# 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

© 2023

Nardos Ayele Ashenafi

ALL RIGHTS RESERVED

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1:

# INTRODUCTION

It is possible to manually switch between multi-modal controllers by using some event trigger that implies mode changes in the dynamics. But this technique poses two main difficulties. First, a system with $k$ contact events can have up to $2^k$ contact combinations. It is quite difficult to parameterize the effects of these contact combinations and find individual controllers for each mode. Second, it is difficult to parameterize the relationship between the states and the conditional that triggers the control switch.

# 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 basic introduction to 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 [2]

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

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 and impact forces that enforce the geometric and kinematic constraints of contact surfaces. 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 Painleve' Paradox [3] in high friction scenarios. The linear complementarity formulation in [2] 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 $\gamma_T$ 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 $\epsilon_N$ and $\epsilon_T$ are the normal and tangential coefficients of restitution respectively. From the dynamics in (2.1), $\xi_N$ and $\xi_T$ can be expressed as an affine function over the contact forces $\lambda_N$ and $\lambda_T$ as follows [2]:

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

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

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

where $\mu$ 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 $\gamma_N$ and $\gamma_T$, respectively, and $\Delta t$ 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 [2] 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 [4] 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** $t \in 0 : \Delta t : T$ **do**

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

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

  $\lambda_N, \lambda_T \leftarrow \text{Lemke}(q(t_M), \dot{q}(t))$ ▷ Lemke [4]

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

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

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

**return** $\phi$

### 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

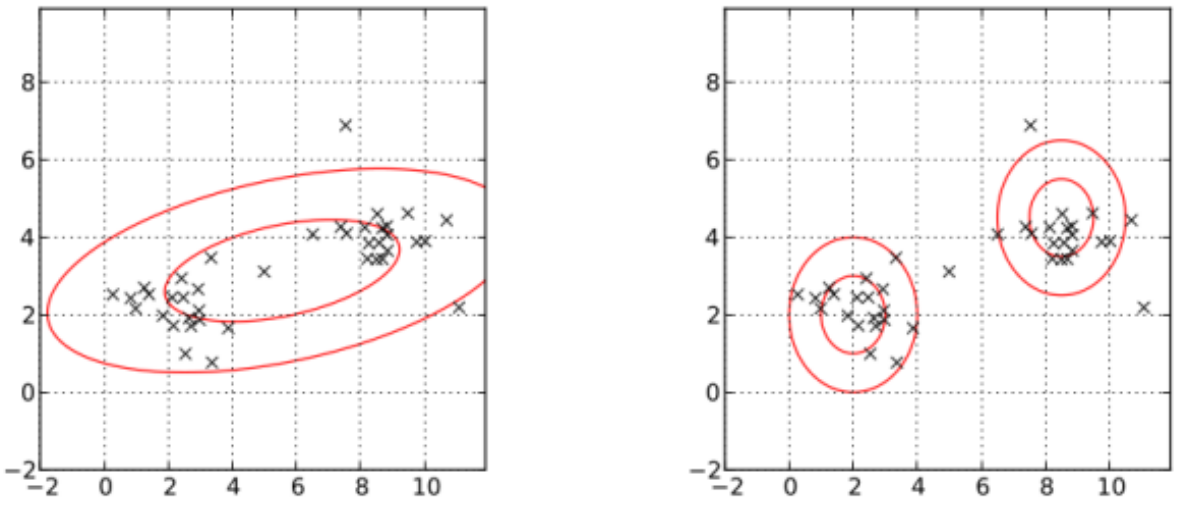



**Figure 2.1: Left: Fit with one Gaussian distribution. Right: Fit with Gaussian mixture of 2 experts [1]**

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 [5]. However, as the observations from the dataset grow large, the Gaussian experts impose large computational and memory overhead [6]. 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 $P(x; \psi) : \mathcal{X} \to \mathbb{R}^{N_F}$ is also a neural net with parameters $\psi$. The objective is to learn the decision parameters $(\psi, \theta)$ from observed data.

The gating network $P(x)$ provides a vector of probabilities, whose components $P(i|x, \psi)$ represent the probability of state $x$ belonging to the state partition $i$. We use a categorical

probability distribution to sample a state partition based on the outputs of the gating network as follows [6]

$$c_i|x, \psi \sim \text{Categorical}(P(x; \psi)) \tag{2.4}$$

where $c_i$ is the bin corresponding to the $i^{th}$ expert and $\psi$ is the parameters of the gating network. The prediction from the mixture of experts is given by the linear combination over the *responsibility* of each state partition.

$$F_o(x) = \sum_{i=1}^{N_F} F_i(x) P(c_i = i|x, \psi)$$

A single controller is active in each state partition hence the prediction $F_o(x)$ is equivalent to

$$j = \underset{i}{\text{argmax}}\{P(c_i = i|x, \psi)\}$$
$$F_o(x) = F_j(x)$$

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 log likelihood function $\ln\{P(y|\theta, \psi)\}$ as [5]

$$\ln\{P(y|\theta, \psi)\} = \ln\Big\{\sum_{i=1}^{N_F} \mathcal{N}(y|F_i(x), s) P(c_i = i|x, \psi)\Big\} \tag{2.5}$$

where $\mathcal{N}$ is the Gaussian distribution with mean $F_i(x)$ and standard deviation $s$. We can

simplify this expression to get

$$\ln\{P(y|\theta,\psi)\} = \sum_{i=1}^{N_F} \ln\Big\{\frac{1}{(2\pi s)^{1/2}}\Big\} - \frac{1}{2s}\|f_i(x) - y\|^2 + \ln\{P(c_i = i|x,\psi)\} \tag{2.6}$$

where $\|F_i(x) - y\|$ is the error in the current prediction. We can remove the constants in (2.6) and define the final form of the likelihood function as

$$\mathbb{L}(y|\theta,\psi) := \sum_{i=1}^{N_F} -\|F_i(x) - y\|^2 + \ln\Big\{P(c_i = i|x,\psi)\Big\}. \tag{2.7}$$

Notice that the likelihood measures how likely the current parameters $(\psi,\theta)$ are to generate the sampled data $y$. The goal is to maximize the likelihood and consequently minimize prediction error. The likelihood (2.7) is maximum when the parameter $\theta_i$ has the lowest prediction error and the highest probability of getting selected by the gating network. The standard EM approach evaluates the expectation of (2.7) over many samples of $x$ and uses gradient-based techniques to iteratively update the parameters $(\psi,\theta)$.

## 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 Section 2.2. The running cost plays the role of *prediction error* in the construction of the likelihood as shown in (2.7). The goal is to learn the decision parameters $(\psi, \theta)$ 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

$$\begin{aligned} \underset{\psi,\theta}{\text{minimize}} \quad & \ell\Big(\phi, u(\phi;\psi,\theta)\Big), \\ \text{subject to} \quad & M(q)\,\mathrm{d}\dot{q} + h(x;\psi,\theta)\,\mathrm{d}t - \mathrm{d}R = 0, \\ & u = \{F_i(x;\theta_i) \mid i \sim \text{Categorical}(P(x;\psi))\} \end{aligned} \tag{2.8}$$

In Section 2.2, we provide a procedure to solve the optimization problem 2.8.

**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.

$$\begin{aligned} \ell(x,u) &= \frac{1}{2}(x-x^*)^\top Q(x-x^*) + \frac{1}{2}u^\top Ru, \\ \ell(\phi,u) &= \int_0^T \ell(x(t),u)\,\mathrm{d}t, \end{aligned} \tag{2.9}$$

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. The corresponding likelihood is given by

$$\mathbb{L}(\phi) = \int_0^T \left( \sum_{i=1}^{N_F} \left[ -\ell\Big(x(t), F_i\Big) + \ln\Big\{ P\Big(c_i = i | x(t), \psi\Big) \Big\} \right] \right) \mathrm{d}t. \tag{2.10}$$

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

$$\begin{aligned} t_{min} &= \underset{t}{\operatorname{argmin}}\ \ell(x(t),u) \\ \mathbb{L}(\phi) &= -\frac{\ell(x(t_{min}),u)}{C} + \sum_{t=0}^{t_{min}} \ln\{P(c_i = i|x(t),\psi)\} \end{aligned} \tag{2.11}$$

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 $T$.

**State Sampling**

We intend to find a solution to the optimization problem in (2.8) 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 [7]. This technique refines the controller on the states most visited under the current parameters $(\psi, \theta)$. 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 compute the running cost as an expectation over $N_{\mathcal{D}} = N_d + N_r$ samples as follows:

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

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

**Training MOE**

We solve the optimization problem given in (2.8) with stochastic gradient descent (SGD). We invoke a variant of SGD known as Adam [8] to efficiently train the parameters with adaptive learning rates $\alpha_i$. The full training procedure is outlined in Algorithm (2). We leverage forward-mode auto-differentiation technique [9] to back-propagate on the gradient of the likelihood with respect to the learned parameters.

**Algorithm 2** Solution to the Optimization Problem (2.8)

$\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ $N_\mathcal{D}$ initial state samples
**while** $i <$ `maximum iteration` **do**
  $J \leftarrow 0$ ▷ Batch loss
  **for** $z_0 \in \mathcal{D}_N$ **do**
    $\phi = \text{Moreau}(x_0, \psi, \theta)$ ▷ Algorithm (1)
    $J \leftarrow J + \mathbb{L}(\phi)/N_\mathcal{D}$ ▷ Batch loss
  $\theta \leftarrow \theta + \alpha_i \partial J / \partial \theta$ ▷ 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** $\theta$

### Back-propagation through Hybrid Systems

The training framework outlined (2.8) 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 $\phi$ with the sampled expert control parameter $\theta_i$. Forward-mode auto-differentiation evaluates the gradient of the accumulated cost with respect to $\theta_i$ 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 $\lambda$ 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 $\theta_i$ 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 expert controllers for the cartpole system enclosed with wall barriers.

### 2.3.1 Stable Switching Between Unstable Systems

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

$$\begin{aligned} \dot{x} = A_1 x = \begin{bmatrix} 0 & -1 \\ 2 & 0 \end{bmatrix} x \\ \dot{x} = A_2 x = \begin{bmatrix} 0 & -2 \\ 1 & 0 \end{bmatrix} x. \end{aligned} \tag{2.12}$$

Even if both systems are unstable, it is possible to find a state-dependent switching rule that makes the resulting switched system stable [10]. We aim to learn the parameters $\psi$ 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 ($2 \to 6 \to 4$ neurons) and an Elu activation function [11]. 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

$$u(\theta_i) = \begin{cases} 0, & \theta_i > \frac{1}{2}, \\ 1, & \theta_i \leq \frac{1}{2}, \end{cases} \tag{2.13}$$

where $u = 0$ corresponds to the first dynamics $\dot{x} = A_1 x$ and $u = 1$ corresponds to $\dot{x} = A_2 x$. We use the SIGMOID activation function to limit $\theta_i$ between 0 and 1. We generate a trajectory following the procedure in Algorithm (3). The likelihood is modified to account for the probabilistic control as

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

**Algorithm 3** Stable Switching between Unstable Systems

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

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

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

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

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

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

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

**return** $\phi$

The resulting switching system is shown in Figure **??**.

### 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 **??**. We demonstrate the performance of the mixture of expert controllers in simulation and real-world experiments. Lastly, we compare

the performance of the mixture of expert controllers against a single swing-up controller.

**System Model**

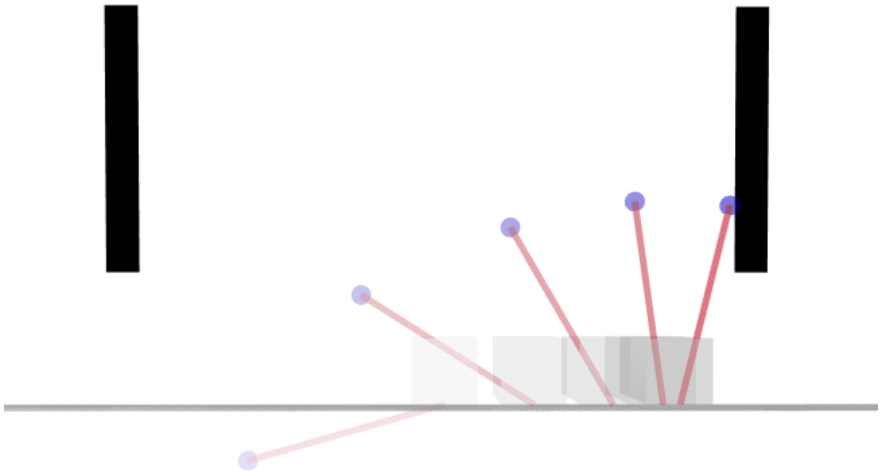
The cartpole system consists of a freely rotating pendulum link, riding on an actuated cart. The setup is enclosed by two rigid walls hanging 0.2m from the bottom of the cart. The objective is to use the control authority on the cart in order to swing-up the passive pendulum to the upright. The pendulum 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 between the cart and the track its sliding on 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

$$
\begin{aligned}
M(q) &= \begin{bmatrix} m_c + m_p & -m_p + l_{cm}\cos(\theta) \\ -m_p l_{cm}\cos(\theta) & m_p l_{cm}^2 + I_1 \end{bmatrix} \\
C(q,\dot{q}) &= \begin{bmatrix} b & m_p l_{cm}\sin(\theta) \\ -m_p \sin(\theta)/2 & 0 \end{bmatrix} \\
G(q) &= \begin{bmatrix} 0 & -m_p g l_{cm}\sin(\theta) \end{bmatrix}^\top \\
B &= [1\ 0]^\top
\end{aligned}
\tag{2.14}
$$

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



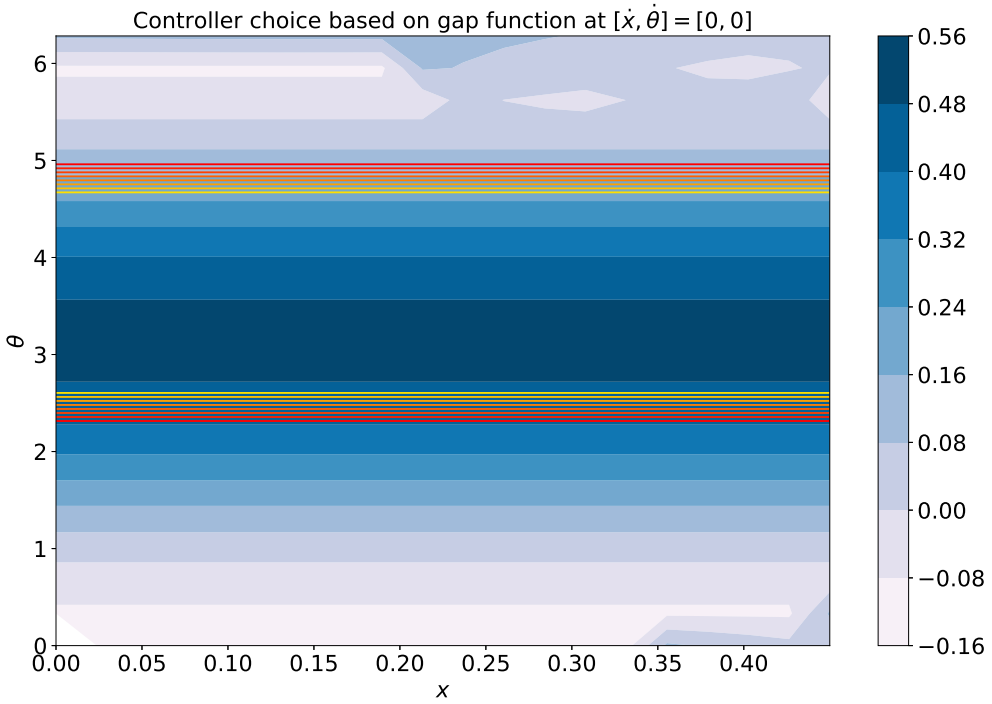



**Figure 2.2: Control switching as a function of gap function. There are total of two experts, one is active between the red boundaries**

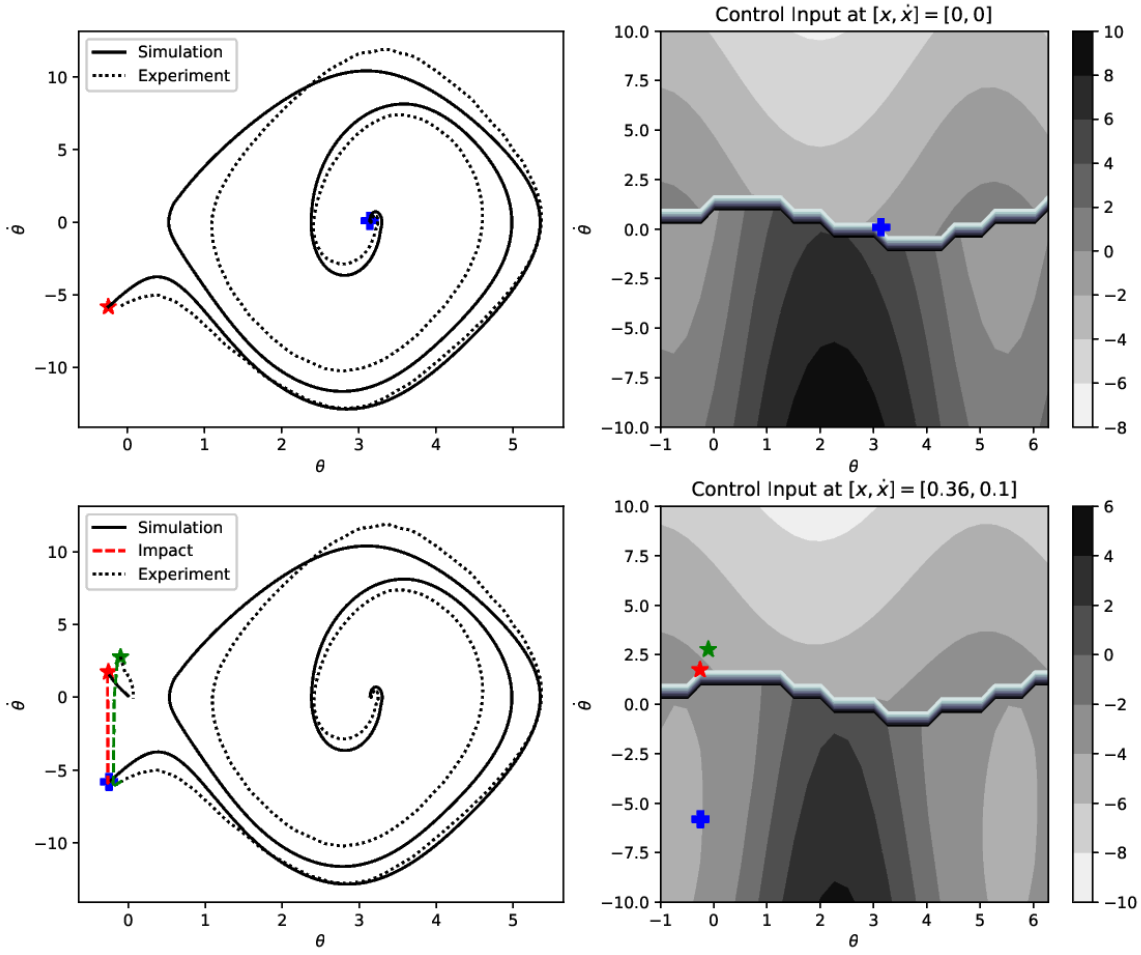



**Figure 2.3: Top row: Pendulum swing-up before impact. Bottom row: Velocity jump and control switching at impact**

## 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 $\Omega(q) \in \mathbb{R}^{m\times n}$ 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 $H_d : \mathcal{X} \to \mathbb{R}$ on the closed-loop system, rendering it passive and therefore stable [12]. 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 $(q^{\star}, p^{*})$. However, the closed-form solution to the partial differential equations (PDEs) in (3.7) is intractable.

**Neural PBC**

The deterministic NeuralPbc framework presented in [13] solves the PDEs (3.7) by rewriting the PBC problem as the following optimization problem

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

where $T > 0$ is the time horizon, $\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is a running cost function to be defined, and $\phi(x_0, u^{\theta}, T)$ is a closed-loop trajectory generated from the initial state $x_0$ under the current control law $u^{\theta}$. 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 perfor-

mance objective $\ell$.

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

**Neural Interconnection and Damping Assignment PBC**

IDAPBC, a variant of PBC, selects a particular structure for $H_d$

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

such that the control input must satisfy the PDEs given by

$$G^\perp \left\{ \nabla_q H - M_d M^{-1} \nabla_q H_d + J_2 M_d^{-1} p \right\} = 0. \tag{3.10}$$

The objective is to learn $V_d$ and the entries of $M_d$ and $J_2$ matrices. Once $V_d$, $M_d$ and $J_2$ are obtained, the energy-shaping control and the damping injection term are given by

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

respectively, where $G^\dagger = \left(G^\top G\right)^{-1} G^\top$.

The closed-form solution to the PDEs in (3.10) is intractable. Hence, the deterministic NEURAL-IDAPBC framework introduced in [14] formulates the following optimization

problem that finds an approximate solution to the PDEs.

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \|l_{\text{IDA}}(x)\|^2 = \left\|G^{\perp}\left\{\nabla_q H - M_d M^{-1}\nabla_q H_d + J_2 M_d^{-1} p\right\}\right\|^2, \\ \text{subject to} \quad & M_d^{\theta} = (M_d^{\theta})^{\top} \succ 0, \\ & J_2^{\theta} = -(J_2^{\theta})^{\top}, \\ & q^{\star} = \underset{q}{\operatorname{argmin}}\, V_d^{\theta}. \end{aligned} \tag{3.12}$$

where $V_d^{\theta}$ and the entries of the $M_d^{\theta}$ and $J_2^{\theta}$ matrices are parameterized by neural networks.

### 3.1.2 Bayesian Learning

The objective of Bayesian learning is to determine a stochastic model (target function) that best fits observed data $\mathcal{D}$ with inherent noise. Let this stochastic target function be represented by $F(x;\theta) : \mathcal{X} \to \mathbb{R}^t$, where $\theta \in \Theta \subset \mathcal{R}^v$ is a multivariate random variable that parameterizes the model. Given prior belief on the distribution of the parameters $p(\theta)$, Bayesian learning finds a posterior distribution $p(\theta \mid \mathcal{D})$ over $\theta$ that maximizes the likelihood of the target function generating the dataset $\mathcal{D}$ [5]. This can be expressed in terms of Bayes' theorem as

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{\int_{\theta} p(\mathcal{D} \mid \theta')p(\theta')d\theta'}, \tag{3.13}$$

where $p(\mathcal{D} \mid \theta)$ is the likelihood function and $p(\mathcal{D})$ is the evidence. While the likelihood and prior distribution can be expressed explicitly, the evidence is intractable. This calls for techniques that approximate or find the exact posterior distribution, some of which are discussed in the following section.

**Posterior Distribution**

Bayesian learning provides various techniques to infer the posterior distribution over the parameters $\theta$. Two of the most famous techniques are discussed as follows.

1. Markov Chain Monte Carlo (MCMC) methods: learn the exact posterior distribution by collecting samples of $\theta$ either through random walk (e.g. Metropolis-Hastings) or following the gradient of the likelihood (e.g. Hamiltonian Monte Carlo). Metropolis-Hastings methods collect samples of $\theta$ from a conditional probability distribution until the samples converge to an equilibrium distribution per the properties of irreducible and aperiodic Markov chains [15]. Hamiltonian Monte Carlo (HMC) method, shown in detail in Algorithm 4, also finds the equilibrium distribution of the Markov chain, but unlike Metropolis-Hastings, it efficiently searches the parameter space through the gradient of the likelihood. In the case of HMC, the Markov chain is generated from two first-order differential equations shown in lines 5-7 of Algorithm 4. While HMC method learns the exact posterior distribution, it has 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): this technique selects a posterior distribution $q(\theta; z)$ from the conjugate families of the likelihood and prior distributions. The goal is to learn the distribution parameters $z$ that minimize the Kullback-Leibler divergence or equivalently maximize the evidence lower bound (Elbo). The Elbo, $\mathcal{L}$, is given by [16]

$$\begin{aligned} \mathcal{L}(\mathcal{D}, z) &= \mathbb{E}_{\theta \sim q}\left[\log(p(\theta, \mathcal{D}; z)) - \log(q(\theta; z))\right], \\ p(\theta, \mathcal{D}; z) &= p(\mathcal{D} \mid \theta; z)p(\theta), \end{aligned} \tag{3.14}$$

**Algorithm 4** Hamiltonian Monte Carlo

Select initial state $\theta$ and momentum $r$ from prior knowledge
Select regularization coefficient $\lambda$
Create a set $\Theta$ to collect samples of $\theta$
Define $p(\theta, \mathcal{D}) \propto \exp(-E(\theta, \mathcal{D})), \quad E(\theta, \mathcal{D}) = \sum_{d \in \mathcal{D}} ||F(x;\theta) - d||^2 + \lambda ||\theta||^2$
**for** $t = 0 : \Delta t : T$ **do**
    $r(t + \Delta t/2) = r(t) - \frac{\Delta t}{2} \frac{\partial E}{\partial \theta_i}(\theta(t), \mathcal{D})$
    $\theta(t + \Delta t) = \theta(t) + \Delta t \, r(t + \Delta t/2)$
    $r(t + \Delta t) = r(t + \Delta t/2) - \frac{\Delta t}{2} \frac{\partial E}{\partial \theta_i}(\theta(t + \Delta t), \mathcal{D})$
    $\nu \sim \text{Uniform}[0, 1]$
    **if** $E(\theta(t + \Delta t), \mathcal{D}) < E(\theta(t), \mathcal{D})$ **then**
        $\Theta \leftarrow \Theta \cup \theta(t + \Delta t)$
    **else if** $\nu < \exp(E(\theta(t), \mathcal{D}) - E(\theta(t + \Delta t), \mathcal{D}))$ **then**
        $\Theta \leftarrow \Theta \cup \theta(t + \Delta t)$
    **else if** $\nu > \exp(E(\theta(t), \mathcal{D}) - E(\theta(t + \Delta t), \mathcal{D}))$ **then**
        Reject $\theta(t + \Delta t)$
**Return** $\Theta$

where $p(\mathcal{D} \mid \theta; z)$ is the likelihood function.

**Remark 1.** *For continuous posterior distribution, the* Elbo *given in equation* (3.33) *is redefined using differential entropy, which expresses the prior and posterior in terms of their probability density functions. In this case, the likelihood* $p(\theta \mid \mathcal{D}; z)$ *is also a probability density function and the* Elbo *is not bounded by zero.*

The power of Bayesian learning lies in its ability to build a target function and make predictions that integrate over uncertainties [17]. These predictions can be found by marginal-

izing the model over the posterior as follows [18].

$$\hat{F}(x) = \frac{1}{N} \sum_{\theta \sim q} F(x, \theta), \tag{3.15}$$

where $N$ is the number of samples drawn from the posterior. Moreover, Bayesian frameworks can quantify the confidence in the predictions through the variance of the predictive distribution, $p(F \mid x, \mathcal{D})$. The variance of $p(F|x, \mathcal{D})$ is given by [18]

$$\Sigma_{F|x,\mathcal{D}} = \frac{1}{N-1} \sum_{\theta \sim q} \left|\left| F(x, \theta) - \frac{1}{N} \sum_{\theta \sim q} F(x, \theta) \right|\right|^2. \tag{3.16}$$

## 3.2 Theoretical Justification of Robustness

In this section, we demonstrate 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 $p$ is uncertain:

$$\begin{cases} \dot{x} = px + u, \ x(0) = x_0 \\ u(x) = \theta x. \end{cases} \tag{3.17}$$

We assume that $p \sim \mathcal{N}\left(\hat{p}, \sigma_p^2\right)$ where $\hat{p}$ designates our best prior point estimate of the system parameter $p$ and $\sigma_p > 0$ quantifies the uncertainty in the knowledge of the system parameter. The controller is set to be linear in the state $x \in \mathbb{R}$ with its only parameter

$\theta \in \mathbb{R}$ to be determined through 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 $\theta$ is

$$\mathcal{J} = \int_0^T \left( \frac{1}{2} q x^2 + \frac{1}{2} r u^2 \right) dt, \tag{3.18}$$

where $T$ is the control horizon and $q \geq 0$ and $r > 0$ are design parameters. We solve the control system (3.17) to find $x(t) = e^{(p+\theta)t}$ and plug this into the performance index (3.18) along with the form selected for the controller. Performing the integration over time and letting $T \to \infty$, assuming that $p + \theta < 0$ then yields the infinite-horizon optimal cost functional

$$\mathcal{J}_\infty = -\frac{1}{4} \frac{q + r\theta^2}{p + \theta}. \tag{3.19}$$

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

$$\begin{aligned} \nabla_\theta \mathcal{J}_\infty &= -\frac{r}{4} \frac{(p+\theta)^2 - (p^2 + {}^q\!/_r)}{(p+\theta)^2} = 0, \\ \therefore \theta^\star &= g(p) := -p - \sqrt{p^2 + {}^q\!/_r}, \\ g^{-1}(\theta) &= \frac{q}{2r\theta} - \frac{\theta}{2}. \end{aligned} \tag{3.20}$$

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

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

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

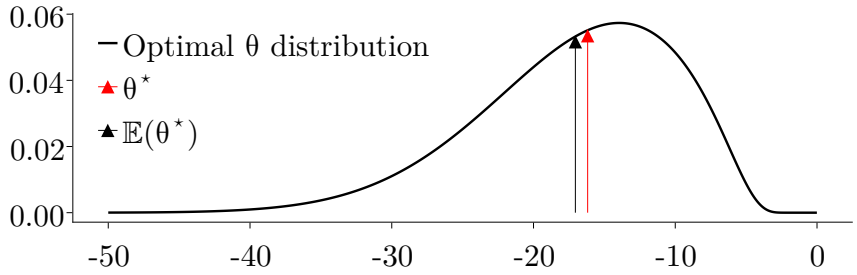



**Figure 3.1: The optimal control parameter distribution given that the system parameter $p$ is normally distributed with mean $\hat{p} = 5$ and $\sigma_p = 5$. The red and black arrows respectively indicate the optimal control parameter without considering the randomness of $p$, and the expected value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

We can further eliminate the control parameter from the expression for the optimal cost function $\mathcal{J}_\infty$ by substituting for $\theta$ from equation (3.20), yielding

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

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

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

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

There are several advantages of employing Bayesian learning to find the optimal control parameter $\theta$ as the toy example in this subsection supports. In order to derive some

quantitative results, let us assign some numerical values to the parameters that define the optimal cost function $(q, r) = (100, 1)$, our best guess $\hat{p} = 5$ of the system parameter $p$ and its standard deviation $\sigma_p = 5$.

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

Finally, Figure 3.1 shows the optimal control parameter distribution given that the system parameter $p$ is normally distributed with mean $\hat{p} = 5$, standard deviation $\sigma_p = 5$. This figure also shows the mean values of the optimal control distribution with the black arrow and the optimal control parameter a deterministic approach would yield in red. We notice that the Bayesian learning that yields the optimal control parameter distribution is more concerned about system stability due to the uncertainty in the parameter $p$, a feat that the deterministic training may not reason about.

### 3.2.2 Optimal Control under Parameter Uncertainty and Measurement Noise

Consider the scenario in which the system (3.17) 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

$$\begin{cases} dx(t) = (p+\theta)x(t)\ dt + \theta\sigma\ dW_t, \\ x(0) = 1, \end{cases} \tag{3.21}$$

where $W$ denotes the Wiener process [19]. The initial state is assumed deterministic and is set to unity for simplicity. The unique solution to this SDE is given by

$$x(t) = e^{(p+\theta)t} + \theta\sigma \int_0^t e^{(p+\theta)(t-s)} dW_s. \tag{3.22}$$

**Lemma 1.** *The conditional expectation* $\mathbb{E}[\mathcal{J} \mid p]$ *of the performance index* (3.18) *given the system parameter* $p$ *is*

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

*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 $p$ constant in this analysis so far. Uncertainty over this variable can be incorporated by taking a further expectation

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

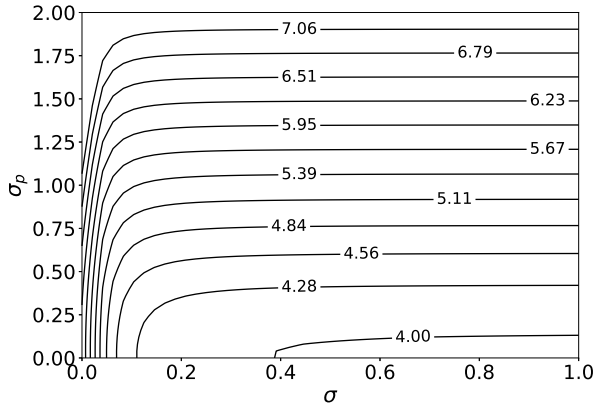



**Figure 3.2: The optimal controller parameter magnitude $|\theta^\star|$.**

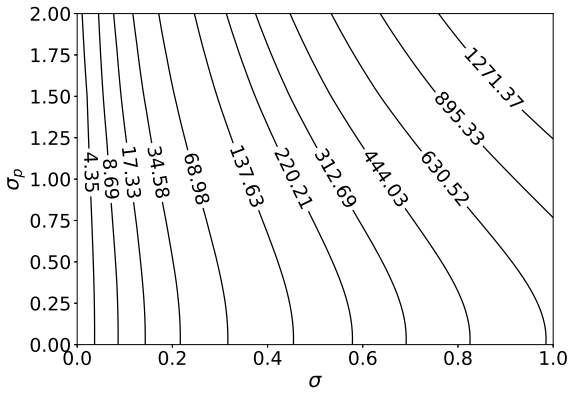



**Figure 3.3: The minimal expected cost $\mathbb{E}[\mathcal{J}]$.**

of $\mathbb{E}_W[\mathcal{J} \mid p]$ over $p$, 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 $\sigma$ and the system parameter $\sigma_p$. The constants we used to generate the data are given by $q = r = 1$ and $T = \hat{p} = 3$. 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 this uncertainty is small. The optimal cost shares this insensitivity for an even wider range of values of $\sigma_p$. 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

### 3.3.1 Control Design for Smooth Dynamical Systems

In this subsection, we formulate the Bayesian learning framework that minimizes the effects of system parameter uncertainties and measurement errors. In this framework, the neural net parameterization of $H_d^\theta$ given in Section **??** is replaced by a Bayesian neural network whose parameters are uncorrelated random variables sampled from a multivariate probability distribution (MPD). The goal is to learn the distribution parameters $z$ of the posterior MPD $q(\theta; z)$ that maximize the ELBO given in (3.33). The computation of the ELBO requires the likelihood function and the prior distribution. The likelihood function is chosen to be a Gaussian probability distribution of the form

$$p(J \mid \theta) = \mathcal{N}(0, s)\,, \tag{3.23}$$

where $s$ is the standard deviation of the likelihood and a hyperparameter. With the choice of the likelihood given in (3.23), maximizing the ELBO pulls the loss $J(\theta)$ to zero. On top of the neural net parameters $\theta$, we wish to learn the posterior over the standard deviation $s$ of the likelihood. Hence, the posterior distribution $q(\xi)$ is an MPD over the multivariate random variable $\xi$, where $\xi = [\theta, s]$ and $z$ is the distribution parameters over $\xi$.

**Remark 2.** *For continuous posterior distribution, the* ELBO *given in equation* (3.33) *is redefined using differential entropy, which expresses the prior and posterior in terms of their probability density functions (PDF). In this case, the likelihood in* (3.23) *is also a PDF and the* ELBO *is not bounded by zero.*

We take two approaches to selecting the distribution parameters $z_0$ of the prior. The

simplest approach is to use an uninformed prior with randomly initialized $z_0$ to start the optimization problem; this choice encourages exploration but has slow convergence properties. The second approach uses an informed prior that warm-starts the Bayesian training around the solution of the deterministic training. To do so, $z_0$ is selected such that the prior distribution is centered around the parameters learned from the deterministic technique discussed in Section **??**.

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

Additionally, we model measurement error by injecting noise into the prediction $\gamma$. This is achieved by replacing the dynamical constraint given in (3.8) with the following stochastic differential equation (SDE).

$$\mathrm{d}x = \left( \begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ G(q) \end{bmatrix} u^\theta(x) \right) \mathrm{d}t + \nabla_x u(x)\, \mathrm{d}W_t, \tag{3.24}$$

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

**Remark 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 $\theta$ 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 [5].*

This Bayesian framework extends the deterministic training process discussed in Section **??**, by replacing lines 7 to 10 of Algorithm **??** with Algorithm 5. Similar to the deterministic training, we use the adjoint method to compute the gradient $\partial\mathcal{L}/\partial z$ through the solution of the SDE given in (3.24). Moreover, we invoke the reparameterization trick of the Automatic Differentiation Variational Inference(ADVI) [20] to compute the gradient of samples $\theta$ with respect to the distribution parameters $z$.

**Algorithm 5** Bayesian Learning

1: $f \leftarrow f(H, G)$ dynamics given by SDE (3.24) with $u = u^\theta$
2: **for** each $x_0 \in$ `batch` **do**
3: Initialize $\mathcal{L} = 0$
4: **for** i=1:N **do**
5: $\theta, s \sim q(\theta; z)$ ▷ Sample parameters of $H_d^\theta$
6: $\zeta \sim \mathcal{N}(\zeta_0, \sigma_\zeta)$ ▷ Sample system parameters
7: $\gamma \leftarrow$ integrate $f$ from $x_0$
8: $p(J, \theta) = \frac{1}{s\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{J(\theta)}{s}\right)^2\right) p(\theta)$
9: $\mathcal{L} \leftarrow \mathcal{L} + \frac{1}{N}\left(\log[p(\ell, \theta)] - \log[q(\theta; z)]\right)$
10: $z \leftarrow z + \alpha_i \left(\partial\mathcal{L}/\partial z\right)$

### 3.3.2 Simple Pendulum

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

$$\mathrm{d}x = \begin{bmatrix} \dot{\vartheta} \\ a \sin\vartheta + u^\theta(x) \end{bmatrix} \mathrm{d}t + \nabla_x u^\theta(x)\, \mathrm{d}W_t, \tag{3.25}$$

where $a = mgl/I$, $\mathrm{d}W_t$ is the Wiener process, $x = (\vartheta, \dot{\vartheta})$ is the state of the pendulum where $\vartheta = 0$ corresponds to the upright position, and the control input $u$ is the torque generated by the actuator. The torque $u^\theta$ 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 value 9.81 $s^{-2}$. To this end, we learn a target function that directly parameterizes the controller $u^\theta$.

**Training**

Algorithm **??** is carried out with expert trajectories generated by the vanilla energy-shaping control (ESC) [21]. The ESC takes the general form

$$u^\theta(\vartheta, \dot{\vartheta}; \theta) = \theta_1 \dot{\vartheta} + \theta_2 \cos(\vartheta)\dot{\vartheta} + \theta_3 \dot{\vartheta}^3, \tag{3.26}$$

where the weights $\{\theta_i\}_{i=1}^3$ satisfy $-\theta_1 = \theta_2 = 2a\theta_3 < 0$ in the vanilla ESC.

The running cost function is chosen to be the loss $J_{track}(\gamma_\perp)$ from homoclinic orbit $\gamma^\star$ provided in (**??**); the corresponding likelihood is given by (**??**). We collect dataset $\mathcal{D}$ of initial states sampled from the expert trajectory, per the state sampling technique discussed in Section **??**. For this particular experiment, we use Hamiltonian Monte Carlo outlined in Algorithm 4 to infer the exact posterior distribution $p(\theta|\mathcal{D})$. We compare the Bayesian policy inferred from the procedure in Algorithm **??** with the deterministic policy, which is simply given by the point-estimates of the expert vanilla ESC in (3.26) [22].

**Simulation Tests**

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

We demonstrate the robustness of the Bayesian policy by comparing $\zeta$, as a function of the system parameter $a$, with that of the deterministic policy. We also investigate the effects of the prior distribution of $\theta$ on the performance of the controllers. In particular, we examine a uniform prior and a Gaussian prior centered around the deterministic solution of (3.26). The comparisons between the controllers from both cases are shown in Figure 3.4.

In addition to uncertainties in $a$, we test the deterministic and Bayesian policies with measurement noise, modelled as a Wiener process with variance 0.0005 rad in the $\vartheta$- and 0.05 ${}^{\text{rad}}\!/_{\text{s}}$ in the $\dot{\vartheta}$-directions. These numbers are chosen to represent a typical error arising from an optical encoder with a resolution of 2048 pulses per revolution and its naive differentiation via backward difference. To capture the influence of measurement noise, we generate 20 trajectories by integrating (3.25) from the same initial states. These trajectories are then used to compute $\zeta$. The effects of noise on $\zeta$ are reflected by the error bands in Figure 3.4.

The results show that Bayesian learning yields controllers that outperform their deterministic counterpart throughout the whole range of $a$. The marginalization method in (3.15) for selecting $\theta$ 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 3.4. This implies that the controllers produced by Bayesian learning

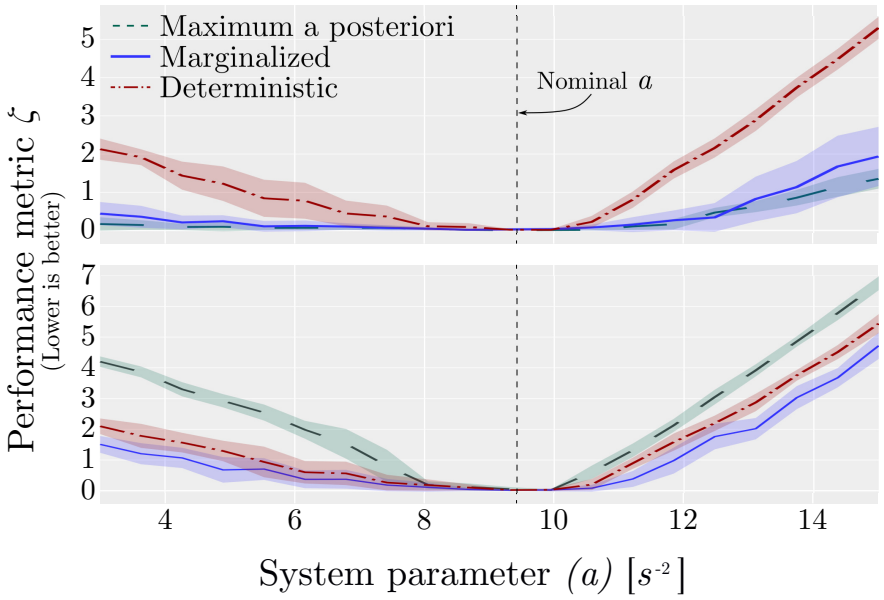



**Figure 3.4: Performance comparisons between deterministic and Bayesian learning methods. The training is initialized with a Gaussian prior (top), and a uniform prior (bottom). The continuous error band is generated by computing $\zeta$ from 20 trajectories of (3.25), starting at the downward equilibrium with a small disturbance. The solid lines represent the mean of $\zeta$. Best viewed in color.**

perform consistently better than the deterministic controller, demonstrating their robustness against model uncertainty and measurement noise.

### 3.3.3 Inertia Wheel Pendulum

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) with $n = 2$ and

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

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 can be written as

$$\mathrm{d}x = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dfrac{mgl\sin(q_1) - u^\theta - b_1\dot{q}_1}{I_1} \\ \dfrac{u^\theta - b_2\dot{q}_2}{I_2} \end{bmatrix} \mathrm{d}t + \nabla_x u^\theta(x)\,\mathrm{d}W_t, \tag{3.27}$$

where the control input $u^\theta$ 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 $H_d^\theta$ is a fully-connected neural network with two hidden layers, each with the ELU activation function [11]. A uniform distribution in $[-2\pi, 2\pi] \times [-2\pi, 2\pi] \times [-10, 10] \times [-10, 10]$ is chosen as the probability distribution from which samples of initial states $x_0$ are drawn for the DAGGER strategy. In each gradient descent step, we sample a batch of 4 initial states $\{x_0\}$ from $x_0 \sim \mathcal{N}(x^\star, \Sigma_0)$ and DAGGER as discussed in Section **??**;

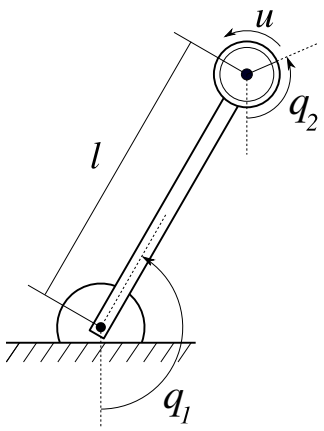



**Figure 3.5: Schematic of the inertia wheel pendulum. Only the joint $q_2$ is actuated, and $q_1$ is not.**

these initial states are integrated forward with a time horizon of $t \in [0, 3]$ seconds. In the Bayesian framework, the standard deviations $\sigma_\zeta$ of system parameters $\zeta = [I_1, I_2, mgl]$ are chosen to be 10% of the nominal system parameters given in Section **??**. Moreover, we train on trajectories per the SDE in (**??**) 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.

### 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

**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 |

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 3.6 shows the performance of deterministic and Bayesian trainings using an accumulated quadratic loss of the form

$$J^T = \frac{1}{2} \int_0^T \left( x^\top Q x + u^\top R u \right) dt. \tag{3.28}$$

The controller learned from the Bayesian training is marginalized over 10 parameters sampled from the posterior. As seen in Figure 3.6, 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 controller is more robust against measurement noise.

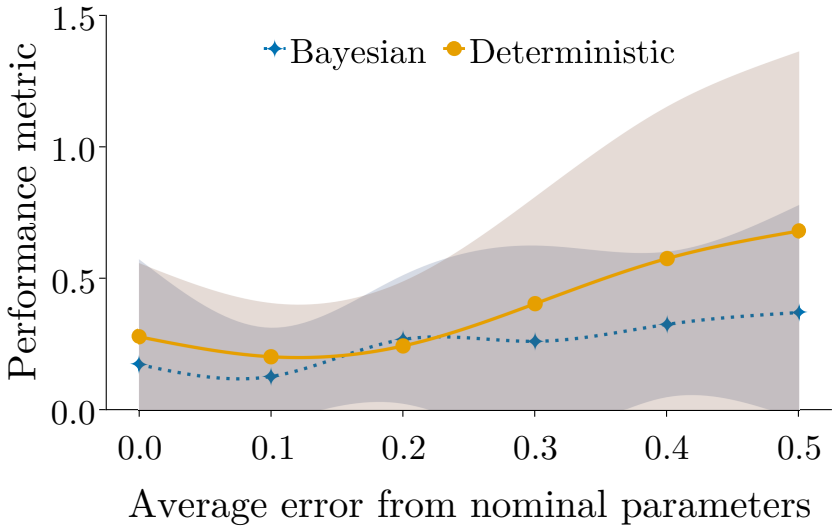



**Figure 3.6: NeuralPBC Performance metric ($J^T$) for various error in system parameters. Measurement noise included as Wiener process with standard deviation of 0.001 and 0.02 on joint angles and velocities, respectively**

**Table 3.2: System parameters used in real-world experiments. The errors in the last column are $\|p_s - p_s^{\mathbf{nom}}\|/\|p_s^{\mathbf{nom}}\|$**

| 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 |

### Hardware Tests

The controllers from deterministic and Bayesian training schemes are evaluated on hardware. 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.

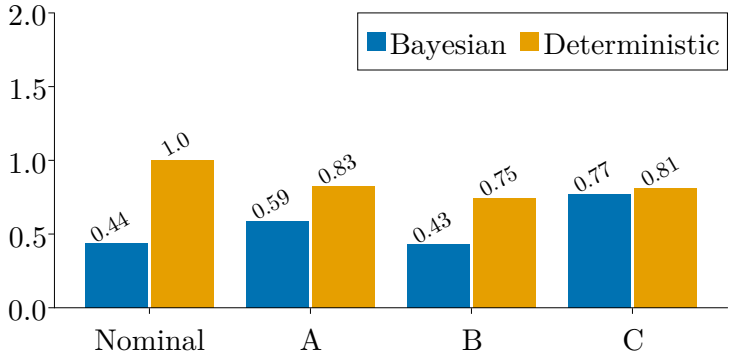



**Figure 3.7: Controller performance for modified system parameters. The performance metric is given by Eq. (3.28). Lower values are better. These results show that controllers trained via Bayesian learning are consistently more robust to errors in system parameters.**

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 (3.28) is the performance metric used to evaluate the controllers. The results are summarized in Figure 3.7.

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.

### 3.3.4 Control Design for Hybrid Dynamical Systems

#### 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 $(q^*, \dot{q}^*)$. 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.

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \int_0^T \ell\left(\phi, u^\theta(\phi)\right)\, \mathrm{d}t, \\ \text{subject to} \quad & M(q)\,\mathrm{d}\dot{q} + h(q,\dot{q},\theta)\,\mathrm{d}t - \mathrm{d}R = 0, \\ & u^\theta = -\Omega^\dagger(\nabla_q H_d^\theta - \nabla_q H), \end{aligned} \tag{3.29}$$

The performance objective $\ell$ is evaluated from closed loop trajectories generated from various initial states. We follow Moreau's time stepping algorithm [2] outline in Algorithm (1) to resolve the complementarity constraint in (2.2) and numerically integrate the measure differential equations (3.34).

We intend to find a solution to the optimization problem in (2.1) for all initial states $(q_0, \dot{q}_0)$ in the state space. To efficiently sample the initial states, we use a combination of greedy and explorative state sampling techniques. The greedy state sampling, commonly known as DAgger [7], generates trajectories using the current $H_d$ parameters and collects $N_d$ samples from the visited states. This allows the training to refine the controller on the states it visits the most. We recover from locally optimal solutions through explorative state sampling, which takes $N_r$ number of random samples from the state space. In a single batch training, we compute the performance objective as an expectation over $N_{\mathcal{D}} = N_d + N_r$ samples as follows:

$$J(\theta) = \mathbb{E}_{q_0,\dot{q}_0 \sim \mathcal{D}_N}[\ell(\phi(t; q_0, \dot{q}_0), u^\theta)]$$

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

As outlined in Algorithm 6, we compose the cost $\ell$ from the $N_{\mathcal{D}}$ closed loop trajectories and update the parameters $\theta$ through stochastic gradient descent (SGD). We compute the

gradient $\partial \ell / \partial \theta$ through auto-differentiation techniques.

**Algorithm 6** Solution to the Optimization Problem (3.29)

$\mathcal{D}_N \leftarrow \{q_0, \dot{q}_0\}_{(N_\mathcal{D})}$ ▷ $N_\mathcal{D}$ initial state samples
**while** $i <$ `maximum iteration` **do**
  $J \leftarrow 0$ ▷ Batch loss
  **for** $(q_0, \dot{q}_0) \sim \mathcal{D}_N$ **do**
    $\phi = \text{Moreau}(q_0, \dot{q}_0)$ ▷ Algorithm (1)
    $J \leftarrow J + \ell(\phi; \theta)/N_\mathcal{D}$ ▷ Batch loss
  $\theta \leftarrow \theta + \alpha_i \partial J / \partial \theta$ ▷ SGD step
  $\mathcal{D}_N \leftarrow \{q_0, \dot{q}_0\}_{(N_\mathcal{D})}$ ▷ New initial state samples
  $i \leftarrow i + 1$
**return** $\theta$

### Bayesian NeuralPbc

Walking machines such as the rimless wheel are bound to experience undetermined external forces from uneven terrain. If the robot does not respond to these external disturbances in real-time, they can cause underperformance or even instability. 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 6 is not sufficient. In the presence of high variance disturbances, the point-estimate parameters $\theta$ under domain randomization are prone to be biased [23]; for instance, if the uncertainty in the terrain elevation is large, the learned parameters $\theta$ 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 $\theta$ via Bayesian inference. Then we address the bias-variance trade-off problem by marginalizing the controller over the distribution of the parameters [5].

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 [18]. 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.

$$\begin{aligned} \underset{P(\theta)}{\text{minimize}} \quad & \int_0^T \ell\left(\phi, u^\theta(\phi)\right)\, \mathrm{d}t, \\ \text{subject to} \quad & M(q)\,\mathrm{d}\dot{q} + h(q,\dot{q},\theta)\,\mathrm{d}t - \mathrm{d}R = 0, \\ & u^\theta = -\Omega^\dagger(\nabla_q H_d^\theta - \nabla_q H), \\ & p_s \sim \mathcal{U}(p_{min}, p_{max}), \\ & \theta \sim P(\theta). \end{aligned} \tag{3.30}$$

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 elevation of the terrain under each spoke, which consequently randomize the gap $g_N$, pre-impact velocities $\dot{q}^-$ and contact forces between each spoke and the ground.

From Bayes' theorem, the posterior distribution $P(\theta|J)$ is a function of the prior distribution $P(\theta)$ and the likelihood function $P(J|\theta)$ [5].

$$P(\theta|J) = \frac{P(J|\theta)P(\theta)}{\int P(J|\theta)P(\theta)\, d\theta} \tag{3.31}$$

The prior can be used to inject any former knowledge about the controller. For instance, an informed prior can be a normal distribution whose mean is the solution to the optimization problem in (3.29). If there is no prior information, $P(\theta)$ can be set as a uniform distribution.

The likelihood function consists of the expected performance $J$ in the following form:

$$P(J|\theta) = \mathcal{N}(0, s), \tag{3.32}$$

where $s$ is the standard deviation of the likelihood.

There are various techniques to find the exact posterior distribution [5], but they have weak scalability for high-dimensional systems. Thus, we use variational inference, a technique that approximates the posterior with the distribution $Q(\theta; z)$ selected from the conjugate family of the prior [16]. The goal is to learn the distribution parameters $z$ of the chosen posterior. The performance objective that minimizes the loss $J$ while also matching the approximate posterior $Q$ to the exact one is

$$\mathcal{L}(J, z) = \mathbb{E}_{\theta \sim Q}[\log(P(J|\theta)P(\theta)) - \log(Q(\theta; z))] \tag{3.33}$$

where $\mathcal{L}(J, z)$ is the evidence lower bound (ELBO) [17].

As outline in Algorithm (7), we update the distribution parameters of the posterior $Q(\theta; z)$ through stochastic gradient descent. We use auto-differentiation techniques to take the gradient of the ELBO with respect to $z$. We invoke the reparameterization trick of the Automatic Differentiation Variational Inference (ADVI) [20] to compute the gradient of (3.33) through the samples drawn from the posterior.

### 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

**Algorithm 7** Solution to the Optimization Problem (3.30)

$\mathcal{D}_N \leftarrow \{q_0, \dot{q}_0\}_{(N_\mathcal{D})}$ ▷ $N_\mathcal{D}$ initial state samples
**while** $i <$ `maximum iteration` **do**
  $\mathcal{L} \leftarrow 0$ ▷ ELBO Loss
  **for** $i = 1 : Q_N$ **do** ▷ Samples to compute (3.33)
    $J \leftarrow 0$ ▷ Batch loss
    $\theta \sim Q(\theta; z)$ ▷ Sample parameters of $H_d^\theta$
    **for** $(q_0, \dot{q}_0) \sim \mathcal{D}_N$ **do**
      $p_s \sim \mathcal{U}(p_{min}, p_{max})$
      $\phi = \text{Moreau}(q_0, \dot{q}_0)$ ▷ 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
  $\mathcal{D}_N \leftarrow \{q_0, \dot{q}_0\}_{(N_\mathcal{D})}$ ▷ New initial state samples
  $i \leftarrow i + 1$
**return** $z$

motor mounted at the hip, which in turn propels the entire wheel. The angle of the torso is characterized by $\varphi$ measured from the outward normal of the runway shown as $\hat{n}$ in Figure **??**. The orientation of the wheel $\beta$ 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

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

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

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 $\Sigma_1$ shown in Figure **??**. The slope of the runway is given by $\gamma$, $g$ is the magnitude of acceleration due to gravity. The Euler-Lagrange equations corresponding to the Lagrangian $\mathcal{L} = \mathcal{K} - \mathcal{P}$ are

$$M(q)\,\mathrm{d}\dot{q} + h(q,\dot{q})\,\mathrm{d}t - \mathrm{d}R = 0, \tag{3.34}$$

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

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

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

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

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 [2] 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

$$J_T = \int_0^T \ell(\phi(t); \theta)\, \mathrm{d}t = \sum_{t=0}^{T} (\dot{x}^* - \dot{x}(t; \theta)) \tag{3.35}$$

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 $\mathcal{U} \sim [0, 3]$ 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 3.9, 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 3.10 shows that a large positive torque is applied if the wheel stumbles backwards.

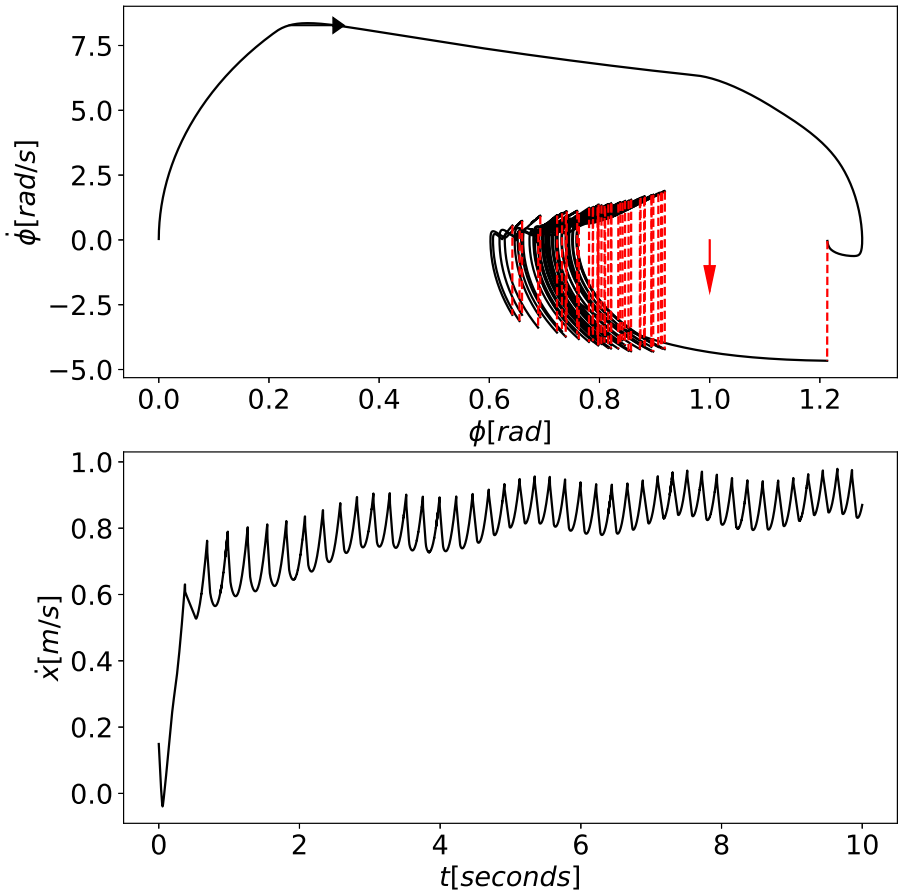



**Figure 3.9: Top: Torso orientation and velocity for 10-second trajectory. The solid black lines show the continuous phases and the dashed red lines show discrete transitions. Bottom: Horizontal hip speed**

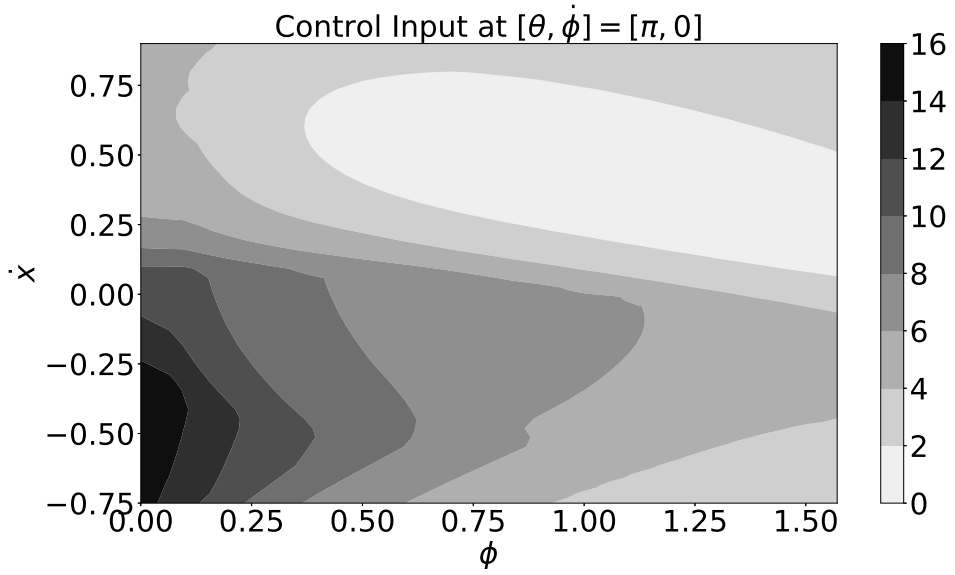



**Figure 3.10: Torque command to torso as a function of torso angle and horizontal hip speed**

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 $(q_0, \dot{q}_0)$ and integrate 10-second trajectories. The cost of each trajectory is calculated as per (3.35). The solid line in Figure **??** shows the average $J_T$ over the 10 trajectories for each $p_{max}$ and the error band represents the standard deviation over the accumulated losses.

### Real-World Experiments

We test the performance of both controllers on the hardware shown in Figure 3.11. 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 [24] 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 3.11: 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 $V_d^\theta$ and the entries of $L_\theta$ and $A_\theta$ matrices with Bayesian neural networks. The goal is to learn the distribution

parameters $z$ of the posterior multivariate probability distribution $q(\theta; z)$ that maximize the ELBO given in (3.33).

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

$$p(\|l_{1k,\mathrm{IDA}}(q)\|^2 + \|l_{2,\mathrm{IDA}}(q)\|^2 \mid \theta) = \mathcal{N}(0, s), \tag{3.36}$$

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 (3.36), maximizing the ELBO in (3.33) coaxes the loss $l_{\mathrm{IDA}}(x)$ to zero.

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

We update the distribution parameters $z$ along the gradient $\partial\mathcal{L}/\partial z$ until the ELBO converges and the objective function of (**??**) reaches the threshold $\epsilon_{tol}$. We invoke the reparameterization trick of the Automatic Differentiation Variational Inference(ADVI) [20] to compute the gradient of samples $\theta$ 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 $\zeta$ from a normal distribution $\mathcal{N}(\zeta_0, \sigma_\zeta)$ centered around the nominal parameter $\zeta_0$, where $\sigma_\zeta$ represents the uncertainty in system parameters. Each time we compute the PDE loss $l_{\text{IDA}}$ for a batch of discrete states sampled from the configuration space $\mathcal{Q}$, we draw a new sample of $\zeta$.

### 3.4.1 Inertia Wheel Pendulum

#### Training

The optimization problem (3.12) is constructed as follows. The potential energy function $V_d^\theta$ 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 $L_\theta$ are part of the parameters $\theta$ 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) [25] 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 $\sigma_\zeta$ of system parameters $\zeta = [I_1, I_2, mgl]$ are chosen to be 10% of the nominal system parameters given in Section **??**. 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.

**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 |

### 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 **??** shows the performance of the controllers. The policy learned from the Bayesian training is marginalized over 10 parameters sampled from the posterior per (3.15).

As seen in Figure **??**, 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.

### 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 (3.28) is used to evaluate the controllers. The results are summarized in Figure 3.13.

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 **??**.

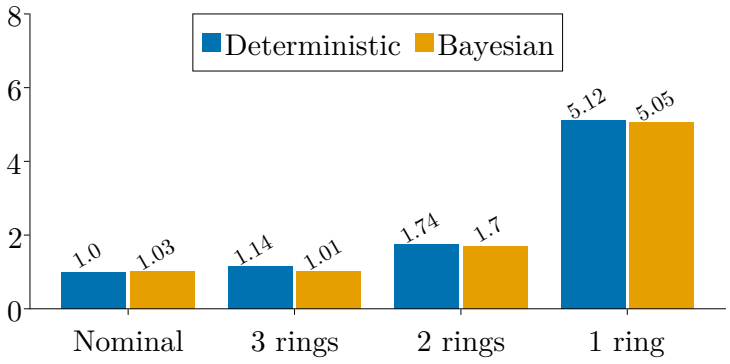



**Figure 3.13: Normalized accumulated cost $J_T$ (lower is better) for modified system parameters. The categories A-C correspond to the parameters shown in Table 3.2.**

## 3.5 Conclusion

,í„ì„ê,

,í„ì„ê„í„ì„ê,

# BIBLIOGRAPHY

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

[2] 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.

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

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

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

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

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

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

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

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

[11] 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.

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

[13] 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.

[14] 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.

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

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

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

[18] 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.

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

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

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

[22] 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.

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

[24] 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.

[25] 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 (3.22) of the SDE (3.21) expression into the performance measure (3.18) yields

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

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

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

□