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

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

***Abstract*— Inspired by passivity-based control (PBC) techniques, we propose a data-driven approach in order to learn a neural net parameterized storage function of underactuated mechanical systems. First, the PBC problem is cast as an optimization problem that searches for point estimates of the neural net parameters. Then, we improve the robustness properties of this deterministic framework against system parameter uncertainties and measurement error by injecting techniques from Bayesian inference. In the Bayesian framework, the neural net parameters are samples drawn from a posterior distribution learned via Variational Inference. We demonstrate the performance of the deterministic and Bayesian trainings on the swing-up task of an inertia wheel pendulum in simulation and real-world experiments.**

I. Introduction

Passivity-based control (PBC) [1], [2] is a methodology that casts the control problem as a search for an interconnection pattern among subsystems such that the overall dynamics exhibits passivity properties, which help infer stability. The standard PBC approach in the robot control domain amounts to shaping the system's overall energy with a fictitious one and injecting damping to achieve asymptotic stability. In practice, the closed-loop energy function is typically chosen as a quadratic function of the error to the desired equilibrium point. However, a limit of this approach is the lack of a clear connection between the form of the desired energy function and the performance objective of a given task. Furthermore, control synthesis in PBC methods relies strongly on the dynamical model, and imperfect compensation of the system's energy may lead to poor performance or even instability. This raises concerns regarding model uncertainties, especially in dynamical systems with large number of parameters [3], [4].

There are techniques that combine tools from optimization, probability theory, and machine learning to learn control strategies from inaccurate system models or even unknown dynamics. One example in this domain is reinforcement learning (RL), wherein the quintessential task is to find a mapping from the observed states to the control input such that the long-term cumulative reward is maximized. Many variants of RL view the dynamical system as a black-box, which may be represented by a real system or a simulation, and rely on repeated interactions with the unknown environment to improve the control law. One way to account for model uncertainties within this method is to perform training directly on the real system. While RL methods evidently offer more flexibility on how they learn from the unknown environment [5], [6], [7], most of them discard any potential geometric or algebraic structures that may be useful in control synthesis. This prohibitively increases the sample complexity and limits the ability to infer stability properties.

Bayesian learning [8], [9] offers an alternative method to simultaneously combat model uncertainties while preserving the useful physical structure in a data-driven framework. For instance, it is used to model disturbances, such as the effect of wind gusts on quadcopters and the motion of other vehicles in autonomous driving [10]. In human-robot interactions, such as prosthetics and rehabilitation devices, Bayesian neural networks can classify and predict motions in order to generate reliable commands for the device [11]. However, the literature on applications of Bayesian learning in the control of uncertain dynamical systems is scarce.

In this work, we present a unified framework that simultaneously combines PBC techniques and rigorously addresses model uncertainties using Bayesian learning. The specific contributions of this work are threefold: (i) Motivated by [12], we incorporate uncertainties into the dynamics and cast the passivity-based control synthesis problem as a stochastic optimization problem. The closed-loop storage (energy-like) function, from which the control law is derived, is not restricted to a certain form and instead represented by a neural network whose parameters are random variables. (ii) We apply Bayesian learning and develop an algorithm that finds a suitable probability distribution of the neural net parameters automatically. In contrast to deterministic optimization, this approach provides a probability distribution over the neural net parameters instead of a point estimate, providing a way to reason about model uncertainties and measurement noise during the learning process. (iii) We demonstrate the efficacy and robustness of our current framework with a comparison against our deterministic framework [12]. The comparison is performed on a benchmark underactuated control problem–the inertia wheel pendulum–both in simulation and real-world experiments.

II. Background

This section provides a brief technical background on which the proposed learning framework is based upon.

A. Passivity-Based Control

Let $x \in \mathcal{X} \subset \mathbb{R}^{2n}$ denote the state of the robot. The state x is represented in terms of the generalized positions and momenta $x = (q, p)$. With $M \succ 0$ denoting the symmetric,

[1]Electrical and Computer Engineering Department
[2]Mechanical and Biomedical Engineering Department
[1,2]Boise State University, Boise, ID 83725, USA

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

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

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 rank $G = m < n$.

The main idea of passivity-based control (PBC) [2] is to design the input u with the objective of imposing a desired storage function $H_d : \mathcal{X} \to \mathbb{R}$ on the closed-loop system, rendering it passive and consequently stable. In the standard formulation of PBC, the control comprises an energy shaping term and a damping injection term:

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

such that the closed loop dynamics satisfies

$$H_d(x(t)) - H_d(x(0)) = \int_0^t u_{di}^\top(\tau) y(\tau) \, \mathrm{d}\tau - d^\star(x(t)),$$

where $d^\star \geq 0$ is the desired damping dissipation. For mechanical systems, one solution to the PBC problem is of the form

$$\begin{aligned} u_{es}(x) &= -G^\dagger \left(\nabla_q H_d - \nabla_q H \right), \\ u_{di}(x) &= -K_v \, y, \end{aligned}$$

where $G^\dagger = \left(G^\top G\right)^{-1} G^\top$, and $K_v \succ 0$ is the damping gain matrix. The choice for H_d must satisfy

$$G^\perp \left(\nabla_q H_d - \nabla_q H \right) = 0,$$

where $G^\perp G = 0$, and H_d has a minimum at the desired equilibrium $x^\star = (q^\star, 0)$.

Controllers designed using PBC techniques are based on a nominal dynamical model (2). For many applications, the uncertainties in system parameters are not negligible. In this work, we attempt to improve the controller's robustness properties by means of Bayesian inference, whose theory we briefly summarize in the following subsection.

B. Bayesian Inference

The objective of Bayesian learning is to generate a stochastic model that best fits observed data. Let this stochastic model be represented by $m(x;\theta)$, where θ is a multivariate random variable that parametrizes the model and x is the input. Given prior belief on the distribution of the parameters $p(\theta)$, Bayesian learning finds a posterior distribution $p(\theta \mid \mathbb{D})$ over θ that maximizes the likelihood of the model matching the underlying source of the dataset [13].

Bayesian learning provides various techniques to find the posterior distribution over θ, one of which is the Markov Chain Monte Carlo (MCMC) methods. MCMC methods learn the exact posterior distribution by collecting samples of θ either through a random walk (e.g. Metropolis-Hastings) or by following the gradient of the likelihood (e.g. Hamiltonian Monte Carlo). Unfortunately, MCMC techniques have weak convergence properties for high-dimensional parameters. Hence, we leverage variational inference (VI), a technique that approximates the posterior $p(x \mid \theta)$ with the distribution $q(\theta; z)$ selected from the conjugate family of the prior [14]. This gradient-based method selects the distribution parameters z of the approximate posterior q that maximize the evidence lower bound (ELBO), $\mathcal{L}(x, z)$:

$$\begin{aligned} \mathcal{L}(x,z) &= \mathbb{E}_{\theta \sim q} \left[\log(p(x,\theta;z)) - \log(q(\theta;z)) \right], \\ p(x,\theta;z) &= p(x \mid \theta; z) p(\theta), \end{aligned} \tag{4}$$

where $p(x \mid \theta; z)$ is the likelihood function.

Bayesian learning techniques can be made more powerful when combined with the expressive power of Bayesian neural networks (BNN), whose weights and biases are samples drawn from a posterior distribution [15]. The BNN architecture comes with many advantages; unlike deterministic neural networks, it can provide a model and characterize the uncertainty of predictions with a small amount of data. The use of prior distribution also behaves like a regularization term that prevents the model from overfitting to the training data [16]. On the flip side, it is more complicated and computationally expensive to learn probability distributions than point estimates.

III. Methods

The goal of this paper is to systematically design robust controllers for underactuated systems by combining the universal approximation capability of neural networks with the intrinsic stabilization properties of PBC. In this approach, which we refer to as NeuralPBC, we employ a neural network H_d^θ as a surrogate for H_d, and the control function is parametrized by its gradient $\nabla_x H_d^\theta$. The control objective is achieved by optimizing the evolution of the closed-loop trajectories of (2).

For the NeuralPBC learning problem, we devise two approaches to find a suitable solution. The first approach obtains a point estimate of the neural net parameters via stochastic gradient descent (SGD). We shall refer to these point estimates as the *deterministic* parameters for the remainder of this document. The second approach obtains a probability distribution for the parameters via Bayesian inference. The latter addresses concerns about model uncertainties and measurement noise.

Remark 1: We provide a thorough comparison of the deterministic and Bayesian solutions to the optimal control of a one-dimensional linear dynamical system that is subject to parameter and measurement uncertainties in [17]. The simplicity of the underlying system facilitates the computation of exact solutions, revealing the fact that the Bayesian solution improves the robustness of the closed-loop system against

both types of uncertainties. This comparison serves as a theoretical justification for the improved robustness properties of controllers obtained via Bayesian learning techniques.

A. NEURALPBC *Problem Statement*

In this framework, rather than limiting H_d to be of a certain form, we leverage the expressive power of neural networks and formulate an optimization problem to automatically come up with a suitable function. We aim to find an energy-like function H_d such that the closed-loop trajectories converge to the vicinity of the desired equilibrium $x^\star$, at which point a linear controller, in particular LQR, is employed for stabilization.

Let $H_d^\theta : \mathcal{X} \to \mathbb{R}$ be a neural network, and $K_v^\theta \in \mathbb{R}^{m\times m}$ be a positive-definite matrix. Let $\phi = \phi(t; x_0, u)$ denote the flow of the equations of motion (2) with the initial condition $x_0 \in \mathcal{X}$. With the decision variables θ comprising the parameters of H_d^θ and the entries of K_v^θ, we formulate the minimization problem:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \int_0^T \ell\left(\phi, u^\theta(\phi)\right)\, \mathrm{d}t, \\ \text{subject to} \quad & \dot{x} = \begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ G(q) \end{bmatrix} u^\theta, \\ & u^\theta = -G^\dagger \nabla_q H_d^\theta - K_v^\theta G^\top \nabla_p H_d^\theta, \end{aligned} \tag{5}$$

where $T > 0$ is the time horizon, and $\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is a running cost function to be defined. Henceforth, we shall refer to each trajectory $\gamma : t \to \phi(t; x_0, u^\theta)$ as a prediction from the initial state x_0 with the current control law.

The running cost function ℓ that yields the objective function of (5) consists of the set distance, ℓ_{set}, between the current prediction γ and the goal set $\mathcal{S}$ containing a neighborhood of $x^\star$; and the terminal loss, ℓ_T, between the final state of the trajectory and $x^\star$. The running cost is the sum of these components:

$$\ell := \ell_{\text{set}}(\gamma) + \ell_T(\gamma), \tag{6}$$

where each term depends on the decision variables through the dependence of the system trajectory on θ.

The set distance ℓ_{set} is constructed by defining a convex open neighborhood $\mathcal{S}$ containing $x^\star$ and computing the set distance of γ to $\mathcal{S}$:

$$\ell_{\text{set}}(\gamma) = \inf_t \{\|a - b\| : a \in \gamma(t),\, b \in \mathcal{S}\}. \tag{7}$$

For instance, the set $\mathcal{S}$ may be chosen as a ball around of radius r around $x^\star$ in the standard norm topology. Here, r becomes a hyperparameter of the training algorithm. With a particular choice of $\mathcal{S}$, if at any point along the prediction γ, a state x is closer than r to $x^\star$, no penalty is incurred by ℓ_{set}. Moreover, we introduce a terminal loss of the form $\ell_T = ||x(T) - x^\star||$. While set distance loss encourages trajectories to approach the neighborhood of $x^\star$ at any point along the prediction, the terminal loss encourages trajectories to remain as close to the desired state as possible at time T.

B. *Sampling of the Initial States*

The objective function of (5) depends on the initial state $x_0 \in \mathcal{X}$ through the prediction $\gamma : t \to \phi(t; x_0, u)$. We are interested in obtaining a controller that performs well for all initial states rather than for a single point. Viewing x_0 as a random variable over the state space $\mathcal{X}$, the objective function of (5) may be expressed as the expectation of the loss over the distribution of x_0:

$$J(\theta) = \mathbb{E}_{x_0 \sim p(\mathbf{x}_0)}\Big[\ell(\phi(t; x_0), u)\Big]. \tag{8}$$

Instead of randomly sampling the entire space, we adopt the state sampling techniques from [18]. Let $\mathcal{D}$ denote the training dataset of size $N_\mathcal{D}$. We randomly alternate between populating $\mathcal{D}$ using (a) the DAGGER strategy [18], and (b) sampling from $x_0 \sim \mathcal{N}(x^\star, \Sigma)$, where the variance Σ is kept small. Sampling initial states from $x_0 \sim \mathcal{N}(x^\star, \Sigma)$ helps recover from locally optimal solutions obtained from trajectories that predominantly miss the goal set $\mathcal{S}$ by a large amount. The DAGGER strategy samples a collection of initial states $\{x_0\}$ from trajectories $\{\gamma\}$ generated by executing the current policy u^θ. This technique helps to efficiently sample the relevant part of the state space. After each iteration through the entire dataset, we substitute N_R points in $\mathcal{D}$ with new samples, keeping an $N_\mathcal{D} - N_\text{R}$ portion of initial states as the *replay buffer*, a common technique used in reinforcement learning algorithms [19].

C. *Deterministic Solution for* NEURALPBC

A solution to the nonlinear program (5) can be obtained using a gradient-based search. In this work, we use the stochastic gradient descent (SGD) method, which is standard in machine learning applications. We invoke the adjoint method [20] to compute $\partial J/\partial\theta$, which depends on the solution $\phi(t; x_0; u^\theta)$ of the ordinary differential equation (ODE) provided in (2). Algorithm 1 provides a summary of our training process.

Algorithm 1 Solution to Nonlinear Program (5)

procedure NEURALPBC(H, G, θ)
  $f \leftarrow f(H, G)$ dynamics given by ODE (2) with $u = u^\theta$
  $\mathcal{D} \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ $N_\mathcal{D}$ samples of x_0 (Section III-B)
  while $i <$ `maxiters` **do**
    for each `batch` $\subset \mathcal{D}$ **do**
      $J \leftarrow 0$ ▷ Batch loss
      for each $x_0 \in$ `batch` **do**
        $\gamma \leftarrow$ integrate f from x_0 with current θ
        $J \leftarrow J + \ell(\gamma; \theta)$ ▷ Eq. (6)
      $\theta \leftarrow \theta + \alpha_i \partial J/\partial\theta$ ▷ SGD step
    $\mathcal{D} \leftarrow \{\mathcal{D}\}_{(1,\ldots,N_\mathcal{D}-N_\text{R})} \cup \{x_0\}_{(N_\text{R})}$ ▷ Replay buffer
    $i \leftarrow i + 1$
  return θ

D. *Bayesian Solution for* NEURALPBC

In this subsection, we formulate the Bayesian learning framework that minimizes the effects of system parameter uncertainties and measurement errors. In this framework, the neural net parameterization of H_d^θ given in Section III-C is

replaced by a Bayesian neural network whose parameters are uncorrelated random variables sampled from a multivariate probability distribution (MPD). The goal is to learn the distribution parameters z of the posterior MPD $q(\theta; z)$ that maximize the ELBO given in (4). The computation of the ELBO requires the likelihood function and the prior distribution. The likelihood function is chosen to be a Gaussian probability distribution of the form

$$p(J \mid \theta) = \mathcal{N}(0, s), \tag{9}$$

where s is the standard deviation of the likelihood and a hyperparameter. With the choice of the likelihood given in (9), maximizing the ELBO pulls the loss $J(\theta)$ to zero.

System parameter uncertainties and measurement noise can deteriorate the performance of controllers when employed on real systems. Hence, in the Bayesian framework, we inject these uncertainties directly into the training loop in order to learn a controller that works for a wide range of system parameters and measurement noise. We model system parameter uncertainties by sampling a set of system parameters ζ from a normal distribution $\mathcal{N}(\zeta_0, \sigma_\zeta)$ centered around a nominal parameter ζ_0. Each time we generate a new trajectory starting from the initial states drawn from the replay buffer $\mathcal{D}$, we draw a new sample of ζ.

Additionally, we model measurement error by injecting noise into the prediction γ. This is achieved by replacing the dynamical constraint given in (5) with the following stochastic differential equation (SDE).

$$\mathrm{d}x = \left(\begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ G(q) \end{bmatrix} u^\theta(x) \right) \mathrm{d}t + \nabla_x u(x)\, \mathrm{d}W_t, \tag{10}$$

where $\mathrm{d}W_t$ is a correlated noise process, such as Wiener process, on the states due to measurement uncertainties, and $\nabla_x u(x)$ is the coefficient of the first-order Taylor approximation of $u^\theta(x)$ around zero noise.

Remark 2: Introducing uncertainties to the deterministic training finds a point estimate of the optimal controller parameter, which may be interpreted as the mean of the optimal posterior distribution that Bayesian learning provides. A point estimate of the learned parameters is prone to be biased (for example, if the uncertainty in system parameters is large, the optimal parameter θ for the true system parameter may be quite far from the deterministic solution). This bias-variance trade-off problem is alleviated by Bayesian inference which allows one to marginalize over the posterior parameter distribution [13].

This Bayesian framework extends the deterministic training process discussed in Section III-C, by replacing lines 7 to 10 of Algorithm 1 with Algorithm 2. Similar to the deterministic training, we use the adjoint method to compute the gradient $\partial\mathcal{L}/\partial z$ through the solution of the SDE given in (10). Moreover, we invoke the reparameterization trick of the Automatic Differentiation Variational Inference(ADVI) [21] to compute the gradient of samples θ with respect to the distribution parameters z.

Algorithm 2 Bayesian Learning

1: $f \leftarrow f(H, G)$ dynamics given by SDE (10) with $u = u^\theta$
2: **for** each $x_0 \in$ `batch` **do**
3: Initialize $\mathcal{L} = 0$
4: **for** i=1:N **do**
5: $\theta, s \sim q(\theta; z)$ ▷ Sample parameters of H_d^θ
6: $\zeta \sim \mathcal{N}(\zeta_0, \sigma_\zeta)$ ▷ Sample system parameters
7: $\gamma \leftarrow$ integrate f from x_0
8: $p(J, \theta) = \frac{1}{s\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{J(\theta)}{s}\right)^2\right) p(\theta)$
9: $\mathcal{L} \leftarrow \mathcal{L} + \frac{1}{N}\left(\log[p(\ell, \theta)] - \log[q(\theta; z)]\right)$
10: $z \leftarrow z + \alpha_i \left(\partial\mathcal{L}/\partial z\right)$

IV. Case Study: Inertia-Wheel Pendulum

In this section, we validate the proposed control design framework on the problem of swinging-up and stabilizing the inverted position of an inertia wheel pendulum (IWP), shown in Fig. 1. We provide experimental results from simulation and real-world hardware in order to thoroughly demonstrate the efficacy and robustness claims of Bayesian inference. We use the deterministic solution for NEURALPBC as the baseline on which we compare the performance of the Bayesian solution.

A. System Model

The IWP mechanism consists of a pendulum with an actuated wheel instead of a static mass. The wheel has mass m, which is connected to a massless rod of length l. The position of the rod is denoted by the angle q_1 measured with respect to the downward vertical position. The position of the wheel q_2 is measured with respect to the vertical line through the center of the wheel.

The Hamiltonian of the IWP is given by Equation (1) with $n = 2$ and

$$M = \begin{bmatrix} I_1 & 0 \\ 0 & I_2 \end{bmatrix}, \; G = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \; V(q) = mgl\,(\cos q_1 - 1),$$

and $p = (I_1\dot{q}_1, I_2\dot{q}_2)$. We denote the state of the system as $x = (q_1, q_2, \dot{q}_1, \dot{q}_2)$. The parameters I_1 and I_2 denote the moment of inertia of the pendulum and the wheel, respectively, and g is the gravitational constant. The equations of motion of the IWP can be written as

$$M \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} -mgl \sin q_1 \\ 0 \end{bmatrix} = Gu, \tag{11}$$

where the control input u is the torque applied to the inertia wheel. The desired equilibrium $x^\star = (q^\star, 0)$ is the origin, which corresponds to the upward position. The nominal system parameters are estimated to be $I_1 = 0.0455$ kg-m^2, $I_2 = 0.00425$ kg-m^2, and $mgl = 1.795$ N-m.

B. Case Study

We begin by demonstrating the NEURALPBC framework outlined in Section III-A on the IWP system. In this setting, the surrogate H_d^θ for the closed-loop energy function is determined by solving the optimization problem (5), and the corresponding control law is applied on the system given

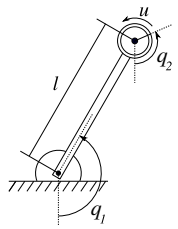

Fig. 1. Schematic of the inertia wheel pendulum. Only the joint q_2 is actuated, and q_1 is not.

TABLE I

NEURALPBC TRAINING SETUP FOR DETERMINISTIC AND BAYESIAN FRAMEWORKS

	Deterministic	Bayesian
H_d neural net size	(6, 12, 3, 1)	(6, 5, 3, 1)
Learned parameters	133	128
Optimizer	ADAM	DecayedAdaGrad
Initial learning rate	0.001	0.01
Replay buffer size	400	50

by (11). We apply this control law in conjunction with the Linear Quadratic Regulator (LQR), the latter of which is activated when the trajectories enter a neighborhood of $x^\star$. Two case studies are performed: 1) solving (5) through stochastic gradient descent (deterministic training from Section III-C), and 2) solving (5) through Bayesian learning (Section III-D). The results from both case studies are compared to showcase the effectiveness of NEURALPBC and the robustness benefits of Bayesian learning.

1) Training Setup: The energy-like function H_d^θ is a fully-connected neural network with two hidden layers, each with the ELU activation function. A uniform distribution in $[-2\pi, 2\pi]\times[-2\pi, 2\pi]\times[-10, 10]\times[-10, 10]$ is chosen as the probability distribution from which samples of initial states x_0 are drawn for the DAGGER strategy. In each gradient descent step, a batch of 4 initial states $\{x_0\}$, generated by the state sampling technique in III-B, are integrated forward with a time horizon of $t \in [0, 3]$ seconds. In the Bayesian framework, the standard deviations σ_ζ of system parameters $\zeta = [I_1, I_2, mgl]$ are chosen to be 10% of the nominal system parameters given in Section IV-A. Moreover, we train on trajectories per the SDE in (10) with measurement error represented by Wiener process with standard deviation of 0.001 and 0.02 on the joint angles and velocities, respectively. The trainings are terminated when the loss function ℓ and the ELBO converge for the deterministic and Bayesian trainings, respectively. The hyperparameters for the deterministic and Bayesian NEURALPBC trainings are shown in Table I. It can be seen that the Bayesian training effectively learns with smaller neural network size and fewer data than the deterministic training.

2) Simulated Experiments: The performance of the controllers obtained from the deterministic and Bayesian trainings are compared as follows. We evaluate the performance of both trainings with parameter uncertainties on I_1, I_2 and mgl. We introduce these uncertainties by moving the average system parameters by $\pm 10\%$ to $\pm 50\%$ with increments of 10%. For each average system parameter, we sample uniformly with a $\pm 5\%$ support around the average system parameters. This helps test the performance of the controller with various combinations of I_1, I_2 and mgl. On top of the system parameter uncertainties, we introduce measurement noise represented by a Wiener process with standard deviation of 0.001 and 0.02 on the joint angles and velocities, respectively. Fig. 2 shows the performance of deterministic and Bayesian trainings using an accumulated quadratic loss of the form

$$J_T = \frac{1}{2}\int_0^T \left(x^\top Q x + u^\top R u\right) dt. \tag{12}$$

The controller derived from the Bayesian training is marginalized over 10 parameters sampled from the posterior distribution. As shown in Fig. 2, the Bayesian training effectively incurs less cost for large error in system parameters. Moreover, the error band on the cost of the Bayesian training is smaller than that of the deterministic training, demonstrating that the marginalized controller is more robust against measurement noise.

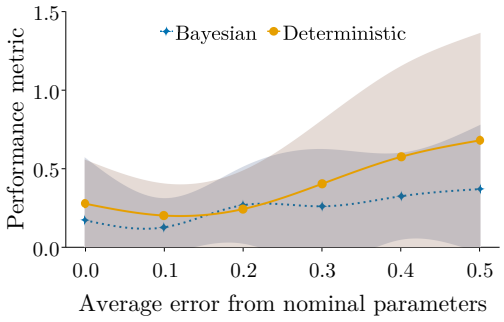

Fig. 2. NEURALPBC Performance metric (J_T) for various error in system parameters. Measurement noise included as Wiener process with standard deviation of 0.001 and 0.02 on joint angles and velocities, respectively

3) Real-World Experiments: The controllers from both training schemes are evaluated on hardware. Since the nominal model (11) neglects the friction in the bearings any contribution to the dynamics by the belt-drive system, the hardware experiments empirically demonstrate the robustness of our controllers against these uncertainties. We further bolster this claim by deliberately modifying the hardware and test the controllers without any additional training. In particular, throughout the experiments, the inertia wheel attached to q_2 is replaced with parts (labelled A-C on Table II) whose mass and inertia are different from the nominal values. The modified system parameters are summarized in Table II.

The system starts from rest at the downward position. A small disturbance in the q_1 direction is introduced to start the swing-up. The state x is recorded and (12) is the performance metric used to evaluate the controllers. The results are summarized in Fig. 3.

In all scenarios, our controllers are able to achieve the control objective despite the errors introduced in the system parameters. These results demonstrate that our approach

TABLE II

SYSTEM PARAMETERS USED IN REAL-WORLD EXPERIMENTS. THE ERRORS IN THE LAST COLUMN ARE $\|p_s - p_s^{\text{NOM}}\|/\|p_s^{\text{NOM}}\|$

Parameter set p_s	I_1	I_2	mgl	Error
Nominal	0.0455	0.00425	1.795	0
A	0.0417	0.00330	1.577	0.122
B	0.0378	0.00235	1.358	0.243
C	0.0340	0.00141	1.140	0.365

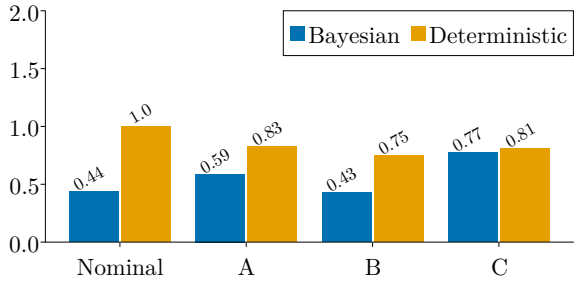

Fig. 3. Controller performance for modified system parameters. The performance metric is given by Eq. (12). Lower values are better. These results show that controllers trained via Bayesian learning are consistently more robust to errors in system parameters.

enables a unified way to tackle nonlinear control problems while simultaneously incorporating prior knowledge and model uncertainties.

C. Discussion

Given a fixed amount of data available during a training session, the deterministic framework has the upper hand since the Bayesian framework needs to learn a whole probability distribution as opposed to a point summary of this whole distribution. However, given a trustworthy prior distribution, Bayesian framework needs much fewer data to come up with solutions that perform well. The Bayesian framework is inherently more robust against parameter uncertainties and measurement errors as long as the controller is computed by marginalizing the probability distribution over the weights of the neural network as this process is less prone to overfitting. Moreover, the Bayesian framework allows for further training of the controller by updating the neural net parameters online; that is, during the operation of the real system by treating the probability distributions learned during simulation as priors.

V. CONCLUSION

This work presents a data-driven passivity based control architecture that encodes the desired Hamiltonian of the system with a neural network. The learning approach efficiently explores the state space using the state sampling technique and enforces a desired behavior through a carefully designed loss function. To improve the robustness properties of the controller, we parameterize the desired Hamiltonian with a Bayesian neural network, whose weights are sampled from a Gaussian posterior learned from Variational Inference techniques. Through the simulation and real-world experiments, we demonstrate that both the deterministic and Bayesian frameworks find control laws that stabilize a desired equilibrium point of a dynamical system. Moreover, the Bayesian framework can handle significant system parameters uncertainties and measurement error.

REFERENCES

[1] 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.

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

[3] Z. Nagy and R. Braatz, "Worst-case and distributional robustness analysis of finite-time control trajectories for nonlinear distributed parameter systems," *IEEE Transactions on Control Systems Technology*, vol. 11, no. 5, pp. 694–704, 2003.

[4] Z. Wu, D. Li, and Y. Chen, "Active disturbance rejection control design based on probabilistic robustness for uncertain systems," *Industrial & Engineering Chemistry Research*, vol. 59, no. 40, pp. 18 070–18 087, 2020.

[5] 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.

[6] 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.

[7] 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.

[8] Y. Gal, R. McAllister, and C. E. Rasmussen, "Improving pilco with bayesian neural network dynamics models," in *Data-Efficient Machine Learning workshop, ICML*, vol. 4, no. 34, 2016, p. 25.

[9] S. Thakur, H. van Hoof, J. C. G. Higuera, D. Precup, and D. Meger, "Uncertainty aware learning from demonstrations in multiple contexts using bayesian neural networks," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 768–774.

[10] D. Sadigh and A. Kapoor, "Safe control under uncertainty," *arXiv preprint arXiv:1510.07313*, 2015.

[11] N. Bu, M. Okamoto, and T. Tsuji, "A hybrid motion classification approach for emg-based human–robot interfaces using bayesian and neural networks," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 502–511, 2009.

[12] W. Sirichotiyakul and A. C. Satici, "Data-driven design of energy-shaping controllers for swing-up control of underactuated robots," in *International Symposium on Experimental Robotics*. Springer, 2020, pp. 323–333.

[13] C. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. Springer New York, 2016.

[14] S. Cohen, "Bayesian analysis in natural language processing," *Synthesis Lectures on Human Language Technologies*, vol. 9, no. 2, pp. 1–274, 2016.

[15] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun, "Hands-on bayesian neural networks–a tutorial for deep learning users," *arXiv preprint arXiv:2007.06823*, 2020.

[16] M. E. Tipping, "Bayesian inference: An introduction to principles and practice in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 41–62.

[17] N. A. Ashenafi, W. Sirichotiyakul, and A. C. Satici, "Robustness of control design via bayesian learning," 2022. [Online]. Available: https://arxiv.org/abs/2205.06896

[18] S. Ross, G. J. Gordon, and J. A. Bagnell, "No-regret reductions for imitation learning and structured prediction," in *In AISTATS*. Citeseer, 2011.

[19] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[20] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," 2018.

[21] A. Kucukelbir, R. Ranganath, A. Gelman, and D. M. Blei, "Automatic variational inference in stan," *arXiv preprint arXiv:1506.03431*, 2015.