]. \tag{11}$$

In this subsection, we formulate a strategy that samples the state-space $\mathcal{X}$ and computes (11) efficiently. We base our sampling method on two key ideas: (i) sampling along trajectories generated via optimization, (ii) utilizing DAgger [19].

In high-dimensional state spaces with complicated dynamics, it may be difficult to recover from locally optimal solutions of (8) if the training set $\{\gamma\}$ predominantly miss the goal set by a large amount. In order to circumvent this problem, early in the learning process, we uniformly sample the state space and produce expert trajectories via trajectory optimization. The points along these trajectories are then used as *initial states* for generating γ to be included in the computation of the loss. This kick-starts the learning process with several samples already close to the goal set.

As the learning proceeds, we gradually switch to the DAgger sampling strategy. In short, instead of using the points sampled along expert trajectories as initial states for generating γ, DAgger uses points along the trajectories produced by the current control policy. This iteratively collects new samples from the regions of the state-space visited by the learned controller.

D. Training the Neural Network

We introduce five hyperparameters during the training process: M, T, N, `epochs`, and `batch_size`. First we accumulate M training samples into a collection $\mathcal{D}$ according to in Section III-C. Then, $\mathcal{D}$ is divided into batches, each containing `batch_size` points. These points are used as initial conditions to generate predictions $\gamma^{(\text{batch})}$ by simulating forward until the time horizon T. The parameter N is the number of sampled states along each prediction γ used to compute the loss. This whole process is repeated until the number of batches exceeds the parameter `epochs`. The training process is summarized in Algorithm 1.

Algorithm 1: Training Process

```
1  if deterministic then input: initial parameters $\theta_0$
2  else if Bayesian then input: prior distribution $p(\theta)$
3  Initialize $\mathcal{D} \leftarrow \emptyset$, a container for initial states $x_0$
4  while $P <$ epochs do
5      $\gamma_0 \leftarrow$ integrate eq. (2) OR solve trajectory
         optimization from random $x_0$ (see Section III-C)
6      $\gamma_0^{(M)} \leftarrow$ pick $M$ states sampled from $\gamma_0$
7      $\mathcal{D} \leftarrow \mathcal{D} \cup \gamma_0^{(M)}$
8      for each batch $\subset \mathcal{D}$ do
9          Initialize batch loss $J = 0$
10         for each $d \in$ batch do
11             $\gamma \leftarrow$ integrate eq. (2) with $x_0 = d$
12             $\gamma^{(N)} \leftarrow$ sample $N$ points along $\gamma$
13             Compute current loss $\ell\big(\gamma^{(N)};\theta\big)$
14             $J \leftarrow J + \ell\big(\gamma^{(N)};\theta\big)$
15         Compute gradient $\partial J/\partial\theta$
16         if deterministic then
17             $\theta \leftarrow$ update $\theta$ (SGD step)
18         else if Bayesian then
19             $p(\theta \mid \mathcal{D}) \leftarrow$ update the posterior over $\theta$
                 using No-U-Turn Sampler (Section II-B)
20         $P \leftarrow P + 1$
   Output: $\theta$ Solution of optimization problem (8); or
           $p(\theta \mid \mathcal{D})$ posterior over $\theta$ (12)
```

E. Robustness Improvement via Bayesian Inference

In this section, we shift our point of view and treat the controller parameters θ as random variables over the sample space $\mathbb{R}^s$. We seek a probability distribution such that values of θ that cause large losses have small probabilities of getting sampled during the MCMC algorithm. It is also desired that the posterior distribution stays close to a Gaussian prior distribution, whose mean is given by the solution, θ_d, of the optimization problem (8). These considerations inform the use of the following form for the distribution of θ [11]

$$\bar{p}(\theta) = W_p\, p(\theta \mid \mathcal{D}) = \exp\left(-J(\theta) - \alpha\|\theta - \theta_d\|^2\right), \quad (12)$$

where W_p is the normalization constant, $J(\theta)$ is the expected loss given in equation (11), and the uniform covariance of the Gaussian prior, α, is a hyperparameter. The main objective is to generate samples from (12) without the knowledge of W_p. In this setting, each state of the Markov Chain in the MCMC algorithm represents a particular parameter vector. The parameters θ are initialized randomly or at the deterministic solution θ_d. The NUTS algorithm [20] allows us to reach the stationary distribution of the Markov Chain, which is given by (12), by construction.

F. From the posterior to the controller

Using the No-U-Turn Sampler in the MCMC algorithm allows us to numerically compute the posterior $p(\theta \mid \mathcal{D})$ over the neural network parameters given the training data $\mathcal{D}$. Unlike the deterministic training, which readily defines the controller by providing its unique parameters, Bayesian learning provides us with a myriad of different parameters, each sampled from the posterior.

It is then up to the user to come up with a strategy to pick a form of the controller from all of these samples. This selection procedure is most naturally formulated in terms of statistical decision theory [21–24]. We provide two particularly attractive methods to select a deterministic controller from the posterior. Each of these methods have performed better than the controller found through deterministic neural network training, as elaborated in Section IV-B.

1) Maximum a posteriori (MAP): From the history of the Markov Chain, we pick the weight that has the maximum log-posterior probability (12).

2) Marginalization: The MCMC algorithm returns samples from the posterior distribution along with an unnormalized probability value for each sample. The normalization constant, W_p may be estimated by summing the associated probability vector. This estimate is then used to numerically marginalize the parameter vector with respect to the posterior

$$u(x) = \frac{1}{W_p}\left(\sum_i \bar{p}(\theta_i) u(x, \theta_i)\right). \quad (13)$$

IV. RESULTS

We first demonstrate the robustness properties of the controllers trained via Bayesian methods on the simple pendulum system. The system is simulated under measurement noise via stochastic differential equations briefly discussed in Section IV-A. Furthermore, the system parameter is varied in order to analyze the closed-loop system response under model uncertainties in Section IV-B. Finally, we employ our deterministic data-driven PBC framework to experimentally stabilize the cart-pole system around its open-loop unstable equilibrium point in Section IV-C.

A. *Quantification of uncertainty and noise*

We provide empirical analyses of how controllers derived from deterministic and Bayesian procedures compare under the presence of model uncertainties and measurement noise. While model uncertainties are analyzed by discretizing the model parameter over its uncertainty range, measurement noise is injected into the system dynamics with the help of measure differential equations and integrated using stochastic calculus [25]. In order to integrate the measure differential equations, we first express the dynamical system in the form

$$\mathrm{d}x = f(x)\ \mathrm{d}t + g(x)\ \mathrm{d}W, \tag{14}$$

where f is the deterministic vector field, $g(x)$ are the vector fields that couple the Wiener noise process W with the system dynamics. We then make use of Julia's stochastic differential equations ecosystem [26] in order to numerically integrate these equations.

B. *Robustness: Deterministic vs. Bayesian*

The objective of this case study is to stabilize the homoclinic orbit of the pendulum whose single parameter a has the nominal value $9.81\ s^{-2}$. Algorithm 1 is carried out with expert trajectories generated by the vanilla energy-shaping control (ESC) [27]. Let $z = (\vartheta, \dot{\vartheta})$ be the state of the pendulum where $\vartheta = 0$ corresponds to the upright position. The ESC takes the general form

$$u(\vartheta, \dot{\vartheta}; \theta) = \theta_1 \dot{\vartheta} + \theta_2 \cos(\vartheta)\dot{\vartheta} + \theta_3 \dot{\vartheta}^3, \tag{15}$$

where the weights $\{\theta_i\}_{i=1}^3$ satisfy $-\theta_1 = \theta_2 = 2a\theta_3 < 0$ in the vanilla ESC. While the deterministic vector field of the simple pendulum is

$$f(\vartheta, \dot{\vartheta}) = \begin{bmatrix} \dot{\vartheta} \\ a \sin\vartheta + u(\vartheta, \dot{\vartheta}; \theta) \end{bmatrix},$$

the noise coupling vector fields are the columns of

$$g(\vartheta, \dot{\vartheta}) = \begin{bmatrix} -\theta_2 \sin(\vartheta)\dot{\vartheta} & 0 \\ 0 & \theta_1 + \theta_2 \cos(\vartheta)\dot{\vartheta} + 3\theta_3 \dot{\vartheta}^2 \end{bmatrix}.$$

The homoclinic orbit of the pendulum is defined by the $2a$-level set of the total energy $\mathcal{H} = {}^1\!/_2 \dot{\vartheta}^2 + a(1 + \cos\vartheta)$. Hence, an appropriate measure of the distance to the homoclinic orbit is given by the absolute value $|\tilde{\mathcal{H}}|$ of the error: $\tilde{\mathcal{H}} = \mathcal{H} - 2a$. We evaluate the performance of a closed-loop system by recording the value $\zeta = \min |\tilde{\mathcal{H}}|$, where the minimum is taken over the last 2 seconds of a 10-second-long trajectory.

The robustness of controllers produced by the Bayesian learning framework are demonstrated by comparing ζ, as a function of the system parameter a, with that of the controller with deterministic parametrization. We also investigate the effects of the prior distribution of θ on the performance of the controllers. In particular, we examine a uniform prior and a Gaussian prior centered around the deterministic solution of (8). The comparisons between the controllers from both cases are shown in Figure 1.

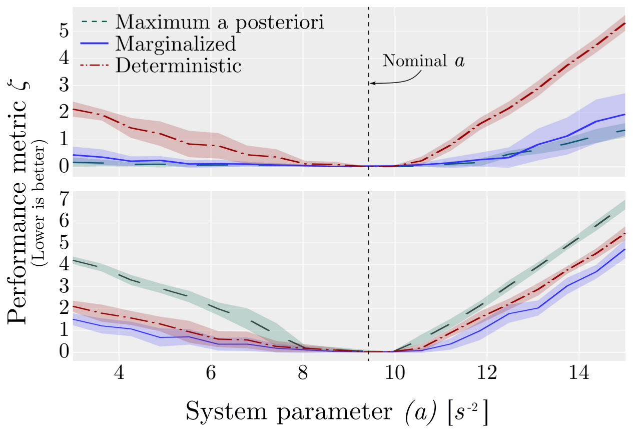

Fig. 1: Performance comparisons between deterministic and Bayesian learning methods. The training is initialized with a Gaussian prior (top), and a uniform prior (bottom). The continuous error band is generated by computing ζ from 20 trajectories of (14), starting at the downward equilibrium with a small disturbance. The solid lines represent the mean of ζ. Best viewed in color.

In addition to uncertainties in a, we inject measurement noise, modelled as a Wiener process with variance 0.0005 rad in the ϑ- and 0.05 ${}^{\mathrm{rad}}\!/_{\mathrm{s}}$ in the $\dot{\vartheta}$-directions. These numbers are chosen to represent a typical error arising from an optical encoder with a resolution of 2048 pulses per revolution and its naive differentiation via backward difference. To capture the influence of measurement noise, we generate 20 trajectories by integrating (14) from the same initial states. These trajectories are then used to compute ζ. The effects of noise on ζ are reflected by the error bands in Figure 1.

The results show that Bayesian learning yields controllers that outperform their deterministic counterpart throughout the whole range of a. The marginalization method for selecting θ from the learned distribution performs best when a uniform prior is used, while the MAP method performs best with the Gaussian prior. We emphasize that the error bands of the marginalized and MAP point estimates stay well below those of the deterministic curve in the top plot of Figure 1. This implies that the controllers produced by Bayesian learning perform consistently better than the deterministic controller, demonstrating their robustness against model uncertainty and measurement noise.

C. *Deterministic Data-Driven PBC for Cart-Pole*

In this subsection, we apply the deterministic optimization framework presented Section III-A to design a swing-up controller for the cart-pole system, a common benchmark underactuated control problem.

The mechanism consists of an unactuated pendulum mounted on an actuated cart. The angular position of the pendulum is denoted by ϑ, with the upright pose at $\vartheta = 0$. The position of the cart, denoted by x, is constrained to $|x| < x_{\max}$. With M the mass of the cart; and m, l,

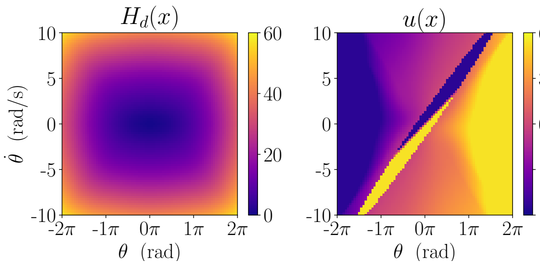

Fig. 2: The learned energy function $H_d^\theta(x)$, and the control policy $u^\theta(x)$ for the cart-pole system, projected onto the ϑ-$\dot{\vartheta}$ plane with $x = \dot{x} = 0$. Best viewed in color.

I_p denoting, respectively, the mass, length, and inertia of the pendulum; the equations of motion are given by [28]

$$I_2\ddot{x} = I_1\big(F_c - B_c\dot{x} - ml\dot{\vartheta}^2 s_\vartheta\big) + ml\big(\hat{m}c_\vartheta s_\vartheta - B_p\dot{\vartheta}c_\vartheta\big),$$
$$I_2\ddot{\vartheta} = \hat{M}\big(\hat{m}s_\vartheta - B_p\dot{\vartheta}\big) + mlc_\vartheta\big(F_c - B_c\dot{x} - ml\dot{\vartheta}^2 s_\vartheta\big),$$

where $\hat{M} = M + m$; $\hat{m} = mgl$; $s_\vartheta = \sin\vartheta$; $c_\vartheta = \cos\vartheta$; $I_1 = I_p + ml^2$; $I_2 = \hat{M}I_p + ml^2\big(M + m\sin^2\vartheta\big)$; F_c is the actuator force acting on the cart; and g is the acceleration due to gravity. The parameters B_p and B_c are friction coefficients of the revolute joint and the cart track, respectively. The force is limited by $|F_c| \le F_{\max}$. For system parameters of the hardware, refer to [28].

The control objective is to swing up the pendulum by moving the cart left and right. The controller must bring the state $\xi = \big(x, \dot{x}, \vartheta, \dot{\vartheta}\big)$ close to the origin, where we switch to a linear controller to stabilize the upright pose.

The neural network H_d^θ has the input dimensions $(4, 64, 64)$ for the respective layers, each with the ELU activation function [29]. There are 4,545 parameters to train with this architecture. The goal set in the computation of ℓ_{set} is $S = \big\{\xi \in \mathbb{R}^4 : \|\xi\|_2 \le 0.1\big\}$. The term $\ell_\perp$ is set to zero to demonstrate that the applicability of our framework is agnostic to the dependency on a nominal trajectory. We perform the training using the procedures outlined in Section III-D with the hyperparameters $M, N = (8, 64)$, $T = 4$ seconds, and batch size of 4. Training converged within 300 epochs.

In simulation studies, the learned controller is evaluated by starting the system from a range of initial states with $\vartheta \in \pm[30, 150]$ degrees, $\dot{\vartheta} \in [-8, 8]$ rad/s, and zero in other dimensions. Fifty samples along each dimension are collected, for a total of 2,500 simulations, all of which end in successful swing-up and stabilization of the origin.

We conduct the experimental studies on the hardware manufactured by Quanser [28]. Figure 3 shows the evolution of the system, recorded from the experiment. The robot starts from the downward pose with a small disturbance in the $\dot{\vartheta}$ direction. The learned controller swings up the pendulum without going over the track limits, and the linear controller catches the swing and stabilizes the upright pose.

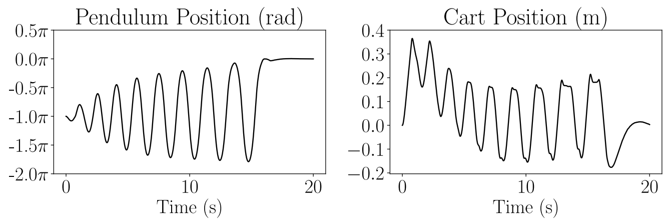

Fig. 3: Experimental results showing time evolution of the closed-loop cart-pole system. The controller is able to swing up the pendulum while maintaining within the track limit of $x_{\max} = 0.4$ m.

D. Updating Controllers with Experimental Data

Bayesian training provides the user an additional method for improving the performance of the controller whenever new data on system parameters becomes available. Given an estimate $\hat{a}$ of the system parameters, a, we assume that a preliminary deterministic and a Bayesian training have been performed, yielding a maximum likelihood solution θ_{ML} and a posterior probability distribution $p(\theta \mid \hat{a})$ over the control parameters, respectively. Note that both solutions depend on the specific belief on the system parameter $\hat{a}$ even though we have only denoted the explicit dependence of the posterior.

System parameters may change due to several reasons, such as external load, system redesign, or simply by a more recent and thorough system identification. Suppose our new belief of the system parameter gets updated to $\hat{a} + \delta a$. We can always perform a new deterministic training taking as the initial parameter vector the current best solution, θ_{ML}. However, this method would completely discard the additional information we have through our approximate knowledge of $p(\theta \mid \hat{a})$.

A more principled update on the parameter vector θ may be achieved that utilizes the estimate of the posterior by performing stochastic gradient updates that are not purely along the gradient of the loss function ℓ (9), but along an appropriate combination of ℓ and the log-posterior $\log p(\theta \mid \hat{a})$. When combining these two pieces of information the goal would be to minimize the component of the negative gradient of ℓ on the negative gradient of the log-posterior as this component tends to decrease the log-posterior the greatest amount.

V. Conclusions

This work presents two data-driven, constructive methods for asymptotically stabilizing a desired equilibrium point of Hamiltonian dynamical systems. The first of these methods employs a deterministic neural network that encodes an energy-like function, from which the controller is derived. Its training is made much more data-efficient in comparison with other state-of-the-art data-driven control synthesis methods by reasoning through a nominal dynamical model of the nonlinear system to be controlled. Our alternative training method leverages Bayesian learning to improve the robustness of the deterministic controller we learn in a data-driven fashion. In this complementary framework, we learn a posterior distribution over the weight vector of a function approximator, such as a neural network, and use a user-specified utility function to make decisions on the selection of the specific weight vector that gets used in the final controller. We find in our simulation experiments that a particularly attractive method of selecting the final form of the controller is by marginalizing over the whole weight vector by using the posterior distribution we learn through an MCMC algorithm.

References

[1] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.

[2] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[4] H. Christensen, N. Amato, H. Yanco, M. Mataric, H. Choset, A. Drobnis, K. Goldberg, J. Grizzle, G. Hager, J. Hollerbach, *et al.*, "A roadmap for us robotics–from internet to robotics 2020 edition," *Foundations and Trends in Robotics*, vol. 8, no. 4, pp. 307–424, 2021.

[5] S. J. Greydanus, M. Dzumba, and J. Yosinski, "Hamiltonian neural networks," 2019.

[6] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, "Lagrangian neural networks," *arXiv preprint arXiv:2003.04630*, 2020.

[7] Y. D. Zhong, B. Dey, and A. Chakraborty, "Symplectic ode-net: Learning hamiltonian dynamics with control," *arXiv preprint arXiv:1909.12077*, 2019.

[8] V. I. Arnold, *Mathematical methods of classical mechanics.* Springer Science & Business Media, 2013, vol. 60.

[9] R. Ortega, A. J. Van Der Schaft, I. Mareels, and B. Maschke, "Putting energy back in control," *IEEE Control Systems Magazine*, vol. 21, no. 2, pp. 18–33, 2001.

[10] A. Van Der Schaft, *L2-gain and passivity techniques in nonlinear control.* Springer, 2000, vol. 2.

[11] C. M. Bishop, "Pattern recognition," *Machine learning*, vol. 128, no. 9, 2006.

[12] R. Dandekar, K. Chung, V. Dixit, M. Tarek, A. Garcia-Valadez, K. V. Vemula, and C. Rackauckas, "Bayesian neural ordinary differential equations," *arXiv preprint arXiv:2012.07244*, 2020.

[13] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *2011 IEEE International Conference on Robotics and Automation.* IEEE, 2011, pp. 1478–1483.

[14] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, S. Thrun, and R. C. Arkin, *Principles of robot motion: theory, algorithms, and implementation.* MIT press, 2005.

[15] A. Shiriaev, J. W. Perram, and C. Canudas-de Wit, "Constructive tool for orbital stabilization of underactuated nonlinear systems: Virtual constraints approach," *IEEE Transactions on Automatic Control*, vol. 50, no. 8, pp. 1164–1176, 2005.

[16] M. W. Spong, P. Corke, and R. Lozano, "Nonlinear control of the reaction wheel pendulum," *Automatica*, vol. 37, no. 11, pp. 1845–1851, 2001.

[17] A. S. Shiriaev and L. B. Freidovich, "Transverse linearization for impulsive mechanical systems with one passive link," *IEEE Transactions on Automatic Control*, vol. 54, no. 12, pp. 2882–2888, 2009.

[18] I. R. Manchester, "Transverse dynamics and regions of stability for nonlinear hybrid limit cycles," *arXiv preprint arXiv:1010.2241*, 2010.

[19] S. Ross, G. J. Gordon, and J. A. Bagnell, "No-regret reductions for imitation learning and structured prediction," *CoRR*, vol. abs/1011.0686, 2010. [Online]. Available: http://arxiv.org/abs/1011.0686

[20] M. D. Hoffman and A. Gelman, "The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo," 2011.

[21] H. Raiffa and R. Schlaifer, "Applied statistical decision theory new york," 2000.

[22] J. M. Bernardo and A. F. Smith, *Bayesian theory.* John Wiley & Sons, 2009, vol. 405.

[23] J. O. Berger, *Statistical decision theory and Bayesian analysis.* Springer Science & Business Media, 2013.

[24] S. Särkkä, *Bayesian filtering and smoothing.* Cambridge University Press, 2013, no. 3.

[25] J. M. Steele, *Stochastic calculus and financial applications.* Springer, 2001, vol. 1.

[26] C. Rackauckas and Q. Nie, "Adaptive methods for stochastic differential equations via natural embeddings and rejection sampling with memory," *Discrete and continuous dynamical systems. Series B*, vol. 22, no. 7, p. 2731, 2017.

[27] R. Tedrake, "Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation," *Course Notes for MIT 6.832*, 2021.

[28] *Linear Servo Base Unit with Inverted Pendulum*, Quanser, 2013, rev. 1.0.

[29] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.