ROBUST CONTROL OF CONTACT-RICH ROBOTS VIA NEURAL BAYESIAN INFERENCE

by

Nardos Ayele Ashenafi

A dissertation

submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

Boise State University

May 2023

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1:

INTRODUCTION

Many robotics applications consist of hybrid systems that exhibit both continuous and discrete state transitions. Common examples of hybrid systems are contact-rich mechanisms such as legged robots and manipulators. These mechanisms experience contact forces from their interaction with the environment, causing them to undergo mode changes. Controlling contact-rich robots comes with two main complications. First, it is difficult to find one optimal controller that can achieve the desired performance in all modes of the hybrid system. For instance, multiagent manipulation [2] uses a group of robots to cooperatively execute a task, such as maneuvering an object in space. As we change the number of robots establishing contact with the object, we find unique optimal control schemes for each contact mode. In these scenarios, it may be best to detect event triggers during mode changes and manually switch between several optimal controllers. However, such techniques do not scale well to contact-rich systems with numerous modes. A dynamical system with k contact events can have up to 2^k contact combinations. It is quite tedious, and in some cases impossible, to parameterize the effects of these contact combinations and find individual controllers for each mode. Moreover, it is difficult to parameterize the relationship between the states and the conditional that triggers the control switch. To address this issue, we propose a data-driven framework that learns a mixture of expert controllers for contact-rich systems. This technique infers the experts and a gating network, which determines the control switching

scheme based on observed states. Moreover, we learn optimal expert controllers by injecting a performance objective that characterizes the desired behavior of the resulting closed-loop system.

The second complication in the control of hybrid systems is that they operate in an environment that is not known completely or modeled accurately. For instance, a legged robot needs to perform robustly on uneven terrain. Similarly, manipulators need to hold a firm grip on objects of all textures and shapes. 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. Model-free reinforcement learning is an example of a technique that relies on repeated interactions with the unknown environment [3, 4, 5]. While this technique offers more flexibility on how the control policies are inferred from unknown dynamics, they do not provide the physical structure required to infer stability properties. On the other hand, data-driven techniques trained in simulation, such as neural passivity-based control (NeuralPbc [6] and Neural-Idapbc [7]) offer more insight on the stability of the system but strongly rely on the dynamical model. The use of inaccurate models may lead to poor performance or even instability. This raises concerns regarding model uncertainties, especially in hybrid dynamical systems.

Bayesian learning (BL) [8, 9] offers an alternative method to simultaneously combat model uncertainties while preserving the useful stability analysis in data-driven frameworks. BL is typically used to characterize uncertainties of a dynamical system with a stochastic model. For instance, [10] models uncertainties caused by disturbances, such as the effect of wind gusts on quadcopters via Bayesian inference. Similar approach is shown in [11, 12], where a stochastic dynamical model is constructed via BL techniques, and utilized in data-driven control synthesis executed in simulation. Adaptive control framework is provided

in [13], 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 passivity-based control (PBC) with the robustness properties of BL.

In this work, we present a unified framework that simultaneously combines data-driven techniques and rigorously addresses model uncertainties using Bayesian learning. We aim to apply BL and develop an algorithm that finds a suitable probability distribution of the policy automatically. In contrast to deterministic optimization, this approach provides a probability distribution over the parameters of the controller instead of point estimates, providing a way to reason about model uncertainties and measurement noise during the learning process. We first demonstrate the efficacy of this technique on smooth systems, such as the simple pendulum and the inertia wheel pendulum, in simulation and real-world experiment. Then we extend the framework to contact-rich systems and evaluate its performance on the rimless wheel, a simplified walking machine.

CHAPTER 2:

SWITCHING CONTROL WITH DEEP-NET MIXTURE OF EXPERTS

2.1 Background

In this section, we provide a brief introduction to contact modeling with the linear complementarity formulation. We also present a summary of the mixture of experts architecture and its uses in machine learning.

2.1.1 Contact Modeling with Linear Complementarity Problem

Suppose a hybrid dynamical system consists of k contact events, each introducing normal contact forces $\lambda_N \in \mathbb{R}^k$ and Coulomb friction forces $\lambda_T \in \mathbb{R}^k$ to the overall system. We can model this hybrid system with the measure equality [14]

$$\begin{aligned} M(q)\,\mathrm{d}\dot{q} + h(q,\dot{q})\,\mathrm{d}t - \mathrm{d}R = 0, \\ h(q,\dot{q}) = C(q,\dot{q})\dot{q} + G(q) - Bu(q,\dot{q}), \end{aligned} \tag{2.1}$$

where $x \in \mathcal{X} \subset \mathbb{R}^{2m}$ consists of robot position $q \in \mathbb{R}^m$ and velocities $\dot{q}$, $M \in \mathbb{R}^{m\times m}$ denotes the positive-definite mass matrix, $C \in \mathbb{R}^{m\times m}$ holds the Coriolis and centripetal terms, and $G \in \mathbb{R}^m$ is the gravitational term. The matrix $B \in \mathbb{R}^{m\times n}$ maps the input $u \in \mathbb{R}^n$ to the generalized coordinates and $\mathrm{d}R \in \mathbb{R}^m$ represents the force measure of contact forces

and Coulomb friction exerted on the system. In contrast to smooth dynamical systems, the hybrid system (2.1) exhibits contact forces that enforce the geometric and kinematic constraints of surfaces in contact. For instance, the contact force between two objects in collision characterizes the no-penetration conditions and the post-impact velocities of the objects. Resolving these contact forces accurately can be difficult and computationally expensive. Most collision simulators work on the kinematic level as opposed to the dynamic level. For instance, there are event detection methods that simply change the velocity of the moving objects at the time of impact. One of the drawbacks of these techniques is finding the exact time the contact occurs in high speed collision. There is the additional difficulty of identifying the Coulomb friction, which is especially prone to Painlevé Paradox [15] in high friction scenarios. The linear complementarity formulation in [14] provides a rigorous technique to resolve contact forces, Coulomb friction and impact forces in a hybrid system. The work presents an optimization problem that searches for *contact force and post-impact velocity* pairs that obey the geometric and kinematic constraints during contacts or impacts.

The linear complementarity formulation is constructed as follows. We define a vector of gap functions $g_N(q) \in \mathbb{R}^k$ that measure the Euclidean distance between the contact surfaces. Let $\gamma_N = \dot{g}_N(q)$ be the normal relative velocity and γ_T the tangential relative velocity between contact surfaces. The linear complementarity formulation imposes a unilateral constraint between contact forces and relative velocities given by:

$$\begin{aligned} 0 \leq \begin{pmatrix} \xi_N(q,\dot{q}) \\ \xi_T(q,\dot{q}) \end{pmatrix} \perp \begin{pmatrix} \lambda_N \\ \lambda_T \end{pmatrix} \geq 0, \\ \begin{pmatrix} \xi_N(q,\dot{q}) \\ \xi_T(q,\dot{q}) \end{pmatrix} = \begin{pmatrix} (1+\epsilon_N)\gamma_N(q,\dot{q}) \\ (1+\epsilon_T)\gamma_T(q,\dot{q}) \end{pmatrix}, \end{aligned} \tag{2.2}$$

where ϵ_N and ϵ_T are the normal and tangential coefficients of restitution respectively. From the dynamics in (2.1), ξ_N and ξ_T can be expressed as an affine function over the contact forces λ_N and λ_T as follows [14]:

$$\begin{pmatrix} \xi_N \\ \xi_R \\ \lambda_L \end{pmatrix} = A \begin{pmatrix} \lambda_N \\ \lambda_R \\ \xi_L \end{pmatrix} + b,$$

$$A = \begin{bmatrix} W_N M^{-1}(W_N - W_T\mu) & W_N M^{-1} W_N & 0 \\ W_T M^{-1}(W_N - W_T\mu) & W_T M^{-1} W_T & I_k \\ 2\mu & -I_k & 0 \end{bmatrix}, \; b = \begin{bmatrix} W_N M^{-1} h\Delta t + (I_k + \epsilon_N)\gamma_N \\ W_T M^{-1} h\Delta t + (I_k + \epsilon_T)\gamma_T \\ 0 \end{bmatrix}$$

$$\begin{pmatrix} \xi_T \\ \lambda_R \\ \lambda_L \end{pmatrix} = \begin{pmatrix} \xi_R - \xi_L \\ \mu\lambda_N + \lambda_T \\ \mu\lambda_N - \lambda_T \end{pmatrix}$$

where μ is the coefficient of friction, I_k is the $k \times k$ identity matrix, W_N and W_T map the velocity vector $\dot{q}$ to γ_N and γ_T, respectively, and Δt is the integration time step. The linear complementarity problem (LCP) (2.2) can be posed as the following feasibility problem:

$$0 \leq \left[A \begin{pmatrix} \lambda_N \\ \lambda_R \\ \xi_L \end{pmatrix} + b \right] \perp \begin{pmatrix} \lambda_N \\ \lambda_R \\ \xi_L \end{pmatrix} \geq 0, \tag{2.3}$$

which can be solved with various optimization techniques.

We follow Moreau's time stepping algorithm [14] outlined in Algorithm (1) to resolve the complementarity constraint in (2.3) and numerically integrate the dynamics (2.1). While the complementarity constraint can be posed as a feasibility problem, the presence of Coulomb friction makes it a non-convex optimization problem. We use a pivoting (basis-exchange) technique called Lemke's algorithm [16] to find the solution to the linear complementarity problem (2.3). This allows us to differentiate the solution to the LCP in our efforts to use machine learning techniques.

Algorithm 1 Moreau's Time Stepping Algorithm

Input: $x(0) = (q(0), \dot{q}(0))$

$\phi \leftarrow [x(0)]$ ▷ Initial States

for $t = 0 : \Delta t : T$ **do** ▷ Time stepping

  $t_M = t + {}^{\Delta t}/_{2}$

  $q(t_M) = q(t) + ({}^{\Delta t}/_{2})\dot{q}(t)$ ▷ Half-time step integration

  $\lambda_N, \lambda_T \leftarrow \text{Lemke}(q(t_M), \dot{q}(t))$ ▷ Lemke [16]

  $\dot{q}(t + \Delta t) = M^{-1}(W_T \lambda_T + W_N \lambda_N + h\Delta t) + \dot{q}(t)$ ▷ Apply contact forces

  $q(t + \Delta t) = q(t_M) + ({}^{\Delta t}/_{2})\dot{q}(t + \Delta t)$

  $\phi \leftarrow \phi \cup [x(t + \Delta t)]$ ▷ Save trajectory

return ϕ

2.1.2 Mixture of Expert Models

The mixture of experts (MOE) architecture is primarily used to learn an ensemble of expert models that best fit high variance or multi-modal datasets. Suppose we are trying to model the source of the dataset shown on the left of Figure 2.1. A single Gaussian distribution provides a poor fit to the multi-modal data. MOE framework allows us to fit multiple uni-modal distributions as shown on the right of Figure 2.1. This technique uses a gating network to divide the input space $x \in \mathcal{X} \subset \mathbb{R}^{2m}$ into state partitions within which a single expert is active. The objective is to learn the expert models and the gating network

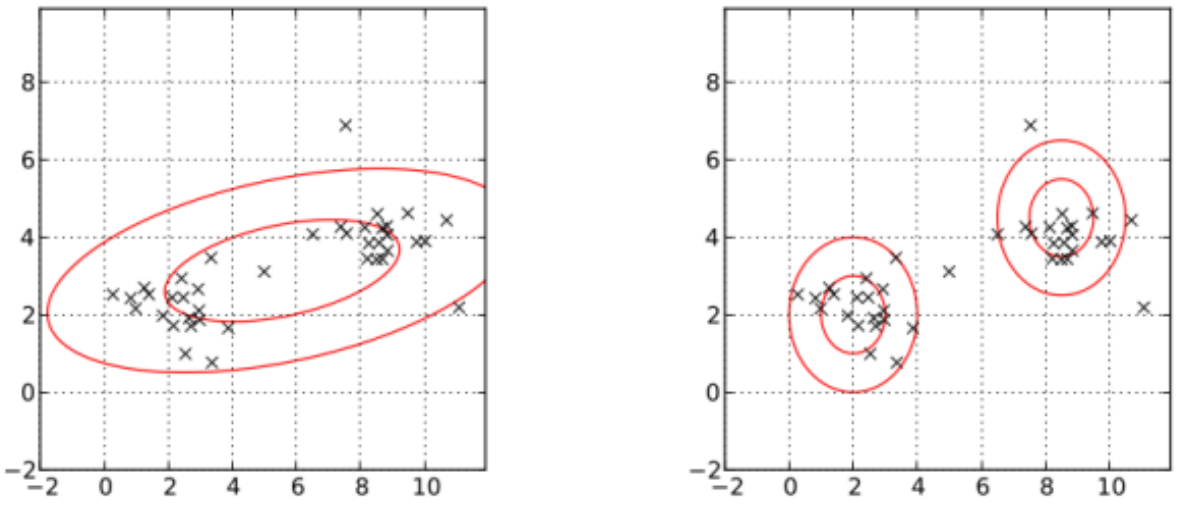

Figure 2.1: Left: Fit with one Gaussian distribution. Right: Fit with Gaussian mixture of 2 experts [1]

that best fit the dataset.

The expert models and the gating networks can take several forms. Gaussian experts and gating networks are commonly used in the MOE framework to find a multi-modal probabilistic model from a small amount of data [17]. However, as the observations from the dataset grow large, the Gaussian experts impose large computational and memory overhead [18]. Moreover, Gaussian gating networks can only provide a quadratic parameterization of the boundaries of the state partitions. Hence, we leverage the universal approximation capabilities of neural networks for both the experts and the gating network.

Let $F(x)$ denote a collection of N_F expert models $F(x) = \{F_1(x), \ldots, F_{N_F}(x)\}$, whose parameters are given by the set $\theta = \{\theta_1, \ldots, \theta_{N_F}\}$. The gating network $\mathbf{P}(x|\psi) := (P_1(x|\psi), \ldots, P_{N_F}(x|\psi))$ is a collection of probabilities, where $P_i(x|\psi)$ denotes the probability of state x belonging to state partition $i, i \in \{1, \ldots, N_F\}$. We parameterize the gating network with a neural network $\mathbf{P}(x|\psi) : \mathcal{X} \to \mathbb{R}^{N_F}$ with parameters ψ, and the output corresponds to the vector $[P_1(x|\psi), \ldots, P_{N_F}(x|\psi)]$. We use a SOFTMAX activation function on the last layer to

ensure that the probabilities $P_i(x|\psi)$ over all state partitions i sum to one. The objective is to learn the decision parameters (ψ, θ) from observed data.

We use a categorical probability distribution to sample a state partition based on the outputs of the gating network as follows [18]

$$i|x, \psi \sim \text{Categorical}(\mathbf{P}(x|\psi)). \tag{2.4}$$

The prediction from the mixture of experts $F(x;\theta)$ is given by the linear combination over the *responsibility* of each state partition.

$$I = \underset{i}{\text{argmax}}\{P_i(x|\psi)\}$$
$$F(x;\theta) = F_I(x;\theta_I)$$

There are several Bayesian inference techniques to learn the parameters of the expert models and the gating network, one of which is *expectation maximization* (EM). In this technique, we compose the expectation over the log likelihood as [17]

$$\mathbb{L}(\mathbb{D}|\theta,\psi) := \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{N_F} -\|F_i(x_j;\theta_i) - y_j\|^2 P_i(x_j|\psi) \tag{2.5}$$

where $\mathbb{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ is a collection of N training datasets, and $\|F_i(x_j;\theta_i) - y_j\|$ is the error in the prediction made by the expert i. Notice that the likelihood measures how likely the current parameters (ψ, θ) are to generate the sampled data $\mathbb{D}$. The goal is to maximize the likelihood and consequently minimize prediction error. The likelihood (2.5) is maximum when the parameter θ_i has the lowest prediction error and the highest probability of getting selected by the gating network. The standard EM approach evaluates (2.5) and

uses gradient-based techniques to iteratively update the parameters (ψ, θ).

2.2 Mixture of Expert Controllers

In this section, we present a data-driven control design framework for hybrid dynamical systems. The objective of this framework is to learn a mixture of expert controllers and their responsibilities as determined by the gating network. This technique allows us to observe the effects of contacts from the closed loop trajectories and learn a switching mechanism to best control the hybrid system in all modes. We also learn optimal expert controllers that achieve a certain performance objective through data-driven techniques.

Let $\phi(x_0, u, T)$ denote a closed loop trajectory with initial state x_0 integrated for the time horizon T. For every state x in the trajectory, the control law first samples state partition (bin) number from the categorical distribution in (2.4). The control input at state x is

$$u(x; \psi, \theta) = \{F_i(x; \theta_i) \mid i \sim \text{Categorical}(P(x|\psi))\}$$

Without prior knowledge injected to the gating network, the samples from the categorical distribution initially explore the performance of most, if not all, of the expert controllers. We use the running cost $\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ to measure the performance of the sampled experts, which we discuss in depth in the next section. The running cost plays the role of *prediction error* in the construction of the likelihood as shown in (2.5). The goal is to learn the decision parameters (ψ, θ) that minimize the running cost for all initial states in the state space. We

pose the search over the parameters of the experts and the gating network as

$$\begin{aligned} \underset{\psi,\theta}{\text{minimize}} \quad & \ell(x,u), \\ \text{subject to} \quad & M(q)\,\mathrm{d}\dot{q} + h(x;\psi,\theta)\,\mathrm{d}t - \mathrm{d}R = 0, \\ & u = \{F_i(x;\theta_i) \mid i \sim \text{Categorical}(P(x|\psi))\} \end{aligned} \tag{2.6}$$

In the coming sections, we provide a procedure to solve the optimization problem (2.6).

Performance Objective

We present two viable choices for the running cost function.

1. Accumulated loss: is the total quadratic loss between the desired state x^* and the states generated under the current control law. We also incur a cost on the control authority as follows.

$$\ell(x,u) = \frac{1}{2}(x-x^*)^\top Q(x-x^*) + \frac{1}{2}u^\top R u, \tag{2.7}$$

where $Q \succ 0$ is positive definite matrix and $R \succeq 0$ is a positive semi-definite matrix. This construction encourages trajectories to reach the desired equilibrium with minimum effort and shortest time.

We construct the corresponding likelihood as follows. At each integration step, we evaluate the performance of each expert as shown in Algorithm 2. The running cost of each expert is weighed by its *responsibility* $P_i(x|\psi)$. The resulting likelihood is given

by

$$\mathbb{L}(\phi) = \int_0^T \left(\sum_{i=1}^{N_F} \Big[-\ell\Big(x(t), F_i\Big) P_i\Big(x(t)|\psi\Big) \Big] \right) \mathrm{d}t. \tag{2.8}$$

Algorithm 2 Accumulated Loss

Input: x_0, θ, ψ

$\mathbb{L} \leftarrow 0$
for $t = 0 : \Delta t : T$ **do**
  for $j = 1 : N_F$ **do** ▷ Evaluate performance of each expert
    $\hat{x} \leftarrow$ `Moreau's one time step`$(x(t), F_j(x(t), \theta_j))$ ▷ Algorithm (1)
    $\mathbb{L} \leftarrow \mathbb{L} - \ell(\hat{x}, F_j) P_j(\hat{x}|\psi)$
  $i \sim \text{Categorical}\Big(P(x(t)|\psi)\Big)$ ▷ Sample a bin number
  $x(t + \Delta t) \leftarrow$ `Moreau's one time step`$(x(t), F_i(x(t), \theta_i))$
return $\phi, \mathbb{L}$

2. Minimum trajectory loss (MTL): While a trivial choice, accumulated loss may not reflect the desired behavior of some dynamical systems. For instance, suppose we want to swing-up the simple pendulum to the upright equilibrium. For an underactuated pendulum, the controller needs to swing about the downward equilibrium, moving the states closer and further away from the upright. Accumulated loss incurs a lot of cost in such scenarios and the control search would get stuck in local minima. In such cases, a successful loss function encourages trajectories that *eventually* lead to a minimum cost. Hence, we compose MTL as

$$\begin{aligned} t_{min} &= \inf_t \{\ell(x, u) : x \in \phi(x_0, u, T)\} \\ \mathbb{L}(\phi) &= -\frac{\ell(x(t_{min}), u)}{C} \sum_{t=0}^{t_{min}} P_i(x(t)|\psi) \end{aligned} \tag{2.9}$$

where $C > 0$ is a normalization factor. Unlike accumulated loss, MTL does not particularly reward low effort or short time trajectories, but it equally rewards two trajectories as long as they both reached the desired state within the time horizon T. Moreover, MTL does not need to evaluate the performance of each expert at every integration step, as shown in Algorithm (3); this greatly reduces the computational cost.

Algorithm 3 Minimum Trajectory Loss

Input: x_0, θ, ψ

$\phi \leftarrow \{x_0\}$

for $t = 0 : \Delta t : T$ **do**

  $i \sim \text{Categorical}\Big(P(x(t)|\psi)\Big)$ ▷ Sample a bin number

  $x(t + \Delta t) \leftarrow$ `Moreau's one time step`$(x(t), F_i(x(t), \theta_i))$ ▷ Algorithm (1)

  $\phi \leftarrow \phi \cup x(t + \Delta t)$ ▷ Save trajectory

$t_{min} = \inf_t \{\ell(x, u) : x \in \phi\}$

$\mathbb{L} = -\ell(\phi(t_{min}), u) \sum_{t=0}^{t_{min}} P_i(\phi(t)|\psi)$

return $\phi, \mathbb{L}$

State Sampling

We intend to find a solution to the optimization problem in (2.6) for all initial states x_0 in the state space. To efficiently sample the initial states, we use a combination of greedy and explorative state sampling techniques. Greedy state sampling, commonly known as DAgger, is a technique adapted from imitation learning [19]. This technique refines the controller on the states most visited under the current parameters (ψ, θ). We first sample several initial states randomly and generate trajectories using the current parameters. Then, we randomly select N_d samples from the visited states. The explorative state sampling technique helps recover from locally optimal solutions. This is achieved by sampling N_r initial states around the neighborhood of the desired state x^*. In a single batch training, we

Algorithm 4 Solution to the Optimization Problem (2.6)

$\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ $N_\mathcal{D}$ initial state samples
while $i <$ `maximum iteration` **do**
  $J \leftarrow 0$ ▷ Batch loss
  for $z_0 \in \mathcal{D}_N$ **do**
    $\phi, \mathbb{L} =$ `Performance objective`(x_0, ψ, θ) ▷ Algorithm (2) or (3)
    $J \leftarrow J + \mathbb{L}/N_\mathcal{D}$
  $\theta \leftarrow \theta + \alpha_i \partial J / \partial \theta$ ▷ SGD step
  $\psi \leftarrow \psi + \alpha_i \partial J / \partial \psi$
  $\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ New initial state samples
  $i \leftarrow i + 1$
return θ

compute the running cost as an expectation over $N_\mathcal{D} = N_d + N_r$ samples as follows:

$$J(\phi, u) = \mathbb{E}_{x_0 \sim \mathcal{D}_N}[\mathbb{L}(\phi(x_0, u, T))]$$

where $\mathcal{D}_N$ is a collection of $N_\mathcal{D}$ initial state samples.

Training MOE

We solve the optimization problem given in (2.6) with stochastic gradient descent (SGD). We invoke a variant of SGD known as Adam [20] to efficiently train the parameters with adaptive learning rates α_i. The full training procedure is outlined in Algorithm (4). We leverage forward-mode auto-differentiation technique [21] to back-propagate on the gradient of the likelihood with respect to the learned parameters.

Back-propagation through Hybrid Systems

The training framework outlined (2.6) allows us to observe the effects of contacts in the closed loop trajectories and infer a controller that either uses the contact to its advantage or minimizes its adverse effects. In this section, we look at the relevant parts of the back-propagation to give insight on how this is achieved. We also show that despite the state jumps in the hybrid dynamics, the derivatives involved in the back-propagation are well-defined.

Suppose we generate a short trajectory ϕ with the sampled expert control parameter θ_i. Forward-mode auto-differentiation evaluates the gradient of the accumulated cost with respect to θ_i as

$$\frac{\partial \ell}{\partial \theta_i} = \sum_{t=0}^{T} \frac{\partial \ell}{\partial x_t} \frac{\partial x_t}{\partial \theta_i},$$

where $R = 0$ for simplicity. Without loss of generality, we take one integration step for the reminder of this discussion. In that step, a contact event is triggered causing the velocities to jump between the initial state x_0 and the following state x_1. Hence, we focus on

$$\frac{\partial \ell}{\partial \theta_i} = \frac{\partial \ell}{\partial x_1} \frac{\partial x_1}{\partial \theta_i},$$

We can expand the gradient further as

$$\frac{\partial \ell}{\partial \theta_i} = \frac{\partial \ell}{\partial x_1} \Big(\frac{\partial x_1}{\partial u} \frac{\partial u}{\partial \theta_i} + \frac{\partial x_1}{\partial \lambda} \frac{\partial \lambda}{\partial \theta_i} \Big),$$

where λ holds the contact forces. We can compute first term from Moreau's integration step

as

$$\frac{\partial x_1}{\partial u}\frac{\partial u}{\partial \theta_i} = \begin{bmatrix} M^{-1}B\Delta t^2/2 \\ M^{-1}B\Delta t \end{bmatrix} \frac{\partial u}{\partial \theta_i},$$

At first glance, the derivative $\frac{\partial x_1}{\partial \lambda}\frac{\partial \lambda}{\partial \theta_i}$ may seem ill-defined due to the discontinuity in the states. A closer observation reveals that $\frac{\partial x_1}{\partial \lambda}$ determines how the *post-impact velocity is affected by the contact forces* . In fact, the derivative can be found from Moreau's integration as

$$\frac{\partial x_1}{\partial \lambda} = \begin{bmatrix} W_N & W_T \end{bmatrix},$$

demonstrating that the gradient is well-defined even if a state jump has occurred. This term is crucial in adjusting the decision parameters in response to how the contact force assists or inhibits the system. If the contact forces affect the *post-impact* velocity such that the resulting generalized coordinates are closer to the desired state x^*, then the gradient $\frac{\partial \ell}{\partial x_1}\frac{\partial x_1}{\partial \lambda}\frac{\partial \lambda}{\partial \theta_i}$ adjusts the parameter θ_i to favor states undergoing contact events. We in fact demonstrate this behavior in simulation and real-world experiments in Section 2.3.2. Conversely, if the contact forces move the states further away from x^*, the gradient leads to control parameters that attempt to recover from the outcomes of the contact events. We also demonstrate this behavior on a walking robot example in Section 3.3.5.

2.3 Experimental Results

We demonstrate the efficacy of the mixture of expert controllers in simulation and real-world experiments. In the first case study, we learn a gating network that switches between two unstable closed-loop systems to result in a piecewise-stable system. Then, we

find switching MOE controllers to swing up the classical cartpole mechanism enclosed with wall barriers.

2.3.1 Stable Switching Between Unstable Systems

Suppose we have two linear closed-loop systems of the form

$$\begin{aligned} \dot{x} = A_1 x = \begin{bmatrix} 0 & -1 \\ 2 & 0 \end{bmatrix} x \\ \dot{x} = A_2 x = \begin{bmatrix} 0 & -2 \\ 1 & 0 \end{bmatrix} x. \end{aligned} \tag{2.10}$$

Even if both systems are unstable, it is possible to find a state-dependent switching rule that makes the resulting switched system stable [22]. We aim to learn the parameters ψ of the gating network $P(x|\psi)$ such that the switching system converges to the desired equilibrium $x^* = (0, 0)$. The gating network is a fully-connected neural net with one hidden layer (2 input states $\rightarrow$6 neurons $\rightarrow$4 outputs) and an ELU activation function [23]. We constrain the maximum number of state partitions to 4. Each state partition has a corresponding controller parameter $\theta_i \in \mathbb{R}$. The control law is a sample from the Bernoulli probability distribution

$$F_i(\theta_i) = \begin{cases} 0, & \theta_i > \frac{1}{2}, \\ 1, & \theta_i \leq \frac{1}{2}, \end{cases} \tag{2.11}$$

where $F_i = 0$ corresponds to the first dynamics $\dot{x} = A_1 x$ and $F_i = 1$ corresponds to $\dot{x} = A_2 x$. We use the SIGMOID activation function to limit θ_i between 0 and 1. We generate a trajectory following the procedure in Algorithm (5). The likelihood is modified to account for the

probabilistic control as

$$\mathbb{L}(\phi) = \int_0^T \left(\sum_{i=1}^{N_F} -\Big[\theta_i \ell\Big(x(t), u(\theta_i)\Big) + (1-\theta_i)\ell\Big(x(t), u(1-\theta_i)\Big)\Big] P\Big(c_i = i | x(t), \psi\Big) \right) \mathrm{d}t.$$

Algorithm 5 Stable Switching between Unstable Systems

Input: $x(0)$

$\phi \leftarrow [x(0)]$ ▷ Initial States

for $t \in 0 : \Delta t : T$ **do**

  $i \sim \text{Categorical}\Big(P(x(t)|\psi)\Big)$ ▷ Sample a bin number

  $F_i \sim \text{Bernoulli}\Big(\text{Sigmoid}(\theta_i)\Big)$

  $x(t+\Delta t) = (1-F_i)A_1 x(t) + F_i\, A_2 x(t)$

  $\phi \leftarrow \phi \cup [x(t+\Delta t)]$

return ϕ

The resulting switching system is shown in Figure 2.2. The training uses only 3 out of the 4 state partitions available. The state partition in Figure 2.2 matches the analytical solution to the stable switching system given as [22]

$$\dot{x} = \begin{cases} A_1 x, \ x_1 x_2 \leq 0, \\ A_2 x, \ x_1 x_2 > 0. \end{cases}$$

where $x = [x_1, x_2]$. The training progress is shown in Figure 2.3. The three rows in the figure depict the performance of the training after 0, 200 and 1400 parameter updates, respectively. The sampled trajectory on the top left figure shows that the initial parameters created unstable switching between the two systems. After only few parameter updates, the training finds a stable switching mechanism, but it does not yet converge to the desired equilibrium x^*. The explorative state sampling technique visits states close to the desired equilibrium, allowing the training to learn the distinct boundaries of each partition at the

origin.

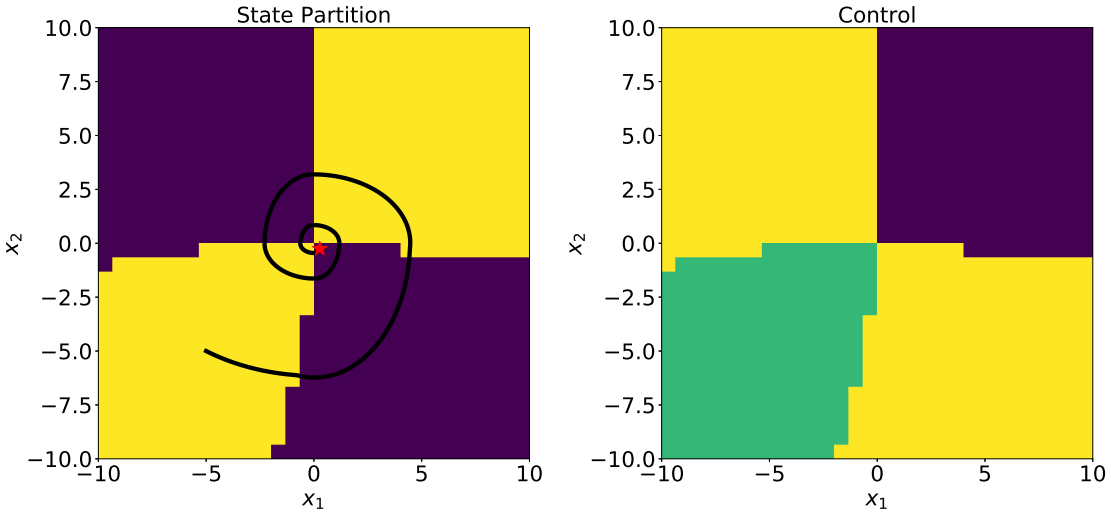

Figure 2.2: Final stable switching system. Left: Control input $F_i = \{0, 1\}$ in the state space. Purple corresponds to $F_i = 0$ or $\dot{x} = A_1 x$. Yellow corresponds to $F_i = 1$ or $\dot{x} = A_2 x$. The trajectory starts at $x_0 = [-5, -5]$ and converges to the origin shown by the red star. Right: State partition from the output of the gating network. The training converged to 3 state partitions that can stabilize the system.

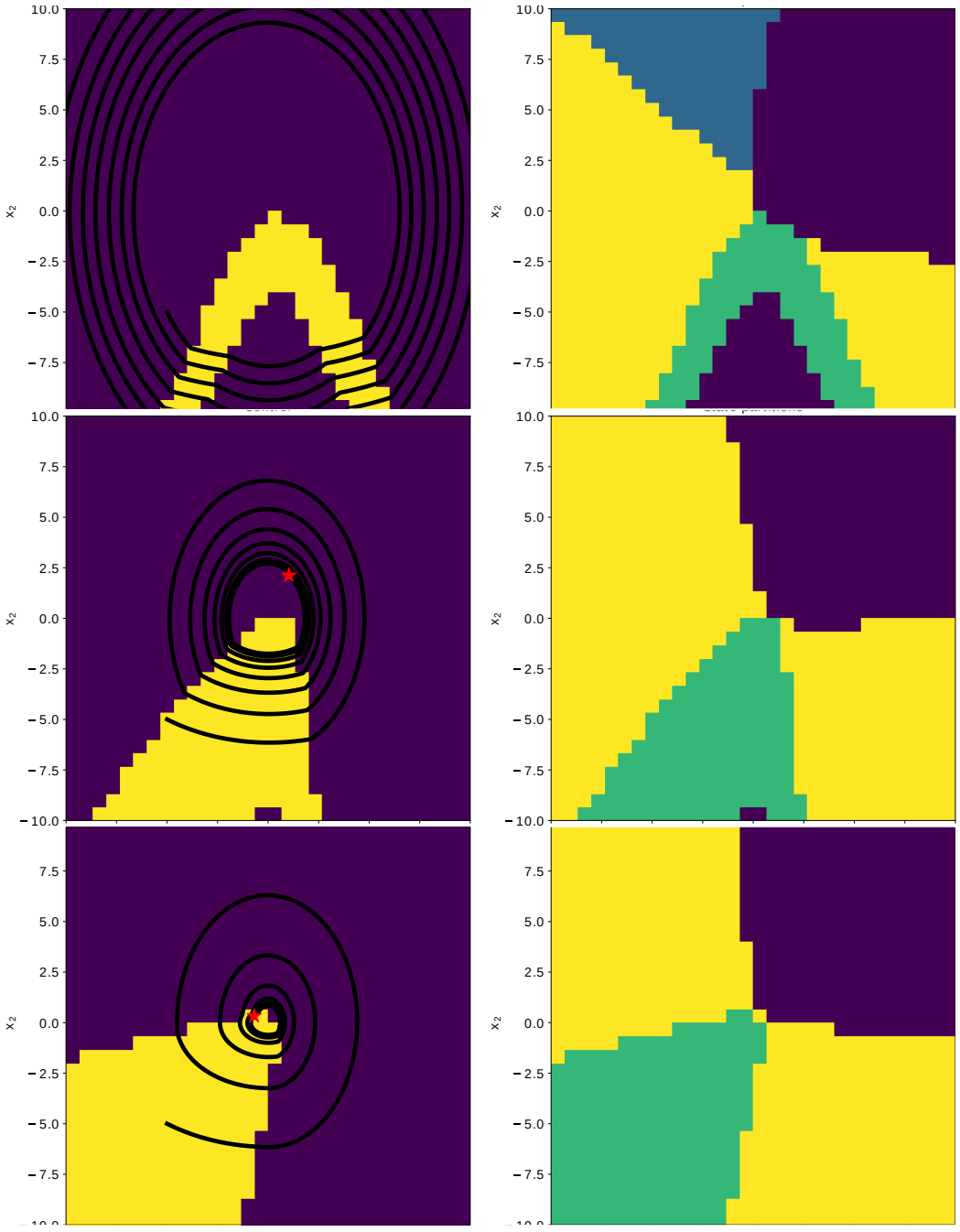

Figure 2.3: Training progress. Top row: the control and state partition at the initial parameters. Second row: after 200 parameter updates. Third row: after 1400 parameter updates. The final solution is shown in Figure 2.2

2.3.2 Cartpole with Wall Contacts

In this section, we take the classical cartpole swing-up problem and introduce contact events from two barriers as shown Figure 2.4. We demonstrate the performance of the mixture of expert controllers in simulation and real-world experiments. Lastly, we compare the performance of the MOE controllers against a single swing-up controller.

System Model

The cartpole system consists of a freely rotating pendulum link hinged on an actuated cart. The setup is enclosed by two rigid walls hanging 0.2m from the bottom of the cart. The objective is to use the control authority on the cart in order to swing-up the passive pendulum to the upright. The pendulum spans length of $l = 0.2$m and its mass $m_c = 0.75$kg is concentrated at the distance $l_{cm} = l/2$ from the hinge. The cart alone has a mass of $m_p = 0.165$ kg. The viscous friction in the cart wheels is characterized by the coefficient $b = 1.2$ $^{\mathrm{N}\cdot\mathrm{sec}}/_{\mathrm{m}}$. The dynamics of the system is given by (2.1) where

$$
\begin{aligned}
M(q) &= \begin{bmatrix} m_c + m_p & -m_p + l_{cm}\cos(\theta_p) \\ -m_p l_{cm}\cos(\theta_p) & m_p l_{cm}^2 + I_1 \end{bmatrix} \\
C(q,\dot{q}) &= \begin{bmatrix} b & m_p l_{cm}\sin(\theta_p) \\ -m_p \sin(\theta_p)/2 & 0 \end{bmatrix} \\
G(q) &= \begin{bmatrix} 0 & -m_p g l_{cm}\sin(\theta_p) \end{bmatrix}^\top \\
B &= [1\ 0]^\top
\end{aligned}
\tag{2.12}
$$

where $q = [x_c; \theta_p]$, x_c is the location of the cart, θ_p is the angle of the pendulum from the vertical and g is the acceleration due to gravity. There are a total of $k = 10$ contact events

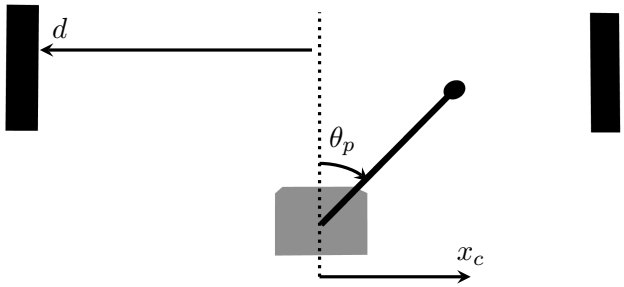

Figure 2.4: Cartpole with wall contacts

between the pendulum and the sides of the walls. We integrate closed-loop trajectories with Moreau time stepping algorithm outlined in Algorithm (1) with an integration time step $\Delta t = 0.001$.

Training

The goal is to learn mixture of expert controllers that stabilize the system at $x^* = (q^*, \dot{q})^* = (0, 0)$. Once the system reaches within a small neighborhood of x^*, we employ Linear Quadratic Regulator (LQR) to stabilize at the desired equilibrium. We use minimum trajectory loss (MTL) discussed in Section 2.2 with time horizon $T = 1.5$s. In each parameter update, we sample $N_{\mathcal{D}} = 4$ initial states through greedy and explorative techniques. The gating network is a fully-connected neural net with two hidden layers (state input $\rightarrow$4 neurons $\rightarrow$3 neurons $\rightarrow$1 output). We constrain the maximum number of state partitions to 3. There are a total of 3 neural net experts, each corresponding to a state partition. The experts are fully-connected neural net with two hidden layers (state input $\rightarrow$10 neurons $\rightarrow$4 neurons $\rightarrow$1 output). The output of the experts correspond to the force applied on the cart.

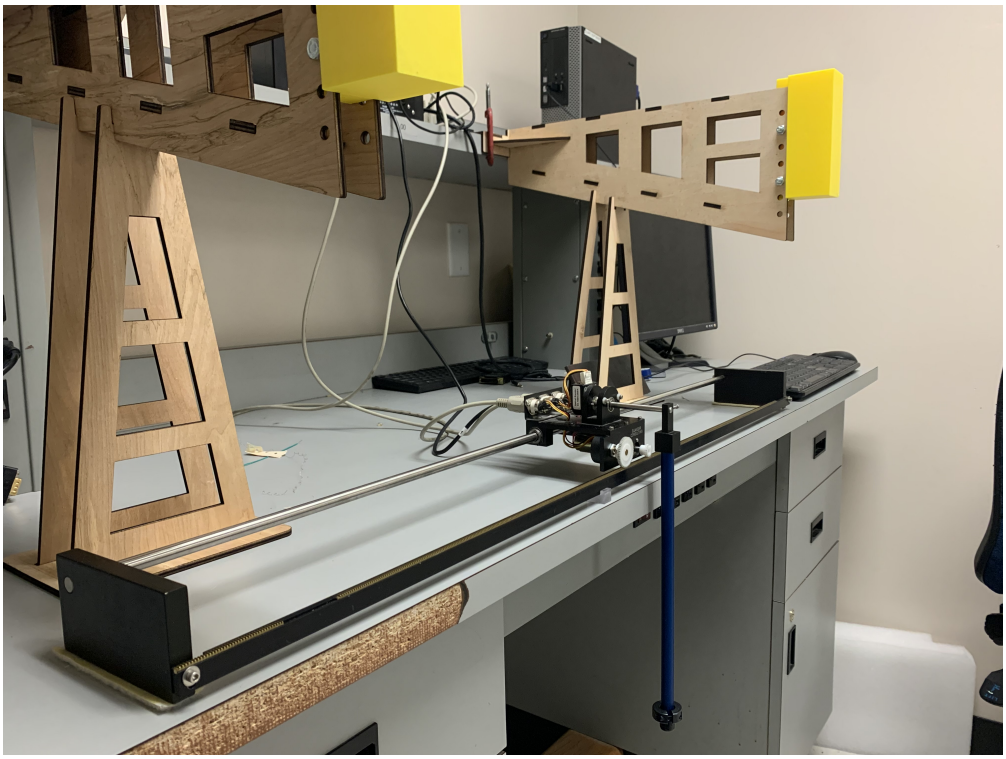

Figure 2.5: Experimental setup of cartpole with wall contacts

Results

We demonstrate the performance of the MOE controllers in simulation and hardware. The hardware is set up as shown in the Figure 2.5. The cart uses a rack and pinion mechanism to translate on the track with zero-slip. One of the wheels of the cart is attached to an optical encoder, from which we can estimate the position and velocity of the cart. There is also an optical encoder rigidly attached to the pendulum link, reporting its orientation. We use MATLAB/Simulink to evaluate the neural nets and pass voltage commands to the DC-motor. We convert the force commands output by the MOE controllers into voltage

commands $V(t)$ as follows.

$$V(t) = \frac{u(x(t); \psi, \theta) + A_m \dot{x}_c}{B_m}$$

where A_m and B_m consist of system parameters of the motor.

Figure 2.6 shows a trajectory generated by the MOE controllers in simulation and hardware. The blue contours represent the level sets of the control input at the pre-impact and post-impact states. The solid line splitting the state space horizontally in half shows the boundary between two state partitions. Even though the gating network can provide up to three state partitions, the training converges to utilizing only two. Figure 2.6 shows that the system successfully avoids contacts during the swing-up phase, which otherwise would have prevented the pendulum from pumping energy from the downward equilibrium. By the time the pendulum approaches the upright equilibrium, it is moving at such high speed that LQR cannot stabilize it. But from several trajectories, we have observed that the system leverages the impact from the wall to lower the speed of the pendulum. Then, the control law switches between experts as the velocity jumps due to impact. The magnitude of the control is such that the MOE controllers apply rapid braking during contact, allowing LQR to catch the pendulum post-impact. The MOE controllers achieve successful swing-up in simulation and real-world, proving the accuracy in the contact modelling and the robustness of the controllers.

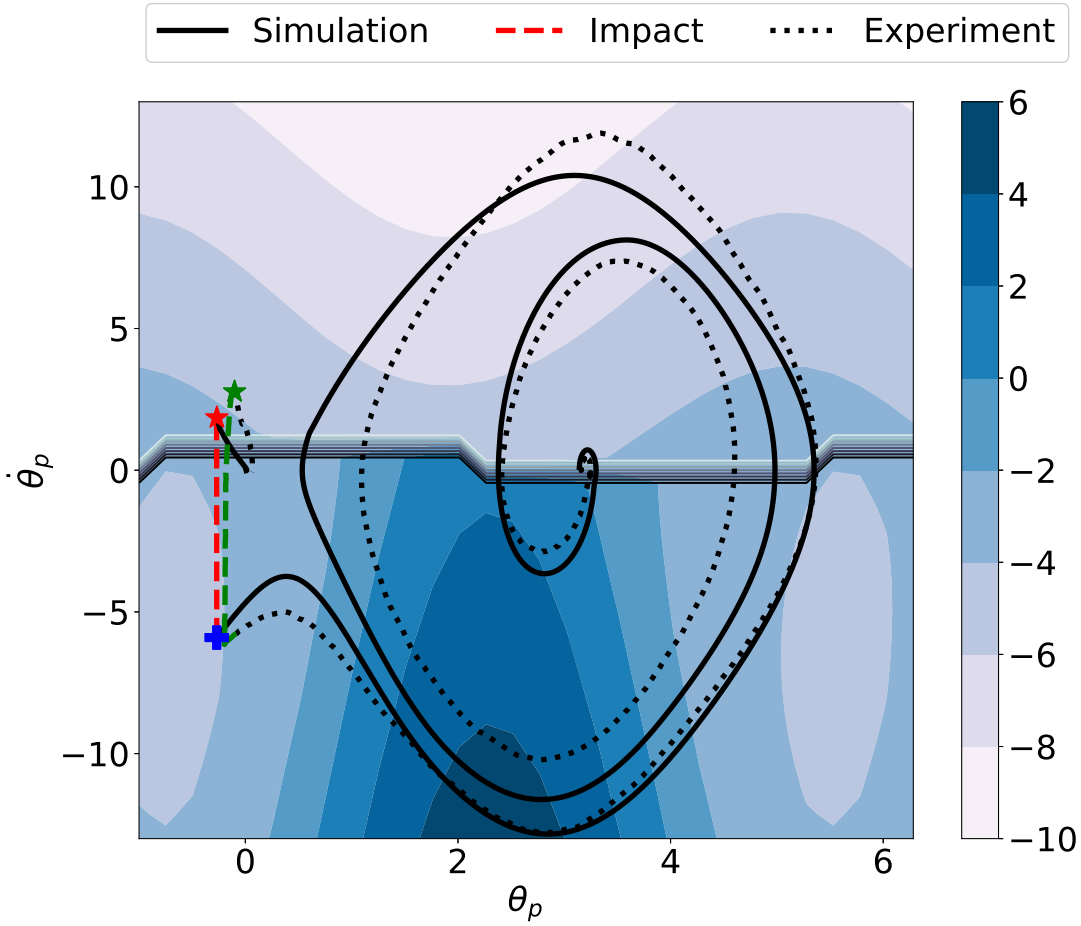

Figure 2.6: A sample trajectory starting from downward equilibrium at rest. The contours correspond to the outputs of the MOE controllers during impact ($x_c = 0.36m, \dot{x}_c = 0^m/_s$). The solid line splitting the state space horizontally in half shows the boundary between the two state partitions. The state partition shows that the controller must switch from one expert to another in order to catch the pendulum post-impact.

Figure 2.7 shows the relationship between the gap function and the state partitions. The red lines represent the boundaries of the state partitions and the contour depicts the level sets of the gap function. Notice that the states within the boundaries of the red lines have large gap values, hence contact events do not occur in this region. This shows

that the gating network has dedicated one expert for states with low risk of contact. The regions outside the red boundaries correspond to the second expert, which is activated to prevent contact during swing-up and to catch the pendulum just after impact, as shown in the trajectory of Figure 2.6. This demonstrates that the state partition depends on the occurrence of contact events. Moreover, the multi-modal nature of MOE controllers allow us to leverage the advantages or prevent the adverse effects of contacts and impacts.

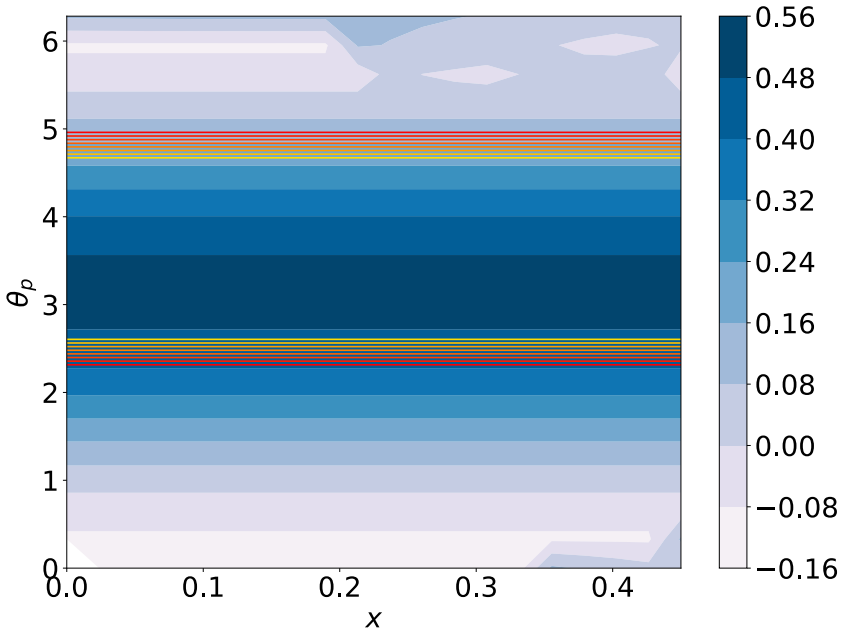

Figure 2.7: The blue contours represent the level sets of the gap function. The red lines show the boundaries of the state partition. There are a total of two experts, one is active between the red boundaries and the other is responsible for regions outside the red lines.

Lastly, we compare the performance of the MOE controllers against a single controller. This controller is parameterized by a neural net (state input $\rightarrow$10 neurons $\rightarrow$4 neurons $\rightarrow$1 output) similar to one of the mixture experts. We train the controller with the same minimum trajectory loss and training parameters as the MOE. Once the controller swings

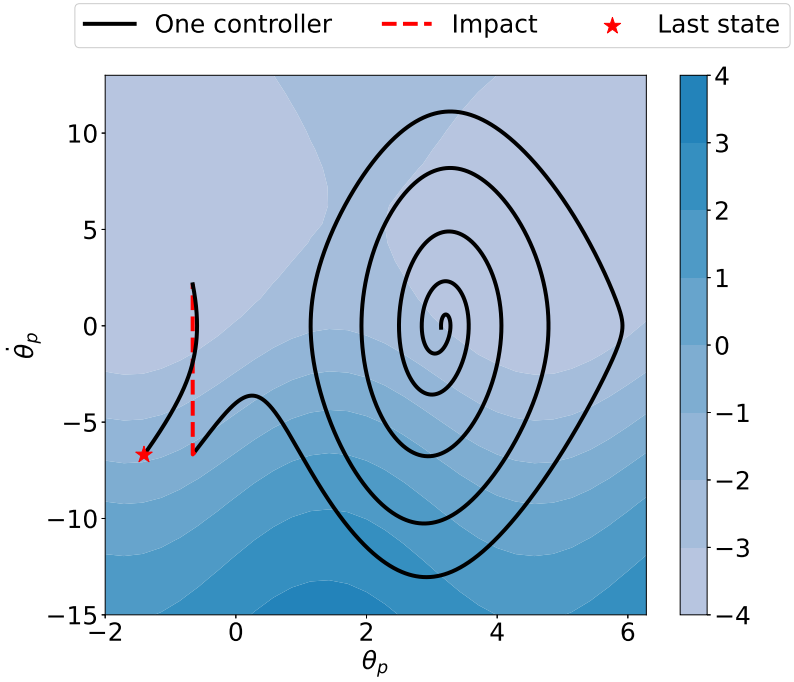

Figure 2.8: A sample trajectory generated by the single controller.

the pendulum to the neighborhood of x^*, we use LQR to stabilize it to the upright. As shown in Figure 2.8, the single controller successfully swings up the pendulum closer to the upright. But the controller is not able to catch the pendulum post-impact. Figure 2.9 shows the performance of the MOE controllers in the same scenario as Figure 2.8. The MOE solution leverages the switching controllers to successfully catch the pendulum post-impact.

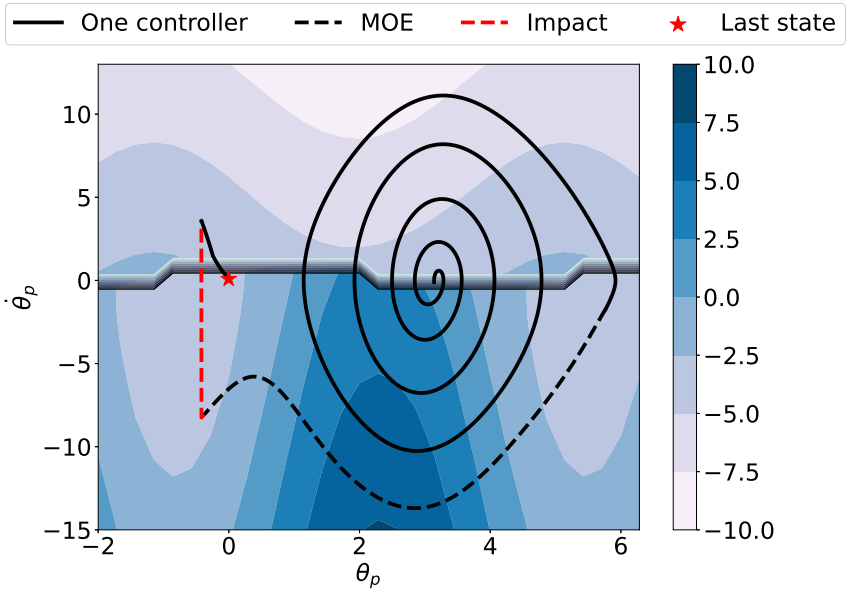

Figure 2.9: The performance of the MOE controllers compared to the single controller in Figure 2.8

.

2.4 Conclusion

CHAPTER 3:

UNCERTAINTY HANDLING VIA NEURAL BAYESIAN INFERENCE

3.1 Background

3.1.1 Passivity-Based Control (PBC)

Suppose we have a robotic system whose Hamiltonian H can be expressed as

$$H(q,p) = \frac{1}{2}p^\top M^{-1}(q)p + V(q), \tag{3.1}$$

where $p \in \mathbb{R}^m$ is the generalized momenta, and $V(q)$ represents the potential energy. Hamilton's equations of motion are given by

$$\begin{aligned} f(x,u) &= \begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ \Omega(q) \end{bmatrix} u, \\ y &= \Omega(q)^\top \dot{q}, \end{aligned} \tag{3.2}$$

where $x = (q, \dot{q})$, $\Omega(q) \in \mathbb{R}^{m \times n}$ is the input matrix, and $u \in \mathbb{R}^n$ is the control input. Passivity-based control leverages the stability properties of passive systems to design a stable

closed-loop system. A mechanical system is considered passive if it is dissipative, i.e.

$$H(x(t_1)) \leq H(x(t_0)) + \int_{t_0}^{t_1} s(u(t), y(t))dt, \tag{3.3}$$

for all initial state $x(t_0)$ and all input u under the supply rate $s = u^\top y$. From Lyapunov stability theory, the system (3.2) is passive and therefore the origin $x = (0, 0)$ is stable because

$$\dot{H} = \frac{\partial H}{\partial x} f(x, u) \leq u^\top y,$$
$$H \geq 0.$$

The objective of passivity-based control (PBC) is to design a control law u that imposes the desired storage function H_d on the closed-loop system, rendering it passive and therefore stable [24]. The dynamics of the resulting closed-loop system is

$$f(x, u) = \begin{bmatrix} \nabla_p H_d \\ -\nabla_q H_d \end{bmatrix} \tag{3.4}$$

and it has a new desired stable equilibrium at x^*.

From (3.2) and (3.4), we can find the energy shaping term that creates a passive closed-loop system as

$$u_{es}(x) = -\Omega^\dagger \left(\nabla_q H_d - \nabla_q H \right). \tag{3.5}$$

where $\Omega^\dagger = \left(\Omega^\top \Omega\right)^{-1} \Omega^\top$. We also introduce a damping term u_{di} that results in an asymptotically stable system. The resulting controller is

$$\begin{aligned} u &= u_{es}(x) + u_{di}(x), \\ u_{di}(x) &= -K_v \, y \end{aligned} \tag{3.6}$$

where $K_v \succ 0$ is the damping gain matrix. From (3.6), we can construct a constraint on the form of H_d as

$$\Omega^{\perp} \left(\nabla_q H_d - \nabla_q H \right) = 0, \tag{3.7}$$

where $\Omega^{\perp}\Omega = 0$ and H_d has a minimum at the desired equilibrium $x^* = (q^\star, p^*)$. However, for high-dimensional systems, the closed-form solution to the partial differential equations (PDEs) in (3.7) may be intractable.

Neural PBC

The deterministic NeuralPbc framework presented in [6] solves the PDEs (3.7) by rewriting the PBC problem as the following optimization scheme:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \int_0^T \ell \left(\phi, u^\theta(\phi) \right) \, \mathrm{d}t, \\ \text{subject to} \quad & f(x,u) = \begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ \Omega(q) \end{bmatrix} u^\theta, \\ & u^\theta = -\Omega^\dagger \nabla_q H_d^\theta - K_v^\theta \Omega^\top \nabla_p H_d^\theta, \end{aligned} \tag{3.8}$$

where $T > 0$ is the time horizon, $\phi(x_0, u^\theta, T)$ is a closed-loop trajectory generated from the initial state x_0 under the current control law u^θ and ℓ is a running cost function that parameterizes the performance of the current control. The NeuralPbc technique adds three important features to the classical PBC framework.

1. The optimization problem finds an approximate solution to the PDEs in (3.7) using stochastic gradient descent.

2. Desired system behavior is explicitly introduced into the optimization via the performance objective ℓ.

3. The framework leverages the universal approximation capabilities of neural networks to parameterize the desired Hamiltonian H_d^θ.

Neural Interconnection and Damping Assignment PBC

IdaPbc, a variant of Pbc, selects a particular structure for H_d

$$H_d(q,p) = \frac{1}{2} p^\top M_d^{-1}(q) p + V_d(q), \tag{3.9}$$

where the minimum of the closed-loop potential energy $V_d(q)$ is at the desired equilibrium coordinate q^*. The resulting passive closed loop system is given by [25]

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & M^{-1}M_d \\ -M_d M^{-1} & J_2(q,p) - \Omega K_v \Omega^\top \end{bmatrix} \begin{bmatrix} \nabla_q H_d \\ \nabla_p H_d \end{bmatrix}, \tag{3.10}$$

where $J_2 = -J_2^\top$ and $M_d \succ 0$ is a positive-definite matrix. We set the closed loop systems in (3.2) and (3.10) equal to each other to find the energy-shaping and the damping control terms as

$$\begin{aligned} u_{es} &= \Omega^\dagger \left(\nabla_q H - M_d M^{-1} \nabla_q H_d + J_2 M_d^{-1} p \right), \\ u_{di} &= -K_v \Omega^\top \nabla_p H_d, \end{aligned} \tag{3.11}$$

We build a constraint from (3.11) to extract V_d and the matrices M_d and J_2. The resulting PDEs are given by

$$\Omega^\perp \left\{ \nabla_q H - M_d M^{-1} \nabla_q H_d + J_2 M_d^{-1} p \right\} = 0. \tag{3.12}$$

Similar to NeuralPbc, the closed-form solution to the PDEs in (3.12) may be intractable.

The deterministic Neural-IdaPbc framework introduced in [7] finds an approximate solution to the PDEs from the following optimization problem.

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \|l_{\text{IDA}}(x)\|^2 = \left\|\Omega^{\perp}\left\{\nabla_q H - M_d M^{-1}\nabla_q H_d + J_2 M_d^{-1} p\right\}\right\|^2, \\ \text{subject to} \quad & M_d^{\theta} = (M_d^{\theta})^{\top} \succ 0, \\ & J_2^{\theta} = -(J_2^{\theta})^{\top}, \\ & q^{\star} = \underset{q}{\text{argmin}}\ V_d^{\theta}, \end{aligned} \tag{3.13}$$

where V_d^{θ} and the entries of the M_d^{θ} and J_2^{θ} matrices are parameterized by neural networks. To enforce the constraints shown in (3.13), we redefine M_d^{θ}, J_2^{θ} and V_d^{θ} as follows. We constrain the positive-definiteness of M_d^{θ} with the Cholesky decomposition

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

where $\delta_M > 0$ is a small constant and I_n is the $n \times n$ identity matrix. The matrix $L_{\theta} \in \mathbb{R}^{n\times n}$ is a lower-triangular matrix whose $n(n+1)/2$ entries are outputs of a neural network. The skew-symmetrix matrix J_2 is constructed as

$$J_2^{\theta}(q,p) = A_{\theta}(q,p) - A_{\theta}^{\top}(q,p)$$

where the entries of $A_{\theta}(q,p)$ are given by neural nets. Lastly, we design a fully-connected neural network for V_d^{θ} such that it has a minimum at $q^* = 0$ as follows. Let V_d^{θ} be a deep

neural net with j layers and all bias terms set to zero. We denote this with

$$V_d^\theta(x) = \Phi\Big(W_j\sigma(W_{j-1}\sigma(\dots W_2\sigma(W_1x)))\Big), \tag{3.14}$$

where W_i holds the weights of layer i and σ is the activation function. The activation function is chosen such that $\sigma(0) = 0$, ensuring that $\Phi(0) = 0$ and $\Phi(x) > 0, x \neq 0$. Several choices for the activation function include Elu, Tanh and Relu.

3.1.2 Bayesian Learning

Suppose we are given a finite dataset from an unknown noise process, for which we are trying to fit a regression model. The goal is not only to predict the average outputs of the source, but also the uncertainty of the noise process. Bayesian learning uses the expressive power of stochastic models to best represent the data source and report the uncertainty in predictions [17].

Let $F(x;\theta)$ denote the stochastic model whose parameters θ are multivariate random variables. These parameters are samples drawn from a posterior probability distribution. The objective is to find the posterior distribution that minimizes the quadratic prediction error $\|F(x;\theta) - Y_x\|^2$, where Y_x is an observation in the training dataset corresponding to the input x. As discussed in Section 2.1.2, we can use *expectation maximization* (EM) to learn the optimal posterior distribution. To do so, we define the likelihood in terms of the quadratic prediction error as

$$P(Y_x|\theta) = \mathcal{N}(Y_x|F(x;\theta), s),$$

Even though the EM approach finds optimal parameters θ that minimize the prediction error, this technique is prone to overfitting. Given finite number of observations, EM finds

low variance posterior distribution *biased to the training data* [17]. The risk of overfitting is especially an issue when we are modelling the noise in the data source, which requires some amount of variance. The solution to this problem involves finding a *bias-variance trade-off*, where the training adjusts the parameters based on the likelihood, but also enforces the posterior to hold some variance to prevent overfitting. Bias-variance trade-off is achieved with the introduction of a prior distribution, which adds a regularization term to the likelihood [17]. In supervised learning, regularization terms restrict the training from learning a complex model, minimizing the risk of overfitting [26]. Similarly, the prior distribution prevents the learned parameters from becoming *overconfident* in their predictions. In this construction, the posterior distribution $P(\theta|Y_x)$ is defined in terms of the likelihood and prior with Bayes' theorem as

$$P(\theta|Y_x) = \frac{P(Y_x|\theta)P(\theta)}{P(Y_x)} = \frac{P(Y_x|\theta)P(\theta)}{\int_\theta P(Y_x|\theta')P(\theta')d\theta'}, \tag{3.15}$$

where $P(\theta)$ is the prior and $P(Y_x)$ is the evidence or the normalization constant of the posterior. The relative weighting between the likelihood and the prior is parameterized by the standard deviation s of the likelihood. The higher the standard deviation, the more weight we give to the regularization enforced by the prior. The rate at which we update the prior distribution also determines the regularization weight. The posterior in (3.15) shows that the likelihood and the prior are in a tug of war. The likelihood pulls the parameters towards minimizing the prediction error, but the prior distribution gives priority to the initial distribution of the decision parameters. This tug of war prevents the posterior from finding a near zero-variance solution, and consequently achieving the bias-variance trade-off.

Posterior Distribution

While the likelihood and prior distributions in (3.15) can be expressed explicitly, the evidence is intractable. We leverage Bayesian inference techniques to approximate or find the exact posterior distribution without the use of the evidence. Two of the most famous techniques are discussed as follows.

1. **Markov Chain Monte Carlo (MCMC) methods**: find the exact posterior distribution from a collection of samples of θ. MCMC methods collect these samples from a proposal distribution $\tilde{Q}(\theta^{(\tau)}|\theta^{(\tau-1)})$, where the sequence of samples θ^{τ} form a Markov Chain [17]. The proposal distribution is known up to its normalization constant and is sufficiently simple to sample from. We accept or reject each candidate sample according to the following rule [17]:

$$\nu \sim \mathcal{U}(0,1),$$
$$A(\theta^{(\tau)}, \theta^{(\tau-1)}) = \min\left(1, \frac{P(Y_x|\theta^{(\tau)})P(\theta^{(\tau)})}{P(Y_x|\theta^{(\tau-1)})P(\theta^{(\tau-1)})}\right),$$

where $\mathcal{U}$ is the uniform distribution. If $A(\theta^{(\tau)}, \theta^{(\tau-1)}) \geq \nu$, then we accept the sample. Otherwise, we discard the candidate and resample from $\tilde{Q}(\theta^{(\tau)}|\theta^{(\tau-1)})$. MCMC methods such as Metropolis-Hastings collect the next sample through random walk [27], while other gradient-based techniques such as Hamiltonian Monte Carlo (HMC) method, efficiently search the parameter space through the gradient of the likelihood. Even though the MCMC methods learn the exact posterior distribution, they have slow convergence properties for high-dimensional parameters. In such cases, techniques such as variational inference compromise accuracy of the posterior distribution for speed of

convergence.

2. **Variational Inference (VI)**: is a gradient-based technique that approximates the posterior with the pre-selected distribution $Q(\theta; z)$. The approximate posterior is selected from the conjugate families of the likelihood and prior distributions. The goal is to learn the distribution parameters z of the approximate posterior that minimize the Kullback-Leibler divergence or equivalently maximize the evidence lower bound (ELBO). The ELBO, $\mathcal{L}$, is given by [28]

$$\mathcal{L}(Y_x, z) = \mathbb{E}_{\theta \sim Q} \left[\log(P(Y_x \mid \theta; z) P(\theta)) - \log(Q(\theta; z)) \right]. \tag{3.16}$$

Remark 1. *For continuous posterior distribution, the* ELBO *given in equation* (3.16) *is redefined using differential entropy, which expresses the prior and posterior in terms of their probability density functions. In this case, the likelihood* $P(\theta \mid Y_x; z)$ *is also a probability density function and the* ELBO *is not bounded by zero.*

Once we find the exact or approximate posterior, the prediction for state x can be found in one of two ways. The first option is to marginalize the model over the posterior as follows [29]

$$\hat{F}(x) = \frac{1}{N_Q} \sum_{\theta \sim Q} F(x; \theta), \tag{3.17}$$

where N_Q is the number of samples drawn from the posterior. The second option only takes one sample from the posterior, and it corresponds to the maximum aposteriori (MAP), i.e.

$$\theta_{MAP} = \underset{\theta}{\operatorname{argmax}}\ P(\theta | Y_x) \tag{3.18}$$

Bayesian frameworks can quantify the confidence in the predictions through the variance [29]

$$\Sigma_{F|x,Y_x} = \frac{1}{N_Q - 1} \sum_{\theta \sim Q} \left\| F(x;\theta) - \frac{1}{N_Q} \sum_{\theta \sim Q} F(x;\theta) \right\|^2 . \tag{3.19}$$

3.2 Theoretical Justification of Robustness

In Section 3.1.2, we discussed how a Bayesian approach to regression achieves bias-variance tradeoff and combats the risk of overfitting. We translate the same idea to the data-driven control design problem. For instance, the NeuralPbc framework discussed in Section 3.1.1 assumes a nominal system model given by $f(x, u)$. The parameters of the desired Hamiltonian are updated from a finite dataset generated from this model. A robust controller would not overfit to a point-esimate with the assumption that the finite trajectory observations represent the real system. System parameter and measurement uncertainties in the real system can generate vastly different trajectories, which the deterministic (point-esimate) controller may find foreign. We propose a Bayesian learning framework where the controllers are given by stochastic functions. In this framework, we achieve the performance objective with wide range of control inputs sampled from the stochastic control, which can be viewed as training an ensemble of controllers. This provides a conservative control policy that is robust to uncertainties.

The variance in the stochastic control is the most beneficial when there is an unknown discrepancy between the nominal model and the real system. In this section, we show the relationship between the model uncertainties and the variance of the stochastic control. We also demonstrate the significant effects of these model discrepancies on the performance objective and the improved robustness properties of Bayesian learning over point-estimates of a policy. This theoretical justification is given by a toy example, where closed-form calculation of the point-estimates and posterior distributions for the optimal controller is

provided.

3.2.1 Optimal Control under Parameter Uncertainty

Let us consider the first-order scalar control system, whose system parameter p_s is uncertain:

$$\begin{cases} \dot{x} = p_s x + u, \ x(0) = x_0 \\ u(x) = \theta x. \end{cases} \tag{3.20}$$

We assume that $p_s \sim \mathcal{N}(\hat{p}_s, \sigma_p)$ where $\hat{p}_s$ designates our best prior point estimate of the system parameter p_s 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 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

$$\mathcal{J} = \int_0^T \left(\frac{1}{2} c x^2 + \frac{1}{2} r u^2 \right) dt, \tag{3.21}$$

where T is the control horizon and $c \geq 0$ and $r > 0$ are design parameters. We solve the control system (3.20) to find $x(t) = e^{(p_s+\theta)t}$ and plug this into the performance index (3.21) along with the form selected for the controller. Performing the integration over time and letting $T \to \infty$, assuming that $p_s + \theta < 0$ then yields the infinite-horizon optimal cost functional

$$\mathcal{J}_\infty = -\frac{1}{4} \frac{c + r\theta^2}{p_s + \theta}. \tag{3.22}$$

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

$$\begin{aligned} \nabla_\theta \mathcal{J}_\infty &= -\frac{r}{4}\frac{(p_s+\theta)^2 - (p^2 + {}^c\!/_r)}{(p_s+\theta)^2} = 0, \\ \therefore \theta^\star &= g(p_s) := -p_s - \sqrt{p_s^2 + {}^c\!/_r}, \\ g^{-1}(\theta) &= \frac{c}{2r\theta} - \frac{\theta}{2}. \end{aligned} \tag{3.23}$$

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

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

where f_p is the Gaussian probability density function with mean $\hat{p}_s$ 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 (3.23), yielding

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

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

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

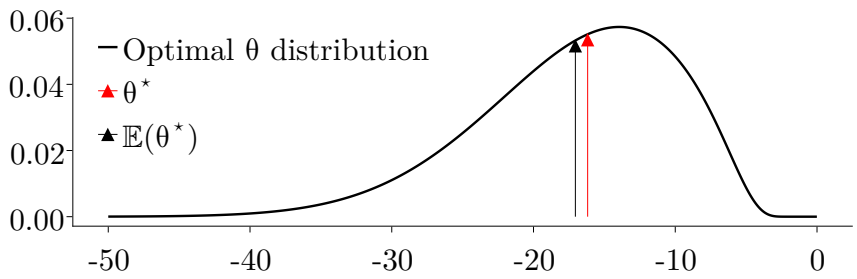

Figure 3.1: The optimal control parameter distribution given that the system parameter p_s is normally distributed with mean $\hat{p}_s = 5$ and $\sigma_p = 5$. The red and black arrows respectively indicate the optimal control parameter without considering the randomness of p_s, and the expected value of the optimal control parameter distribution.

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

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 $(c, r) = (100, 1)$, our best guess $\hat{p}_s = 5$ of the system parameter p_s 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_s 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_s.

Finally, Figure 3.1 shows the optimal control parameter distribution given that the system parameter p_s is normally distributed with mean $\hat{p}_s = 5$, standard deviation $\sigma_p = 5$. This figure also 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_s, a feat that the deterministic training may not reason about.

3.2.2 Optimal Control under Parameter Uncertainty and Measurement Noise

Consider the scenario in which the system (3.20) is also subject to measurement errors; that is, our measurement model for the state x is probabilistic and is distributed according to the Gaussian $\mathcal{N}(x, \sigma^2)$. Since the controller uses this measurement to determine its action, the closed-loop system has to be modelled as a stochastic differential equation (SDE), given by

$$\begin{cases} \mathrm{d}x(t) = (p_s + \theta)x(t)\ \mathrm{d}t + \theta\sigma\ \mathrm{d}W_t, \\ x(0) = 1, \end{cases} \tag{3.24}$$

where W denotes the Wiener process [30]. The initial state is assumed deterministic and is set to unity for simplicity. The unique solution to this SDE is given by

$$x(t) = e^{(p_s+\theta)t} + \theta\sigma \int_0^t e^{(p_s+\theta)(t-s)} dW_s. \tag{3.25}$$

Lemma 1. *The conditional expectation* $\mathbb{E}[\mathcal{J} \mid p_s]$ *of the performance index* (3.21) *given the*

system parameter p_s is

$$\mathbb{E}\left[\mathcal{J} \mid p_s\right] = -\frac{1}{4}\frac{c + r\theta^2}{p_s + \theta}\left[\theta^2\sigma^2 T + \left(1 - e^{2T(p_s+\theta)}\right)\left(1 + \frac{1}{2}\frac{\theta^2\sigma^2}{p_s + \theta}\right)\right].$$

Proof. The proof may be found in the appendix. □

It is easily shown that this quantity is positive for all $T > 0$. Furthermore, it blows up as the horizon T is extended to infinity. This is not surprising since a nonzero measurement noise causes the state to oscillate around the origin, rather than asymptotically converging to it, incurring nonzero cost all the while.

We have kept the system parameter p_s constant in this analysis so far. Uncertainty over this variable can be incorporated by taking a further expectation

$$\mathbb{E}[\mathcal{J}] := \mathbb{E}_{p_s}\left[\mathbb{E}_W\left[\mathcal{J} \mid p_s\right]\right],$$

of $\mathbb{E}_W\left[\mathcal{J} \mid p_s\right]$ over p_s, which must be accomplished numerically as it does not admit a closed-form expression.

We can then minimize $\mathbb{E}[\mathcal{J}]$ over the control parameter in order to study the effects of both kinds of uncertainties on the optimal controller. Such a study is provided in Figures 3.2 and 3.3, where we have plotted the optimal control parameter $\theta^\star$ and the minimal expected cost $\mathbb{E}[\mathcal{J}]$ as a function of the standard deviations of the measurement noise σ and the system parameter σ_p. The constants we used to generate the data are given by $c = r = 1$ and $T = \hat{p}_s = 3$. Our first observation is that the magnitude of the optimal control parameter is an increasing function of system parameter uncertainty and a decreasing function of measurement uncertainty. Our second observation is that if the measurement noise is small, then the optimal control parameter is insensitive to system parameter uncertainty as long as

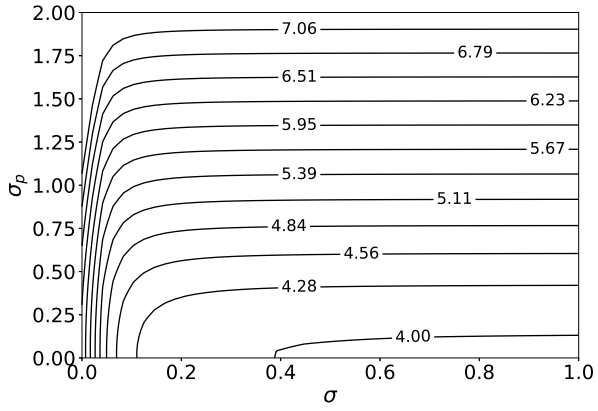

Figure 3.2: The optimal controller parameter magnitude $|\theta^\star|$.

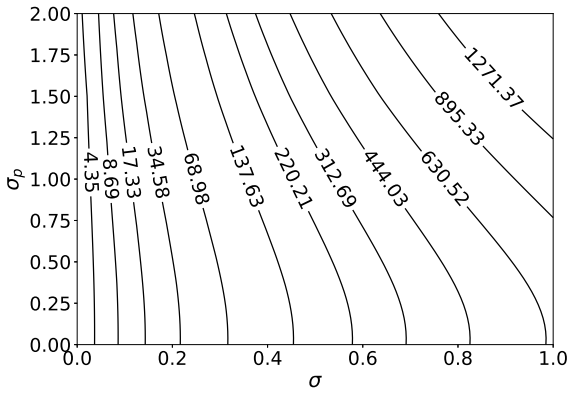

Figure 3.3: The minimal expected cost $\mathbb{E}[\mathcal{J}]$.

this uncertainty is small. The optimal cost shares this insensitivity for an even wider range of values of σ_p. In a similar vein, if the uncertainty in the system parameter is large, then the optimal control parameter is insensitive to the magnitude of the measurement noise. However, the optimal cost is still sensitive to this quantity.

3.3 Bayesian Neural PBC

In this section, we present a unified framework that simultaneously combines the NeuralPbc technique and rigorously addresses model uncertainties using Bayesian learning. Motivated by [31], 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. 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. We demonstrate the efficacy and robustness of our current framework with a comparison against the deterministic framework [31]. The comparison is performed on benchmark underactuated control problems–the simple pendulum and the inertia wheel pendulum–both in simulation and real-world experiments.

3.3.1 Control Design for Smooth Dynamical Systems

In this section, we formulate the Bayesian learning framework that minimizes the effects of system parameter uncertainties and measurement errors for smooth dynamical systems. In this framework, the neural net parameterization of H_d^θ given in Section 3.1.1 is replaced by a Bayesian neural network whose weights and biases are samples drawn from a posterior distribution. The goal is to learn this posterior that achieves the performance objective under system parameter and measurement uncertainties.

Let $\phi(x_0, u^\theta, T)$ denote a closed loop trajectory integrated from the Hamilton's equation of motion in (3.2). The trajectory starts from the initial state x_0 and spans for the time horizon T. We sample the initial state x_0 through greedy and explorative state sampling techniques discussed in Section 2.2. The control law u^θ consists of the energy shaping and damping terms found from the desired Hamiltonian H_d^θ as shown in equation (3.6). To generate this trajectory, we first sample the parameters θ from the prior distribution $P(\theta)$.

We take two approaches to selecting the prior distribution. The simplest approach is to use an uninformed prior given by a uniform distribution; 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, the prior distribution is a Gaussian probability distribution centered around the parameters learned from the deterministic NeuralPbc technique discussed in Section 3.1.1.

With the trajectories generated from the prior distribution samples, we compose a

performance objective as follows.

$$J(\phi, u^\theta) = \mathbb{E}_{x_0 \sim \mathcal{D}_N}[\ell(\phi(x_0, u^\theta, T), u^\theta)],$$

where $\mathcal{D}_N$ is a collection of initial states sampled with explorative and greedy state sampling techniques (Section 2.2), and ℓ is the running cost of the trajectories. We provide three options for the running cost in the next section. In order to update the posterior distribution based on the performance of the generated trajectories, we compose the likelihood function as

$$P(J \mid \theta) = \mathcal{N}(0, s), \tag{3.26}$$

where s is the standard deviation of the likelihood. Notice that the likelihood is maximum when the expectation of the running cost J is minimum.

Remark 2. *On top of the neural net parameters θ, we can learn the posterior over the standard deviation s of the likelihood. This automatically finds the weighting between the likelihood and prior distributions, achieving optimal bias-variance trade-off. In this scenario, the posterior distribution is a multivariate probability distribution over the parameters θ and s.*

From here, we can use Hamiltonian Monte Carlo or variational inference (Section 3.1.2) to find the posterior distribution from the likelihood and the prior. We first pose the search

over the posterior as the following optimization problem.

$$\begin{aligned}
\underset{P(\theta|J)}{\text{minimize}} \quad & J(\phi, u^\theta) = \mathbb{E}_{x_0 \sim \mathcal{D}_N}[\ell(\phi(x_0, u^\theta, T), u^\theta)], \\
\text{subject to} \quad & \begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ \Omega(q) \end{bmatrix} u^\theta, \\
& u^\theta = -\Omega^\dagger(\nabla_q H_d^\theta - \nabla_q H), \\
& p_s \sim \mathcal{N}(\hat{p}_s, \sigma_p), \\
& \theta \sim P(\theta|J).
\end{aligned} \tag{3.27}$$

We expand on the use of HMC and variational inference to solve the optimization problem.

1. **Hamiltonian Monte Carlo (HMC)**: represents the posterior with a large collection of samples denoted by Θ. We draw the first sample $\theta^{(\tau=0)}$ from the prior distribution and construct the likelihood from the performance objective as $P(J|\theta^{(\tau=0)})$. The likelihood and the prior distributions are known in closed form, hence the joint distribution $P(\theta^{(\tau=0)}, J)$ is given by

$$P(\theta^{(\tau=0)}, J) = P(J|\theta^{(\tau=0)})P(\theta^{(\tau=0)}).$$

   We generate the next sample from a Markov Chain, which is given the following first order differential equations [17]

$$\begin{aligned}
m^{\tau+\Delta\tau/2} &= m^\tau - \frac{\Delta\tau}{2}\frac{\partial P(\theta^\tau, J)}{\partial \theta^\tau} \\
\theta^{\tau+\Delta\tau} &= \theta^\tau + m^{\tau+\Delta\tau/2}\,\Delta t \\
m^{\tau+\Delta\tau} &= m(\tau + {}^{\Delta\tau}/_2) - \frac{\Delta t}{2}\frac{\partial P(\theta^{\tau+\Delta\tau}, J)}{\partial \theta^{\tau+\Delta\tau}}
\end{aligned} \tag{3.28}$$

where $\tau = 0$ for the first sample and $\Delta\tau$ is the integration time step. This integration is commonly known as *leap frog discretization*. We accept or reject the sample $\theta^{\tau+\Delta\tau}$ based on the rule:

$$\nu \sim \mathcal{U}(0,1),$$
$$A(\theta^{(\tau+\Delta\tau)}, \theta^{(\tau)}) = \min\left(1, \frac{P(\theta^{(\tau+\Delta\tau)}, J)}{P(\theta^{(\tau)}, J)}\right),$$

If $A(\theta^{(\tau+\Delta\tau)}, \theta^{(\tau)}) \geq \nu$, then we accept the sample. Otherwise, we discard the candidate and resample from the prior distribution. The complete procedure is outlined in Algorithm (6) We keep collecting these samples until the average accumulated loss converges to the minimum achievable value.

Once we obtain the collection samples, we use them to evaluate the controller as follows. From the collection Θ, we sample N_Q parameters in a uniform fashion. We evaluate the neural net H_d^θ and the corresponding energy shaping and damping control for each sample. Then, we marginalize over the control law as

$$u(x) = \frac{1}{N_Q} \sum_{\theta \sim P(\theta|J)} u(x;\theta).$$

We repeat this computation for every state in a trajectory.

Algorithm 6 Bayesian NeuralPbc via Hamiltonian Monte Carlo

$\Theta \leftarrow \{\}$ ▷ Collection of samples
Define prior $P(\theta)$
while $i <$ `maximum iteration` **do**
  $\tau = 0, \theta^0 \sim P(\theta)$, $m^0 = 0$, accepted=true
  while accepted **do**
    $\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ $N_\mathcal{D}$ initial state samples
    $p_s \sim \mathcal{N}(\hat{p}_s, \sigma_p)$ ▷ Sample a system parameter
    $J^\tau \leftarrow 0$, $J^{\tau+\Delta\tau} \leftarrow 0$
    for $(x_0) \sim \mathcal{D}_N$ **do**
      $\phi \leftarrow \phi(x_0, u^{\theta^\tau}, T)$ ▷ Generate trajectory
      $J^\tau \leftarrow J^\tau + \ell(\phi; \theta^\tau)/N_\mathcal{D}$ ▷ Batch loss of current parameter
    $m^{\tau+\Delta\tau}, \theta^{\tau+\Delta\tau} \leftarrow$ `leap frog discretization`(m^τ, θ^τ) ▷ Equation (3.28)
    for $(x_0) \sim \mathcal{D}_N$ **do**
      $\phi \leftarrow \phi(x_0, u^{\theta^{\tau+\Delta\tau}}, T)$ ▷ Generate trajectory
      $J^{\tau+\Delta\tau} \leftarrow J^{\tau+\Delta\tau} + \ell(\phi; \theta^{\tau+\Delta\tau})/N_\mathcal{D}$ ▷ Batch loss of next parameter
    $\nu \sim \mathcal{U}(0, 1)$
    $A(\theta^{(\tau+\Delta\tau)}, \theta^{(\tau)}) = \min\left(1, \frac{P(\theta^{(\tau+\Delta\tau)}, J^{(\tau+\Delta\tau)})}{P(\theta^{(\tau)}, J^{(\tau)})}\right)$
    if $A(\theta^{(\tau+\Delta\tau)}, \theta^{(\tau)}) \geq \nu$ **then**
      $\Theta \leftarrow \Theta \cup \theta^{(\tau+\Delta\tau)}$ ▷ Accept sampled parameter
      $\tau = \tau + \Delta\tau$
    else
      $i \leftarrow i + 1$, accepted=false ▷ Reject sampled parameter
Return Θ

2. **Variation Inference (VI)**: intends to learn the distribution parameters z of the

pre-selected (approximate) posterior $Q(\theta; z)$. Unlike HMC, we have a closed form probability distribution for the posterior. We draw several samples from the current posterior and evaluate its performance through the joint distribution $P(\theta, J)$. Given the likelihood and the prior distributions, variational inference constructs the ELBO as

$$\mathcal{L}(J, z) = \mathbb{E}_{\theta \sim Q} \left[\log(P(J|\theta)P(\theta)) - \log(Q(\theta; z)) \right]. \tag{3.29}$$

We use stochastic gradient descent to iteratively update the distribution parameters as follows:

$$z \leftarrow z + \frac{\partial \mathcal{L}(J, z)}{\partial z}$$

The full variational inference training process is shown in Algorithm (7).

The gradient $\partial \mathcal{L}/\partial z$ holds a rather complex form for two reasons. First, the ELBO is evaluated from trajectories integrated from the differential equations in (3.2). We use a combination of adjoint method and auto-differentiation techniques [32] to compute the gradient $\partial \mathcal{L}/\partial z$ through the trajectories. Secondly, computing $\partial \mathcal{L}/\partial z$ requires the derivative of the sample θ with respect to the distribution parameters z, which is intractable. We handle this complication by invoking the reparameterization trick of the Automatic Differentiation Variational Inference(ADVI) [33].

Performance Objective

The cost function J helps impose various desired behaviors in the learned controller. In this section, we present three performance objectives, their corresponding likelihood functions and the desired behavior they impose.

Algorithm 7 Bayesian NeuralPbc via variational inference

$\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ $N_\mathcal{D}$ initial state samples
while $i <$ `maximum iteration` **do**
  $\mathcal{L} \leftarrow 0$ ▷ Elbo Loss
  for $i = 1 : N_Q$ **do** ▷ Samples to compute (3.29)
    $J \leftarrow 0$ ▷ Batch loss
    $\theta \sim Q(\theta; z)$ ▷ Sample parameters of H_d^θ
    for $(x_0) \sim \mathcal{D}_N$ **do**
      $p_s \sim \mathcal{N}(\hat{p}_s, \sigma_p)$ ▷ Sample a system parameter
      $\phi \leftarrow \phi(x_0, u^\theta, T)$ ▷ Generate trajectory
      $J \leftarrow J + \ell(\phi; \theta)/N_\mathcal{D}$
    $\mathcal{L} \leftarrow \mathcal{L} + \frac{1}{N_Q}\left(\log[P(J|\theta)P(\theta)] - \log[Q(\theta; z)]\right)$
  $z \leftarrow z + \alpha_i \partial\mathcal{L}/\partial z$ ▷ SGD step
  $\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ New initial state samples
  $i \leftarrow i + 1$
return z

1. **Trajectory tracking**: Let $x^\star$ denote the desired equilibrium of a dynamical system and $\phi(x_0, u^\theta, T)$ represent a prediction from the initial condition x_0 with the current control law u^θ. The objective of this task is to find a closed-loop controller that can track an expert trajectory $\phi^\star$ obtained from a path planner. To impose this behavior, the running cost is a function of the distance from the current trajectory ϕ to $\phi^\star$, defined in terms of the transverse coordinates $\phi_\perp - \phi^\star_\perp$ along the preferred orbit (shown in Figure 3.4), using the ideas outlined in [34, 35]. In this setting, the prediction converges to $\phi^\star$ if and only if $\phi_\perp - \phi^\star_\perp \xrightarrow{t\to\infty} 0$. In order to encourage this behavior, we define the cost function as

$$J_{track}(\phi_\perp) = \sum_{x_\perp \in \phi_\perp,\, x^\star_\perp \in \phi^\star_\perp} ||x_\perp - x^\star_\perp||^2 \tag{3.30}$$

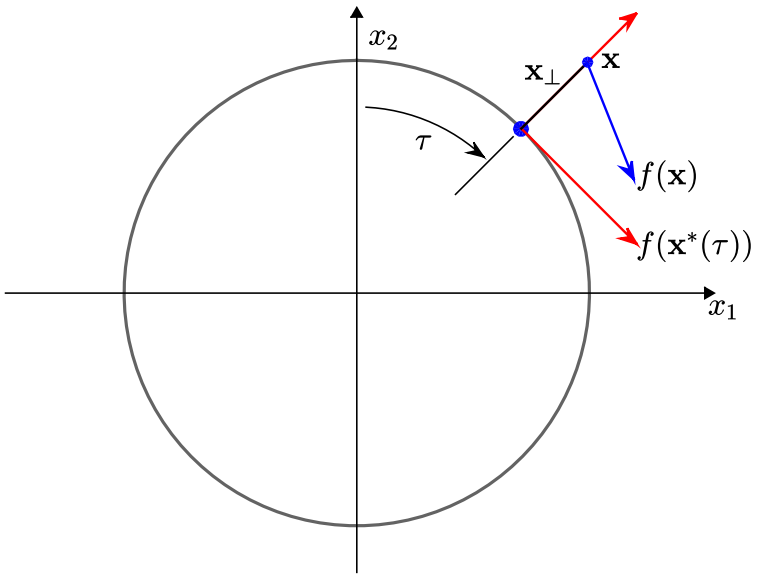

Figure 3.4: Transverse Coordinates

Thus, the likelihood can be constructed to minimize the cost J_{track} as follows.

$$P(J_{track}|\theta) = \prod_{x_\perp \in \phi_\perp,\, x_\perp^\star \in \phi_\perp^\star} \mathcal{N}(x_\perp^\star, s), \tag{3.31}$$

2. **Set distance loss**: Let $\mathcal{S}$ represent a small convex neighborhood containing $x^\star$. The objective is to find a policy that pulls trajectories to the goal set $\mathcal{S}$. The cost function suitable for this task is set distance, J_{set}, between the current prediction ϕ and the goal set $\mathcal{S}$:

$$J_{\text{set}}(\phi) = \inf_t \{\|a - b\| \colon a \in \phi(t),\, b \in \mathcal{S}\}. \tag{3.32}$$

For instance, the set $\mathcal{S}$ may be chosen as a ball 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 ϕ, a state x is closer than

r to $x^\star$, no penalty is incurred by J_{set}. This construction has the same advantages as the Minimum Trajectory Loss (MTL) discussed in Section 2.2. The corresponding likelihood function is given by

$$P(J_{\text{set}}|\theta) = \mathcal{N}(0, s). \tag{3.33}$$

3. **Terminal loss**: encourages trajectories to remain as close to the desired state as possible at time T. Terminal loss, ℓ_T, is the distance between the final state of the prediction ϕ and $x^\star$, which is given by

$$J_T(\phi) = \ell_{\text{set}}(\phi) + \ell_T(\phi). \tag{3.34}$$

The corresponding likelihood function is given by

$$P(J_T|\theta) = \mathcal{N}(0, s). \tag{3.35}$$

Uncertainty Modeling

In Section 3.2, we have shown the effects of system model uncertainties on the performance of a controller. Moreover, in Section 3.1.2, we have discussed how the variance in the stochastic control can prevent the Bayesian training from overfitting on inaccurate observations. In this section, we give more structure to the desired variance of the Bayesian control. We rigorously model the system parameter and measurement uncertainties of the system in order to guide the variance of the controller. 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 pa-

rameter uncertainties by sampling a set of system parameters p_s from a normal distribution $\mathcal{N}(\hat{p}_s, \sigma_p)$ centered around a nominal parameter $\hat{p}_s$. Additionally, we model measurement error by injecting noise into the prediction ϕ. This is achieved by replacing the ordinary differential equation given in (3.2) with the following stochastic differential equation (SDE).

$$\mathrm{d}x = \left(\begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ \Omega(q) \end{bmatrix} u^\theta(x) \right) \mathrm{d}t + \nabla_x u(x)\, \mathrm{d}W_t, \tag{3.36}$$

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. The resulting Bayesian NeuralPbc problem is given by

$$\begin{aligned} \underset{P(\theta|J)}{\text{minimize}} \quad & J(\phi, u) = \mathbb{E}_{x_0 \sim \mathcal{D}_N}[\ell(\phi(x_0, u^\theta, T), u^\theta)], \\ \text{subject to} \quad & \mathrm{d}x = \left(\begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ \Omega(q) \end{bmatrix} u^\theta(x) \right) \mathrm{d}t + \nabla_x u(x)\, \mathrm{d}W_t, \\ & u^\theta = -\Omega^\dagger(\nabla_q H_d^\theta - \nabla_q H), \\ & p_s \sim \mathcal{N}(\hat{p}_s, \sigma_p), \\ & \theta \sim P(\theta|J). \end{aligned} \tag{3.37}$$

Remark 3. *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 [17].

3.3.2 Simple Pendulum

In this section, we demonstrate the robustness properties of the controllers trained via Bayesian methods on the simple pendulum system. The system is simulated under measurement noise via stochastic differential equations (3.36). Furthermore, the system parameter is varied in order to analyze the closed-loop system response under model uncertainties.

The equation of motion of a simple pendulum with measurement noise is given by

$$\mathrm{d}x = \begin{bmatrix} \dot{q} \\ a \sin(q) + u^{\theta}(x) \end{bmatrix} \mathrm{d}t + \nabla_x u^{\theta}(x)\, \mathrm{d}W_t, \tag{3.38}$$

where $a = mgl/I$, $\mathrm{d}W_t$ is the Wiener process, $x = (q, \dot{q})$ is the state of the pendulum where $q = 0$ corresponds to the upright position, and the control input u is the torque generated by the actuator. The torque u^{θ} is limited by $|u| \leq u_{\max}$. The maximum torque $u_{\max}$ available is such that the upward equilibrium point cannot be reached by just rotating in one direction, as the gravitational force overcomes the motor torque eventually. The controller has to be clever enough to overcome gravitational forcing with a combination of built-up momentum and torque bandwidth.

The objective of this case study is to stabilize the homoclinic orbit of the pendulum whose single parameter a has the nominal and yet unconfirmed value 9.81 s^{-2}. To this end, we learn a Bayesian control u^{θ} that can stabilize the pendulum even with system parameter uncertainties.

Training

The goal of this training is to track the expert trajectories generated by the vanilla energy-shaping control (ESC) [36]. The ESC takes the general form

$$u^{\theta}(q, \dot{q}; \theta^{e}) = \theta_1^{e}\dot{q} + \theta_2^{e}\cos(q)\dot{q} + \theta_3^{e}\dot{q}^3, \tag{3.39}$$

where θ^e represents the parameters of the expert. The weights $\{\theta_i^e\}_{i=1}^3$ satisfy $-\theta_1^e = \theta_2^e = 2a\theta_3^e < 0$ in the vanilla ESC. The Bayesian control is also linear over the decision parameters, i.e.

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

and unlike ESC, the learned parameters θ are samples drawn from the posterior.

The running cost function is chosen to be the loss $J_{track}(\phi_\perp)$ from homoclinic orbit $\phi^\star$ provided in (3.30); the corresponding likelihood is given by (3.31). We collect dataset $\mathcal{D}_N$ of initial states sampled with greedy and explorative techniques. For this particular experiment, we use Hamiltonian Monte Carlo outlined in Algorithm 6 to infer the exact posterior distribution. We compare the Bayesian policy with the deterministic policy, which is simply given by the point-estimates of the expert vanilla ESC in (3.39) [37].

Simulation Tests

The homoclinic orbit of the pendulum is defined by the $2a$-level set of the total energy $\mathcal{H} = {}^1\!/_2\dot{q}^2 + a(1+\cos q)$. Hence, an appropriate measure of the distance to the homoclinic orbit is given by the absolute value $|\tilde{\mathcal{H}}|$ of the error: $\tilde{\mathcal{H}} = \mathcal{H} - 2a$. We evaluate the performance of

a closed-loop system by recording the value $\zeta = \min |\tilde{\mathcal{H}}|$, where the minimum is taken over the last 2 seconds of a 10-second-long trajectory.

We demonstrate the robustness of the Bayesian policy by comparing ζ, as a function of the system parameter a, with that of the deterministic policy. We also investigate the effects of the prior distribution of θ on the performance of the controllers. In particular, we examine a uniform prior and a Gaussian prior centered around the deterministic solution of (3.39). The comparisons between the controllers from both cases are shown in Figure 3.5.

In addition to uncertainties in a, we test the deterministic and Bayesian policies with measurement noise, modeled as a Wiener process with standard deviation 0.0005 rad in the q-direction and 0.05 $^{\text{rad}}/_{\text{s}}$ in the $\dot{q}$-direction. These numbers are chosen to represent a typical error arising from an optical encoder with a resolution of 2048 pulses per revolution and its naive differentiation via backward difference. To capture the influence of measurement noise, we generate 20 trajectories by integrating (3.38) from the same initial states. These trajectories are then used to compute ζ. The effects of noise on ζ are reflected by the error bands in Figure 3.5.

The results, shown in Figure 3.5, demonstrate that Bayesian learning yields controllers that outperform their deterministic counterpart throughout the whole range of a. The marginalization method in (3.17) for selecting θ from the learned distribution performs best when a uniform prior is used, while the MAP method performs best with the Gaussian prior. We emphasize that the error bands of the marginalized and MAP point estimates stay well below those of the deterministic curve in the top plot of Figure 3.5. This implies that the controllers produced by Bayesian learning perform consistently better than the deterministic controller, demonstrating their robustness against model uncertainty and measurement noise.

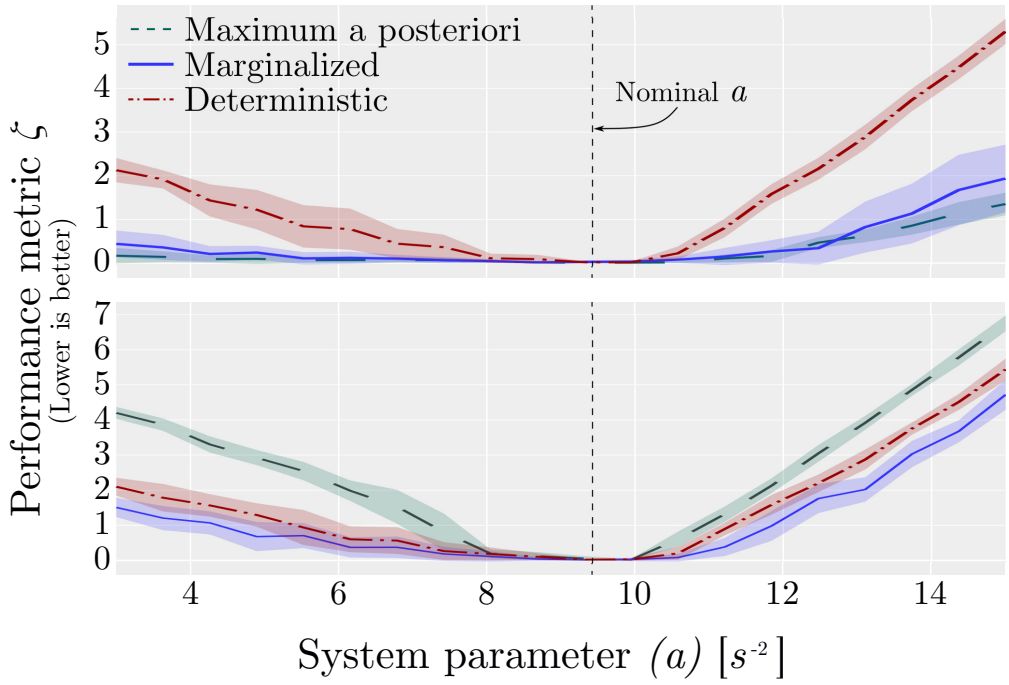

Figure 3.5: Performance comparisons between deterministic and Bayesian learning methods. The training is initialized with a Gaussian prior (top), and a uniform prior (bottom). The continuous error band is generated by computing ζ from 20 trajectories of (3.38), starting at the downward equilibrium with a small disturbance. The solid lines represent the mean of ζ. Best viewed in color.

3.3.3 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. 3.6. 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.

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 (3.1), where

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

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 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 with measurement noise can be written as

$$\mathrm{d}x = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dfrac{mgl\sin(q_1) - u^\theta - b_1\dot{q}_1}{I_1} \\ \dfrac{u^\theta - b_2\dot{q}_2}{I_2} \end{bmatrix} \mathrm{d}t + \nabla_x u^\theta(x)\,\mathrm{d}W_t, \tag{3.40}$$

where the control input u^θ is the torque applied to the inertia wheel and $\{b_i\}_{i=1}^2$ are friction coefficients. The desired equilibrium $x^\star$ 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.

Training

The energy-like function H_d^θ is a fully-connected neural network with two hidden layers, each with the Elu activation function [23]. A uniform distribution in $[-2\pi, 2\pi] \times [-2\pi, 2\pi] \times$

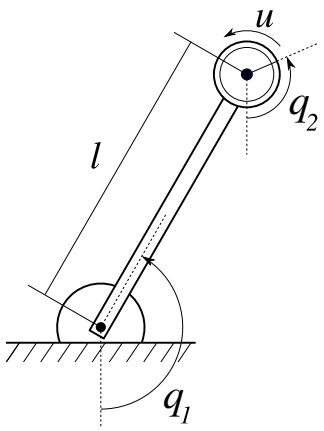

Figure 3.6: Schematic of the inertia wheel pendulum. Only the joint q_2 is actuated, and q_1 is not.

$[-10, 10] \times [-10, 10]$ 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, we sample a batch of 4 initial states $\{x_0\}$ from greedy and explorative state sampling techniques; these initial states are integrated forward with a time horizon of $t \in [0, 3]$ seconds. In the Bayesian framework, the standard deviations σ_p of system parameters $p_s = [I_1, I_2, mgl]$ are chosen to be 10% of the nominal system parameters. Moreover, we train on trajectories per the SDE in (3.36) with measurement error represented by Wiener process with standard deviation of 0.001 and 0.02 on the joint angles and velocities, respectively.

We use variational inference to estimate a Gaussian posterior distribution over uncorrelated parameters. The trainings are terminated when the loss function $J(\gamma) = J_{set}(\gamma) + J_T(\gamma)$ and the ELBO converge for the deterministic and Bayesian trainings, respectively. The hyperparameters for the deterministic and Bayesian NEURALPBC trainings are shown in Table 3.1. It can be seen that the Bayesian training effectively learns with smaller neural network size than the deterministic training.

Table 3.1: 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

Simulation Tests

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. Figure 3.7 shows the performance of deterministic and Bayesian trainings using an accumulated quadratic loss of the form

$$J^T = \frac{1}{2} \int_0^T \left(x^\top Q x + u^\top R u \right) dt. \tag{3.41}$$

The controller learned from the Bayesian training is marginalized over 10 parameters sampled from the posterior. As seen in Figure 3.7, the Bayesian training effectively collects 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; this shows that the marginalized

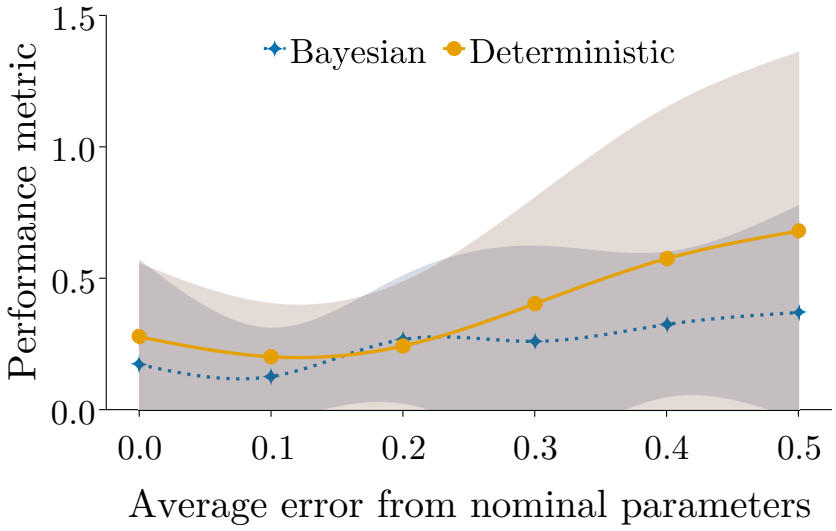

Figure 3.7: NeuralPBC Performance metric (J^T) for various error in system parameters. Measurement noise included as Wiener process with standard deviation of 0.001 **and** 0.02 **on joint angles and velocities, respectively**

controller is more robust against measurement noise.

Hardware Tests

The controllers from deterministic and Bayesian training schemes are evaluated on the hardware shown in Figure 3.8. We deliberately modify 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 3.2) whose mass and inertia are different from the nominal values. The modified system parameters are summarized in Table 3.2.

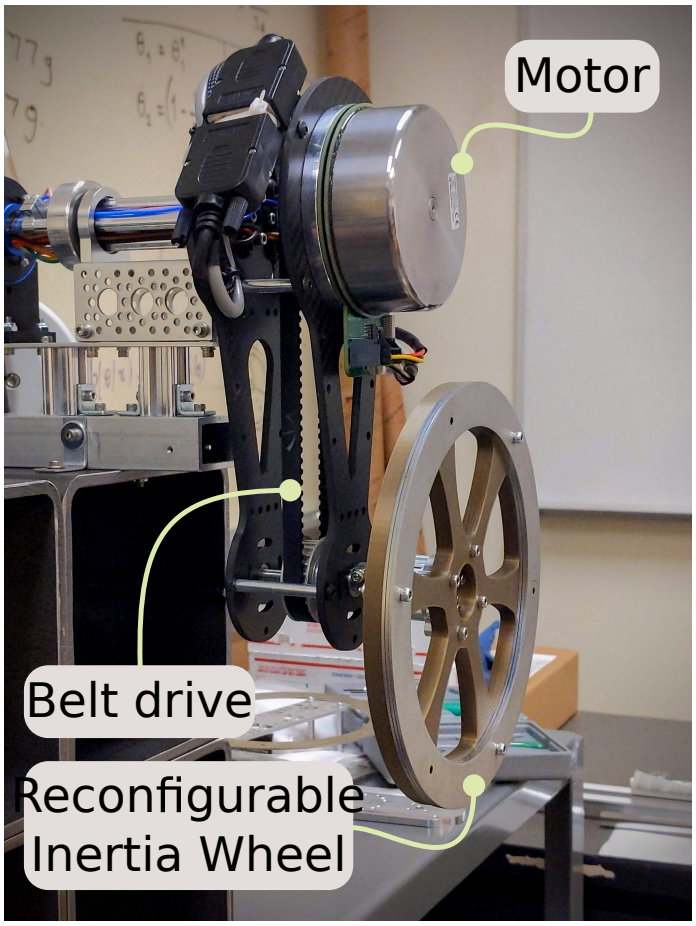

Figure 3.8: Inertia Wheel Pendulum Hardware

Table 3.2: 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

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 (3.41) is the performance metric used to evaluate the controllers. The results are summarized in Figure 3.9.

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

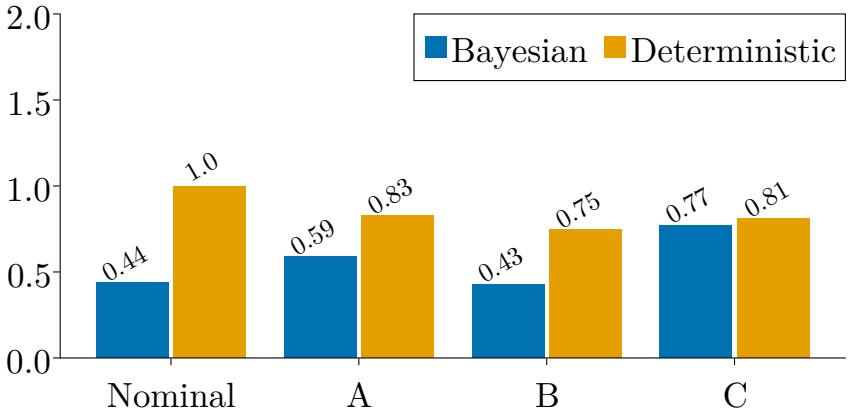

Figure 3.9: Controller performance for modified system parameters. The performance metric is given by Eq. (3.41). Lower values are better. These results show that controllers trained via Bayesian learning are consistently more robust to errors in system parameters.

3.3.4 Control Design for Hybrid Dynamical Systems

In Section 3.1.1, we constructed the properties of passivity on the Hamiltonian of a smooth dynamical system. For hybrid systems, we can deconstruct the concept of passivity into flow-passive and jump-passive [38]. Flow-passive implies that the continuous phase of

the dynamics with Hamiltonian H is dissipative, i.e.,

$$H(x(t_1)) \leq H(x(t_0)) + \int_{t_0}^{t_1} s(u(t), y(t))dt, \tag{3.42}$$

under the supply rate $s = u^\top y$. In order for a hybrid system to be passive, the conditions in (3.42) must hold during discrete state transitions as well. In particular, the class of mechanical systems exhibiting contacts, impacts and Coulomb friction obey the dissipative property under the same Hamiltonian as [39]

$$H(x^+) \leq H(x^-),$$

where x^- and x^+ are connected with the jump rule. Hence, the discrete transitions undergoing elastic or inelastic collisions are jump-passive. For instance, in the case of well-defined contact events, the post-impact Hamiltonian is given by

$$H(x^+) = \epsilon^2 H(x^-),$$

where ϵ is the coefficient of restitution, and it takes values between 0 and 1. This demonstrates that it is possible to find a single Hamiltonian H_d that renders the closed-loop hybrid system flow- and jump-passive. Thus, we translate the passivity-based control framework in Section 3.3.1 to hybrid systems. We aim to find the desired Hamiltonian H_d, whose stable equilibrium is at x^*.

In this section, we extend the techniques of deterministic and Bayesian NeuralPbc to hybrid dynamical systems. We introduce a contact model of the hybrid dynamics into the deterministic NeuralPbc framework and infer a controller that leverages the advantages

of contact events and/or minimizes its adverse effects. Moreover, we inject uncertainties in contact forces and post-impact velocities to NeuralPbc in order to design a robust controller through Bayesian learning.

Deterministic NeuralPBC

In this framework, we aim to find a passive closed loop system for the hybrid dynamics (2.1) whose desired stable equilibrium is at $x^* = (q^*, \dot{q}^*)$. We formulate this problem as a search over the point-estimate parameters of H_d given by a neural network. This training framework can be posed as the following optimization problem.

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \int_0^T \ell\left(\phi, u^\theta(\phi)\right)\, \mathrm{d}t, \\ \text{subject to} \quad & M(q)\,\mathrm{d}\dot{q} + h(q,\dot{q},\theta)\,\mathrm{d}t - \mathrm{d}R = 0, \\ & u^\theta = -\Omega^\dagger(\nabla_q H_d^\theta - \nabla_q H), \end{aligned} \tag{3.43}$$

The performance objective ℓ is evaluated from closed loop trajectories ϕ using the current control law. We follow Moreau's time stepping algorithm [14] outlined in Algorithm (1) to resolve the complementarity constraint in (2.2) and integrate the closed-loop dynamics. We sample $N_{\mathcal{D}}$ initial states through greedy and explorative state sampling techniques discussed in Section 2.2. As outlined in Algorithm 8, we compose the cost ℓ from the $N_{\mathcal{D}}$ closed loop trajectories. The performance objective is chosen according to the desired system behavior as discussed in Section 3.3.1. We update the parameters θ through stochastic gradient descent (SGD). We compute the gradient $\partial\ell/\partial\theta$ through auto-differentiation techniques.

Algorithm 8 Solution to the Optimization Problem (3.43)

$\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ $N_\mathcal{D}$ initial state samples
while $i <$ `maximum iteration` **do**
  $J \leftarrow 0$ ▷ Batch loss
  for $(x_0) \sim \mathcal{D}_N$ **do**
    $\phi = \text{Moreau}(x_0)$ ▷ Algorithm (1)
    $J \leftarrow J + \ell(\phi; \theta)/N_\mathcal{D}$ ▷ Batch loss
  $\theta \leftarrow \theta + \alpha_i \partial J / \partial \theta$ ▷ SGD step
  $\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ New initial state samples
  $i \leftarrow i + 1$
return θ

Bayesian NeuralPBC

Hybrid systems such as walking machines and manipulators perform constant interaction with their environment. In many cases, the exact parameters of the environment are not known. For instance, walking machines operate on uneven terrain, where the exact elevation and friction coefficients of the runway may be unknown. Manipulators interact with objects of different textures and friction, which we cannot simply infer from sensors. In order to learn a controller robust against these uncertainties, we inject domain randomization on the state of the environment during the training process. Unfortunately, simply introducing random environmental conditions to the training outlined in Algorithm 8 is not sufficient. In the presence of high variance disturbances, the point-estimate parameters θ under domain randomization are prone to be biased [40]; for instance, if the uncertainty in the terrain elevation is large, the learned parameters θ may be far from the optimal controller corresponding to the true elevation. To combat this issue, we propose a probabilistic framework, where we learn a posterior probability distribution over the parameters θ via Bayesian inference. Then we address the bias-variance trade-off problem by marginalizing the controller over the

distribution of the parameters [17].

In the probabilistic framework, we parameterize the desired Hamiltonian H_d with a Bayesian neural network, whose weights and biases are samples drawn from a posterior probability distribution [29]. The objective is to find the posterior distribution $P(\theta|J)$ that achieves the performance objective for various environmental conditions. This framework can be summarized by the following optimization problem.

$$\begin{aligned} \underset{P(\theta|J)}{\text{minimize}} \quad & J(\phi, u) = \mathbb{E}_{x_0 \sim \mathcal{D}_N}[\ell(\phi(x_0, u^\theta, T), u^\theta)], \\ \text{subject to} \quad & M(q)\,\mathrm{d}\dot{q} + h(q, \dot{q}, \theta)\,\mathrm{d}t - \mathrm{d}R = 0, \\ & u^\theta = -\Omega^\dagger(\nabla_q H_d^\theta - \nabla_q H), \\ & p_s \sim \mathcal{U}(p_{min}, p_{max}), \\ & \theta \sim P(\theta|J). \end{aligned} \tag{3.44}$$

The random variable $p_s \in \mathbb{R}^N$ is sampled from N uncorrelated uniform probability distributions $\mathcal{U} \sim [p_{min}, p_{max}]$ with lower bound p_{min} and upper bound p_{max}. The magnitude of the samples p_s determine the parameters of the environment with which we are interacting. For walking robots, p_s can be the elevation of the terrain; in this case, sampling p_s randomizes the gap g_N, which determines the pre-impact velocities $\dot{q}^-$ and contact forces between the robot and the ground.

We solve the optimization problem in (3.44) with the procedure outlined in Algorithm (9). We follow the initial state sampling techniques and the performance objectives discussed in detail in Section 3.3.1. In particular, we use variational inference to learn the distribution parameters of the pre-selected posterior $Q(\theta; z)$.

Algorithm 9 Solution to the Optimization Problem (3.44)

$\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ $N_\mathcal{D}$ initial state samples
while $i <$ `maximum iteration` **do**
  $\mathcal{L} \leftarrow 0$ ▷ ELBO Loss
  for $i = 1 : Q_N$ **do** ▷ Samples to compute (3.16)
    $J \leftarrow 0$ ▷ Batch loss
    $\theta \sim Q(\theta; z)$ ▷ Sample parameters of H_d^θ
    for $(x_0) \sim \mathcal{D}_N$ **do**
      $p_s \sim \mathcal{U}(p_{min}, p_{max})$
      $\phi = \text{Moreau}(x_0)$ ▷ Algorithm (1)
      $J \leftarrow J + \ell(\phi; \theta)/N_\mathcal{D}$
    $\mathcal{L} \leftarrow \mathcal{L} + \frac{1}{Q_N} \left(\log[P(J|\theta)P(\theta)] - \log[Q(\theta; z)]\right)$
  $z \leftarrow z + \alpha_i \partial\mathcal{L}/\partial z$ ▷ SGD step
  $\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ New initial state samples
  $i \leftarrow i + 1$
return z

3.3.5 Rimless Wheel

This mechanism consists of a rimless wheel in the plane with a set of N spokes and a torso freely rotating about a pin joint located at the center of the wheel. The mass of the wheel m_1 is concentrated at the hip and spans a radius of l_1. The torso has its mass m_2 concentrated at a distance of l_2 from the hip. We actuate the torso angle with the motor mounted at the hip, which in turn propels the entire wheel. The angle of the torso is characterized by φ measured from the outward normal of the runway shown as $\hat{n}$ in Figure 3.10. The orientation of the wheel β is measured from $\hat{n}$ to a datum spoke.

Rimless wheel is a system that undergoes phases of continuous flows and discrete transitions, resulting in a hybrid dynamical system with two modes. We construct a dynamical model for such system with the Lagrangian approach. The kinetic and potential energies of the system are given by

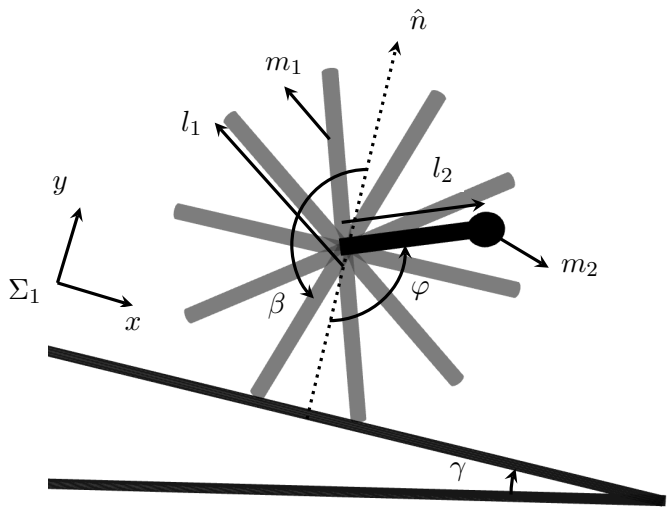

Figure 3.10: Rimless wheel with torso; depicted with $N = 10$ spokes.

$$\mathcal{K} = \frac{1}{2}m_t(\dot{x} + \dot{y})^2 + \frac{1}{2}I_1\dot{\beta}^2 + \frac{1}{2}(m_2 l_2^2 + I_2)\dot{\varphi}^2 + m_2 l_2(c_\varphi \dot{x} + s_\varphi \dot{y})\dot{\varphi}$$

$$\mathcal{P} = m_t g(-x s_\gamma + y c_\gamma) - m_2 g l_2 c_{\varphi\gamma}$$

where $c_a := \cos(a), s_a := \sin(a), c_{ab} := \cos(a-b), s_{ab} := \sin(a-b)$. The total mass of the mechanism is given by m_t; I_1 and I_2 are moments of inertia of the wheel and torso respectively. The position vector (x, y) represents the location of the hip with respect to the frame Σ_1 shown in Figure 3.10. The slope of the runway is given by γ, g is the magnitude of acceleration due to gravity. The Euler-Lagrange equations corresponding to the Lagrangian $\mathcal{L} = \mathcal{K} - \mathcal{P}$ are

$$M(q)\,\mathrm{d}\dot{q} + h(q, \dot{q})\,\mathrm{d}t - \mathrm{d}R = 0, \tag{3.45}$$

$$h(q,\dot{q}) = C(q,\dot{q})\dot{q} + G(q) - Bu(q,\dot{q}),$$

$$M(q) = \begin{bmatrix} m_t & 0 & m_2 l_2 c_\varphi & 0 \\ 0 & m_t & m_2 l_2 s_\varphi & 0 \\ m_2 l_2 c_\varphi & m_2 l_2 s_\varphi & I_2 + m_2 l_2^2 & 0 \\ 0 & 0 & 0 & I_1 \end{bmatrix},$$

$$C(q,\dot{q}) = m_2 l_2 \begin{bmatrix} 0 & 0 & -s_\varphi \dot{\varphi} & 0 \\ 0 & 0 & c_\varphi \dot{\varphi} & 0 \\ -s_\varphi \dot{\varphi} & c_\varphi \dot{\varphi} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$G(q) = g \begin{bmatrix} -m_t s_\gamma & m_t c_\gamma & m_2 l_2 s_{\phi,\gamma} & 0 \end{bmatrix}^\top$$

where $q = (x, y, \varphi, \beta)$, $B = \begin{bmatrix} 0 & 0 & 1 & -1 \end{bmatrix}^\top$, $u(q,\dot{q})$ is the torque applied to the torso and $\mathrm{d}R$ represents the force measure of contact forces and Coulomb friction exerted on the spokes by the ground. We use the linear complementarity formulation [14] to resolve the contact forces and Coulomb friction contributing to the discrete transitions.

Training

Both the deterministic and Bayesian frameworks learn H_d given by a fully-connected neural network with a total of 113 weights and biases. A single parameter update consists of a batch of 4 initial states sampled through greedy or explorative techniques. Each initial state is integrated forward for time horizon of 5 seconds. The objective of the trainings is to use the control authority on the torso to move the robot at a constant hip speed. The

performance objective is given by the accumulated loss

$$J_T = \sum_{t=0}^{T} (\dot{x}^* - \dot{x}(t;\theta)) \tag{3.46}$$

where $\dot{x}^*$ is a constant 1 $^{\mathrm{m}}/_{\mathrm{s}}$. For the Bayesian training, we generate random terrain elevation parameters p_s from a uniform probability distribution $\mathcal{U} \sim [0, 2]$ centimeters.

Simulated Experiments

We first show the performance of the deterministic NeuralPbc training on a level ground. As shown on the bottom plot of Figure 3.11, the robot starts from rest and slowly reaches the desired hip speed. The top plot shows that the controller learned to apply large torque when the wheel is at rest and then the torso maintains a constant angle, which is sufficient to create constant hip speed on level ground. During discrete transitions (shown with dashed-red lines), the angular velocity of the torso counteracts the forward motion of the wheel, but a large torque is applied quickly to maintain the torso angle. The torque plot on Figure 3.12 shows that a large positive torque is applied if the wheel stumbles backwards.

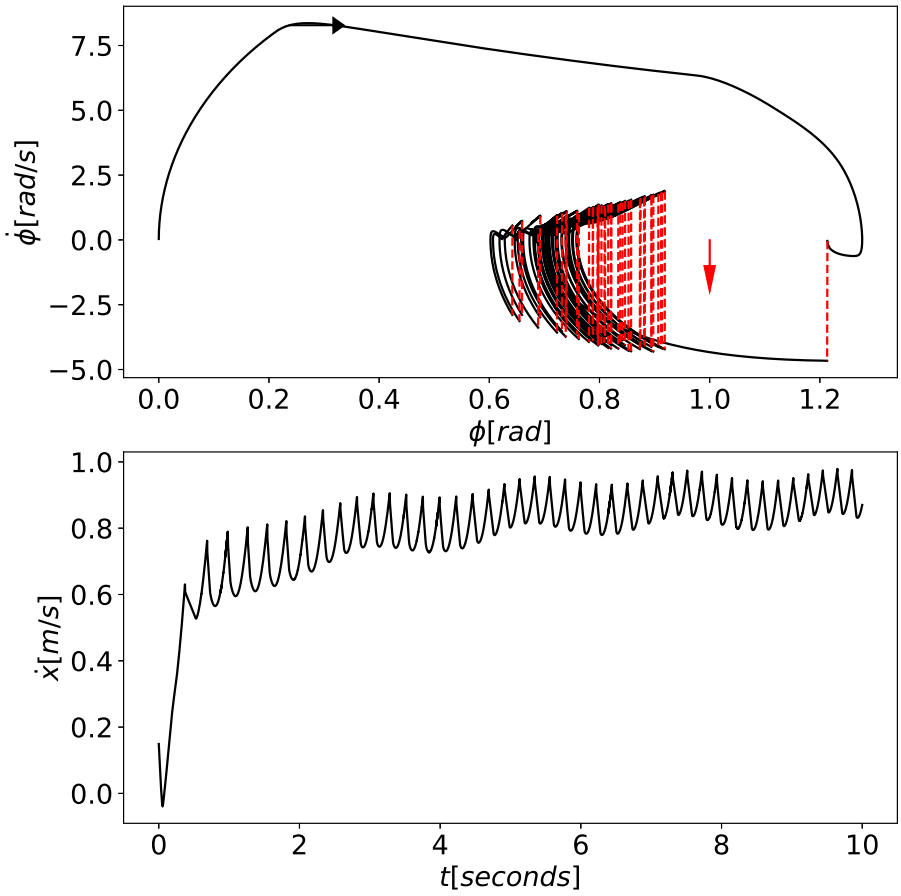

Figure 3.11: Top: Torso orientation and velocity for 10-second trajectory. The solid black lines show the continuous phases and the dashed red lines show discrete transitions. Bottom: Horizontal hip speed

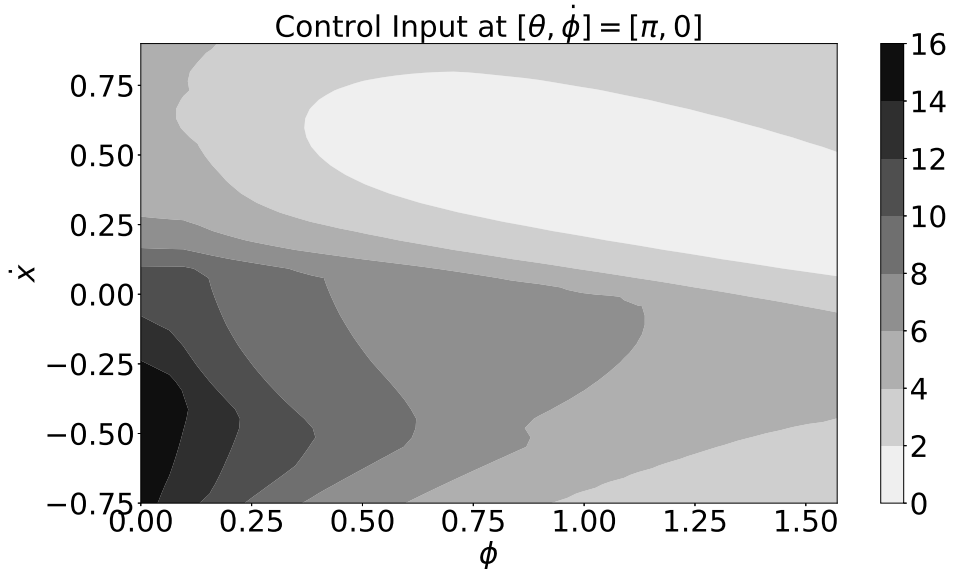

Figure 3.12: Torque command to torso as a function of torso angle and horizontal hip speed

We compare the performance of the two controllers as follows. Similar to the Bayesian training, we sample the terrain elevation from $\mathcal{U} \sim [0, p_{max}]$ where $p_{max} = [0, 0.5, 1, 1.5, 2]$ centimeters. For each value of p_{max}, we generate 10 random initial states x_0 and integrate 10-second trajectories. The cost of each trajectory is calculated as per (3.46). The bars in Figure 3.13 show the average J_T over the 10 trajectories for each p_{max}.

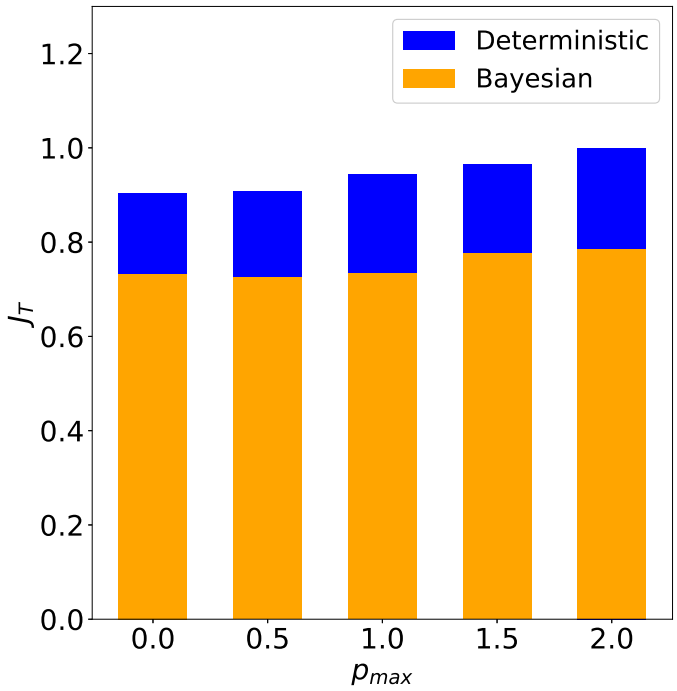

Figure 3.13: Comparison of deterministic and Bayesian control for the rimless wheel.

Real-World Experiments

We test the performance of both controllers on the hardware shown in Figure 3.14. The robot consists of two set of 10 spokes for stability. The torso holds two ODrive v3.6 brushless DC motors, which actuate the drive shaft through a belt-drive system. The end of the Aluminum spokes land on preloaded springs to reduce vibration, while the rubber feet ensure no-slip condition. The incremental capacitive encoders attached to the motors report the orientation of the spokes. We use IMU readings fused with Mahony filter [41] to estimate the pitch of the torso. A 24V battery pack is placed at the bottom edge of the torso to power the motors, a Raspberry Pi 3B and a Teensy microcontroller. We evaluate

the neural networks on a laptop and exchange sensor readings and torque commands with the Raspberry Pi via ROS wireless communication protocols.

Figure 3.14: Rimless Wheel Assembly

3.4 Bayesian Neural Interconnection and Damping Assignment PBC

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. We invoke variational inference to

find the approximate posterior over the parameters θ. The goal is to learn the distribution parameters z of the posterior multivariate probability distribution $Q(\theta; z)$ that maximize the ELBO given in (3.16).

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 (3.12). Then, the likelihood is given by

$$P(\|l_{\text{IDA}}(x)\|^2 \mid \theta) = \mathcal{N}(0, s), \tag{3.47}$$

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 (3.47), maximizing the ELBO in (3.16) coaxes the loss $l_{\text{IDA}}(x)$ to zero.

We update the distribution parameters z along the gradient $\partial\mathcal{L}/\partial z$ until the ELBO converges and the objective function $\|l_{\text{IDA}}(x)\|^2$ reaches the threshold ϵ_{tol}. We invoke the reparameterization trick of the Automatic Differentiation Variational Inference(ADVI) [33] 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 p_s from a normal distribution $\mathcal{N}(\hat{p}_s, \sigma_p)$ centered around the nominal parameter $\hat{p}_s$, where σ_p 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, we draw a new sample of p_s.

Table 3.3: Neural-IDAPBC training setup for deterministic and Bayesian frameworks

	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

3.4.1 Inertia Wheel Pendulum

Training

The optimization problem (3.13) is constructed as follows. The potential energy function V_d^θ is a fully-connected neural network with two hidden layers, each of which has the activation function ELU. The closed-loop mass matrix is constructed according to the Cholesky decomposition $M_d^\theta = L_\theta^\top L_\theta$, 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 is independent of q for this system. The parameters of the surrogates are initialized according to the Glorot (Xavier) [42] scheme. The optimization problem is solved over a uniform discretization of $q = (q_1, q_2) \in [-2\pi, 2\pi] \times [-50, 50]$.

In the deterministic setting, the nominal system parameters reported in Table 3.2 are used for $H(q, p)$ during training. In the Bayesian setting, the standard deviations σ_p of system parameters $p_s = [I_1, I_2, mgl]$ are chosen to be 10% of the nominal system parameters given in Section 3.3.3. We use variational inference to estimate a Gaussian posterior distribution over uncorrelated parameters. After training, both settings use the nominal values for the computation of $H(q, p)$ in the control synthesis given by Equation (3.11). A summary of the hyperparameters for both the deterministic and Bayesian methods are given in Table 3.3.

Simulation Tests

The performance of the controllers obtained from the deterministic and Bayesian trainings are compared as follows. Similar to the NeuralPbc simulation tests, we introduce system parameter 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 m_3. Figure 3.15 shows the performance of the controllers. The policy learned from the Bayesian training is marginalized over 10 parameters sampled from the posterior per (3.17).

As seen in Figure 3.15, 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.

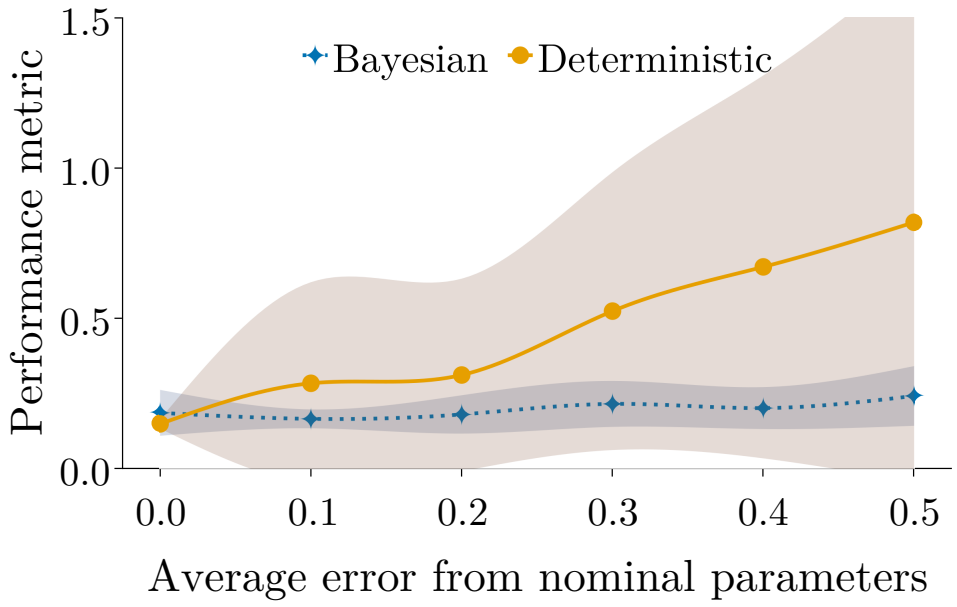

Figure 3.15: Accumulated quadratic cost (J^T) for a range of error in system parameters. Lower values correspond to better controller performance.

Hardware Tests

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 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 3.2). The state x is recorded, and the performance metric (3.41) is used to evaluate the controllers. The results are summarized in Figure 3.16.

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 3.2.

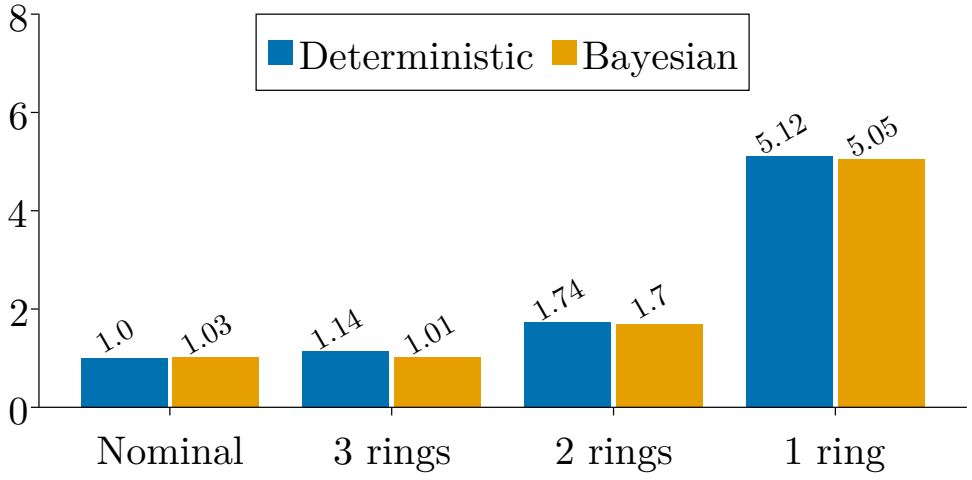

Figure 3.16: Normalized accumulated cost J_T (lower is better) for modified system parameters. The categories A-C correspond to the parameters shown in Table 3.2.

3.5 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 our framework finds 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.

BIBLIOGRAPHY

[1] J. McGonagle, A. Dobre, and G. Pilling, "Gaussian mixture model." [Online]. Available: https://brilliant.org/wiki/gaussian-mixture-model/

[2] N. A. Ashenafi and A. C. Satici, "Nonholonomic cooperative manipulation in the plane using linear complementarity formulation," in *2021 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2021, pp. 634–639.

[3] 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.

[4] 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.

[5] 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.

[6] N. A. Ashenafi, W. Sirichotiyakul, and A. C. Satici, "Robust passivity-based control of underactuated systems via neural approximators and bayesian inference," in *2022 IEEE Conference on Decision and Control (under review)*, 2022.

[7] W. Sirichotiyakul, N. A. Ashenafi, and A. C. Satici, "Robust interconnection and damping assignment passivity-based control via neural bayesian inference," in *2022 IEEE Transactions on Automatic Control (under review)*, 2022.

[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] 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.

[12] 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*, ser. Proceedings of Machine Learning Research, A. Singh and

J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 914–922. [Online]. Available: https://proceedings.mlr.press/v54/linderman17a.html

[13] 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)*. IEEE, 2020, pp. 4093–4099.

[14] C. Glocker and C. Studer, "Formulation and preparation for numerical evaluation of linear complementarity systems in dynamics," *Multibody System Dynamics*, vol. 13, no. 4, pp. 447–463, 2005.

[15] F. Génot and B. Brogliato, "New results on painlevé paradoxes," *European Journal of Mechanics-A/Solids*, vol. 18, no. 4, pp. 653–677, 1999.

[16] V. Acary and B. Brogliato, *Numerical methods for nonsmooth dynamical systems: applications in mechanics and electronics*. Springer Science & Business Media, 2008.

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

[18] T. Härkönen, S. Wade, K. Law, and L. Roininen, "Mixtures of gaussian process experts with smc^2," *arXiv preprint arXiv:2208.12830*, 2022.

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

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] J. Revels, M. Lubin, and T. Papamarkou, "Forward-mode automatic differentiation in julia," *arXiv preprint arXiv:1607.07892*, 2016.

[22] D. Liberzon, *Switching in systems and control*. Springer, 2003, vol. 190.

[23] 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.

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

[25] 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.

[26] C. F. G. D. Santos and J. P. Papa, "Avoiding overfitting: A survey on regularization methods for convolutional neural networks," *ACM Computing Surveys (CSUR)*, vol. 54, no. 10s, pp. 1–25, 2022.

[27] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov chain Monte Carlo in practice*. CRC press, 1995.

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

[29] 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.

[30] L. C. Evans, *An introduction to stochastic differential equations*. American Mathematical Soc., 2012, vol. 82.

[31] 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.

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

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

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

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

[36] R. Tedrake, *Underactuated Robotics*, 2022. [Online]. Available: http://underactuated.mit.edu

[37] W. Sirichotiyakul, N. A. Ashenafi, and A. C. Satici, "Robust data-driven passivity-based control of underactuated systems via neural approximators and bayesian inference," in *2022 American Control Conference*, 2022.

[38] R. Naldi and R. G. Sanfelice, "Passivity-based control for hybrid systems with applications to mechanical systems exhibiting impacts," *Automatica*, vol. 49, no. 5, pp. 1104–1116, 2013.

[39] A. J. Van Der Schaft and H. Schumacher, *An introduction to hybrid dynamical systems*. Springer, 2007, vol. 251.

[40] N. A. Ashenafi, W. Sirichotiyakul, and A. C. Satici, "Robustness of control design via bayesian learning," *arXiv preprint arXiv:2205.06896*, 2022.

[41] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on automatic control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[42] 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*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

APPENDIX

Expectation of the performance index

Proof of Lemma 1. Substituting the solution (3.25) of the SDE (3.24) expression into the performance measure (3.21) yields

$$\mathcal{J} = -\frac{1}{4}\frac{c+r\theta^2}{p_s+\theta}\left(1+e^{2T(p_s+\theta)}\right) + (c+r\theta^2)\theta\sigma\int_0^T e^{(p_s+\theta)t}\int_0^t e^{(p_s+\theta)(t-s)}dW_s dt + \frac{1}{2}(c+r\theta^2)\theta^2\sigma^2\int_0^T\left(\int_0^t e^{(p_s+\theta)(t-s)}dW_s\right)^2 dt$$

The conditional expectation of this quantity given the system parameter p_s under the distribution induced by the Wiener process may be computed in closed-form using Itô calculus:

$$\begin{aligned}
\mathbb{E}_W\left[\mathcal{J} \mid p_s\right] &= -\frac{1}{4}\frac{c+r\theta^2}{p_s+\theta}\left(1-e^{2T(p_s+\theta)}\right) + (c+r\theta^2)\theta\sigma\int_0^T e^{(p_s+\theta)t}\mathbb{E}_W\left[\int_0^t e^{(p_s+\theta)(t-s)}dW_s \,\middle|\, p_s\right]dt + \\
&\quad \frac{1}{2}(c+r\theta)^2\theta^2\sigma^2\int_0^T \mathbb{E}_W\left[\left(\int_0^t e^{(p_s+\theta)(t-s)}dW_s\right)^2 \,\middle|\, p_s\right]dt \\
&= -\frac{1}{4}\frac{c+r\theta^2}{p_s+\theta}\left(1-e^{2T(p_s+\theta)}\right) + \frac{1}{2}(c+r\theta^2)\theta^2\sigma^2\int_0^T\left(\int_0^t e^{2(p_s+\theta)(t-s)}ds\right)dt \\
&= -\frac{1}{4}\frac{c+r\theta^2}{p_s+\theta}\left(1-e^{2T(p_s+\theta)}\right) + \frac{1}{2}(c+r\theta^2)\theta^2\sigma^2\int_0^T -\frac{1}{2(p_s+\theta)}\left(1-e^{2T(p_s+\theta)}\right)dt \\
&= -\frac{1}{4}\frac{c+r\theta^2}{p_s+\theta}\left[\theta^2\sigma^2 T + \left(1-e^{2T(p_s+\theta)}\right)\left(1+\frac{1}{2}\frac{\theta^2\sigma^2}{p_s+\theta}\right)\right].
\end{aligned}$$

$\square$