Robust Interconnection and Damping Assignment Passivity-Based Control via Neural Bayesian Inference

Abstract

Passivity-based control provides a systematic way to stabilize a nonlinear dynamical system by rendering it passive with respect to a desired storage function. However, obtaining the storage function and the control law requires a closed-form solution to nonlinear partial differential equations, which is intractable for a general robotic system. This method also relies heavily on the dynamical model of the system; thus model uncertainties can deteriorate the performance and may even lead to instability. In order to obtain the storage function and the control law, we formulate a neural network optimization problem that leads to a solution to the matching equations while preserving the port Hamiltonian structure, capturing the inherent stability properties of IDAPBC [1]. Moreover, we merge the robustness properties of Bayesian learning with IDAPBC to find robust controllers for underactuated mechanical systems. A successful combination of Bayesian learning and IDAPBC is achieved by parameterizing the control policy with samples drawn from a posterior probability distribution learned via variational inference. We demonstrate the performance of our approach on the swing-up task of an inertia wheel pendulum, both in simulation and on real-world experiments.

Index Terms

Nonlinear systems, machine learning, robust control, optimization, robotics

I. INTRODUCTION

The essence of passivity-based control (PBC) is to view dynamical systems as subsystems that exchange energy with one another. The control problem is cast as a search for an interconnection pattern such that the overall energy function exhibits desired asymptotic stability properties [2]. The interconnection and damping assignment (IDA) variant [1] is among the most popular PBC contenders, thanks to its applicability to a large class of underactuated mechanical systems [3], [4], [5]. This method targets the closed-loop dynamics of the port-Hamiltonian form [6], with the desired energy function that mimics that of another fictitious mechanical system.

As with other branches of nonlinear control, e.g. feedback linearization and optimal control, the construction of controllers in PBC frameworks explicitly depends on the solution to a set of nonlinear partial differential equations (PDE). In general, the solutions are non-trivial, and obtaining them in closed-form is, in general, intractable. Besides the complication of solving PDEs, using a nominal dynamical model in control synthesis always raises concerns regarding model uncertainties. This issue is magnified in dynamical systems with large number of parameters [7], [8], [9]. Although passive systems have some inherent robustness properties [6], imperfect compensation of the system's energy in PBC may lead to instability.

It is becoming increasingly common to use a combination of tools from optimization, probability theory, and machine learning to formulate control strategies from inaccurate system models or even unknown dynamics. One example in this domain is reinforcement learning (RL) [10], which seeks to find an approximate solution to the Hamilton-Jacobi-Bellman equation, the quintessential task of optimal control. Many variants of RL take a black-box view to the dynamical system, which may be represented by a real system or a simulation. A natural way to account for model uncertainties in this learning method is to perform training directly on the real system. However, to avoid damage, training may require conservative state constraints, consequently limiting the capabilities of the learning techniques [11], [12]. While RL methods evidently offer more applicability to a wide range of dynamical systems [13], [14], [15], 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 of the closed-loop system.

In contrast to black-box approaches, constructing learning frameworks based on prior knowledge and physical principles have been shown to be much more efficient. For example, in [16], a learning framework for discovering Hamiltonian dynamics is proposed, and the controller is synthesized using passivity-based techniques. However, these model-based approaches do not provide a direct method to address uncertainties in the dynamical model from which they are formulated.

Bayesian learning (BL) offers a way to simultaneously combat model uncertainties while preserving the useful physical structure in a data-driven framework. A common approach is shown in [17], [18], [19], where a stochastic model of the dynamical system is constructed via Bayesian learning techniques, and utilized in control synthesis. For instance, BL is used to model uncertainties caused by disturbances, such as the effect of wind gusts on quadcopters and the motion of other vehicles in autonomous driving [17]. The authors of [18] perform Bayesian learning online to characterize time-varying kinematic and dynamical models of industrial robots. Another approach to combating model uncertainties is to surpass the process of learning the stochastic model and instead directly learn the controller through BL. Adaptive control framework is provided in [20], where the search for the control is given by a quadratic program that imposes Lyapunov stability constraint for safety critical systems. This technique uses BL to infer a controller through interactions with an unknown dynamics, while maintaining the algebraic structure of a stable system. Inspired by this technique, we merge the structure and stability properties of PBC with the robustness properties of BL.

In this work, we present a unified data-driven framework that simultaneously combine passivity theory and rigorously address model uncertainties using Bayesian learning. The contributions of this work is two-fold. The first contribution is the development of an optimization framework that automatically achieves the main objective of IdaPbc, easing the burden of solving nonlinear PDEs. In particular, our approach finds appropriate surrogates to the solutions of the matching PDEs without destroying the passivity structure, intrinsically preserving stability properties. The second contribution is to infer robustness properties of our controllers using Bayesian learning, which provides a probability distribution over the parameters of the surrogates representing the energy function. Validations of our methods are provided on a benchmark nonlinear control problem–the inertia wheel pendulum–both in simulation and on real hardware.

II. Background

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

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, 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 is the $n \times n$ identity matrix, and $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input. The system (2) is *underactuated* if $\operatorname{rank} G = m < n$.

The central idea of passivity-based control [6] 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 interconnection and damping assignment (IDAPBC) [1], the closed-loop dynamics to chosen as the port-controlled Hamiltonian (PCH) form:

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & M^{-1}M_d \\ -M_d M^{-1} & J_2(q,p) - G K_v G^\top \end{bmatrix} \begin{bmatrix} \nabla_q H_d \\ \nabla_p H_d \end{bmatrix}, \tag{3}$$

where $J_2 = -J_2^\top$, and the storage function H_d is another Hamiltonian, which is quadratic in the system momenta:

$$H_d(q,p) = \frac{1}{2} p^\top M_d^{-1}(q) p + V_d(q), \tag{4}$$

with $M_d(q) \succ 0$ denoting the closed-loop, positive definite mass matrix and $V_d : \mathbb{R}^n \to \mathbb{R}$ is the closed-loop potential energy function that satisfies

$$q^\star = \operatorname*{argmin}_q \; V_d(q). \tag{5}$$

The control law that achieves the objective of IDAPBC comprises an energy-shaping term u_{es} and a damping injection term u_{di}, i.e.

$$u = u_{es}(q,p) + u_{di}(q,p). \tag{6}$$

The energy-shaping term requires a solution to

$$G u_{es} = \nabla_q H - M_d M^{-1} \nabla_q H_d + J_2 M_d^{-1} p. \tag{7}$$

If system is underactuated, G is not invertible, and Equation (7) cannot be uniquely solved. This leads to the constraints that must be satisfied for any choice of u_{es}:

$$G^\perp \left\{ \nabla_q H - M_d M^{-1} \nabla_q H_d + J_2 M_d^{-1} p \right\} = 0. \tag{8}$$

Equation (8) is a set of nonlinear partial differential equations (PDE) parametrized by M_d, V_d, and J_2. The skew-symmetric matrix J_2 serves as a free parameter to ease obtaining a solution to (8). The success of the IDAPBC approach hinges on the ability to solve this set of PDEs. Once a solution is obtained, the energy shaping term of the control is

$$u_{es} = G^{\dagger}\left(\nabla_q H - M_d M^{-1}\nabla_q H_d + J_2 M_d^{-1} p\right), \tag{9}$$

where $G^{\dagger} = \left(G^\top G\right)^{-1} G^\top$. With $K_v \succ 0$ a user-selected gain matrix, the damping injection term is given as

$$u_{di} = -K_v G^\top \nabla_p H_d. \tag{10}$$

Proposition 1: The closed-loop Hamiltonian H_d in IDAPBC is, by construction, a Lyapunov function for the closed-loop system. The time-derivative of H_d is

$$\begin{aligned}\dot{H}_d &= \left(\nabla_q H_d\right)^\top \dot{q} + \left(\nabla_p H_d\right)^\top \dot{p}\\ &= -\left(\nabla_p H_d\right)^\top \left(J_2 - G K_v G^\top\right)\nabla_p H_d\\ &\le -\lambda_{\text{min}}\{K_v\}\left|\left(\nabla_p H_d\right)^\top G\right|^2 \le 0,\end{aligned}$$

where the last inequality follows from $J_2 = -J_2^\top$ and $K_v \succ 0$. Therefore, as long as V_d is bounded from below and the conditions (5) and (8) are satisfied, $(q^\star, 0)$ is a stable equilibrium of (3).

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 make rigor the controller's robustness properties by means of Bayesian learning, whose theory we briefly summarize in the following subsection.

B. Bayesian Learning

Bayesian learning is a technique that generates a stochastic model $m(x;\theta)$ that best fits a dataset $\mathbb{D}$ with inherent noise. The vector θ is a multivariate random variable that holds the parameters of the model and x is the input. Given a structure of the model and a prior belief $p(\theta)$ on the distribution of the parameters, the main task is to learn a posterior distribution $p(\theta \mid \mathbb{D})$ that maximizes the likelihood of the model generating the dataset $\mathbb{D}$ [21].

Researchers have explored various techniques to find the posterior distribution over the parameters θ. Commonly used approaches include Markov Chain Monte Carlo (MCMC) methods, which learn the exact posterior distribution by collecting samples of θ. Some MCMC techniques, such as Gibbs Sampling, collect samples of the posterior through random walk, while others, such as Hamiltonian Monte Carlo, follow the gradient of the likelihood to sample the next parameter θ. Even though, MCMC techniques learn the exact posterior, they have slow convergence properties for high-dimensional parameters. Hence, we utilize variational inference (VI), a technique that approximates the posterior $p(\theta \mid \mathbb{D})$ with some distribution $q(\theta; z)$ and learns the distribution parameters z. This gradient-based method selects the values of z that maximize the evidence lower bound (ELBO), $\mathcal{L}(x, z)$, defined by [22]

$$\begin{aligned} \mathcal{L}(x,z) &= \mathbb{E}_{\theta\sim q}\left[\log(p(x,\theta;z)) - \log(q(\theta;z))\right], \\ p(x,\theta;z) &= p(x \mid \theta; z)p(\theta;z), \end{aligned} \tag{11}$$

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

The power of Bayesian learning lies in its ability to build a model and make predictions that integrate over all uncertainties [23]. These predictions can be found by marginalizing the model over the posterior [24]

$$\hat{m} = \frac{1}{N}\sum_{\theta\sim q} m(x,\theta), \tag{12}$$

where N is the number of samples drawn from $q(\theta; z)$.

In this work, we leverage the universal approximation capabilities of BNN to parametrize the stochastic model $m(x;\theta)$. In the BNN architecture, the weights and biases of the neural network are samples drawn from a posterior probability distribution, which is inferred through Bayesian learning techniques [24]. There are many advantages to learning the distribution of the neural net parameters over point estimates; for instance, Bayesian learning can infer a model and characterize uncertainty of predictions with small amount of data [24]. Prior knowledge can be explicitly injected into the Bayesian training in the form of prior distribution. Moreover, the use of prior distribution behaves like a regularization term that prevents the model from overfitting to the training data [21]. On the flip side, it is more complicated and computationally expensive to learn probability distributions than point estimates.

III. METHODS

We present the data-driven framework that achieves the main objective of IDAPBC systematically. In Section III-A, we begin by casting the problem of finding a solution to the matching

PDEs given by (8) as an optimization problem. This problem is formulated such that the validity of the relevant physical quantities and the passive structure of IDAPBC are preserved, facilitating stability analysis. In Section III-C, we develop two algorithms that automatically solve the proposed optimization problem. The first algorithm provides a deterministic solution to (8). The second algorithm leverages Bayesian learning to incorporate uncertainties in the open-loop Hamiltonian, and produces a posterior probability distribution over the parameters of the solution to (8).

A. Problem Statement

Control synthesis in the IDAPBC framework relies on a set of functions $M_d(q) : \mathbb{R}^n \to \mathbb{R}^{n\times n}$, $V_d(q) : \mathbb{R}^n \to \mathbb{R}$, and $J_2(q,p) : \mathbb{R}^{2n} \to \mathbb{R}^{n\times n}$ satisfying the PDEs (8). Obtaining the solutions in closed-form is intractable in general. We seek to obtain their numerical approximations, represented by the surrogates M_d^θ, V_d^θ, J_2^θ, where θ denotes the parameters to be learned. Define the loss function $l_{\text{IDA}}(x)$ as the left side of the PDE given by (8):

$$l_{\text{IDA}}(x) = G^\perp \left\{ \nabla_q H - M_d^\theta M^{-1} \nabla_q H_d^\theta + J_2^\theta M_d^{\theta^{-1}} p \right\},$$

where $H_d^\theta(q,p) = \frac{1}{2} p^\top \left(M_d^\theta\right)^{-1} p + V_d^\theta(q)$. The goal is to search for θ such that l_{IDA} is minimized over an appropriate region of the state space $\mathcal{X}$. The set of θ is constrained by the positive-definiteness of M_d^θ, the isolated minimum of V_d^θ at $q^\star$, and the skew-symmetry of J_2^θ. The task of finding an approximate solution to (8) hence reduces to a finite-dimensional optimization problem:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & J = \left\| l_{\text{IDA}}(x) \right\|^2, \\ \text{subject to} \quad & M_d^\theta = \left(M_d^\theta\right)^\top \succ 0, \\ & J_2^\theta = -\left(J_2^\theta\right)^\top, \\ & q^\star = \underset{q}{\text{argmin}}\, V_d^\theta. \end{aligned} \tag{13}$$

As $J \to 0$, the solution H_d^θ of the optimization problem (13) converges to a Lyapunov function of the closed-loop system by construction. Leveraging the well-known fact that solutions of ordinary differential equations depend continuously on their parameters, we make the connection between stability and controllers derived from learning algorithms in the following proposition.

Proposition 2: Suppose the optimization problem (13) has an optimal solution at $\theta^\star$ with the optimal value $J^\star = 0$. The control law u^θ is a continuous function of the objective value J and the solution θ.

Proof: See Appendix A. ■

This implies that for a sufficiently small error $\|\theta - \theta^\star\|$, the control law u^θ will still steer the trajectories to pass through a small neighborhood of $x^\star$, at which point standard linear control techniques may be employed to achieve asymptotic stability. Further exposition of this discussion is presented in Appendix B.

We proceed by showing how the objective function of (13) can be expressed in a more parsimonious form, which only depends on the generalized coordinates $q \in \mathcal{Q} \subset \mathcal{X}$ instead of $(q,p) \in \mathcal{X}$. This halves the sample complexity of the training algorithm. Following [1], the PDE (8) can be written as:

$$0 = G^\perp \Big\{ \nabla_q \left(p^\top M^{-1} p\right) - M_d M^{-1} \nabla_q \left(p^\top M_d^{-1} p\right) + 2 J_2 M_d^{-1} p \Big\}, \tag{14}$$

$$0 = G^\perp \Big\{ \nabla_q V - M_d M^{-1} \nabla_q V_d \Big\}. \tag{15}$$

The PDE (15) is already expressed only in terms of q. To reduce (14), we use with the fact that for any $z \in \mathbb{R}^n$ and any $n \times n$ symmetric matrix A,

$$\begin{aligned} \nabla_q \left(z^\top A(q) z\right) &= \Big[\nabla_q \left(A(q) z\right) \Big]^\top z \\ &= \left[\sum_{k=1}^{n} \nabla_q \left(A_{(\cdot,k)}\right) z_k \right]^\top z, \end{aligned}$$

where $A_{(\cdot,k)}$ denotes the kth column of A. The PDE (14) is equivalent to the following n-PDEs expressed only in terms of the independent variable q:

$$G^\perp \Big\{ \Big[\nabla_q \Big(M^{-1}_{(\cdot,k)} \Big) \Big]^\top - M_d M^{-1} \Big[\nabla_q \big(M_d\big)^{-1}_{(\cdot,k)} \Big]^\top + U_k \big(M_d\big)^{-1} \Big\} = 0, \tag{16}$$

where the matrix $U_k = -U_k^\top \in \mathbb{R}^{n\times n}$ is defined for each $k \in \{1,\dots,n\}$ as $2J_2(q,p) =$

$\sum_{k=1}^{n} U_k(q)p_k$. The optimization problem (13) is restated as

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \sum_{k=1}^{n} \|l_{1k,\text{IDA}}(q)\|^2 + \|l_{2,\text{IDA}}(q)\|^2 , \\ \text{subject to} \quad & M_d^\theta = \left(M_d^\theta\right)^\top \succ 0, \\ & U_k^\theta = -\left(U_k^\theta\right)^\top, \;\; k = 1, \dots, n, \\ & q^\star = \underset{q}{\text{argmin}}\, V_d^\theta, \end{aligned} \tag{17}$$

where $l_{1k,\text{IDA}}$ and $l_{2,\text{IDA}}$ are defined according to (16) and (15), respectively.

B. Constraints

In this subsection, we describe how the constraints of (17) are enforced. The positive-definiteness of M_d^θ is achieved by leveraging the Cholesky decomposition

$$M_d^\theta(q) = L_\theta(q)L_\theta^\top(q) + \epsilon I_n, \tag{18}$$

where $\epsilon > 0$ is a small constant, I_n is an $n \times n$ identity matrix, and $L_\theta \in \mathbb{R}^{n\times n}$ is a lower-triangular matrix whose $n(n+1)/2$ entries are outputs of a neural network. The skew-symmetric matrix U_k^θ is constructed by taking an $n \times n$ matrix A_θ, whose entries are outputs of a neural network, and computing

$$U_k^\theta(q) = A_\theta(q) - A_\theta^\top(q). \tag{19}$$

The condition (5) can be guaranteed by enforcing nonnegativity on V_d^θ, particularly $V_d^\theta(q^\star) \equiv 0$ and $V_d^\theta(q) > 0$ for all $q \neq q^\star$. Without loss of generality, we assume $q^\star = 0$. Let V_d^θ be a deep neural network with zero bias, i.e., with W_i denoting the weights of the ith layer,

$$V_d^\theta(x) = \Phi\Big(W_j\sigma(W_{j-1}\sigma(\dots W_2\sigma(W_1x)))\Big). \tag{20}$$

V_d^θ meets the requirements as long as $\sigma(0) \equiv 0$; $\Phi(0) \equiv 0$; σ, Φ are differentiable; and $\Phi(z) > 0,\ z \neq 0$. Common choices for the activation function σ satisfying this requirement include, e.g., RELU, ELU [25], and TANH.

Remark 1: A sum of squares (SoS) polynomial is also a suitable surrogate for V_d. These polynomials can be parametrized as $s(x) = m^\top(x)Pm(x)$, $P \succ 0$, where m is the vector of monomials up to degree d. The search for a SoS polynomial that minimizes (17) is equivalent to finding P in the convex set of positive semidefinite matrices [26], which is computationally less expensive.

C. Solving the Optimization Problem

We develop two approaches to find a suitable solution to the optimization problem (17). 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 learning. The latter addresses concerns about uncertainties in the model. We provide a theoretical justification of the Bayesian approach in Section IV.

The methods mentioned thus far sum up to several possible combinations of techniques for finding H_d^θ. While any combination can lead to a satisfactory control law, we have found that certain combinations complement each other better than others. Section III-C.3 highlights each method's strengths and provides some guidelines for choosing the most appropriate combination for different use cases.

1) Deterministic Solution: We invoke SGD to find a solution to the nonlinear program (17) over a given discretization of the configuration space $\mathcal{Q}$. The SGD iterations are repeated until the objective function reaches a threshold $\epsilon_{\text{tol}} > 0$, or until the number of iterations exceeds a desired limit. Optionally, the problem (17) can be solved in two subsequent stages: first minimize $l_{1k,\text{IDA}}$ for M_d^θ, U_k^θ; then $l_{2,\text{IDA}}$ for V_d^θ.

If the surrogate for V_d is parametrized by an SoS polynomial, as noted in Remark 1, the second stage reduces to a search for the polynomial coefficients in the form of positive semidefinite matrices. This becomes a *convex* optimization problem that can be solved very efficiently.

The gradient of the objective function with respect to the parameters θ is obtained using the reverse-mode automatic differentiation (AD). For scalar functions with many parameters, reverse-mode AD performs fewer computations than the forward-mode counterpart, at the expense of higher memory consumption [27]. Our application requires relatively small number of parameters θ for the surrogates, thus the reduced computation time significantly outweighs the memory usage trade off.

2) Bayesian Solution: In this subsection, we formulate a Bayesian learning framework that tackles the adverse effects of system parameter uncertainties. We parametrize the function V_d^θ and the entries of L_θ and A_θ matrices with Bayesian neural networks. The goal is to learn the distribution parameters z of the posterior multivariate probability distribution $q(\theta; z)$ that maximize the ELBO given in (11).

The computation of the ELBO requires the likelihood function and the prior distribution. In order to compute the likelihood, we first draw samples of θ from the posterior $q(\theta; z)$, and evaluate the PDEs given in (15) and (16) over discretized states within the configuration space $\mathcal{Q}$. Then, the likelihood is given by

$$p(\|l_{1k,\text{IDA}}(q)\|^2 + \|l_{2,\text{IDA}}(q)\|^2 \mid \theta) = \mathcal{N}(0, s), \tag{21}$$

where $\mathcal{N}$ represents the Gaussian probability distribution, and s is a hyperparameter that represents the standard deviation of the likelihood. With the choice of the likelihood function given in (21), maximizing the ELBO in (11) coaxes the loss $l_{\text{IDA}}(x)$ to zero.

We take two approaches to selecting the distribution parameters z_0 of the prior $p(\theta; z)$. The simplest approach is to use an uninformed prior with randomly initialized z_0 to start the optimization problem; this choice encourages exploration but has slow convergence properties. The second approach uses an informed prior that warm-starts the Bayesian training around the solution of the deterministic training. To do so, z_0 is selected such that the prior distribution is centered around the parameters learned from the deterministic technique discussed in Section III-C.1.

We update the distribution parameters z along the gradient $\partial\mathcal{L}/\partial z$ until the ELBO converges and the objective function of (17) reaches the threshold ϵ_{tol}. We invoke the reparameterization trick of the Automatic Differentiation Variational Inference(ADVI) [28] to compute the gradient of samples θ with respect to the distribution parameters z.

System parameter uncertainties can deteriorate the performance of controllers 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. To model these uncertainties, we sample a set of system parameters ζ from a normal distribution $\mathcal{N}(\zeta_0, \sigma_\zeta)$ centered around the nominal parameter ζ_0, where σ_ζ represents the uncertainty in system parameters. Each time we compute the PDE loss l_{IDA} for a batch of discrete states sampled from the configuration space $\mathcal{Q}$, we draw a new sample of ζ.

3) Comparisons of Methods: Table I summarizes the advantages and disadvantages of the two main controller design approaches we develop in this work. One of the main thrusts of this work is to design a low-level controller for stabilizing underactuated robotic systems using a nominal model. However, using a nominal model always raises questions about whether the proposed

TABLE I: Comparison of the approaches

	Methods implemented	
Case	Deterministic	Bayesian
Robustness	✗	✓
Computation cost	✓	✗
Model selection	✗	✓
Prior knowledge	✓	✓✓
Overfitting	✗	✓

framework will work satisfactorily on a real implementation. In order to address these issues, we invoke Bayesian learning framework on top of the original deterministic pipeline. The Bayesian framework is inherently more robust against parameter uncertainties 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 [29]. 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. Prior knowledge may be incorporated in both the deterministic and Bayesian training; however, the latter is much more conducive and explicit in introducing this knowledge.

Given a fixed amount of data to be used 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, it is well-known given a trustworthy prior knowledge (expressed by the prior distribution), Bayesian framework has lower data requirements to reach solutions that perform well [29].

As an added bonus, even though we have not tackled this aspect in this work, the Bayesian framework also provides avenues to decide which neural network architecture is best for a given dynamical system model from the data generated by simulation. In the Bayesian statistics literature, this is known as Bayesian model selection and a numerical approximation of the *evidence* function may be used to determine how many units and how many layers the neural net should have for good control performance of a given robotic system [29].

IV. Theoretical Justification and Motivation for Bayesian Learning

One of our primary contributions in this work is to demonstrate the improved robustness properties of Bayesian learning over point-estimates of the neural-network-based Lyapunov function and controller. In this section, we demonstrate this claim on a toy example, where closed-form calculation of the point-estimates and posterior distributions for the optimal controller may be undertaken.

A. Optimal Control of an Uncertain Control System

Let us consider the first-order scalar control system, whose drift vector field is uncertain:

$$\begin{cases} \dot{x} = px + u, \\ u(x) = \theta x, \\ x(0) = x_0. \end{cases} \tag{22}$$

We assume that $p \sim \mathcal{N}\left(\hat{p}, \sigma_p^2\right)$ where $\hat{p}$ designates our best prior point estimate of the system parameter p and $\sigma_p > 0$ quantifies the uncertainty in the knowledge of the system parameter. The controller is set to be linear in the state $x \in \mathbb{R}$ with its only parameter $\theta \in \mathbb{R}$ to be determined through learning/optimization. Without loss of generality, we will take the initial condition $x_0 = 1$. The performance index to be optimized for determining the best control parameter θ is

$$\mathcal{J} = \int_0^T \left(\frac{1}{2} q x(t)^2 + \frac{1}{2} r u(t)^2\right) dt, \tag{23}$$

where T is the control horizon and $q \geq 0$ and $r > 0$ are design parameters. We solve the control system (22) to find $x(t) = e^{(p+\theta)t}$ and plug this into the performance index (23) along with the form selected for the controller. Performing the integration over time and letting $T \to \infty$, assuming that $p + \theta < 0$ then yields the infinite-horizon optimal cost functional

$$\mathcal{J}_\infty = -\frac{1}{4} \frac{q + r\theta^2}{p + \theta}. \tag{24}$$

The optimal control parameter θ may be found as the appropriate root of the variation $\mathcal{J}_{\infty,\theta}$ of $\mathcal{J}_\infty$.

$$\begin{aligned} \mathcal{J}_{\infty,\theta} &= -\frac{r}{4}\frac{(p+\theta)^2 - (p^2 + {}^q/_r)}{p+\theta} = 0, \\ \therefore \theta^\star &= g(p) := -p - \sqrt{p^2 + {}^q/_r}, \\ g^{-1}(\theta) &= \frac{q}{2r\theta} - \frac{\theta}{2}. \end{aligned} \tag{25}$$

The fact that $p \sim \mathcal{N}(\hat{p}, \sigma_p^2)$ implies that the optimal control parameter has the probability density function

$$\begin{aligned} f_{\theta^\star}(\theta^\star) &= f_p\left(g^{-1}(\theta^\star)\right)\left|\frac{d}{d\theta}g^{-1}(\theta^\star)\right| \\ &= \frac{1}{\sigma_p\sqrt{2\pi}}\left(\frac{1}{2}\left(1 + \frac{q}{r\theta^{\star 2}}\right)\right) \times \\ &\quad \exp\left\{-\frac{1}{2\sigma_p^2}\left(\frac{q}{2r\theta^\star} - \frac{\theta^\star}{2} - \hat{p}\right)^2\right\}, \end{aligned}$$

where f_p is the Gaussian probability density function with mean $\hat{p}$ and variance σ_p^2.

We can further eliminate the control parameter from the expression for the optimal cost function $\mathcal{J}_\infty$ by substituting for θ from equation (25), yielding

$$\begin{aligned} \mathcal{J}_\infty^\star =& h(p) := \frac{r}{2}\left(p + \sqrt{p^2 + {}^q/_r}\right), \\ h^{-1}(\mathcal{J}^\star) =& \frac{\mathcal{J}^\star}{r} - \frac{q}{4\mathcal{J}^\star}. \end{aligned}$$

Hence, the distribution of the optimal cost conditioned on the system parameter p is given by

$$\begin{aligned} f_{\mathcal{J}^\star}(\mathcal{J}^\star) &= f_p\left(h^{-1}(\mathcal{J}^\star)\right)\left|\frac{d}{d\theta}h^{-1}(\mathcal{J}^\star)\right| \\ &= \frac{1}{\sigma_p\sqrt{2\pi}}\left(\frac{1}{r} + \frac{q}{4\mathcal{J}^{\star 2}}\right) \times \\ &\quad \exp\left\{-\frac{1}{2\sigma_p^2}\left(\frac{\mathcal{J}^\star}{r} - \frac{q}{4\mathcal{J}^\star} - \hat{p}\right)^2\right\}. \end{aligned}$$

Notice that the distribution of both the optimal control parameter and the optimal cost are elements of the exponential family that are not Gaussian.

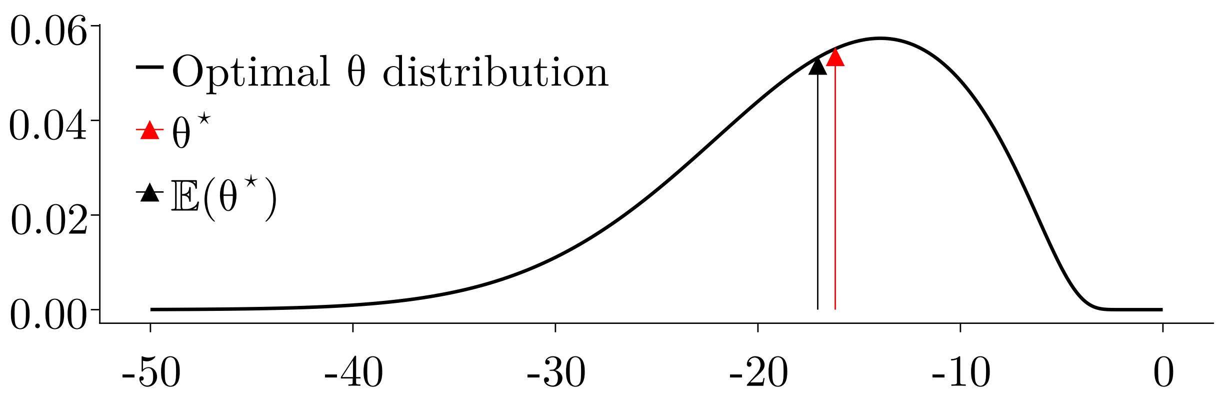


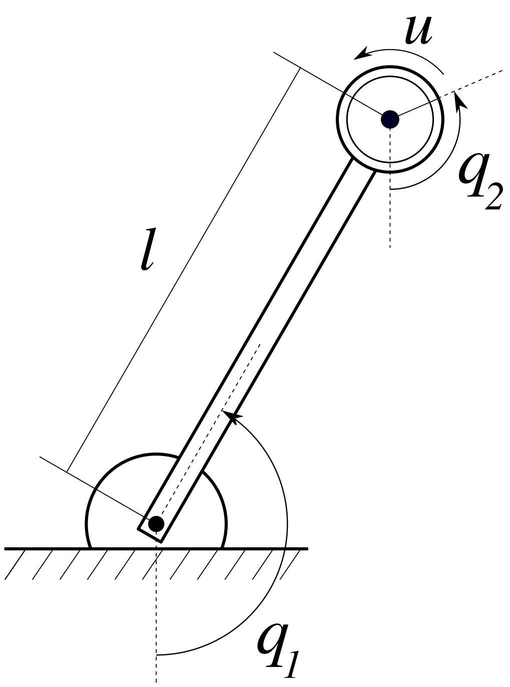
Fig. 1: The optimal control parameter distribution given that the system parameter p is normally distributed with mean $\hat{p} = 5$ and $\sigma_p = 5$. The red and black arrows respectively indicate the optimal control parameter without considering the randomness of p, and the expected value of the optimal control parameter distribution.

B. Takeaways from the Toy Problem

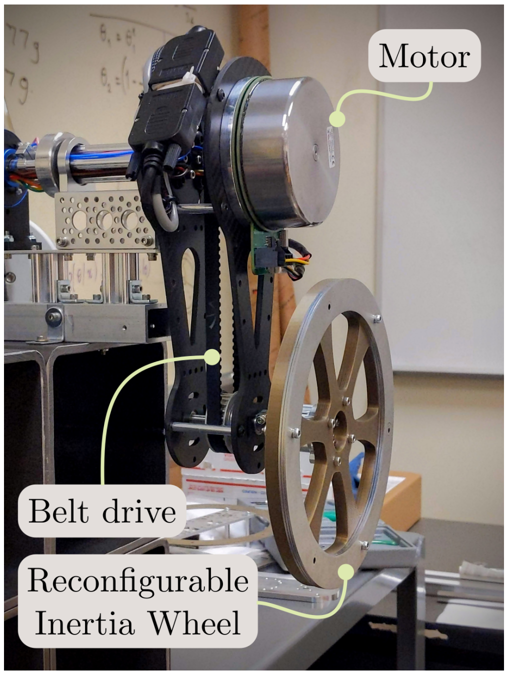
There are several advantages of employing Bayesian learning to find the optimal control parameter θ as the toy example in this subsection supports. In order to derive some quantitative results, let us assign some numerical values to the parameters that define the optimal cost function $(q, r) = (100, 1)$, our best guess $\hat{p} = 5$ of the system parameter p and its standard deviation $\sigma_p = 5$.

The optimal control parameter and cost derived for this system whose model is assumed to be known perfectly are given by $\hat{\theta}^\star = -16.180$ with the corresponding estimated cost $\hat{\mathcal{J}}^\star = 8.090$. This deterministic performance estimate turns out to be *overconfident* when uncertainties in the system parameter are present. For example, if the prior knowledge on the distribution of the system parameter p is utilized, the expected value of the controller parameter is found as $\mathbb{E}(\theta^\star) = -17.046$ and the corresponding expected cost is $\mathbb{E}(\mathcal{J}) = 8.523$. The controller from the deterministic training/optimization is not only overconfident about its performance; but also is less robust against modeling errors, as the Bayesian learning yields a closed-loop stable system for a wider range of values of p.

Finally, Figure 1 shows the optimal control parameter distribution given that the system parameter p is normally distributed with mean $\hat{p} = 5$, standard deviation $\sigma_p = 5$. This figure also



(a) Schematic



(b) Hardware implementation

Fig. 2: Schematic of the inertia wheel pendulum. The joint q_2 is actuated, and q_1 is not. On hardware, q_2 is belt-driven by a motor mounted along the axis of rotation of q_1.

shows the mean values of the optimal control distribution with the black arrow and the optimal control parameter a deterministic approach would yield in red. We notice that the Bayesian learning that yields the optimal control parameter distribution is more concerned about system stability due to the uncertainty in the parameter p, a feat that the deterministic training may not reason about.

V. 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. 2. We provide experimental results from simulation and real-world hardware in order to thoroughly demonstrate the efficacy and robustness claims of Bayesian learning.

A. System Model

The IWP is a simple pendulum with an actuated wheel instead of a static mass. The wheel has mass m, which is connected to a rod of length l. The position of the rod is denoted by the angle θ_1 measured with respect to the downward vertical position. The position of the wheel θ_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

$$M = \begin{bmatrix} I_1 & 0 \\ 0 & I_2 \end{bmatrix}, \; G = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \; V(q) = m_3 \left(\cos q_1 - 1\right),$$

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 parameter I_1 denotes the moment of inertia of the pendulum, I_2 is the moment of inertia of the rotating wheel, $m_3 = mgl$ with g the acceleration due to gravity. The equations of motion of the IWP can be written in standard form as

$$\begin{bmatrix} I_1 & 0 \\ 0 & I_2 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} + \begin{bmatrix} -m_3 \sin q_1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} u, \tag{26}$$

where the control input u is the torque applied on q_2.

The joint q_2 in our hardware implementation is actuated by a Nanotec DFA90 brushless DC motor through a belt drive system. The actuator torque is controlled by Maxon EPOS2 70/10. The link attached to q_2 is reconfigurable so that multiple combinations of system parameters are possible. We perform experiments with a total of 4 combinations of system parameters. The system parameters for each of these configurations are summarized in Table II.

B. Training Setup

We validate the proposed framework on the task of regulating the upright equilibrium of the IWP, which corresponds to the origin, i.e. $x^\star = (q^\star, 0) = 0$. The control objective is to ensure that all closed-loop trajectories of (26) pass through a small neighborhood of $x^\star$, at which point a linear stabilizing controller is activated to asymptotically stabilize the system at $x^\star$. In particular, we apply the controllers derived from the proposed learning framework in conjunction with the standard linear quadratic regulator (LQR).

The optimization problem (17) is constructed as follows. The potential energy function V_d^θ is a fully-connected neural network given by Eq. (20) with $\Phi(z) = \|z\|_2^2$ and two hidden layers, each

TABLE II: System parameters of the IWP. The errors in the last column are normalized with respect to the nominal

Parameter set p_s	I_1	I_2	m_3	Error
Nominal	0.0455	0.00425	1.795	0
Type A	0.0417	0.00330	1.577	0.122
Type B	0.0378	0.00235	1.358	0.243
Type C	0.0340	0.00141	1.140	0.365

TABLE III: Training setup

	Deterministic	Bayesian
Neural net size	(2, 8, 4, 1)	(2, 8, 6, 1)
# of parameters	56	150
Optimizer	ADAM	DecayedAdaGrad
Initial learning rate	0.001	0.01

of which has the activation function ELU. The closed-loop mass matrix is constructed according to Eq. (18), where the components of L_θ are part of the parameters θ to be optimized. We choose $J_2^\theta = 0$, as the mass matrix independent of q for this system. The parameters of the surrogates are initialized according to the Glorot (Xavier) [30] scheme. The optimization problem is solved over a uniform discretization of $q = (q_1, q_2) \in [-2\pi, 2\pi] \times [-50, 50]$.

We obtain two control laws from the proposed framework by performing the training in two settings: 1) stochastic gradient descent (Section III-C.1), which provides a point estimate of θ; and 2) Bayesian learning (Section III-C.2), which provides a probability distribution over θ. In the deterministic setting, the nominal system parameters reported in Table II are used for $H(q,p)$ during training. In the Bayesian setting, the training is performed with the system parameters $\zeta = [m_3, I_1, I_2]$ treated as random variables sampled from $\mathcal{N}(\zeta_0, \sigma_\zeta)$, where the mean ζ_0 is a vector of nominal parameters given in Table II. The standard deviations σ_ζ of the system parameters ζ are chosen to be 10% of the nominal parameters. After training, both settings use the nominal values for the computation of $H(q,p)$ in the control synthesis given by Equations (9) and (10). A summary of the hyperparameters for both the deterministic and Bayesian methods

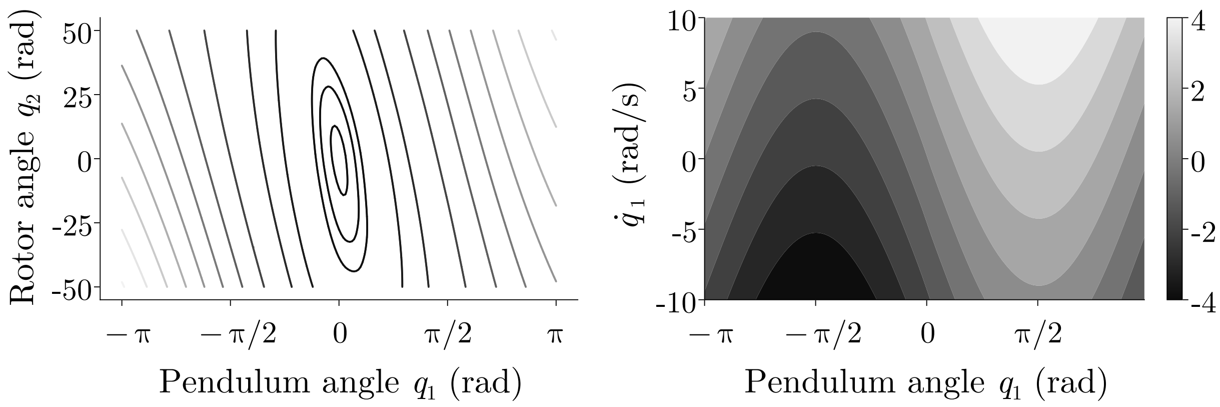


Fig. 3: The left plot shows the level sets of the closed-loop potential energy $V_d^\theta(q)$, which has an isolated minimum at the desired equilibrium $q^\star = 0$. The right plot shows the control effort u^θ derived from H_d^θ, projected on q_1-$\dot{q}_1$ plane with $q_2 = \dot{q}_2 = 0$. The controller correctly commands torque in the appropriate direction, pushing the trajectories to $x^\star$.

are given in Table III.

C. Simulated Experiments

We first evaluate the characteristics of the desired Hamiltonian H_d^θ obtained from the deterministic training. Figure 3 shows the level sets of $V_d^\theta(q)$, which has an isolated minimum at $q^\star = 0$ by construction. This guarantees that the condition (5) is always satisfied, even when the training has not yet converged. Figure (4) shows the evolution of the system (26) under the application of the learned controller u^θ. The closed-loop trajectories converge to $x^\star$ as desired, with the Hamiltonian H_d^θ decreasing along these trajectories, satisfying the criterion of a Lyapunov function. These results show that the proposed optimization framework automatically and correctly identifies a valid closed-loop Hamiltonian for IdaPbc. Our framework preserves the passivity structure, and hence its inherent stability properties, providing an efficient way to connect learning-based control laws with classical Lyapunov stability theory.

The performance of the controllers obtained from the deterministic and Bayesian trainings are compared as follows. We introduce the uncertainties on I_1, I_2 and m_3 by representing them with a uniformly distributed random variable $p_{\text{sys}} \sim \mathcal{U}[-0.05\mu_p, 0.05\mu_p]$, where μ_p differ from the nominal values by $\pm10\%$ to $\pm50\%$ in increments of 10%, for a total of 20 distributions

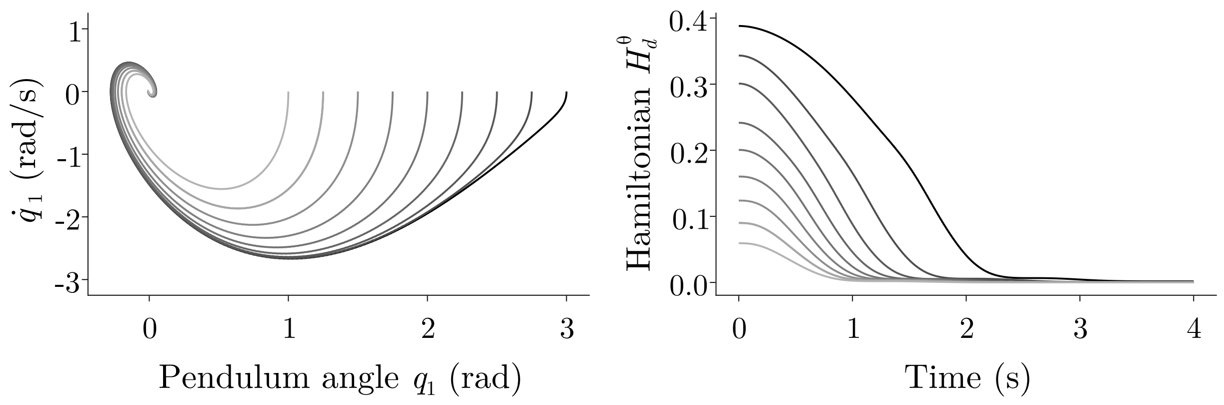


Fig. 4: The time evolution of the system under the application of the controller derived from the learned Hamiltonian H_d^θ. The right plot shows the function H_d^θ decreasing along the closed-loop trajectories, satisfying Lyapunov's criteria.

with different values of μ_p. For each distribution, we draw 20 samples and simulate the system forward under the application of u^θ. This helps test the performance of the controller with various combinations of I_1, I_2 and m_3. Figure 5 shows the performance of the controllers. The metric is an accumulated quadratic loss of the form

$$J_T = \frac{1}{2}\int_0^T \left(x^\top Q x + u^\top R u\right) dt, \tag{27}$$

where T is the time at which the switch to LQR occurs. The policy learned from the Bayesian training is marginalized over 10 parameters sampled from the posterior per Eq. (12). As seen in Figure 5, trajectories from the Bayesian controller incur much lower cost than the deterministic counterpart throughout a wide range of errors in system parameters. Moreover, we observe that the error band on the cost corresponding to Bayesian training is narrower. These results show that controllers trained via Bayesian learning are consistently more robust to errors in system parameters.

D. Real-World Experiments

The hardware experiments are designed to further demonstrate the robustness of our controllers against model uncertainties, which include errors in the parameters, friction in the bearings, and any contribution to the dynamics from the belt-drive system. We deliberately modify the

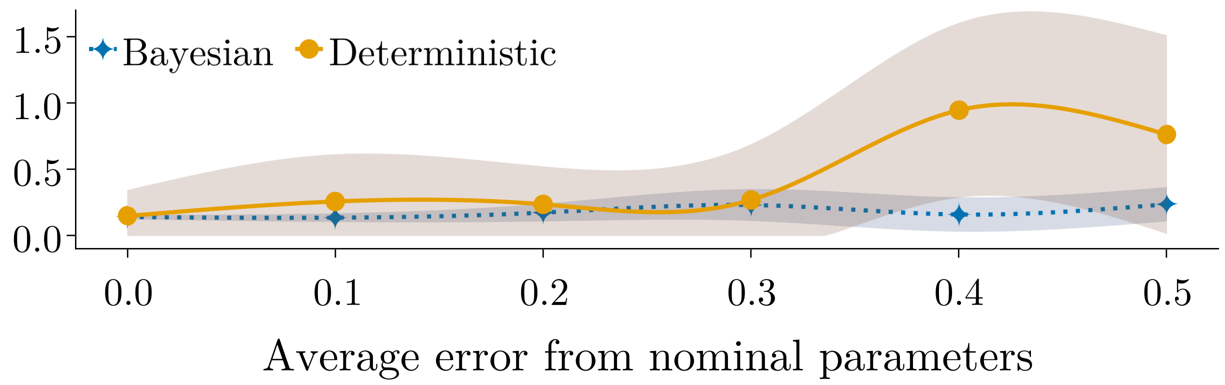


Fig. 5: Accumulated quadratic cost (J_T) for a range of error in system parameters. Lower values correspond to better controller performance.

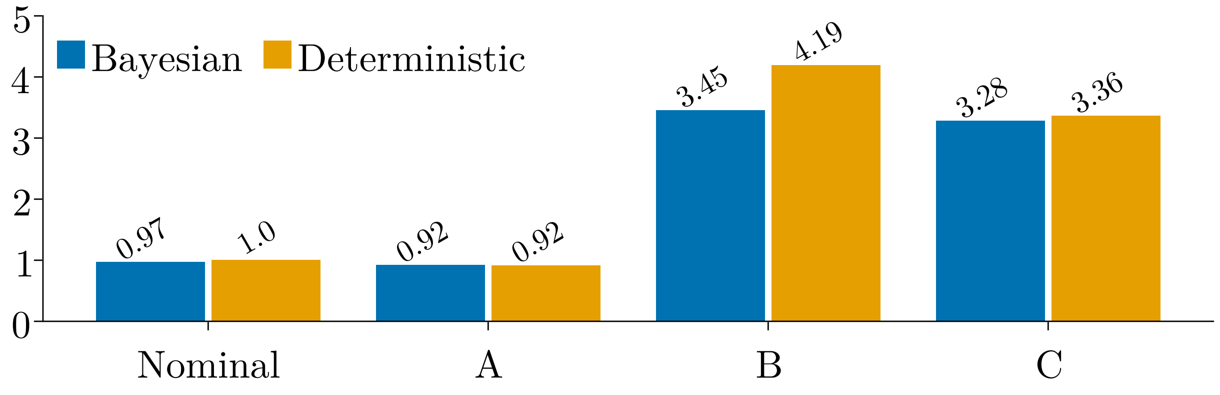


Fig. 6: Normalized accumulated cost J_T (lower is better) for modified system parameters. The categories A-C correspond to the parameters shown in Table II.

hardware to create large errors in the model parameters and test the controllers without any additional training. In particular, the inertia wheel attached to q_2 is replaced with parts whose mass and inertia values differ from the nominal values (see Table II).

The experiment starts with the system at rest in the downward pose, i.e. $x = (\pi, 0, 0, 0)$. A small disturbance in the q_1 direction is introduced to start the swing-up. The state x is recorded, and the performance metric (27) is used to evaluate the controllers. The results are summarized in Figure 6.

In all scenarios, we recorded a 100% success rate in the swing-up task despite the errors introduced in the system parameters. Furthermore, we observe that the controller from Bayesian training consistently outperforms the deterministic counterpart, supporting the theoretical justification discussed in Section IV.

VI. Conclusion

This paper introduces a data-driven framework that automatically finds a robust control policy for a class of underactuated mechanical systems. The first contribution is the construction of an optimization problem that leverages the IdaPbc methodology without destroying the passivity structure and other physical constraints. Our formulation facilitates stability analysis, providing a transparent connection between classical Lyapunov stability and control laws obtained via learning algorithms.

Leveraging the universal approximation capability of neural networks, we solve the optimization problem that yields an approximate solution of the nonlinear PDE relevant to the IdaPbc problem. As the optimization converges to a faithful solution, we show that the corresponding control law is guaranteed to steer the trajectories to pass through a neighborhood of the desired equilibrium, at which point standard linear control techniques may be employed for asymptotic stability.

The second contribution is the development of a training algorithm that accounts for system model uncertainties by treating the neural net parameters as random variables and infer their posterior distribution using Bayesian techniques. A detailed discussion justifying how Bayesian learning improves the controller's robustness is provided.

As an example, we apply the proposed framework on a benchmark nonlinear, underactuated control problem–the inertia wheel pendulum–which is related to a family of applications, e.g. the attitude control of satellites and walking robots. With relatively minimal computation, the learning framework yields a faithful closed-loop Hamiltonian, and the corresponding control law successfully stabilize the system about the desired equilibrium. We validate the controller's performance and robustness properties both in simulation and on hardware.

References

[1] R. Ortega, M. W. Spong, F. Gómez-Estern, and G. Blankenstein, "Stabilization of a class of underactuated mechanical systems via interconnection and damping assignment," *IEEE transactions on automatic control*, vol. 47, no. 8, pp. 1218–1233, 2002.

[2] 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.

[3] J. A. Acosta, R. Ortega, A. Astolfi, and A. D. Mahindrakar, “Interconnection and damping assignment passivity-based control of mechanical systems with underactuation degree one,” *IEEE Transactions on Automatic Control*, vol. 50, no. 12, pp. 1936–1955, 2005.

[4] A. D. Mahindrakar, A. Astolfi, R. Ortega, and G. Viola, “Further constructive results on interconnection and damping assignment control of mechanical systems: The acrobot example,” *International Journal of Robust and Nonlinear Contro*, vol. 16, no. 14, pp. 671–685, 2006.

[5] G. Viola, R. Ortega, R. Banavar, J. Á. Acosta, and A. Astolfi, “Total energy shaping control of mechanical systems: simplifying the matching equations via coordinate changes,” *IEEE Transactions on Automatic Control*, vol. 52, no. 6, pp. 1093–1099, 2007.

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

[7] 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.

[8] R. F. Stengel, L. R. Ray, and C. I. Marrison, “Probabilistic evaluation of control system robustness,” *International Journal of Systems Science*, vol. 26, no. 7, pp. 1363–1382, 1995.

[9] 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. 18070–18087, 2020.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[11] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[12] M.-A. Beaudoin and B. Boulet, “Structured learning of safety guarantees for the control of uncertain dynamical systems,” *arXiv preprint arXiv:2112.03347*, 2021.

[13] 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.

[14] 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.

[15] 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.

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

[17] D. Sadigh and A. Kapoor, “Safe control under uncertainty,” *arXiv preprint arXiv:1510.07313*, 2015.

[18] T. Shen, Y. Dong, D. He, and Y. Yuan, “Online identification of time-varying dynamical systems for industrial robots based on sparse bayesian learning,” *Science China Technological Sciences*, vol. 65, no. 2, pp. 386–395, 2022.

[19] S. Linderman, M. Johnson, A. Miller, R. Adams, D. Blei, and L. Paninski, “Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, vol. 54, pp. 914–922, PMLR, 20–22 Apr 2017.

[20] D. D. Fan, J. Nguyen, R. Thakker, N. Alatur, A.-a. Agha-mohammadi, and E. A. Theodorou, “Bayesian learning-based

adaptive control for safety critical systems," in *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 4093–4099, IEEE, 2020.

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

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

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

[24] 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.

[25] 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.

[26] M.-D. Choi, T. Y. Lam, and B. Reznick, "Sums of squares of real polynomials," in *Proceedings of Symposia in Pure mathematics*, vol. 58, pp. 103–126, American Mathematical Society, 1995.

[27] A. Griewank *et al.*, "On automatic differentiation," *Mathematical Programming: recent developments and applications*, vol. 6, no. 6, pp. 83–107, 1989.

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

[29] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[30] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[31] P. Hartman, *Ordinary differential equations*. SIAM, 2002.

Appendix

A. Proof of Proposition 2

Proof: Let $\mathbb{U} = \mathbb{U}(\theta)$ denote the right side of equation (7). In this notation, the objective function of (13) is the squared norm of $G^{\perp}\mathbb{U}$. Let $(\cdot)^{(k)}$ denote the k^{th} iteration of the optimization algorithm. Since $J^{(k)} \to 0$, $\mathbb{U}^{(k)} \to \mathbb{U}^{\star}$, and $\theta^{(k)} \to \theta^{\star}$, there exists an integer $K > 0$ such that when $k > K$, the following are true:

1) $0 \leq J^{(k)} < \delta_1$,
2) $0 \leq \left\|\mathbb{U}^{(k)} - \mathbb{U}^{\star}\right\| < \delta_2$,
3) $0 \leq \left\|\theta^{(k)} - \theta^{\star}\right\| < \delta_3$.

Since $G^{\dagger}(q) = \left(G^{\top}(q)G(q)\right)^{-1} G^{\top}(q)$ is a continuous function of q, and $u_{es}^{\theta} = G^{\dagger}\mathbb{U}$ is a linear in $\mathbb{U}$, it follows that for all $\epsilon > 0$, $\exists\delta_2 > 0$ such that $\left\|G^{\dagger}\mathbb{U}^{(k)} - G^{\dagger}\mathbb{U}^{\star}\right\| < \epsilon$ whenever $\left\|\mathbb{U}^{(k)} - \mathbb{U}^{\star}\right\| < \delta_2$. The claim of the proposition is demonstrated by noting that $u_{es}^{\theta} = G^{\dagger}\mathbb{U}$ and $u_{es}^{\theta^{\star}} = G^{\dagger}\mathbb{U}^{\star}$. ■

B. Stability of The Control System Given by (2) Under The Control Law u^θ

Proposition 3: The Hamiltonian system (2) enters a neighborhood of $(q^\star, 0)$ upon the application of u^θ as long as the optimal value of the optimization problem (13) is sufficiently small.

Proof: Let $\theta^\star$ denote an optimal solution of the problem (13) so that $G^\perp \mathbb{U}^{\theta^\star} = 0$, and let the corresponding control law be denoted by $u^{\theta^\star}$. By Proposition 1, the control law $u^{\theta^\star}$ asymptotically stabilizes $x^\star = (q^\star, 0)$. By Proposition 2, we know that u^θ is a continuous function of the optimal value J. It is well-known that the solution of the dynamical system (2) is a continuous function of u, hence it is also a continuous function of the parameters θ [31].

Combining these continuity results, we conclude that there exists a time horizon $T > 0$ such that the flow $\phi\left(t; u^\theta(x)\right)$ of the ODE (2) under the application of u^θ satisfies $\phi(T) \in B_r(x^\star)$, where $B_r(x)$ denotes a ball of radius r around x. In this context, the radius r is a function of the tolerance of the optimization algorithm. ■

Proposition 4: The learning-based IDAPBC control law u^θ can be combined with a linear stabilizing controller $\hat{u}$, such as the Linear Quadratic Regulator (LQR), in order to asymptotically stabilize system (2) at $x^\star$.

Proof: In Proposition 3 we have shown that the closed-loop trajectories of (2) passes through a neighborhood $B_r(x^\star)$, and r is a continuous function of the optimization precision. Suppose the region of attraction of the linear control law $\hat{u}(x)$ contains $B_{\hat{r}}(x^\star)$. Choose δ_2 sufficiently small such that $\exists T > 0$ with $\phi(T) \in B_{\hat{r}}(x^\star)$. This guarantees that the trajectories of (2) asymptotically converge to $x^\star$ as $t \to \infty$ under the application of u^θ whenever $x \notin B_{\hat{r}}(x^\star)$ and $\hat{u}$ whenever $x \in B_{\hat{r}}(x^\star)$. ■