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

ABSTRACT

We provide several data-driven control design frameworks for contact-rich robotic systems. These systems exhibit continuous state flows and discrete state transitions, which are governed by distinct equations of motion. Hence, it is difficult to design a single policy that can control the system in all modes. Typically, hybrid systems are controlled by multi-modal policies, each manually triggered based on observed states. However, as the number of potential contacts increase, the number of policies can grow exponentially and the control-switching scheme becomes too complicated to parameterize. To address this issue, we design contact-aware data-driven controllers given by deep-net mixture of experts. This architecture learns switching-control scheme that can achieve the desired overall performance of the system, and a gating network, which determines the region of validity of each expert, based on the observed states

Additionally, we address the adverse effects of model uncertainties in the control of contact-rich robots. Lack of accurate environmental models can misrepresent the effects of contact forces on the system. Policies designed from such models can lead to poor performance or even instability. In particular, we demonstrate the effects of system parameter uncertainties and measurement errors on the overall performance of the system. Then, we design data-driven stochastic controllers that combine the stability properties of passivity-based control with the robustness properties of Bayesian learning.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1:

INTRODUCTION

Many robotics applications consist of hybrid systems that exhibit both continuous state flows and discrete state transitions. Common examples of hybrid systems are contact-rich mechanisms such as legged robots and manipulators. These mechanisms experience contact forces from their interaction with the environment, causing them to undergo mode changes. For example, a humanoid bipedal robot consists of two potential contacts between the legs and the ground. If we only observe the contact forces exerted on one of the legs, we can find a total of three modes [2]. In the first mode, called the *swing mode*, the leg swings forward in the air while balancing off of the other; this phase is governed by a continuous dynamics with no contact forces on one of the legs. Then follows the *heel strike mode*, where the leg impacts the ground, causing a discrete state transition. Lastly, the leg goes into the *stance mode*, where it leverages the contact force and friction from the ground to balance the rest of the mechanism. Each one of these modes have a distinct dynamic behavior governed by unique equations of motion.

Controlling such multi-modal systems comes with two main complications. First, it may be impossible to find a single policy that can achieve the desired performance in all modes of the hybrid system. In these scenarios, we can detect event triggers during mode changes and switch between several controllers. An example of such technique is fuzzy control [3, 4], which is commonly used to generate if-then rules for the control of bipedal

robots. However, such techniques do not scale well to contact-rich systems with numerous modes. An example of such system is multiagent manipulation [5], which uses a group of robots to cooperatively execute a task, such as maneuvering an object in space. As we change the number of robots establishing contact with the object, we find new modes of the hybrid system. A cooperative manipulation with k potential contacts can have up to 2^k contact combinations. It is quite tedious, and in some cases impossible, to parameterize the effects of these contact combinations and find individual controllers for each mode. Moreover, it is difficult to parameterize the relationship between the states and the conditions that trigger the control switch. To address this issue, we propose a data-driven framework that learns a mixture of expert controllers for contact-rich systems. This technique infers the experts and a gating network, which determines the control switching scheme based on observed states. Moreover, we learn optimal expert controllers by injecting a performance objective that characterizes the desired behavior of the resulting closed-loop system.

The second complication in the control of hybrid systems is that they operate in an environment that is not known completely or modeled accurately. For instance, a legged robot needs to be robust enough to be able to perform satisfactorily on uneven terrain. Similarly, manipulators need to hold a firm grip on objects of all textures and shapes. There are techniques that combine tools from optimization, probability theory, and machine learning to learn control strategies from inaccurate system models or even unknown dynamics. Model-free reinforcement learning is an example of a technique that relies on repeated interactions with the unknown environment [6, 7, 8]. While this technique offers more flexibility on how the control policies are inferred from unknown dynamics, they do not provide the physical structure required to infer stability properties. On the other hand, data-driven techniques trained in simulation, such as neural passivity-based control (NeuralPbc [9]

and Neural-Idapbc [10]) offer more insight on the stability of the system but strongly rely on the dynamical model. The use of inaccurate models may lead to poor performance or even instability.

Bayesian learning (BL) [11, 12] offers an alternative method to simultaneously combat model uncertainties while preserving the useful stability properties in the NeuralPbc and Neural-Idapbc frameworks. BL is typically used to characterize uncertainties of a dynamical system with a stochastic model. For instance, the framework presented in [13] models uncertainties caused by disturbances, such as the effect of wind gusts on quadcopters, via Bayesian inference. A similar approach is shown in [14, 15], where a stochastic dynamical model is constructed via BL techniques, and utilized in data-driven control synthesis executed in simulation. Adaptive control framework is provided in [16], where the search for the control is given by a quadratic program that imposes Lyapunov stability constraint for safety critical systems. This technique uses BL to infer a controller through interactions with unknown dynamics, while maintaining the algebraic structure of a stable system. Inspired by this technique, we merge the structure and stability properties of passivity-based control (PBC) with the robustness properties of BL.

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

ing machine that still represents the difficulties in controlling hybrid systems in uncertain environments.

CHAPTER 2:
SWITCHING CONTROL WITH DEEP-NET MIXTURE OF EXPERTS

2.1 Background

In this section, we provide a brief introduction to contact modeling with the linear complementarity formulation. We also present a summary of the mixture of experts architecture and its uses in machine learning. In Section 2.3, we utilize the high-fidelity models obtained from the linear complementarity formulation in order to design contact-aware mixture of expert controllers through data-driven techniques.

2.1.1 Contact Modeling with Linear Complementarity Problem

Suppose a hybrid dynamical system consists of k potential contacts, 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. The contact forces in this hybrid system enforce the geometric and kinematic constraints of surfaces in contact. For instance, the contact force between two objects in collision characterizes the no-penetration conditions and the post-impact velocities of the objects. An accurate contact model identifies the contact forces necessary to obey these kinematic constraints, however resolving the 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 [17] 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. The linear complementarity formulation in [18] provides a rigorous technique to resolve contact forces, Coulomb friction and impact forces in a hybrid system. This formulation 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 from kinematic and dynamic constraints of contact, which we discuss as follows. We begin by introducing the variables necessary to define the kinematic constraints of a contact-rich system. Suppose we have a contact-rich mechanism whose states $x \in \mathcal{X} \subset \mathbb{R}^{2m}$ consists of generalized positions $q \in \mathbb{R}^m$ and velocities $\dot{q}$. Let $g_N(q) \in \mathbb{R}^k$ denote a vector of gap functions that measure the normal distance between the contact surfaces. The normal and tangential relative velocities between the contact surfaces are given by $\gamma_N = \dot{g}_N(q)$ and γ_T, respectively. We can express the relative velocities γ_N and γ_T in terms of the generalized velocities as

$$\gamma_N = W_N \dot{q},\ \gamma_T = W_T \dot{q}, \tag{2.1}$$

where W_N and W_T are the linear mappings from the generalized coordinates to the local contact coordinates. The kinematic constraints of contact provide a connection between pre- and post-impact velocities throughout this formulation, thus we introduce the following notations. The pre- and post-impact generalized velocities are denoted by $\dot{q}^-$ and $\dot{q}^+$,

respectively, and their corresponding relative velocities are given by

$$\gamma_N^+ = W_N \dot{q}^+,\ \gamma_T^+ = W_T \dot{q}^-,$$
$$\gamma_N^- = W_N \dot{q}^-,\ \gamma_T^- = W_T \dot{q}^-.$$

The kinematic constraint of contact states that two rigid bodies undergoing contact must always maintain a normal distance such that $g_N \geq 0$. Moreover, in the presence of contacts, the post-impact velocities can be found from the pre-impact velocities as:

$$\begin{aligned} \gamma_N^+ &= -\epsilon_N \gamma_N^-, \\ \gamma_T^+ &= -\epsilon_T \gamma_T^-, \end{aligned} \tag{2.2}$$

where $\epsilon_N \in \mathbb{R}^k$ and $\epsilon_T \in \mathbb{R}^k$ are diagonal matrices consisting of the normal and tangential coefficients of restitution, respectively.

The dynamic constraints of contact-rich system are given by the model [18]

$$\begin{aligned} M(q)\,\mathrm{d}\dot{q} + h(q,\dot{q})\,\mathrm{d}t - \mathrm{d}R &= 0, \\ h(q,\dot{q}) &= C(q,\dot{q})\dot{q} + G(q) - Bu(q,\dot{q}), \end{aligned} \tag{2.3}$$

where $M \in \mathbb{R}^{m\times m}$ 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 $u \in \mathcal{U} \subset \mathbb{R}^n$ to the generalized coordinates. The force measure $\mathrm{d}R$ contains the contact forces as

$$\mathrm{d}R = W_N\,\mathrm{d}\lambda_N + W_T\,\mathrm{d}\lambda_T,$$

where W_N and W_T are the projection matrices that map the effect of the normal and tangential contact forces, respectively, to the generalized coordinates. The vectors $\mathrm{d}\lambda_N$ and $\mathrm{d}\lambda_T$ consist of the normal and tangential contact impulse measures, respectively. In the presence of impacts, we integrate the contact measures over a singleton time t as $\int_{\{t\}}(\mathrm{d}\lambda_N, \mathrm{d}\lambda_T) = (\lambda_N(t), \lambda_T(t))$ in order to obtain the impulsive contact forces. In the case of persisting contact forces, the contact impulse measures evaluate to $(\mathrm{d}\lambda_N, \mathrm{d}\lambda_T) = (\dot{\lambda}_N, \dot{\lambda}_T)$, where $\dot{\lambda}_N$ and $\dot{\lambda}_T$ hold the normal and the tangential contact forces, respectively. The dynamic constraint in (2.3) characterizes how the local contact forces affect the dynamics of the overall system.

Remark 1. *Unlike the classical second order equations of motion $M(q)\ddot{q} + h(q, \dot{q}) = 0$, the measure differential inclusion in* (2.3) *can characterize the behavior of the system under impact forces. Notice that in the presence of impacts, the velocity $\dot{q}$ is not continuous for all time, thus the acceleration $\ddot{q}$ does not exist everywhere. For further understanding of how the measure equality can accurately represent the impact dynamics, we refer the reader to [19].*

We first motivate the linear complementarity formulation for normal contact forces, where the system has no Coulomb friction. The linear complementarity formulation imposes a unilateral constraint between contact forces and relative velocities given by:

$$\begin{aligned} &0 \leq \xi_N(q, \dot{q}) \perp \lambda_N \geq 0, \\ &\xi_N(q, \dot{q}) := \gamma_N^+ + \epsilon_N \gamma_N^-. \end{aligned} \tag{2.4}$$

where $0 \leq a \perp b \geq 0$ denotes $a \geq 0, b \geq 0$ and $a^\top b = 0$. The physical interpretation of (2.4) can be given as follows. In the presence of contacts and impacts, the post-impact velocities can be found from pre-impact velocities by (2.2). In this scenario, the quantity ξ_N evaluates

Table 2.1: Linear Complementarity Formulation: Possible contact scenarios

Scenario	g_N	γ_N^-	γ_N^+	ξ_N	λ_N
No contact	$g_N \leq 0$	$\gamma_N > 0$	γ_N	$\gamma_N + \epsilon_N \gamma_N$	0
Contact or impact	$g_N \leq 0$	$\gamma_N \leq 0$	$-\epsilon_N \gamma_N$	0	$\lambda_N \geq 0$

to zero, because

$$\xi_N = \gamma_N^+ + \epsilon_N \gamma_N^- = -\epsilon_N \gamma_N^- + \epsilon_N \gamma_N^- = 0.$$

The complementarity constraint in (2.4) states that when two surfaces come into contact, the resultant between pre- and post-impact velocities, given by ξ_N must be zero, and in the meantime, the contact forces can take positive values. Conversely, if there are no potential contacts, the relative velocities are continuous, i.e. $\gamma_N^+ = \gamma_N^- = \gamma_N$. In this case, the complementarity constraint states that ξ_N can be non-zero, which is analogues to contact surfaces approaching each other or moving away from each other, but the normal contact forces λ_N must be zero. There exists no scenario when both the contact force λ_N and ξ_N are both positive, hence $\xi_N^\top \lambda_N = 0$ must always hold. This concept is summarized as shown in Table 2.1.

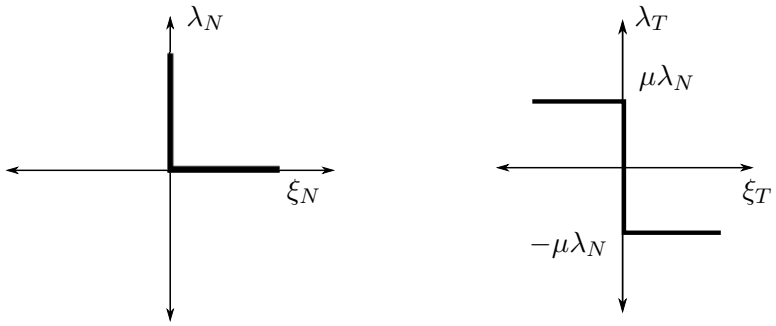


Figure 2.1: Left: Complementarity condition for normal contact forces. Right: Characteristics of Coulomb friction as a function of ξ_T

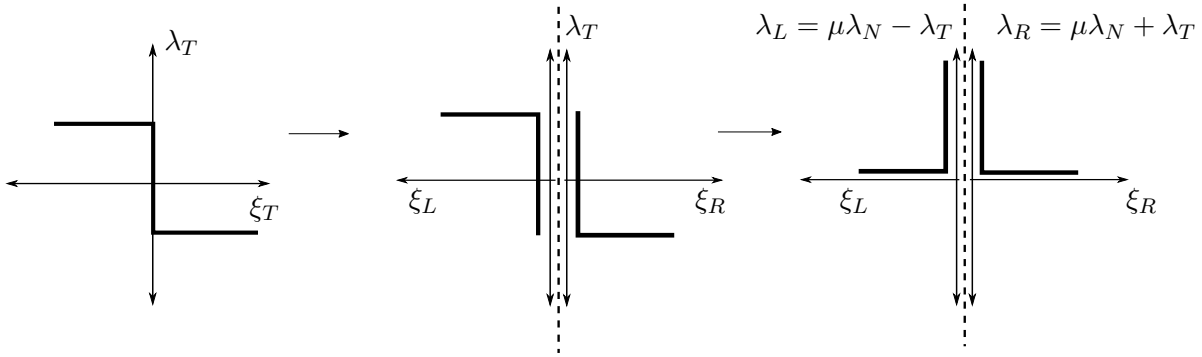


Figure 2.2: Construction of complementarity condition for Coulomb friction

From (2.1) and (2.3), we can express ξ_N as

$$\begin{aligned}
\xi_N &= W_N^\top \dot{q}^+ + \epsilon_N W_N^\top \dot{q}^-, \\
&= W_N^\top M^{-1}[-h\Delta t + W_N \lambda_N] + (1+\epsilon_N) W_N^\top \dot{q}^-, \\
&= \underbrace{W_N^\top M^{-1} W_N}_{A_N} \lambda_N \underbrace{- W_N^\top M^{-1} h\Delta t + (1+\epsilon_N) W_N^\top \dot{q}^-}_{b_N}, \\
&= A_N \lambda_N + b_N,
\end{aligned}$$

where $\dot{q}^-$ and $\dot{q}^+$ represent the pre- and post-impact velocities of the system, and Δt is the integration time-step for the discretization of (2.3). Notice that ξ_N is a linear function of the contact forces λ_N, which allows us to pose the search for λ_N as the following quadratic optimization problem:

$$\begin{aligned}
\underset{\lambda_N}{\text{minimize}} \quad & (A_N \lambda_N + b_N)^\top \lambda_N, \\
\text{subject to} \quad & A_N \lambda_N + b_N \geq 0, \lambda_N \geq 0.
\end{aligned} \tag{2.5}$$

If the linear complementarity problem (LCP) in (2.5) has a feasible solution, the objective

function evaluates to zero.

With this understanding in mind, we extend the complementarity condition in (2.4) to a system with normal and tangential contact forces. Unfortunately, the complementarity relationship between λ_N and ξ_N given in Table 2.1 does not directly translate to the tangential components λ_T and ξ_T. To best explain the reason, consider the contact forces applied on a box sliding on a flat surface with Coulomb friction. The properties of the contact forces exerted on the sliding box are depicted in Figure 2.1. On the left, we provide a visual representation of the complementarity constraint between λ_N and ξ_N. On the right, we have the relationship between Coulomb forces λ_T and ξ_T. The right figure shows that if the box is sliding to the right ($\xi_T > 0$), the tangential force acts to the left, resisting the motion of the box. If the box moves to the left, the tangential forces apply resistive force to the right. The maximum λ_T applied on the system is $\mu\lambda_N$, where μ is the coefficient of friction. Notice, the relationship between λ_T and ξ_T is not complementary. However, the plot of λ_T can be split into two components λ_R and λ_L as shown in Figure 2.2, which individually resemble the complementarity properties of λ_N and ξ_N in Figure 2.1. The new quantities λ_R and λ_L are defined as [18]

$$\lambda_R := \mu\lambda_N + \lambda_T,$$
$$\lambda_L := \mu\lambda_N - \lambda_T,$$

and the corresponding complementarity condition becomes

$$0 \leq \begin{pmatrix} \xi_R(q, \dot{q}) \\ \xi_L(q, \dot{q}) \end{pmatrix} \perp \begin{pmatrix} \lambda_R \\ \lambda_L \end{pmatrix} \geq 0, \tag{2.6}$$

where $\xi_T = \xi_R - \xi_L$.

From the dynamics in (2.3), ξ_N, ξ_R and ξ_L can be expressed as an affine function of the contact forces λ_N, λ_R and λ_L. This allows us to express the complementarity constraints in (2.4) and (2.6) as a quadratic function of the contact forces. The affine function that relates ξ_N, ξ_R, ξ_L with $\lambda_N, \lambda_R, \lambda_L$ is given by [18]:

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

where the values of A and b are extracted from the dynamics. We refer the reader to [18] for detailed derivation of the forms of A and b, and present the results as

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

where I_k is the $k \times k$ identity matrix, and Δt is the integration time step. The linear complementarity problem (LCP) (2.4) can be posed as the following feasibility problem:

$$0 \leq \begin{bmatrix} A \begin{pmatrix} \lambda_N \\ \lambda_R \\ \xi_L \end{pmatrix} + b \end{bmatrix} \perp \begin{pmatrix} \lambda_N \\ \lambda_R \\ \xi_L \end{pmatrix} \geq 0, \tag{2.7}$$

which we can solve for $\lambda_N, \lambda_R, \xi_L$ with various optimization techniques.

We follow Moreau's time stepping algorithm [18] outlined in Algorithm (1) to numerically integrate the dynamics (2.3). In this procedure, we first use the kinematics of the

system to evaluate the position vector q after a half-time step as

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

This allows us to check the gap functions for possible penetration between contact surfaces at time $t + {}^{\Delta t}/_{2}$. If all the gap functions are positive, we determine that there are no contact forces applied. On the other hand, if any of the gap functions are non-positive, we compose a complementarity constraint for the active contacts as given in (2.7). 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 [20] to find the solution to the linear complementarity problem (2.7). The fact that Lemke's algorithm may be automatically differentiated to provide the gradients of the pertinent variables allows us to seamlessly integrate the solution to the differential equations into machine learning algorithms. We substitute the contact forces λ_N, λ_T provided by Lemke's algorithm into the equations of motion (2.3) to compute the post-impact velocities as follows:

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

This procedure is repeated for time horizon T.

2.1.2 Mixture of Expert Models

The mixture of experts (MoE) framework is a technique primarily used to learn an ensemble of regression models (experts) that best fit high variance or multi-modal datasets, such as the one shown in Figure 2.3a [1]. This technique provides a way to train several specialized expert models simultaneously, where each expert is well curated for a cluster of

Algorithm 1 Moreau's Time Stepping Algorithm

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

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

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

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

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

  $\lambda_N, \lambda_T \leftarrow \text{Lemke}(q(t_M), \dot{q}(t))$ ▷ Lemke [20]

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

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

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

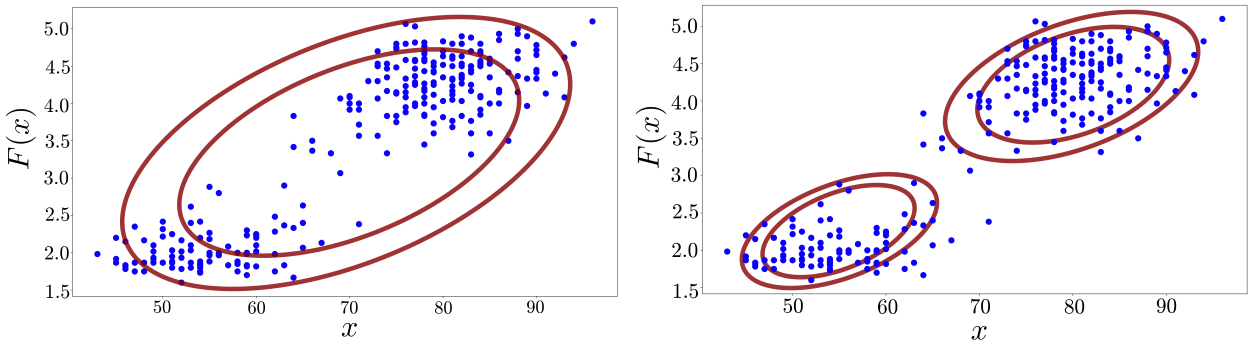
return ϕ

datasets as seen in Figure 2.3b. The MoE architecture uses a routing function called a *gating network* to allocate the appropriate local expert for each input data [21]. The objective is to learn the parameters of each local experts and the gating network to best fit the dataset.

Let $F(x;\theta)$ denote a collection of N_F expert models $F(x;\theta) := \{F_1(x;\theta_1), \ldots, F_{N_F}(x;\theta_{N_F})\}$, whose parameters are given by the set $\theta = \{\theta_1, \ldots, \theta_{N_F}\}$. The gating network is responsible for dividing the input space $\mathcal{X} \subset \mathbb{R}^m$ into *state partitions*, and assigning local expert models capable of providing specialized predictions for each partition. We represent the gating network with the discrete probability distribution $\mathbf{P}(x|\psi) := (P_1(x|\psi), \ldots, P_{N_F}(x|\psi))$, where $P_i(x|\psi)$ denotes the probability of state x belonging to the state partition $\mathcal{X}^i \subset \mathcal{X}$ with the index $i \in \{1, \ldots, N_F\}$. In the standard MoE framework [22], the prediction $u(x)$ of the MoE is given by

$$u(x) = \sum_{i=1}^{N_F} F_i(x;\theta_i) P_i(x|\psi), \tag{2.8}$$

which requires evaluating all the experts for each input x. We can reduce the computation cost of (2.8) by utilizing the output of the single best expert as determined by the gating



(a) Multi-modal dataset fit with one model **(b) Multi-modal dataset fit with MoE**

Figure 2.3: Comparison of multi-modal dataset fit with one regression model and MoE

network[23]

$$u(x) = \{F_a(x; \theta_a) \mid a = \underset{i}{\text{argmax}}\{P_i(x|\psi)\}\}. \tag{2.9}$$

Model Structure: The expert models and the gating network can take several forms. Gaussian process (GP) models are commonly used in the MoE framework to infer a multi-modal probabilistic model from a small amount of data [21]. Despite the expressive power and tractability of GP experts, the inference procedure requires repeated matrix inversions that scale cubically with the size of the dataset [24]. In order to circumvent the large computational and memory overhead while also preserving the expressive power of GP experts, we leverage the universal approximation capabilities of neural networks for both the experts and the gating network. For regression problems that require the flexibility of nonlinear models, the experts can be given by deep neural-nets with point-estimate parameters, which can be extended to probabilistic models with the use of Bayesian neural networks, whose weights and biases are given by probability distributions [25]. Similarly, the gating network can be given by a neural network $\mathbf{P}(x|\psi) : \mathcal{X} \to \mathbb{R}^{N_F}$ with parameters ψ, and the output

corresponds to the vector $[P_1(x|\psi), \ldots, P_{N_F}(x|\psi)]$. In order to ensure that the probabilities $P_i(x|\psi)$ over all state partitions i sum to one, we use the SOFTMAX activation function [26] on the last layer of the gating network.

Training: Given the training dataset $\mathbb{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ with N input state-label pairs, we can use gradient-based techniques to find the optimal parameters (ψ, θ) that best fit the dataset [23]. In such techniques, we construct the cost function we wish to minimize as

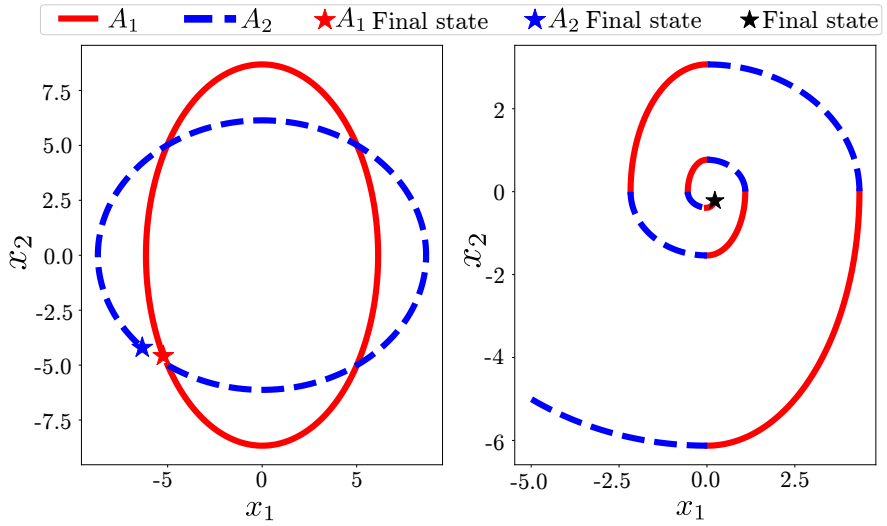
$$\mathbb{L}(\mathbb{D}) = \sum_{j=1}^{N} \sum_{i=1}^{N_F} \|F_i(x_j; \theta_i) - y_j\| \; P_i(x_j, \psi), \tag{2.10}$$

where $\|F_i(x_j; \theta_i) - y_j\|$ is the error in the prediction made by the expert i. Notice that the cost function (2.10) is minimum when the parameter θ_i has the lowest prediction error and the highest probability of getting selected by the gating network. So long as the complexity of the experts and the cost function allow for the pertinent gradients $\partial\mathbb{L}/\partial\psi$, $\partial\mathbb{L}/\partial\theta$ to be evaluated, we can invoke stochastic gradient descent (SGD) to update the decision parameters as follows:

$$\psi \leftarrow \psi - \frac{\partial \mathbb{L}}{\partial \psi},$$
$$\theta \leftarrow \theta - \frac{\partial \mathbb{L}}{\partial \theta}.$$

2.2 Motivating Application: *Switching Linear System*

In the following discussion, we present an example to motivate and lay the foundation for the use of MoE in the control design problem. In particular, we propose a data-driven technique to automatically seek switching controllers for multi-modal systems. Suppose we



(a) Two marginally stable closed-loop systems **(b) Asymptotically stable switching**

Figure 2.4: Stable switching between two marginally stable systems

have two linear systems of the form

$$\begin{aligned} \dot{x} = A_1 x &= \begin{bmatrix} 0 & -1 \\ 2 & 0 \end{bmatrix} x, \\ \dot{x} = A_2 x &= \begin{bmatrix} 0 & -2 \\ 1 & 0 \end{bmatrix} x, \end{aligned} \tag{2.11}$$

where each system is marginally stable as shown in Figure 2.4a. Although the individual systems are not asymptotically stable, it is possible to find a state-dependent switching rule that makes the resulting switched system stable[27] (Figure 2.4b). We aim to learn a gating network $\mathbf{P}(x|\psi)$ to automatically divide the state space into partitions and identify which of the two systems to execute in each state partition, with the goal of asymptotically stabilizing the origin.

Akin to the regression problem in Section 2.1.2, the training dataset consists of *input state-label* pairs, where the labels are the performances of the trajectories generated under the current control law. In the case of the switching-control problem, we generate a trajectory and the corresponding performance metric (labels) as follows. Starting from some initial state $x(t=0)$, we sample a state partition index i from the categorical distribution, whose probabilities are provided by the gating network:

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

Given the partition index i, the expert (control law) is given by a sample from the Bernoulli probability distribution

$$F_i(\theta_i) = \begin{cases} 0, & \theta_i > \frac{1}{2}, \\ 1, & \theta_i \leq \frac{1}{2}, \end{cases} \tag{2.12}$$

where $F_i = 0$ corresponds to the first dynamics $\dot{x} = A_1 x$ and $F_i = 1$ corresponds to $\dot{x} = A_2 x$. The parameter θ_i of the expert is to be learned, and it determines which of the two experts to execute in each partition. In order to ensure that the parameter θ_i of the expert serves as the probability of the Bernoulli distribution, we use the SIGMOID function[26] to limit θ_i between 0 and 1. The next state $x(t + \Delta t)$ in the trajectory is obtained from the following integration scheme:

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

We repeat this process to generate a trajectory for the time-horizon T. The performance of

the trajectory generated under the current parameters (ψ, θ) can be quantified by the metric ℓ as

$$\ell(x(t+\Delta t)) := \frac{1}{2}\|x(t+\Delta t)\|^2.$$

In Section 2.3.1, we generalize the performance metrics to be applicable to various dynamical systems and discuss how we can encode desired characteristics of the controller. From the performance metric ℓ, we can construct the cost function $\mathbb{L}$ similar to the standard MoE framework in (2.10) as

$$\mathbb{L}\Big(\{x(0), \ldots, x(T)\}\Big) = \sum_{t=0}^{T} \sum_{i=1}^{N_F} \ell_i\Big(x_i(t+\Delta t)\Big)\ P_i\Big(x(t), \psi\Big).$$

In the upcoming sections, we generalize the MoE control-search problem and provide techniques to efficiently learn the optimal decision parameters from appropriate cost functions.

2.3 Mixture of Expert Controller

Based on the motivating example provided in Section 2.2, we present a generalized data-driven control design framework for hybrid dynamical systems. In this framework, the controller is given by deep-net mixture of experts $F(x;\theta)$, and the control switching scheme is governed by the gating network $\mathbf{P}(x|\psi)$. This technique allows us to observe the effects of mode changes from the closed-loop trajectories and learn a switching mechanism to best control the hybrid system across modes. The objective is to learn the parameters θ_i of each expert and the gating network ψ that can achieve the desired performance.

Let $\phi(x_0, u, T)$ denote a closed-loop trajectory generated from a hybrid dynamical model starting from initial state x_0. We represent the dynamics of a hybrid system with the

differential inclusions[**?**]

$$\begin{cases} \dot{x} \in f(x,u), & x \in C, \\ x^+ \in g(x,u), & x \in D, \end{cases} \tag{2.13}$$

where $x \in \mathbb{R}^m$ is the state vector, and $u \in \mathbb{R}^n$ is the input. The set-valued mappings $f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ denote the flow and jump maps, respectively, where C and D are subsets of $\mathbb{R}^m$ consisting of the feasible states under the flow and jump rules, respectively. The notation x^+ indicates the state resulted by the jump rule g.

For every state x in a trajectory, the control law first samples state partition index i from a categorical distribution and evaluates the corresponding expert as

$$u(x;\psi,\theta) = \{F_i(x;\theta_i) \mid i \sim \text{Categorical}(\mathbf{P}(x|\psi))\}. \tag{2.14}$$

We use the metric $\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ to measure the performance of the sampled experts, which we discuss in depth in Section 2.3.1. The goal is to learn the decision parameters (ψ, θ) that minimize the metric ℓ for all initial states in the state space. We pose the search over the parameters of the experts and the gating network as the following optimization problem.

$$\begin{aligned} \underset{\psi,\theta}{\text{minimize}} \quad & \int_0^T \ell(x(t),u)\,\mathrm{d}t, \\ \text{subject to} \quad & \begin{cases} \dot{x} \in f(x,u), & x \in C, \\ x^+ \in g(x,u), & x \in D, \end{cases} \\ & u = \{F_i(x;\theta_i) \mid i \sim \text{Categorical}(\mathbf{P}(x|\psi))\}. \end{aligned} \tag{2.15}$$

In Section 2.3.3, we provide a procedure to solve the optimization problem (2.15) via stochastic gradient descent.

Remark 2. *Without prior knowledge injected to the gating network, the samples from the categorical distribution in* (2.14) *initially explore the performance of most, if not all, of the expert controllers. As the parameters converge to their optimal values, the samples from the categorical distribution correspond to the indices of the single best experts and the control law in* (2.14) *is equivalent to* (2.9)*.*

2.3.1 Performance Objective

We present two viable choices for the running cost function.

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

$$\ell(x, u) = \frac{1}{2}(x - x^*)^\top \mathcal{Q}(x - x^*) + \frac{1}{2}u^\top \mathcal{R}u, \tag{2.16}$$

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

We modify the likelihood function $\mathbb{L}$ constructed in (2.10) to incorporate the accumulated loss $\ell(x, u)$. The quadratic loss in (2.16) plays the role of prediction error in the likelihood function. Similar to the regression problem provided in Section 2.1.2, we want the likelihood to be maximum when the running cost incurred by expert i is low and the responsibility of the expert is high. We can obtain this characteristics by

Algorithm 2 Accumulated Loss

Input: x_0, θ, ψ

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

designing the likelihood as

$$\mathbb{L}(\phi) = \sum_{t=0}^{T} \sum_{i=1}^{N_F} -\ell\Big(x(t + \Delta t), F_i\Big) P_i\Big(x(t)|\psi\Big). \tag{2.17}$$

Algorithm 2 outlines how we construct the likelihood from a trajectory. In this procedure we check the performance of each expert at every integration step. To do so, starting at initial state x_0, we integrate the dynamics in (2.3) for one time step Δt using all the experts in $F(x_0; \theta)$. We retrieve all the states $\{x_i(\Delta t), i \in \{1, \dots N_F\}\}$ obtained from the integration and evaluate the running cost $\ell(x_i(\Delta t), F_i)$ incurred by each expert. Each running cost $\ell(x_i(\Delta t), F_i)$ is weighed by its *responsibility* $P_i(x_0|\psi)$ and summed across experts to get the likelihood. Notice that by checking the running cost of each expert for every state, the computation of the likelihood is prone to the curse of dimensionality. We minimize the amount of computation needed to compose the likelihood by selecting one state from the collection $\{x_i(\Delta t), i \in \{1, \dots N_F\}\}$ to continue the integration. We select the expert responsible for generating the next state $x(\Delta t)$ from the categorical distribution (**??**). This process is repeated for every time step in the trajectory.

2. **Minimum trajectory loss (MTL)**: In some cases, accumulated loss may not reflect the desired behavior of the 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 construct the minimum trajectory loss (MTL), which is composed of the lowest cost incurred across the entire trajectory and the responsibilities of the experts that led to the minimum cost. The resulting likelihood $\mathbb{L}$ is given by

$$\begin{aligned} t_{min} &= \inf_t \{\ell(x(t), u) : x(t) \in \phi(x_0, u, T)\} \\ \mathbb{L}(\phi) &= -\frac{\ell(x(t_{min}), u)}{C} \sum_{t=0}^{t_{min}} P_i(x(t)|\psi) \end{aligned} \tag{2.18}$$

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 reach the desired state within the time horizon T. Moreover, MTL does not need to evaluate the performance of each expert at every integration step, as shown in Algorithm (3); this greatly reduces the computational cost.

2.3.2 State Sampling

We intend to find a solution to the optimization problem in (2.15) for all initial states x_0 in the state space. To do so, we generate the running cost ℓ from *a batch of initial states* and update the parameters (ψ, θ) *iteratively* via stochastic gradient descent. We need to

Algorithm 3 Minimum Trajectory Loss

Input: x_0, θ, ψ

$\phi \leftarrow \{x_0\}$

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

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

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

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

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

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

return $\mathbb{L}$

collect the batch of initial states in an efficient manner. It is computationally expensive to merely discretize the vast state space into batches of initial states. 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 [28]. This method collects states most visited under the current parameters (ψ, θ) and concentrates on refining the performance of the controller on these states. For instance, suppose we are solving the optimization in (2.15) to obtain a controller that swings up an underactuated simple pendulum to the upright. Initially, the parameters (ψ, θ) may result in a controller that swings the pendulum to the downward equilibrium irrespective of where it started. Thus, it is most efficient to first expose the training to the cost incurred by visiting the downward equilibrium. In this technique, we first sample several initial states randomly and generate trajectories using the current parameters. Then, we randomly select N_d samples from the states visited in the trajectory. At first, the N_d samples mostly consist of states near the downward equilibrium. As the parameter update continues, DAgger starts sampling states that are closer to the upright equilibrium. This efficient state exposition is pivotal for the convergence to the optimal parameters.

The explorative state sampling technique exposes the training to the rewards of approaching and remaining close to x^*. It also uses random sampling to explore new control strategies and recover from locally optimal solutions. This method collects N_r initial states around the neighborhood of the desired equilibrium by drawing samples from the normal distribution $x_0 \sim \mathcal{N}(x^*, \delta)$, whose mean is x^* and the standard deviation is a small constant δ. The combination of greedy and explorative state sampling techniques helps to quickly refine the performance of the current controller and converge to the desired behavior. In a single batch training, we compute the running cost as an expectation over $N_{\mathcal{D}} = N_d + N_r$ samples as follows:

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

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

2.3.3 Training Mixture of Expert Controller

We solve the optimization problem in (2.15) following the procedure outlined in Algorithm (4). At the beginning of the training, we add a batch of initial states x_0 to the collection $\mathcal{D}_N$. These initial states are obtained from the greedy and explorative state sampling techniques. For every initial state in the batch, we generate a trajectory for time horizon T. The running cost of these trajectories are given by the accumulated cost (Algorithm (2)) or minimum trajectory loss (Algorithm (3)), depending on the desired closed-loop behavior. We average the running cost of each of the trajectories and perform *expectation maximization* with respect to the decision parameters (ψ, θ). We invoke a variant of stochastic gradient descent known as ADAM [29] to efficiently train the parameters with adaptive learning rates α_i. We leverage auto-differentiation techniques to back-propagate on

Algorithm 4 Solution to the Optimization Problem (2.15)

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

the gradient of the likelihood with respect to the learned parameters. In particular, we use forward-mode auto-differentiation [30] to take the gradient through the trajectory generated from Moreau's integration scheme. Although not explored in this work, it is possible to design adjoint methods for hybrid systems to efficiently back-propagate on the likelihood through reverse-mode auto-differentiation techniques.

2.3.4 Back-propagation through Hybrid Dynamics

The training framework outlined (2.15) allows us to observe the effects of contacts in the closed-loop trajectories and infer a controller that either uses the contact to its advantage or minimizes its adverse effects. In this section, we look at the relevant parts of the back-propagation to give insight on how this is achieved. We also show that despite the state jumps in the hybrid dynamics, the derivatives involved in the back-propagation are well-defined.

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

respect to θ_i as

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

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

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

We can expand the gradient further as

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

where λ holds the contact forces. We can compute the 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, it may seem the derivative $\frac{\partial x_1}{\partial \lambda} \frac{\partial \lambda}{\partial \theta_i}$ does not exist 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 exists even if a state jump has occurred. This term is crucial in adjusting the decision parameters in response to how the contact force assists or inhibits the system. If the contact forces affect the *post-impact* velocity such that the resulting generalized coordinates are closer to the desired state x^*, then the gradient $\frac{\partial \ell}{\partial x_1}\frac{\partial x_1}{\partial \lambda}\frac{\partial \lambda}{\partial \theta_i}$ adjusts the parameter θ_i to favor states undergoing contact events. Indeed, we demonstrate this behavior in simulation and real-world experiments in Section 2.4.2. 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 4.1.3.

2.4 Experimental Results

We demonstrate the efficacy of the mixture of expert controller in simulation and real-world experiments. In the first case study, we learn a gating network that switches between two unstable closed-loop systems to result in a piecewise-stable system. Then, we find switching MoE controller to swing up the classical cartpole mechanism enclosed with wall barriers.

2.4.1 Stable Switching Between Unstable Systems

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

$$\begin{aligned} \dot{x} = A_1 x = \begin{bmatrix} 0 & -1 \\ 2 & 0 \end{bmatrix} x, \\ \dot{x} = A_2 x = \begin{bmatrix} 0 & -2 \\ 1 & 0 \end{bmatrix} x. \end{aligned} \tag{2.19}$$

Even if the individual systems are not stable, it is possible to find a state-dependent switching rule that makes the resulting switched system stable [27]. We find the stable switching scheme through the MoE framework discussed in Section 2.1.2. We aim to learn the parameters ψ of the gating network $\mathbf{P}(x|\psi)$ and the expert parameters θ_i such that the switching system converges to the desired equilibrium $x^* = (0, 0)$. The gating network is a fully-connected neural net with one hidden layer (2 input states $\rightarrow$ 6 neurons $\rightarrow$ 4 outputs) and an ELU activation function [31]. We constrain the maximum number of state partitions to 4. Each state partition has a corresponding controller parameter $\theta_i \in \mathbb{R}$. The control law is a sample from the Bernoulli probability distribution

$$F_i(\theta_i) = \begin{cases} 0, & \theta_i > \frac{1}{2}, \\ 1, & \theta_i \leq \frac{1}{2}, \end{cases} \tag{2.20}$$

where $F_i = 0$ corresponds to the first dynamics $\dot{x} = A_1 x$ and $F_i = 1$ corresponds to $\dot{x} = A_2 x$. We use the SIGMOID activation function to limit θ_i between 0 and 1.

We generate a trajectory following the procedure in Algorithm (5). For the initial state x_0, we sample a state partition i from the categorical distribution (**??**). Then, we sample from the Bernoulli distribution F_i in (2.20) using the parameter θ_i corresponding to the sampled state partition i. The expert F_i executes one of the two closed loop systems. We repeat this process for time horizon T.

The likelihood in (2.17) is used as the performance objective with $\mathcal{Q} = 3I_2, \mathcal{R} = 0$, where I_2 is 2×2 identity matrix. We use forward-mode auto-differentiation in combination with a variant of stochastic gradient descent known as ADAM [29] to compute the gradient $\partial\mathbb{L}/\partial\psi, \partial\mathbb{L}/\partial\theta_i$ and update the decision parameters. Notice that when evaluating the gradient of $\partial\mathbb{L}/\partial\theta_i$, the derivative $\partial F_i(\theta_i)/\partial\theta_i$ is not well-defined. Thus, we explicitly introduce the parameter

Algorithm 5 Stable Switching between Unstable Systems

Input: $x(0)$

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

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

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

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

  $x(t+\Delta t) = (1-F_i)A_1x(t) + F_i\ A_2x(t)$

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

return ϕ

θ_i into the likelihood without disrupting the structure of the original likelihood (2.17) as follows:

$$\mathbb{L}(\phi) = \sum_{t=0}^{T}\sum_{i=1}^{N_F} -\Bigg[\theta_i \ell\Big(x(t+\Delta t), F_i(\theta_i)\Big) + (1-\theta_i)\ell\Big(x(t+\Delta t), F_i(1-\theta_i)\Big)\Bigg] P_i\Big(x(t)|\psi\Big).$$

The response of the learned switching system is shown in Figure 2.5. The training uses only 3 out of the 4 state partitions available. The state partition in Figure 2.5 matches the analytical solution to the stable switching system given as [27]

$$\dot{x} = \begin{cases} A_1x, \ x_1x_2 \leq 0, \\ A_2x, \ x_1x_2 > 0, \end{cases}$$

where $x = [x_1, x_2]$. The training progress is shown in Figure 2.6. The three rows in the figure depict the performance of the training after 0, 200 and 1400 parameter updates, respectively. The sampled trajectory on the top left figure shows that the initial parameters created unstable switching between the two systems. After only few parameter updates, the training finds a stable switching mechanism, but it does not yet converge to the desired

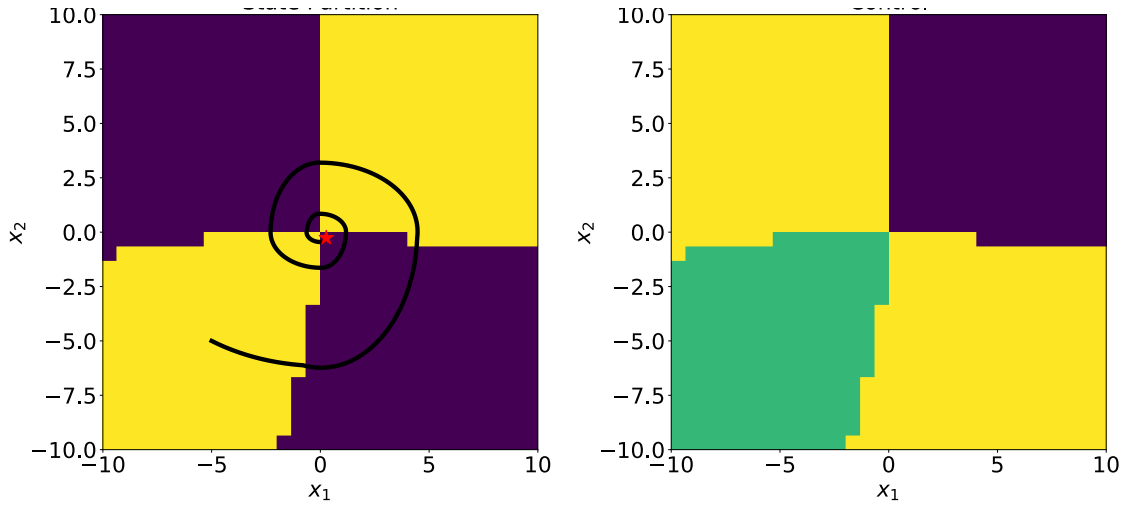


Figure 2.5: Final stable switching system. Left: Control input $F_i = \{0, 1\}$ in the state space. Purple corresponds to $F_i = 0$ or $\dot{x} = A_1 x$. Yellow corresponds to $F_i = 1$ or $\dot{x} = A_2 x$. The trajectory starts at $x_0 = [-5, -5]$ and converges to the origin shown by the red star. Right: State partition from the output of the gating network. The training converged to 3 state partitions that can stabilize the system.

equilibrium x^*. The explorative state sampling technique visits states close to the desired equilibrium, allowing the training to learn the distinct boundaries of each partition at the origin.

2.4.2 Cartpole with Wall Contacts

In this section, we take the classical cartpole swing-up problem and introduce potential contacts from two barriers as shown Figure 2.7. The potential contacts serve as a way to convert the standard cartpole system into a multi-modal dynamics. The objective is to swing-up the pendulum on the cart in the presence of contacts and impacts. We apply the MoE framework to train switching expert controllers and a gating network that governs the switching scheme. We demonstrate the performance of the mixture of expert controller in simulation and real-world experiments. Lastly, we compare the performance of the MoE

controller against a single swing-up controller.

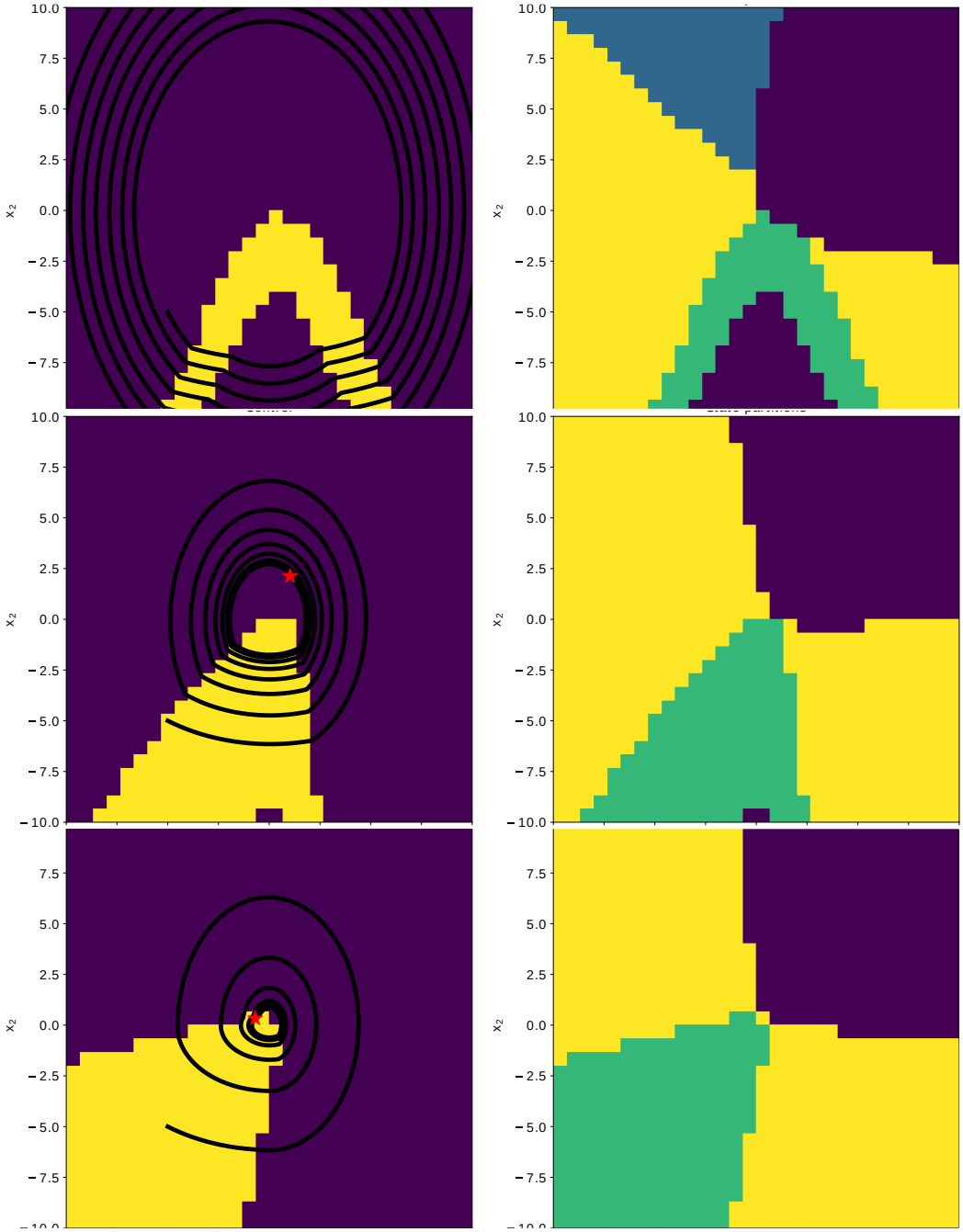


Figure 2.6: Training progress. Top row: the control and state partition at the initial parameters. Second row: after 200 parameter updates. Third row: after 1400 parameter updates. The final solution is shown in Figure 2.5

System Model

The cartpole system consists of a freely rotating pendulum link hinged on an actuated cart. The setup is enclosed by two rigid walls hanging 0.2m from the bottom of the cart. The objective is to use the control authority on the cart in order to swing-up the passive pendulum to the upright. The pendulum spans length of $l = 0.2$m and its mass $m_p = 0.75$kg is concentrated at the distance $l_{cm} = {}^{l}/_{2}$ from the hinge. The cart alone has a mass of $m_c = 0.165$ kg. The viscous friction in the cart wheels is characterized by the coefficient $b = 1.2\ {}^{\text{N}\cdot\text{sec}}/_{\text{m}}$. The dynamics of the system is given by (2.3) where

$$
\begin{aligned}
M(q) &= \begin{bmatrix} m_c + m_p & -m_p + l_{cm}\cos(\theta_p) \\ -m_p l_{cm}\cos(\theta_p) & m_p l_{cm}^2 + I_p \end{bmatrix}, \\
C(q,\dot{q}) &= \begin{bmatrix} b & m_p l_{cm}\sin(\theta_p) \\ -m_p \sin(\theta_p)/2 & 0 \end{bmatrix}, \\
G(q) &= \begin{bmatrix} 0 & -m_p g l_{cm}\sin(\theta_p) \end{bmatrix}^\top, \\
B &= [1\ 0]^\top,
\end{aligned}
\tag{2.21}
$$

where $q = (x_c, \theta_p)$, x_c is the location of the cart, θ_p is the angle of the pendulum from the vertical. The moment of inertia of the pendulum is given by I_p and g is the acceleration due to gravity. There are a total of $k = 10$ potential contacts between the pendulum and the sides of the walls. We integrate closed-loop trajectories with Moreau time stepping algorithm outlined in Algorithm (1) with an integration time step $\Delta t = 0.001$.

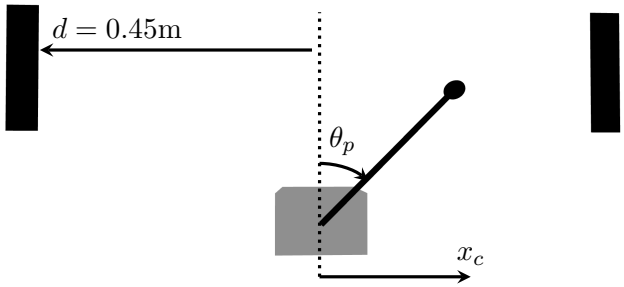


Figure 2.7: Cartpole with wall contacts

Training

We aim to learn the parameters of the experts θ_i and the gating network ψ that swing up the cartpole to the desired state $x^* = (q^*, \dot{q})^* = (0, 0)$ under contacts and impacts. Once the system reaches within a small neighborhood of x^*, we employ Linear Quadratic Regulator (LQR) to stabilize at the desired equilibrium. The gating network is a fully-connected neural net with two hidden layers (state input $\rightarrow$ 4 neurons $\rightarrow$ 3 neurons $\rightarrow$ 3 output). We constrain the maximum number of state partitions to 3. There are a total of 3 experts F_i, each corresponding to a state partition. The experts are fully-connected neural nets with two hidden layers (state input $\rightarrow$ 10 neurons $\rightarrow$ 4 neurons $\rightarrow$ 1 output). The output of the experts correspond to the force applied on the cart. We use minimum trajectory loss (MTL) discussed in Section 2.3.1 with time horizon $T = 1.5$s. In each parameter update, we sample $N_{\mathcal{D}} = 4$ initial states through greedy and explorative techniques.

Results

We demonstrate the performance of the MoE controller in simulation and hardware. The hardware is set up as shown in the Figure 2.8. The cart uses a rack and pinion mechanism

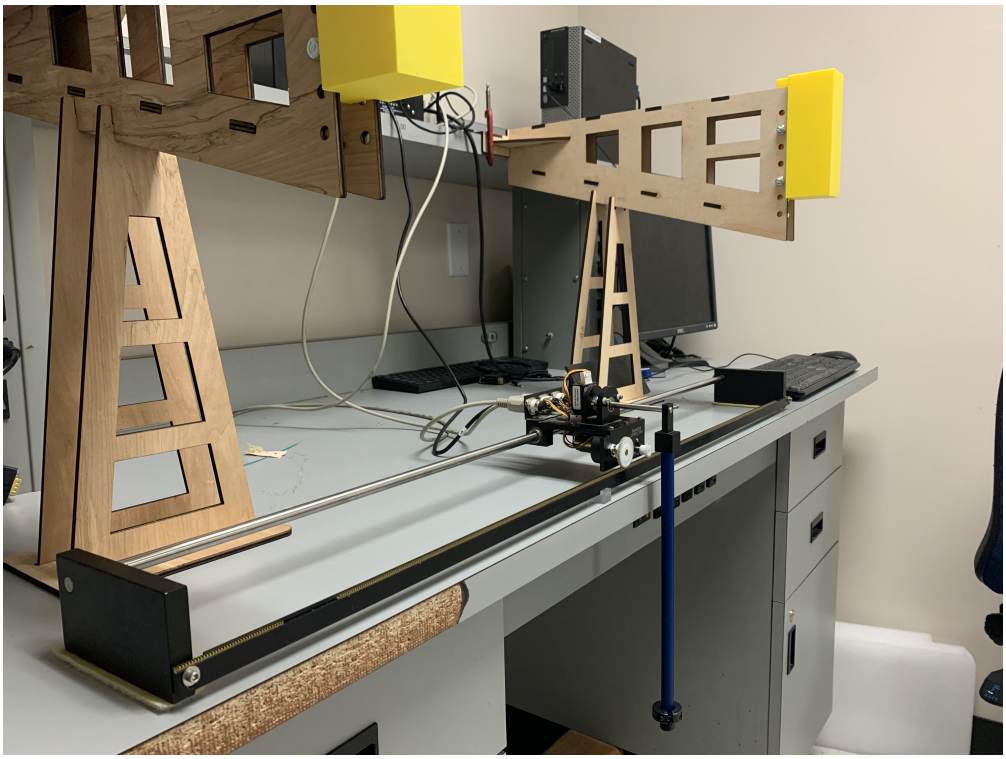


Figure 2.8: Experimental setup of cartpole with wall contacts

to translate on the track with zero-slip. One of the wheels of the cart is attached to an optical encoder, from which we can estimate the position and velocity of the cart. There is also an optical encoder rigidly attached to the pendulum link, reporting its orientation. We use MATLAB/Simulink to evaluate the neural nets and pass voltage commands to the DC-motor. We convert the force commands output by the MoE controller into voltage commands $V(t)$ as follows.

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

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

Figure 2.9 shows a trajectory generated by the MoE controllers in simulation and hard-

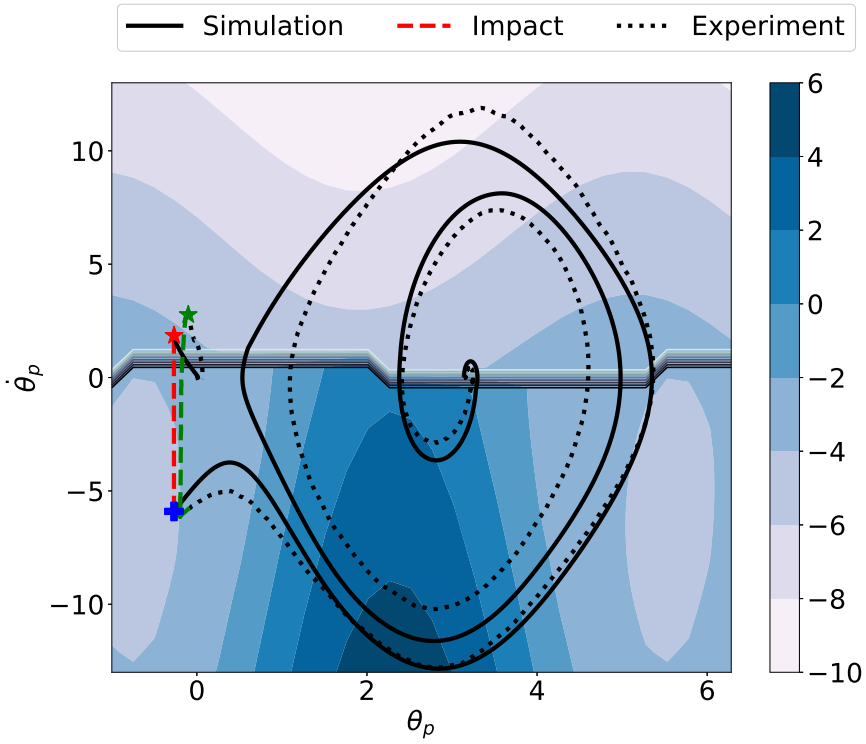


Figure 2.9: A sample trajectory starting from downward equilibrium at rest. The contours correspond to the outputs of the MoE controller during impact ($x_c = 0.36m, \dot{x}_c = 0^m/_s$). The solid line splitting the state space horizontally in half shows the boundary between the two state partitions. The state partition shows that the controller must switch from one expert to another in order to catch the pendulum post-impact.

ware. The blue contours represent the level sets of the control input at the pre-impact and post-impact states. The solid line splitting the state space horizontally in half shows the boundary between two state partitions. Even though the gating network can provide up to three state partitions, the training converges to utilizing only two. Figure 2.9 shows that the system successfully avoids contacts during the swing-up phase, which otherwise would have prevented the pendulum from pumping energy from the downward equilibrium. By the time the pendulum approaches the upright equilibrium, it is moving at such high speed that LQR

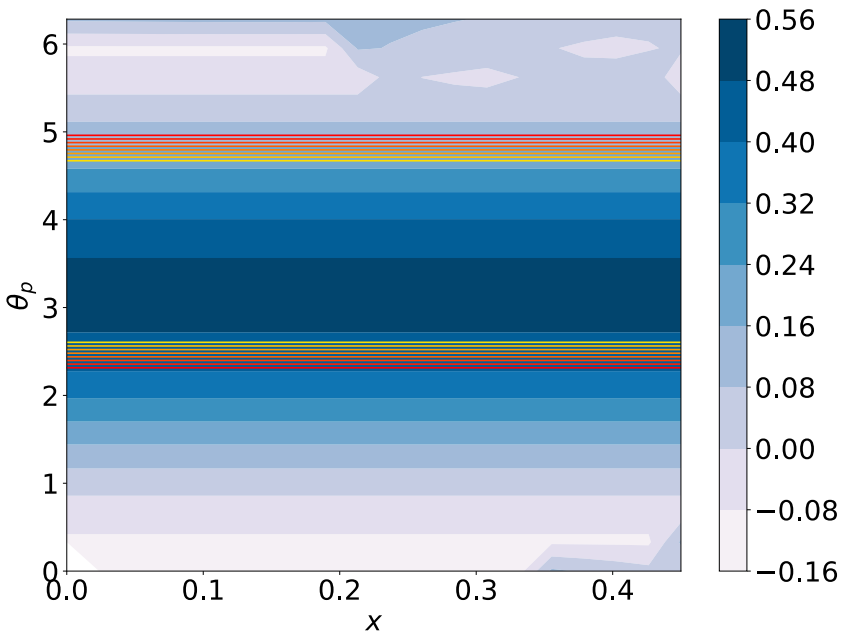


Figure 2.10: The blue contours represent the level sets of the gap function. The red lines show the boundaries of the state partition. There are a total of two experts, one is active between the red boundaries and the other is responsible for regions outside the red lines.

cannot stabilize it. However, we have observed from several trajectories that the system leverages the impact from the wall to lower the speed of the pendulum. Then, the control law switches between experts as the velocity jumps due to impact. The magnitude of the control is such that the MoE controller applies rapid braking during contact, allowing LQR to catch the pendulum post-impact. The MoE controller achieves successful swing-up in simulation and real-world, proving the accuracy in the contact modelling and the robustness of the controllers.

Figure 2.10 shows the relationship between the selected expert and the occurrence of potential contacts. The red lines represent the boundaries of the state partitions and the contour depicts the level sets of the shortest gap between the pendulum and the walls. Notice

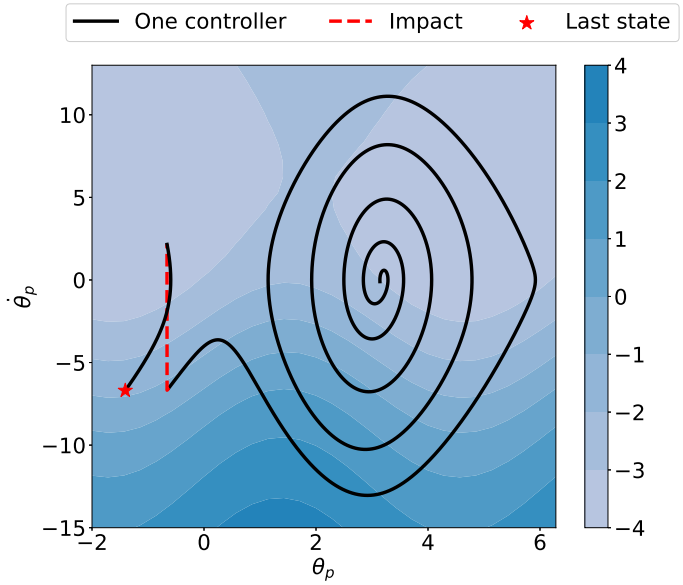


Figure 2.11: A sample trajectory generated by the single controller.

that the states within the boundaries of the red lines have large gap values, hence contact events do not occur in this region. This shows that the gating network has dedicated one expert for states with low risk of contact. The regions outside the red boundaries correspond to the second expert, which is activated to prevent contact during swing-up and to catch the pendulum just after impact, as shown in the trajectory of Figure 2.9. This demonstrates that the state partition depends on the occurrence of potential contacts. Moreover, the gating network dedicates an expert to leverage the advantages or prevent the adverse effects of contacts and impacts.

Lastly, we compare the performance of the MoE controller against a single controller. This controller is parameterized by a neural net (state input $\rightarrow$ 10 neurons $\rightarrow$ 4 neurons $\rightarrow$ 1 output) similar to one of the mixture experts. We train the controller with the same minimum trajectory loss and training parameters as the MoE. Once the controller swings the pendulum to the neighborhood of x^*, we use LQR to stabilize it to the upright. As shown in

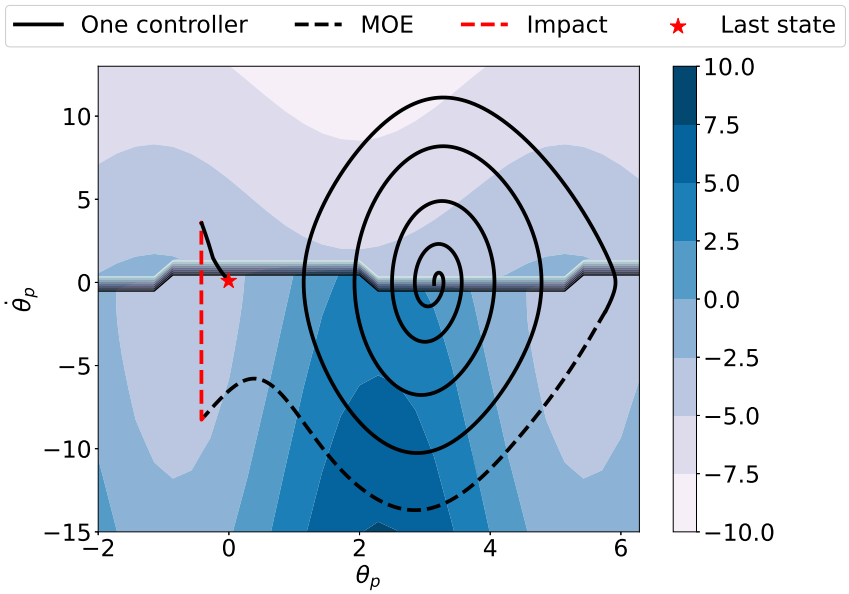


Figure 2.12: The performance of the MoE controller compared to the single controller in Figure 2.11

.

Figure 2.11, the single controller successfully swings up the pendulum closer to the upright. However, the controller is not able to catch the pendulum post-impact, because the velocity of the pendulum is still too high for LQR to stabilize it. On the other hand, Figure 2.12 shows the performance of the MoE controller in the same scenario. The MoE solution leverages the switching controllers to apply rapid braking post-impact, which lowers the velocity of the pendulum. This assists the LQR in catching the pendulum at the appropriate speed. This demonstrates the advantages of switching controllers in presence of multi-modal contact-rich systems.

2.5 Conclusion

In this chapter, we provide a data-driven control design that reasons about the effects of contact forces on the hybrid system. We incorporate accurate system model in the training

via linear complementarity formulation, and infer mixture of expert controllers. The learning framework also provides a gating network, which selects an expert for every observed state. From simulation and real-world experiments, we demonstrate that the learned policy leverages the advantages of contact in some states and minimizes its adverse effects in others. Moreover, the gating network is able to characterize the states prone to contact, and selects the expert trained to react to the outcomes of potential contacts.

CHAPTER 3:

UNCERTAINTY HANDLING VIA NEURAL BAYESIAN INFERENCE

3.1 Background

In this section, we provide a brief summary to passivity-based control and the recent data-driven variant our group has introduced in [9, 10, 32]. We also motivate the advantages of the Bayesian learning framework in uncertainty modeling and robust control.

3.1.1 Passivity-Based Control (PBC)

Suppose we have a robotic system whose Hamiltonian $H : \mathcal{X} \to \mathbb{R}$ can be expressed as

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

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

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

where $x = (q, p)$, $\Omega(q) \in \mathbb{R}^{m \times n}$ is the input matrix, and $u \in \mathbb{R}^n$ is the control input. A mechanical system is considered passive if it is dissipative with respect to a *storage function* H, 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 states $x(t_0)$ and all input u under the supply rate $s = u^\top y$ [33]. There exists a strong connection between the passivity and stability properties of a dynamical system. The system (3.2) has an asymptotically stable equilibrium at the origin if it is *strictly* passive, i.e.,

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

We refer the reader to [34] for a detailed proof on the connection between passivity and stability.

The objective of passivity-based control (PBC) is to design a control law u that imposes the desired storage function H_d on the closed-loop system, rendering it passive and therefore stable [33]. The dynamics of the closed-loop system with the desired storage function H_d can be given as

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} \nabla_p H_d \\ -\nabla_q H_d \end{bmatrix}, \tag{3.4}$$

and it has a new desired stable equilibrium at x^*. From (3.2) and (3.4), we can find the

energy shaping control that results in a passive closed-loop system as

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

where $\Omega^{\dagger} = \left(\Omega^{\top}\Omega\right)^{-1} \Omega^{\top}$. To obtain an asymptotically stable system, we can add a damping term u_{di} to the control as follows.

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

where $K_v \succ 0$ is the damping gain matrix. The goal is to characterize the storage function H_d of the closed-loop system, whose desired equilibrium is at x^*, and to extract the energy-shaping control that morphs the open-loop system (3.2) to (3.4). To do so, we expose an inherent constraint on the form of H_d from (3.5) as

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

where $\Omega^{\perp}\Omega = 0$. Hence, we can obtain H_d from the solution to the partial differential equations (PDEs) in (3.7). In many cases, it is tedious and computationally expensive to extract H_d from the PDEs. Moreover, the closed-form solution to the PDEs may be intractable, especially in high-dimensional systems. A novel data-driven framework is presented in [9, 10, 32], where we find a solution to the PDEs in an iterative manner via stochastic gradient descent. We present a brief summary of this technique as follows.

Neural PBC

The deterministic NeuralPbc framework presented in [9] solves the PDEs (3.7) by rewriting the PBC problem as the following optimization scheme:

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

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

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

2. Desired system behavior is explicitly introduced into the optimization via the performance objective ℓ, which allows us to find a preferred solution to the PDEs.

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

Neural Interconnection and Damping Assignment PBC

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

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

where the minimum of the closed-loop potential energy $V_d(q)$ is at the desired equilibrium position q^*. The resulting passive closed loop system is given by [35]

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

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

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

The goal is to select V_d and the matrices M_d, J_2 that result in a passive closed-loop system. There are a set of PDEs constructed from (3.11) that constrain the forms of V_d, M_d, J_2, and they are given by

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

Similar to NeuralPbc, the closed-form solution to the PDEs in (3.12) may be intractable. The framework presented in [10] addresses this issue through the Neural-IdaPbc architecture, which we summarize as follows.

The deterministic NEURAL-IDAPBC framework introduced in [10] finds an approximate solution to the PDEs from the following optimization problem.

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

where V_d^{θ} and the entries of the M_d^{θ} and J_2^{θ} matrices are parameterized by neural networks. To enforce the constraints shown in (3.13), we redefine M_d^{θ}, J_2^{θ} and V_d^{θ} as follows. We rewrite the desired mass matrix M_d^{θ} using the Cholesky decomposition as

$$M_d^{\theta} = L_{\theta}(q)L_{\theta}^{\top}(q) + \delta_M I_n, \tag{3.14}$$

where $\delta_M > 0$ is a small constant and I_n is the $n \times n$ identity matrix. The matrix $L_{\theta} \in \mathbb{R}^{n\times n}$ is a lower-triangular matrix whose $n(n+1)/2$ entries are outputs of a neural network. The form of M_d^{θ} in (3.14) preserves the positive-definiteness of the mass matrix. The skew-symmetric matrix J_2 is constructed as

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

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

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

$$V_d^\theta(x) = \Phi\Big(W_j\sigma(W_{j-1}\sigma(\dots W_2\sigma(W_1x)))\Big), \tag{3.15}$$

where W_i holds the weights of layer i and σ is the activation function. The activation function is chosen such that $\sigma(0) = 0$, ensuring that $\Phi(0) = 0$ and $\Phi(x) > 0, x \neq 0$. Several choices of activation functions that satisfy these properties include ELU, TANH and RELU.

3.1.2 Bayesian Learning

Suppose we are given a finite dataset with inherent noise, for which we are trying to fit a regression model. Consider an example of recognizing handwritten digits from images. The task is to build a model that takes in input images and identifies the digits written on those images. We train the model on finite labelled dataset, which consists of inherent noise due to the differences in individual handwriting. In practical applications such as this one, the training data comprises only a fraction of all possible ways to write digits. However, we can still use machine learning techniques to recognize patterns in the strokes and make generalized predictions on new dataset. It is also most beneficial to report the uncertainty associated with each prediction in order to inform the user of outlier images present in the testing dataset. This can all be achieved by probabilistic learning, which uses the expressive power of *stochastic models* to generate generalized predictions from finite dataset and report the uncertainty in these predictions [1].

In the probabilistic learning framework, the stochastic models are constructed as follows. Let $F(x;\theta)$ denote the stochastic model whose parameters θ are multivariate random variables. A common approach is to pre-select the posterior probability distribution $P(\theta;z)$ over the parameters θ and learn the distribution parameters z that minimize the prediction

error $\sum_{(x_j,y_j)\in\mathbb{D}}\|F(x_j;\theta)-y_j\|$, where $\mathbb{D}=\{(x_1,y_1),\ldots,(x_N,y_N)\}$ is the training dataset with inputs x_j and observations y_j. *Expectation maximization* (EM) is a gradient-based technique commonly used to learn z that maximizes the likelihood function, and consequently minimizes the prediction error [1]. To do so, we define the likelihood in terms of the prediction error as

$$P(\mathbb{D}|\theta)=\prod_{j=1}^{N}\mathcal{N}(\|F(x_j;\theta)-y_j\|\mid 0,s),$$

where $\mathcal{N}$ is a Gaussian distribution with mean zero and standard deviation s. Depending on the complexity of the stochastic model, we can take analytical gradients of $P(\theta;z)$ with respect to z or leverage auto-differentiation techniques [30]. Once we find the optimal z values that maximize the likelihood, we can make predictions for new data inputs by drawing samples from the distribution $P(\theta;z)$ and marginalizing over the model as follows:

$$\hat{F}(x)=\frac{1}{N_\theta}\sum_{\theta\sim P(\theta;z)}F(x;\theta),$$

where N_θ is the number of samples drawn from $P(\theta;z)$. The uncertainty associated with each prediction is given by [25]

$$\Sigma_{F|x,\mathbb{D}}=\frac{1}{N_\theta-1}\sum_{\theta\sim P(\theta;z)}\left|\left|F(x;\theta)-\hat{F}(x)\right|\right|^2. \tag{3.16}$$

Bias-variance trade-off

Even though the EM approach finds optimal parameters that minimize the prediction error, this technique is prone to overfitting. Given finite number of observations, EM finds low variance distribution $P(\theta;z)$ *biased to the training data* [1]. There are two side effects

to such biased trainings. First, in an attempt to absolutely minimize the prediction error $\|F(x_j;\theta) - y_j\|$, the stochastic model can take a complex form and overfit to the training dataset. However, overfitting to the training data reduces the accuracy of the predictions made by the model. This is best shown in Figure 3.1, where the blue circles correspond to the training data and the red line depicts the average prediction made by the regression model $F(x;\theta)$. The right figure represents a complex model $F(x;\theta)$ that gives minimum prediction error on the training data but does not accurately represent the data source, which is given by the green curve. On the other hand, the left figure shows a regression model that has not overfit to the training dataset. By maintaining some minimal prediction error, the model manages to make better predictions on new datasets. Secondly, stochastic models are most useful to quantify the uncertainty in each prediction. When the model overfits to the training data, it collapses the posterior to a near-zero-variance distribution. In such cases, the prediction uncertainty in (3.16) collapses to zero. Hence, the model makes predictions with absolute certainty, overlooking to report that the overfit model makes inaccurate predictions. We call such overfit models with near-zero prediction uncertainty as *overconfident* models.

The solution to the overfitting problem involves finding a *bias-variance trade-off*, where the training adjusts the parameters based on the likelihood, but also enforces the posterior to hold some variance in order to prevent overfitting. Bias-variance trade-off is achieved with the introduction of a prior distribution, which adds a regularization term to the likelihood [1]. In supervised learning, regularization terms restrict the training from learning a complex model, minimizing the risk of overfitting [36]. Similarly, the prior distribution prevents the learned parameters from becoming *overconfident* in their predictions. In this construction, the posterior distribution $P(\theta|\mathbb{D})$ is defined in terms of the likelihood and prior with Bayes'

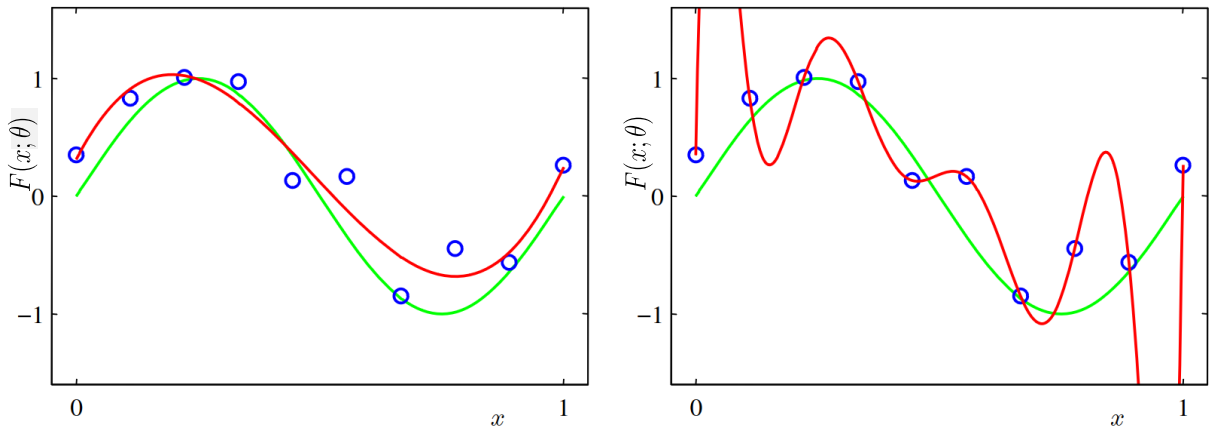


Figure 3.1: Comparison between model biased to the training data (right) and model that achieves bias-variance trade-off (left) [1]. Blue circles represent training data, the green curve is the original data source for which we are learning a regression model, the red line is the learned model $F(x;\theta)$

.

theorem as

$$P(\theta|\mathbb{D}) = \frac{P(\mathbb{D}|\theta)P(\theta)}{P(\mathbb{D})} = \frac{P(\mathbb{D}|\theta)P(\theta)}{\int_\theta P(\mathbb{D}|\theta')P(\theta')d\theta'}, \tag{3.17}$$

where $P(\theta)$ is the prior and $P(\mathbb{D})$ is the evidence or the normalization constant of the posterior. The relative weighting between the likelihood and the prior is parameterized by the standard deviation s of the likelihood. The higher the standard deviation, the more weight we give to the regularization enforced by the prior. The rate at which we update the prior distribution also determines the regularization weight. The posterior in (3.17) shows that the likelihood and the prior are in a tug of war. The likelihood pulls the parameters towards minimizing the prediction error, but the prior distribution gives priority to the initial distribution of the decision parameters. This tug of war prevents the posterior from finding a near zero-variance solution, and consequently achieving the bias-variance trade-off.

Posterior Distribution

While the likelihood and prior distributions in (3.17) can be expressed explicitly, the evidence $P(\mathbb{D})$ is typically intractable. We leverage Bayesian inference techniques to approximate or find the exact posterior distribution without the use of the evidence. Two of the most famous techniques are discussed as follows.

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

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

where $\mathbb{U}$ is the uniform distribution. If $A(\theta^{(\tau)}, \theta^{(\tau-1)}) \geq \nu$, then we accept the sample. Otherwise, we discard the candidate and resample from $\tilde{Q}(\theta^{(\tau)}|\theta^{(\tau-1)})$. MCMC methods such as Metropolis-Hastings collect the next sample through random walk [37], while other gradient-based techniques such as Hamiltonian Monte Carlo (HMC) method, efficiently search the parameter space through the gradient of the likelihood. Even though the MCMC methods learn the exact posterior distribution, they have slow convergence properties for high-dimensional parameters. In such cases, techniques such as variational inference compromise accuracy of the posterior distribution for speed of

convergence.

2. **Variational Inference (VI)**: is a gradient-based technique that approximates the posterior with the pre-selected distribution $Q(\theta; z)$. The approximate posterior is selected from the conjugate families of the likelihood and prior distributions. The goal is to learn the distribution parameters z of the approximate posterior that minimize the Kullback-Leibler divergence or equivalently maximize the evidence lower bound (ELBO) [38]. The ELBO, $\mathcal{L}$, is given by

$$\mathcal{L}(\mathbb{D}, z) = \mathbb{E}_{\theta \sim Q} \left[\log(P(\mathbb{D} \mid \theta) P(\theta)) - \log(Q(\theta; z)) \right]. \tag{3.18}$$

Remark 3. *For continuous posterior distribution, the* ELBO *given in equation* (3.18) *is redefined using differential entropy, which expresses the prior and posterior in terms of their probability density functions. In this case, the likelihood* $P(\mathbb{D} \mid \theta)$ *is also a probability density function and the* ELBO *is not bounded by zero.*

Once we find the exact or approximate posterior, the prediction for state x can be found in one of two ways. The first option is to marginalize the model over the posterior as follows [25]

$$\hat{F}(x) = \frac{1}{N_\theta} \sum_{\theta \sim Q} F(x; \theta), \tag{3.19}$$

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

$$\theta_{MAP} = \underset{\theta}{\operatorname{argmax}}\ P(\theta|\mathbb{D}). \tag{3.20}$$

3.2 Theoretical Justification of Robustness

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

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

3.2.1 Optimal Control under Parameter Uncertainty

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

$$\begin{cases} \dot{x} = p_s x + u,\ x(0) = x_0, \\ u(x) = \theta x. \end{cases} \tag{3.21}$$

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

$$\mathcal{J} = \int_0^T \left(\frac{1}{2} c x^2 + \frac{1}{2} r u^2 \right) dt, \tag{3.22}$$

where T is the control horizon and $c \geq 0$ and $r > 0$ are design parameters. We solve the control system (3.21) to find $x(t) = e^{(p_s+\theta)t}$ and plug this into the performance index (3.22) along with the form selected for the controller. Performing the integration over time and letting $T \to \infty$, assuming that $p_s + \theta < 0$ then yields the infinite-horizon optimal cost functional

$$\mathcal{J}_\infty = -\frac{1}{4} \frac{c + r\theta^2}{p_s + \theta}. \tag{3.23}$$

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

$$\begin{aligned} \nabla_\theta \mathcal{J}_\infty &= -\frac{r}{4}\frac{(p_s+\theta)^2 - (p^2 + {}^c\!/_r)}{(p_s+\theta)^2} = 0, \\ \therefore \theta^\star &= g(p_s) := -p_s - \sqrt{p_s^2 + {}^c\!/_r}, \\ g^{-1}(\theta) &= \frac{c}{2r\theta} - \frac{\theta}{2}. \end{aligned} \tag{3.24}$$

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

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

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

We can further eliminate the control parameter from the expression for the optimal cost function $\mathcal{J}_\infty$ by substituting for θ from equation (3.24), yielding

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

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

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

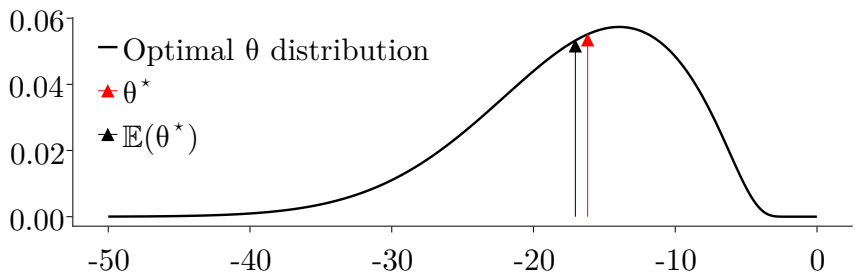


Figure 3.2: The optimal control parameter distribution given that the system parameter p_s is normally distributed with mean $\hat{p}_s = 5$ and $\sigma_p = 5$. The red and black arrows respectively indicate the optimal control parameter without considering the randomness of p_s, and the expected value of the optimal control parameter distribution.

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

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

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

yields a closed-loop stable system for a wider range of values of p_s.

Finally, Figure 3.2 shows the optimal control parameter distribution given that the system parameter p_s is normally distributed with mean $\hat{p}_s = 5$, standard deviation $\sigma_p = 5$. This figure also shows the mean values of the optimal control distribution with the black arrow and the optimal control parameter a deterministic approach would yield in red. We notice that the Bayesian learning that yields the optimal control parameter distribution is more concerned about system stability due to the uncertainty in the parameter p_s, a feat that the deterministic training may not reason about.

3.2.2 Optimal Control under Parameter Uncertainty and Measurement Noise

Consider the scenario in which the system (3.21) is also subject to measurement errors; that is, our measurement model for the state x is probabilistic and is distributed according to the Gaussian $\mathcal{N}(x, \sigma)$. Since the controller uses this measurement to determine its action, the closed-loop system has to be modelled as a stochastic differential equation (SDE), given by

$$\begin{cases} \mathrm{d}x(t) = (p_s + \theta)x(t)\ \mathrm{d}t + \theta\sigma\ \mathrm{d}W_t, \\ x(0) = 1, \end{cases} \tag{3.25}$$

where W denotes the Wiener process [39]. The initial state is assumed deterministic and is set to unity for simplicity. The unique solution to this SDE is given by

$$x(t) = e^{(p_s+\theta)t} + \theta\sigma \int_0^t e^{(p_s+\theta)(t-s)} dW_s. \tag{3.26}$$

Lemma 1. *The conditional expectation* $\mathbb{E}[\mathcal{J} \mid p_s]$ *of the performance index* (3.22) *given the*

system parameter p_s is

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

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

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

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

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

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

We can then minimize $\mathbb{E}[\mathcal{J}]$ over the control parameter in order to study the effects of both kinds of uncertainties on the optimal controller. Such a study is provided in Figures 3.3 and 3.4, where we have plotted the optimal control parameter $\theta^\star$ and the minimal expected cost $\mathbb{E}[\mathcal{J}]$ as a function of the standard deviations of the measurement noise σ and the system parameter σ_p. The constants we used to generate the data are given by $c = r = 1$ and $T = \hat{p}_s = 3$. Our first observation is that the magnitude of the optimal control parameter is an increasing function of system parameter uncertainty and a decreasing function of measurement uncertainty. Our second observation is that if the measurement noise is small, then the optimal control parameter is insensitive to system parameter uncertainty as long as

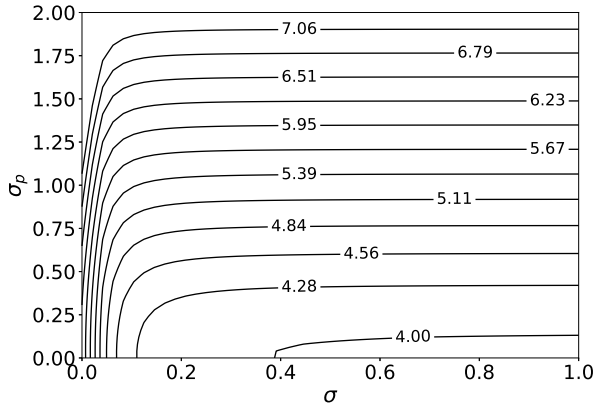


Figure 3.3: The optimal controller parameter magnitude $|\theta^\star|$.

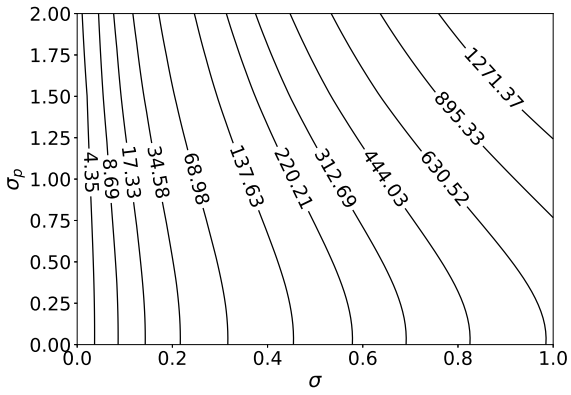


Figure 3.4: The minimal expected cost $\mathbb{E}[\mathcal{J}]$.

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

3.3 Bayesian Neural PBC

In this section, we present a unified framework that simultaneously combines the NeuralPbc technique and rigorously addresses model uncertainties using Bayesian learning. Motivated by [40], we incorporate uncertainties into the dynamics and cast the passivity-based control synthesis problem as a stochastic optimization problem. The closed-loop storage (energy-like) function, from which the control law is derived, is not restricted to a certain form and instead represented by a neural network whose parameters are random variables. We apply Bayesian learning and develop an algorithm that finds a suitable probability distribution of the neural net parameters automatically. In contrast to deterministic optimization, this approach provides a probability distribution over the neural net parameters instead of a point estimate, providing a way to reason about model uncertainties and measurement noise

during the learning process. We demonstrate the efficacy and robustness of our current framework with a comparison against the deterministic framework [40]. The comparison is performed on benchmark underactuated control problems–the simple pendulum and the inertia wheel pendulum–both in simulation and real-world experiments.

3.3.1 Control Design for Smooth Dynamical Systems

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

Let $\phi(x_0, u^\theta, T)$ denote a closed loop trajectory integrated from the Hamilton's equation of motion in (3.2). The trajectory starts from the initial state x_0 and spans for the time horizon T. We sample the initial state x_0 through greedy and explorative state sampling techniques discussed in Section 2.3.2. The control law u^θ consists of the energy shaping and damping terms found from the desired Hamiltonian H_d^θ as shown in equation (3.6). To generate this trajectory, we first sample the parameters θ from the prior distribution $P(\theta)$.

We take two approaches to selecting the prior distribution. The simplest approach is to use an uninformed prior given by a uniform distribution; this choice encourages exploration but has slow convergence properties. The second approach uses an informed prior that warm-starts the Bayesian training around the solution of the deterministic training. To do so, the prior distribution is a Gaussian probability distribution centered around the parameters learned from the deterministic NeuralPbc technique discussed in Section 3.1.1.

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

performance objective as follows.

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

where $\mathcal{D}_N$ is a collection of initial states sampled with explorative and greedy state sampling techniques (Section 2.3.2), and ℓ is the running cost of the trajectories. We provide three options for the running cost in the upcoming section titled *Performance Objective*. In order to update the posterior distribution based on the performance of the generated trajectories, we compose the likelihood function as

$$P(J \mid \theta) = \mathcal{N}(J|0, s), \tag{3.27}$$

where $\mathcal{N}$ is a Gaussian distribution with mean zero and standard deviation s. Notice that the likelihood is maximum when the expectation of the running cost J is minimum.

Remark 4. *On top of the neural net parameters θ, we can learn the posterior over the standard deviation s of the likelihood. This automatically finds the weighting between the likelihood and prior distributions, achieving optimal bias-variance trade-off. In this scenario, the posterior distribution is a multivariate probability distribution over the parameters θ and s.*

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

over the posterior as the following optimization problem.

$$\begin{aligned}
\underset{P(\theta|J)}{\text{minimize}} \quad & J(\phi, u^\theta) = \mathbb{E}_{x_0 \in \mathcal{D}_N}[\ell(\phi(x_0, u^\theta, T), u^\theta)], \\
\text{subject to} \quad & \begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ \Omega(q) \end{bmatrix} u^\theta, \\
& u^\theta = -\Omega^\dagger(\nabla_q H_d^\theta - \nabla_q H), \\
& p_s \sim \mathcal{N}(\hat{p}_s, \sigma_p), \\
& \theta \sim P(\theta|J).
\end{aligned} \tag{3.28}$$

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

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

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

   We generate the next sample from a Markov Chain, which is given by the following first order differential equations [1]

$$\begin{aligned}
m^{\tau+\Delta\tau/2} &= m^\tau - \frac{\Delta\tau}{2}\frac{\partial P(\theta^\tau, J)}{\partial \theta^\tau}, \\
\theta^{\tau+\Delta\tau} &= \theta^\tau + m^{\tau+\Delta\tau/2}\,\Delta\tau, \\
m^{\tau+\Delta\tau} &= m(\tau + {}^{\Delta\tau}/_2) - \frac{\Delta\tau}{2}\frac{\partial P(\theta^{\tau+\Delta\tau}, J)}{\partial \theta^{\tau+\Delta\tau}},
\end{aligned} \tag{3.29}$$

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

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

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

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

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

We repeat this computation for every state in a trajectory.

Algorithm 6 Bayesian NeuralPbc via Hamiltonian Monte Carlo

$\Theta \leftarrow \{\}$ ▷ Collection of samples
Define prior $P(\theta)$
while $i <$ `maximum iteration` **do**
    $\tau = 0, \theta^0 \sim P(\theta)$, $m^0 = 0$, accepted=true
    while accepted **do**
        $\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ $N_\mathcal{D}$ initial state samples
        $p_s \sim \mathcal{N}(\hat{p}_s, \sigma_p)$ ▷ Sample a system parameter
        $J^\tau \leftarrow 0$, $J^{\tau+\Delta\tau} \leftarrow 0$
        for $x_0 \in \mathcal{D}_N$ **do**
            $\phi \leftarrow \phi(x_0, u^{\theta^\tau}, T)$ ▷ Generate trajectory
            $J^\tau \leftarrow J^\tau + \ell(\phi; \theta^\tau)/N_\mathcal{D}$ ▷ Batch loss of current parameter
        $m^{\tau+\Delta\tau}, \theta^{\tau+\Delta\tau} \leftarrow$ `leap frog discretization`(m^τ, θ^τ) ▷ Equation (3.29)
        for $x_0 \in \mathcal{D}_N$ **do**
            $\phi \leftarrow \phi(x_0, u^{\theta^{\tau+\Delta\tau}}, T)$ ▷ Generate trajectory
            $J^{\tau+\Delta\tau} \leftarrow J^{\tau+\Delta\tau} + \ell(\phi; \theta^{\tau+\Delta\tau})/N_\mathcal{D}$ ▷ Batch loss of next parameter
        $\nu \sim \mathbb{U}(0, 1)$
        $A(\theta^{(\tau+\Delta\tau)}, \theta^{(\tau)}) = \min\left(1, \frac{P(\theta^{(\tau+\Delta\tau)}, J^{(\tau+\Delta\tau)})}{P(\theta^{(\tau)}, J^{(\tau)})}\right)$
        if $A(\theta^{(\tau+\Delta\tau)}, \theta^{(\tau)}) \geq \nu$ **then**
            $\Theta \leftarrow \Theta \cup \theta^{(\tau+\Delta\tau)}$ ▷ Accept sampled parameter
            $\tau = \tau + \Delta\tau$
        else
            $i \leftarrow i + 1$, accepted=false ▷ Reject sampled parameter
Return Θ

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

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

$$\mathcal{L}(J, z) = \mathbb{E}_{\theta \sim Q} \left[\log(P(J|\theta)P(\theta)) - \log(Q(\theta; z)) \right]. \tag{3.30}$$

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

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

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

The gradient $\partial \mathcal{L}/\partial z$ holds a rather complex form for two reasons. First, the ELBO is evaluated from trajectories integrated from the differential equations in (3.2). We use a combination of adjoint method and auto-differentiation techniques [41] to compute the gradient $\partial \mathcal{L}/\partial z$ through the trajectories. Secondly, computing $\partial \mathcal{L}/\partial z$ requires the derivative of the sample θ with respect to the distribution parameters z, which is intractable. We handle this complication by invoking the reparameterization trick of the Automatic Differentiation Variational Inference(ADVI) [42].

Performance Objective

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

Algorithm 7 Bayesian NeuralPbc via variational inference

$\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ $N_\mathcal{D}$ initial state samples
while $i <$ `maximum iteration` **do**
  $\mathcal{L} \leftarrow 0$ ▷ Elbo Loss
  for $i = 1 : N_\theta$ **do** ▷ Samples to compute (3.30)
    $J \leftarrow 0$ ▷ Batch loss
    $\theta \sim Q(\theta; z)$ ▷ Sample parameters of H_d^θ
    for $x_0 \in \mathcal{D}_N$ **do**
      $p_s \sim \mathcal{N}(\hat{p}_s, \sigma_p)$ ▷ Sample a system parameter
      $\phi \leftarrow \phi(x_0, u^\theta, T)$ ▷ Generate trajectory
      $J \leftarrow J + \ell(\phi; \theta)/N_\mathcal{D}$
    $\mathcal{L} \leftarrow \mathcal{L} + \frac{1}{N_\theta}\left(\log[P(J|\theta)P(\theta)] - \log[Q(\theta; z)]\right)$
  $z \leftarrow z + \alpha_i \partial\mathcal{L}/\partial z$ ▷ SGD step
  $\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ New initial state samples
  $i \leftarrow i + 1$
return z

1. **Trajectory tracking**: Let $x^\star$ denote the desired equilibrium of a dynamical system and $\phi(x_0, u^\theta, T)$ represent a prediction from the initial condition x_0 with the current control law u^θ. The objective of this task is to find a closed-loop controller that can track an expert trajectory $\phi^\star$ obtained from a path planner. To impose this behavior, the running cost is a function of the distance from the current trajectory ϕ to $\phi^\star$, defined in terms of the transverse coordinates $\phi_\perp - \phi^\star_\perp$ along the preferred orbit (shown in Figure 3.5), using the ideas outlined in [43, 44]. In this setting, the prediction converges to $\phi^\star$ if and only if $\phi_\perp - \phi^\star_\perp \xrightarrow{t\to\infty} 0$. In order to encourage this behavior, we define the cost function as

$$\begin{aligned} J_{track} &= \mathbb{E}_{x_0 \in \mathcal{D}_N}[\ell_{track}(\phi(x_0, u^\theta, T), u^\theta)], \\ \ell_{track} &= \sum_{x_\perp \in \phi_\perp,\, x^\star_\perp \in \phi^\star_\perp} ||x_\perp - x^\star_\perp||\,. \end{aligned} \tag{3.31}$$

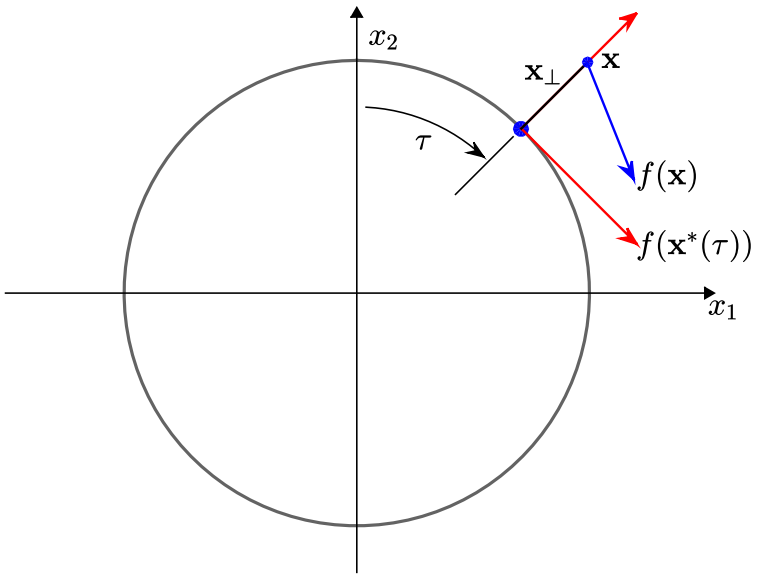


Figure 3.5: Transverse Coordinates

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

$$P(J_{track}|\theta) = \prod_{x_\perp \in \phi_\perp,\, x_\perp^\star \in \phi_\perp^\star} \mathcal{N}(\|x_\perp - x_\perp^\star\| \mid 0, s), \tag{3.32}$$

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

$$\begin{aligned} J_{\text{set}} &= \mathbb{E}_{x_0 \in \mathcal{D}_N}[\ell_{\text{set}}(\phi(x_0, u^\theta, T), u^\theta)], \\ \ell_{\text{set}} &= \inf_t \{\|a - b\| \colon a \in \phi(t),\, b \in \mathcal{S}\}. \end{aligned} \tag{3.33}$$

For instance, the set $\mathcal{S}$ may be chosen as a ball of radius r around $x^\star$. Here, r becomes a hyperparameter of the training algorithm. With a particular choice of $\mathcal{S}$, if at any

point along the prediction ϕ, a state x is closer than r to $x^\star$, no penalty is incurred by J_{set}. This construction has the same advantages as the Minimum Trajectory Loss (MTL) discussed in Section 2.3.1. The corresponding likelihood function is given by

$$P(J_{\text{set}}|\theta) = \mathcal{N}(J_{\text{set}} \mid 0, s). \tag{3.34}$$

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

$$\begin{aligned} J_T &= \mathbb{E}_{x_0 \in \mathcal{D}_N}[\ell_T(\phi(x_0, u^\theta, T), u^\theta)], \\ \ell_T &= \|x(T) - x^*\|. \end{aligned} \tag{3.35}$$

The corresponding likelihood function is given by

$$P(J_T|\theta) = \mathcal{N}(J_T \mid 0, s). \tag{3.36}$$

Uncertainty Modeling

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

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

$$\mathrm{d}x = \left(\begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ \Omega(q) \end{bmatrix} u^\theta(x) \right) \mathrm{d}t + \nabla_x u(x)\, \mathrm{d}W_t, \tag{3.37}$$

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

$$\begin{aligned} \underset{P(\theta|J)}{\text{minimize}} \quad & J(\phi, u^\theta) = \mathbb{E}_{x_0 \in \mathcal{D}_N}[\ell(\phi(x_0, u^\theta, T), u^\theta)], \\ \text{subject to} \quad & \mathrm{d}x = \left(\begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ \Omega(q) \end{bmatrix} u^\theta(x) \right) \mathrm{d}t + \nabla_x u(x)\, \mathrm{d}W_t, \\ & u^\theta = -\Omega^\dagger(\nabla_q H_d^\theta - \nabla_q H), \\ & p_s \sim \mathcal{N}(\hat{p}_s, \sigma_p), \\ & \theta \sim P(\theta|J). \end{aligned} \tag{3.38}$$

Remark 5. *Introducing uncertainties to the deterministic training finds a point estimate of the optimal controller parameter, which may be interpreted as the mean of the optimal posterior distribution that Bayesian learning provides. A point estimate of the learned parameters is prone to be biased (for example, if the uncertainty in system parameters is large, the optimal parameter θ for the true system parameter may be quite far from the determinis-*

tic solution). This bias-variance trade-off problem is alleviated by Bayesian inference which allows one to marginalize over the posterior distribution [1].

3.3.2 Control Design for Hybrid Dynamical Systems

In Section 3.1.1, we constructed the properties of passivity on the Hamiltonian of a smooth dynamical system. For hybrid systems, we can deconstruct the concept of passivity into flow-passive and jump-passive [45]. Flow-passive implies that the continuous phase of the dynamics with Hamiltonian H is dissipative, i.e.,

$$H(x(t_1)) \leq H(x(t_0)) + \int_{t_0}^{t_1} s(u(t), y(t))dt, \tag{3.39}$$

under the supply rate $s = u^\top y$. In order for a hybrid system to be passive, the conditions in (3.39) must hold during discrete state transitions as well. In particular, the class of mechanical systems exhibiting contacts, impacts and Coulomb friction obey the dissipative property under the same Hamiltonian as [46]

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

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

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

where ϵ is the coefficient of restitution, and it takes values between 0 and 1. This demonstrates that it is possible to find a single Hamiltonian H_d that renders the closed-loop hybrid

system flow- and jump-passive. Thus, we translate the passivity-based control framework in Section 3.3.1 to hybrid systems. We aim to find the desired Hamiltonian H_d, whose stable equilibrium is at x^*.

In this section, we extend the techniques of deterministic and Bayesian NeuralPbc to hybrid dynamical systems. We introduce a contact model of the hybrid dynamics into the deterministic NeuralPbc framework and infer a controller that leverages the advantages of potential contacts and/or minimizes its adverse effects. Moreover, we inject uncertainties in contact forces and post-impact velocities to NeuralPbc in order to design a robust controller through Bayesian learning.

Deterministic NeuralPBC

In this framework, we aim to find a passive closed loop system for the hybrid dynamics (2.3) whose desired stable equilibrium is at $x^* = (q^*, \dot{q}^*)$. We formulate this problem as a search over the point-estimate parameters of H_d given by a neural network. This training framework can be posed as the following optimization problem.

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \ell\left(\phi, u^\theta\right), \\ \text{subject to} \quad & M(q)\,\mathrm{d}\dot{q} + h(q,\dot{q},\theta)\,\mathrm{d}t - \mathrm{d}R = 0, \\ & u^\theta = -\Omega^\dagger(\nabla_q H_d^\theta - \nabla_q H), \end{aligned} \tag{3.40}$$

The performance objective ℓ is evaluated from closed loop trajectories ϕ using the current control law. We follow Moreau's time stepping algorithm [18] outlined in Algorithm (1) to resolve the complementarity constraint in (2.4) and integrate the closed-loop dynamics. We sample $N_{\mathcal{D}}$ initial states through greedy and explorative state sampling techniques discussed in Section 2.3.2. As outlined in Algorithm 8, we compose the cost ℓ from the $N_{\mathcal{D}}$ closed loop

trajectories. The performance objective is chosen according to the desired system behavior as discussed in Section 3.3.1. We update the parameters θ through stochastic gradient descent (SGD). We compute the gradient $\partial\ell/\partial\theta$ through auto-differentiation techniques.

Algorithm 8 Solution to the Optimization Problem (3.40)

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

Bayesian NeuralPBC

Hybrid systems such as walking machines and manipulators perform constant interaction with their environment. In many cases, the exact parameters of the environment are not known. For instance, walking machines operate on uneven terrain, where the exact elevation and friction coefficients of the runway may be unknown. Manipulators interact with objects of different textures and friction, which we cannot simply infer from sensors. In order to learn a controller robust against these uncertainties, we inject domain randomization on the state of the environment during the training process. Unfortunately, simply introducing random environmental conditions to the training outlined in Algorithm 8 is not sufficient. In the presence of high variance disturbances, the point-estimate parameters θ under domain randomization are prone to be biased [47]. Let us take a walking robot as an example; if the uncertainty in the terrain elevation is large, the learned parameters θ may be far from

the optimal controller corresponding to the true elevation. To combat this issue, we propose a probabilistic framework, where we learn a posterior probability distribution over the parameters θ via Bayesian inference. Then we address the bias-variance trade-off problem by marginalizing the controller over the distribution of the parameters [1].

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 [25]. The objective is to find the posterior distribution $P(\theta|J)$ that achieves the performance objective for various environmental conditions. This framework can be summarized by the following optimization problem.

$$\begin{aligned} \underset{P(\theta|J)}{\text{minimize}} \quad & J(\phi, u) = \mathbb{E}_{x_0 \in \mathcal{D}_N}[\ell(\phi(x_0, u^\theta, T), u^\theta)], \\ \text{subject to} \quad & M(q)\,\mathrm{d}\dot{q} + h(q, \dot{q}, \theta)\,\mathrm{d}t - \mathrm{d}R = 0, \\ & u^\theta = -\Omega^\dagger(\nabla_q H_d^\theta - \nabla_q H), \\ & p_s \sim \mathbb{U}(p_{min}, p_{max}), \\ & \theta \sim P(\theta|J). \end{aligned} \tag{3.41}$$

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

We solve the optimization problem in (3.41) with the procedure outlined in Algorithm (9). We follow the initial state sampling techniques and the performance objectives discussed in detail in Section 3.3.1. In particular, we use variational inference to learn the

distribution parameters of the pre-selected posterior $Q(\theta; z)$.

Algorithm 9 Solution to the Optimization Problem (3.41)

$\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ $N_\mathcal{D}$ initial state samples
while $i <$ `maximum iteration` **do**
  $\mathcal{L} \leftarrow 0$ ▷ Elbo Loss
  for $i = 1 : N_\theta$ **do** ▷ Samples to compute (3.30)
    $J \leftarrow 0$ ▷ Batch loss
    $\theta \sim Q(\theta; z)$ ▷ Sample parameters of H_d^θ
    for $(x_0) \sim \mathcal{D}_N$ **do**
      $p_s \sim \mathbb{U}(p_{min}, p_{max})$ ▷ Sample a system parameter
      $\phi = \text{Moreau}(x_0)$ ▷ Algorithm (1)
      $J \leftarrow J + \ell(\phi; \theta)/N_\mathcal{D}$
    $\mathcal{L} \leftarrow \mathcal{L} + \frac{1}{N_\theta} (\log[P(J|\theta)P(\theta)] - \log[Q(\theta; z)])$
  $z \leftarrow z + \alpha_i \partial\mathcal{L}/\partial z$ ▷ SGD step
  $\mathcal{D}_N \leftarrow \{x_0\}_{(N_\mathcal{D})}$ ▷ New initial state samples
  $i \leftarrow i + 1$
return z

3.4 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 in the IdaPbc architecture. We parametrize the function V_d^θ and the entries of L_θ and A_θ matrices with Bayesian neural networks. We invoke variational inference to find the approximate posterior over the parameters θ. The goal is to learn the distribution parameters z of the posterior multivariate probability distribution $Q(\theta; z)$ that maximize the Elbo given in (3.18). We pose the search over the parameters z

as the following optimization problem.

$$
\begin{aligned}
\underset{z}{\text{minimize}} \quad & \|l_{\text{IDA}}(x)\|^2 = \left\|\Omega^{\perp}\left\{\nabla_q H - M_d M^{-1}\nabla_q H_d + J_2 M_d^{-1} p\right\}\right\|^2, \\
\text{subject to} \quad & M_d^{\theta} = L_{\theta}(q)L_{\theta}^{\top}(q) + \delta_M I_n, \\
& J_2^{\theta}(q,p) = A_{\theta}(q,p) - A_{\theta}^{\top}(q,p), \\
& q^{\star} = \underset{q}{\text{argmin}}\, V_d^{\theta}, \\
& p_s \sim \mathcal{N}(\hat{p}_s, \sigma_p), \\
& \theta \sim Q(\theta; z).
\end{aligned}
\tag{3.42}
$$

The computation of the ELBO requires the likelihood function and the prior distribution. In order to compute the likelihood, we first draw samples of θ from the posterior $Q(\theta; z)$, and evaluate the PDEs given in (3.12). Then, the likelihood is given by

$$
P(\|l_{\text{IDA}}(x)\| \mid \theta) = \mathcal{N}\left(\|l_{\text{IDA}}(x)\| \mid 0, s\right), \tag{3.43}
$$

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 (3.43), maximizing the ELBO in (3.18) coaxes the loss $l_{\text{IDA}}(x)$ to zero.

We update the distribution parameters z along the gradient $\partial\mathcal{L}/\partial z$ until the ELBO converges and the objective function $\|l_{\text{IDA}}(x)\|^2$ reaches the threshold ϵ_{tol}. We invoke the reparameterization trick of the Automatic Differentiation Variational Inference(ADVI) [42] to compute the gradient of samples θ with respect to the distribution parameters z.

System parameter uncertainties can deteriorate the performance of controllers employed on real systems. Hence, in the Bayesian framework, we inject these uncertainties directly

into the training loop in order to learn a controller that works for a wide range of system parameters. To model these uncertainties, we sample a set of system parameters p_s from a normal distribution $\mathcal{N}(\hat{p}_s, \sigma_p)$ centered around the nominal parameter $\hat{p}_s$, where σ_p represents the uncertainty in system parameters. Each time we compute the PDE loss l_{IDA} for a batch of discrete states sampled from the configuration space, we draw a new sample of p_s.

CHAPTER 4:

EXPERIMENTAL RESULTS

In this chapter, we provide simulated and real-world experiments that demonstrate the robustness properties of the Bayesian framework presented in Chapter 3. We present three sets of experiments for the Bayesian NeuralPbc approach. First, we learn a swing-up controller for the simple pendulum and evaluate its performance under system parameter and measurement uncertainties in simulation. In the second set of experiments, we tackle the swing-up task for the inertia wheel pendulum and demonstrate the efficacy of the Bayesian controller in simulation and real-world experiments. Lastly, we demonstrate the performance of the deterministic and Bayesian NeuralPbc frameworks for hybrid dynamical systems on the rimless wheel, in simulation and real-world experiments. For the Bayesian Neural-IdaPbc framework, we provide simulated and hardware experiments for the swing-up task of the IWP.

4.1 Bayesian Neural PBC

4.1.1 Simple Pendulum

In this section, we demonstrate the robustness properties of the controllers trained via Bayesian NeuralPbc on the simple pendulum system. The system is simulated under measurement noise via stochastic differential equations (3.37). Furthermore, the system parameter is varied in order to analyze the closed-loop system response under model uncer-

tainties.

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

$$\mathrm{d}x = \begin{bmatrix} \dot{q} \\ a\sin(q) + u^{\theta}(x) \end{bmatrix} \mathrm{d}t + \nabla_x u^{\theta}(x)\,\mathrm{d}W_t, \tag{4.1}$$

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

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

Training

The goal of this training is to track the expert trajectories generated by the vanilla energy-shaping control (ESC) [2]. The ESC takes the general form

$$u^{\theta}(q, \dot{q}; \theta^e) = \theta_1^e \dot{q} + \theta_2^e \cos(q)\dot{q} + \theta_3^e \dot{q}^3, \tag{4.2}$$

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

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

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

The running cost function is chosen to be the loss $J_{track}(\phi_\perp)$ from homoclinic orbit $\phi^\star$ provided in (3.31); the corresponding likelihood is given by (3.32). We collect dataset $\mathcal{D}_N$ of initial states sampled with greedy and explorative techniques. For this particular experiment, we use Hamiltonian Monte Carlo outlined in Algorithm 6 to infer the exact posterior distribution. We compare the Bayesian policy with the deterministic policy, which is simply given by the point-estimates of the expert vanilla ESC in (4.2) [32].

Simulation Tests

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

We demonstrate the robustness of the Bayesian policy by comparing ζ, as a function of the system parameter a, with that of the deterministic policy. We also investigate the effects of the prior distribution of θ on the performance of the controllers. In particular, we examine a uniform prior and a Gaussian prior centered around the deterministic solution of (4.2). The comparisons between the controllers from both cases are shown in Figure 4.1.

In addition to uncertainties in a, we test the deterministic and Bayesian policies with measurement noise, modeled as a Wiener process with standard deviation 0.0005 rad in the q-direction and 0.05 $^{\text{rad}}/_{\text{s}}$ in the $\dot{q}$-direction. These numbers are chosen to represent a typical error arising from an optical encoder with a resolution of 2048 pulses per revolution and its naive differentiation via backward difference. To capture the influence of measurement noise, we generate 20 trajectories by integrating (4.1) from the same initial states. These trajectories are then used to compute ζ. The effects of noise on ζ are reflected by the error bands in Figure 4.1.

The results, shown in Figure 4.1, demonstrate that Bayesian learning yields controllers that outperform their deterministic counterpart throughout the whole range of a. The marginalization method in (3.19) for selecting θ from the learned distribution performs best when a uniform prior is used, while the MAP method performs best with the Gaussian prior. We emphasize that the error bands of the marginalized and MAP point estimates stay well below those of the deterministic curve in the top plot of Figure 4.1. This implies that the controllers produced by Bayesian learning perform consistently better than the deterministic controller, demonstrating their robustness against model uncertainty and measurement noise.

4.1.2 Inertia Wheel Pendulum

In this section, we validate the Bayesian NeuralPbc framework on the problem of swinging-up and stabilizing the inverted position of an inertia wheel pendulum (IWP), shown in Fig. 4.2. We provide experimental results from simulation and real-world hardware in order to thoroughly demonstrate the efficacy and robustness claims of Bayesian inference. We use the deterministic solution for NeuralPbc as the baseline on which we compare the performance of the Bayesian solution.

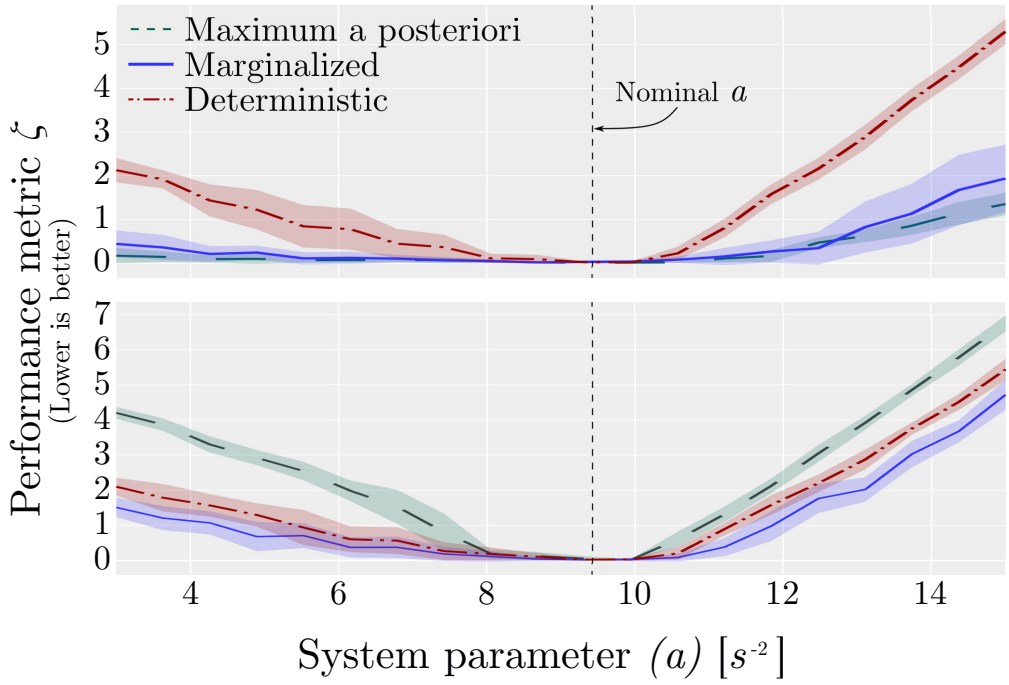


Figure 4.1: Performance comparisons between deterministic and Bayesian learning methods. The training is initialized with a Gaussian prior (top), and a uniform prior (bottom). The continuous error band is generated by computing ζ from 20 trajectories of (4.1), starting at the downward equilibrium with a small disturbance. The solid lines represent the mean of ζ. Best viewed in color.

System Model

The IWP mechanism consists of a pendulum with an actuated wheel instead of a static mass. The wheel has mass m, which is connected to a massless rod of length l. The position of the rod is denoted by the angle q_1 measured with respect to the downward vertical position. The position of the wheel q_2 is measured with respect to the vertical line through the center of the wheel. The Hamiltonian of the IWP is given by Equation (3.1), where

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

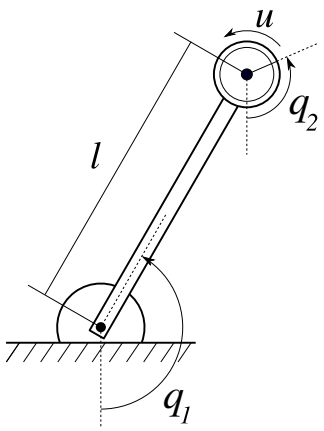


Figure 4.2: Schematic of the inertia wheel pendulum. Only the joint q_2 is actuated, and q_1 is not.

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

$$\mathrm{d}x = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dfrac{mgl\sin(q_1) - u^\theta - b_1\dot{q}_1}{I_1} \\ \dfrac{u^\theta - b_2\dot{q}_2}{I_2} \end{bmatrix} \mathrm{d}t + \nabla_x u^\theta(x)\,\mathrm{d}W_t, \tag{4.3}$$

where the control input u^θ is the torque applied to the inertia wheel and $\{b_i\}_{i=1}^2$ are friction coefficients. The desired equilibrium $x^\star$ is the origin, which corresponds to the upward position. The nominal system parameters are estimated to be $I_1 = 0.0455$ kg-m^2, $I_2 = 0.00425$ kg-m^2, and $mgl = 1.795$ N-m.

Table 4.1: 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

Training

The energy-like function H_d^θ is a fully-connected neural network with two hidden layers, each with the Elu activation function [31]. 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 greedy and explorative state sampling techniques; these initial states are integrated forward with a time horizon of $t \in [0, 3]$ seconds. In the Bayesian framework, the standard deviations σ_p of system parameters $p_s = [I_1, I_2, mgl]$ are chosen to be 10% of the nominal system parameters. Moreover, we train on trajectories per the SDE in (3.37) 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 4.1. 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 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 4.3 shows the performance of deterministic and Bayesian trainings using an accumulated quadratic loss of the form

$$J^T = \frac{1}{2} \int_0^T \left(x^\top \mathcal{Q} x + u^\top \mathcal{R} u \right) dt. \tag{4.4}$$

The controller learned from the Bayesian training is marginalized over 10 parameters sampled from the posterior. As seen in Figure 4.3, 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.

Hardware Tests

The controllers from deterministic and Bayesian training schemes are evaluated on the hardware shown in Figure 4.4. We deliberately modify the hardware and test the controllers without any additional training. In particular, throughout the experiments, the inertia wheel

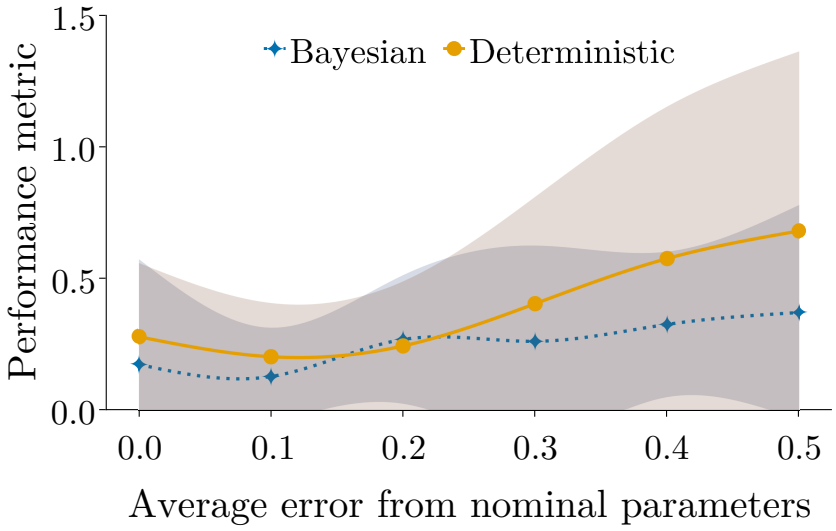


Figure 4.3: NeuralPBC Performance metric (J^T) for various error in system parameters. Measurement noise included as Wiener process with standard deviation of 0.001 **and** 0.02 **on joint angles and velocities, respectively**

attached to q_2 is replaced with parts (labelled A-C on Table 4.2) whose mass and inertia are different from the nominal values. The modified system parameters are summarized in Table 4.2.

Table 4.2: System parameters used in real-world experiments. The errors in the last column are $\|p_s - \hat{p}_s\|/\|\hat{p}_s\|$

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

The system starts from rest at the downward position. A small disturbance in the q_1 direction is introduced to start the swing-up. The state x is recorded and (4.4) is the per-

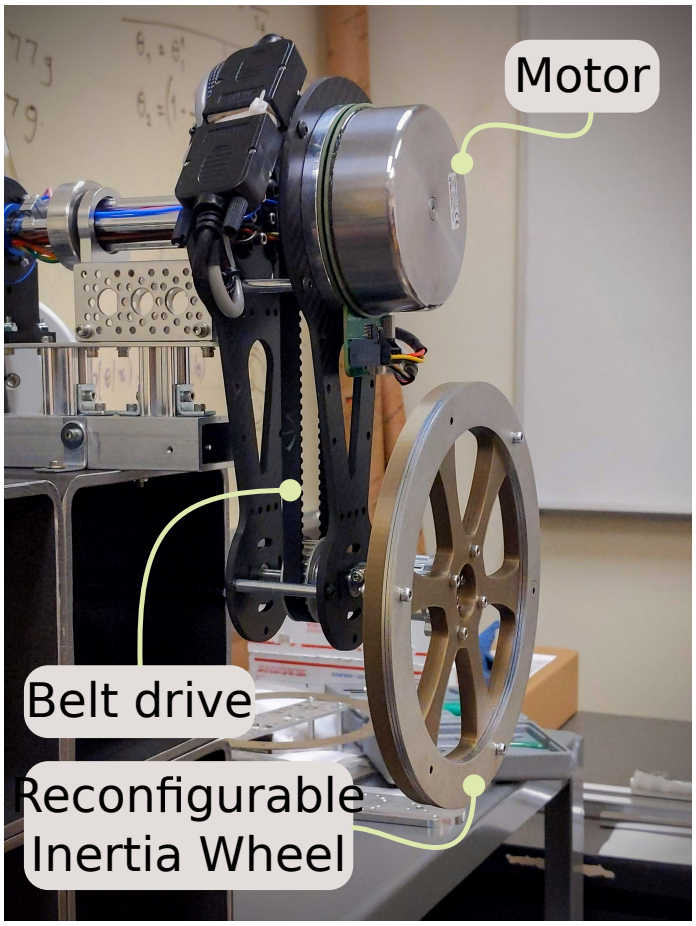


Figure 4.4: Inertia Wheel Pendulum Hardware

formance metric used to evaluate the controllers. The results are summarized in Figure 4.5.

In all scenarios, our controllers are able to achieve the control objective despite the errors introduced in the system parameters. These results demonstrate that our approach enables a unified way to tackle nonlinear control problems while simultaneously incorporating prior knowledge and model uncertainties.

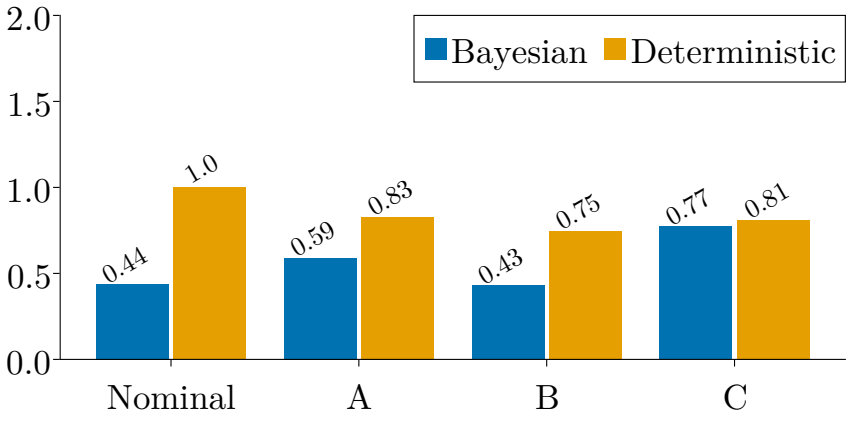


Figure 4.5: Controller performance for modified system parameters. The performance metric is given by Eq. (4.4). Lower values are better. These results show that controllers trained via Bayesian learning are consistently more robust to errors in system parameters.

4.1.3 Rimless Wheel

This mechanism consists of a rimless wheel in the plane with a set of N spokes and a torso freely rotating about a pin joint located at the center of the wheel. The mass of the wheel m_1 is concentrated at the hip and spans a radius of l_1. The torso has its mass m_2 concentrated at a distance of l_2 from the hip. We actuate the torso angle with the motor mounted at the hip, which in turn propels the entire wheel forward or back. The angle of the torso is characterized by φ measured from the outward normal of the runway shown as $\hat{n}$ in Figure 4.6. The orientation of the wheel β is measured from $\hat{n}$ to a datum spoke.

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

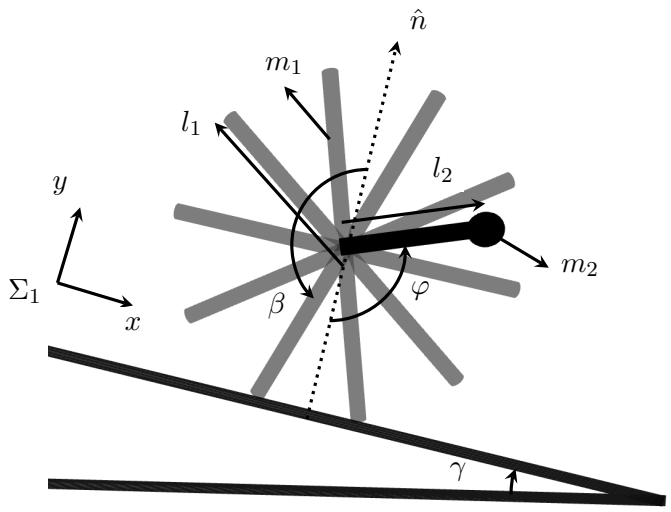


Figure 4.6: Rimless wheel with torso; depicted with $N = 10$ spokes.

the system are given by

$$\mathcal{K} = \frac{1}{2}m_t(\dot{x}+\dot{y})^2 + \frac{1}{2}I_1\dot{\beta}^2 + \frac{1}{2}(m_2l_2^2 + I_2)\dot{\varphi}^2 + m_2l_2(c_\varphi\dot{x} + s_\varphi\dot{y})\dot{\varphi},$$
$$\mathcal{P} = m_tg(-xs_\gamma + yc_\gamma) - m_2gl_2c_{\varphi\gamma},$$

where $c_a := \cos(a), s_a := \sin(a), c_{ab} := \cos(a-b), s_{ab} := \sin(a-b)$. The total mass of the mechanism is given by m_t; I_1 and I_2 are moments of inertia of the wheel and torso respectively. The position vector (x, y) represents the location of the hip with respect to the frame Σ_1 shown in Figure 4.6. The slope of the runway is given by γ, g is the magnitude of acceleration due to gravity. The Euler-Lagrange equations corresponding to the Lagrangian $\mathcal{L} = \mathcal{K} - \mathcal{P}$ are

$$M(q)\,\mathrm{d}\dot{q} + h(q,\dot{q})\,\mathrm{d}t - \mathrm{d}R = 0, \tag{4.5}$$

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

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

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

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

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 Moreau's time stepping algorithm (Algorithm (1)) to resolve the contact forces and Coulomb friction.

Training

The objective of this case study is to use the control authority on the torso to move the robot at a constant hip speed. The performance objective is given by the accumulated loss

$$J_T = \sum_{t=0}^{T} \|\dot{x}_c^* - \dot{x}_c(t;\theta)\| \tag{4.6}$$

where $\dot{x}_c^*$ is a constant 1 $^{\text{m}}/_{\text{s}}$. The deterministic NeuralPbc framework learns the parameters of H_d, which is given by a fully-connected neural network with a total of 113 weights and biases. We observe the cost (4.6) incurred by trajectories generated under the current controller parameters θ and update the decision parameters iteratively via stochastic gradient descent. A single parameter update consists of 4 initial states sampled through greedy or explorative techniques (Section 2.3.2). Each initial state is integrated forward for time horizon of 5 seconds.

In the Bayesian framework, we use variational inference to learn a Gaussian posterior distribution over the parameters of H_d. The decision parameters z are the distribution parameters of the Gaussian posterior. For the Bayesian training, we generate random terrain elevation parameters $p_s \in \mathbb{R}^N$ from a uniform probability distribution $p_s \sim \mathbb{U}(0\text{cm}, 2\text{cm})$. The components of the vector p_s represent the elevation of the terrain under each spoke. We sample a new set of p_s for every trajectory, which exposes the training to the effects of various environmental conditions. The controller learned from the Bayesian training is marginalized according to (3.19) over $N_\theta = 15$ samples drawn from the posterior.

Simulated Experiments

We first show the performance of the deterministic NeuralPbc training on a level ground. As shown on the bottom plot of Figure 4.7, 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 4.8 shows that a large positive torque is applied if the wheel stumbles backwards.

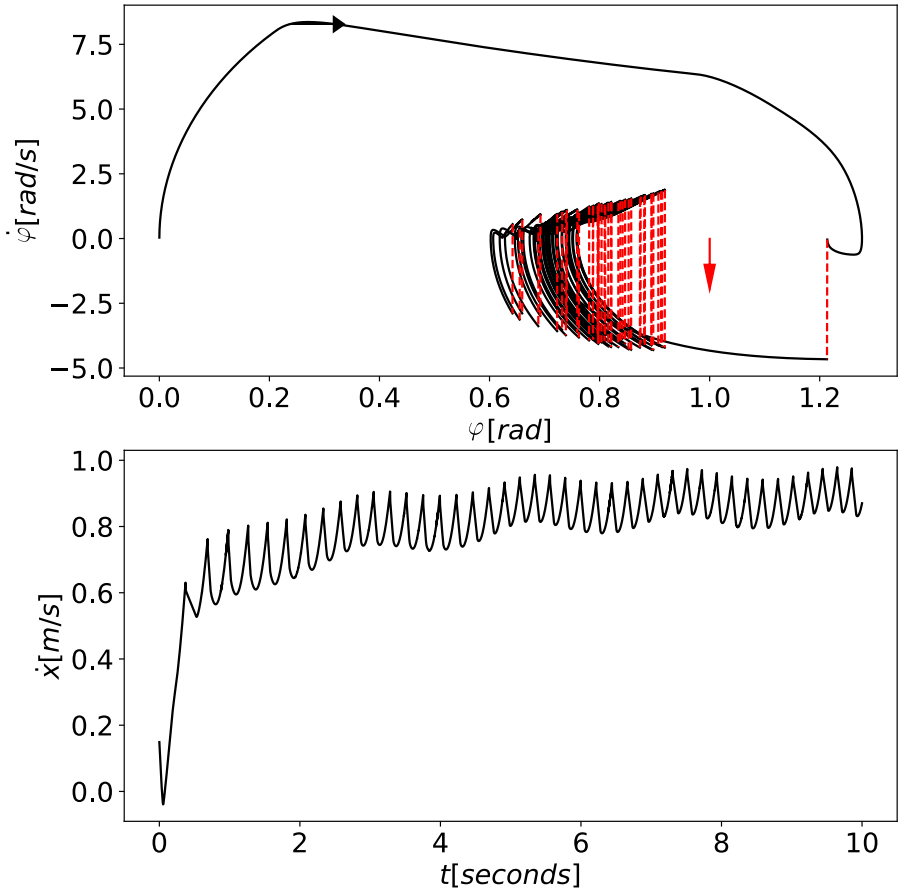


Figure 4.7: Top: Torso orientation and velocity for 10-second trajectory. The solid black lines show the continuous phases and the dashed red lines show discrete transitions. Bottom: Horizontal hip speed

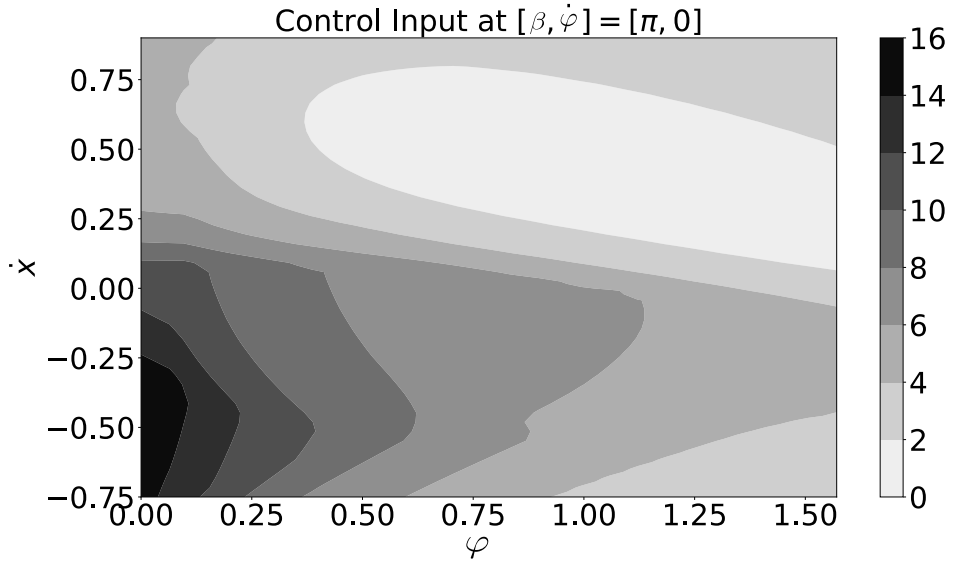


Figure 4.8: Torque command to torso as a function of torso angle and horizontal hip speed

We compare the performance of the deterministic and Bayesian controllers as follows. Similar to the Bayesian training, we sample the terrain elevation from $p_s \sim \mathbb{U}(0, p_{max})$ where $p_{max} = [0, 0.5, 1, 1.5, 2]$ centimeters. As the value of p_{max} increases, the difference in elevation between two consecutive spokes increases. For instance, for $p_{max} = 0.5$, one spoke can be on a level ground and the adjacent spoke can be up to 0.5 cm higher. As p_{max} increases to 2cm, the neighboring spokes can have up to 2cm elevation difference, which requires a large torque to overcome the steep rise in elevation. For each value of p_{max}, we generate 10 random initial states x_0 and integrate 10-second-long trajectories. The cost J_T of each trajectory is calculated as per (4.6). The bars in Figure 4.9 show the average J_T over the 10 trajectories for each p_{max}. Notice that the Bayesian training consistently incurs less cost in all elevation differences compared to its deterministic counterpart. The deterministic training accumulates more cost with each increment in the value of p_{max}, while the Bayesian

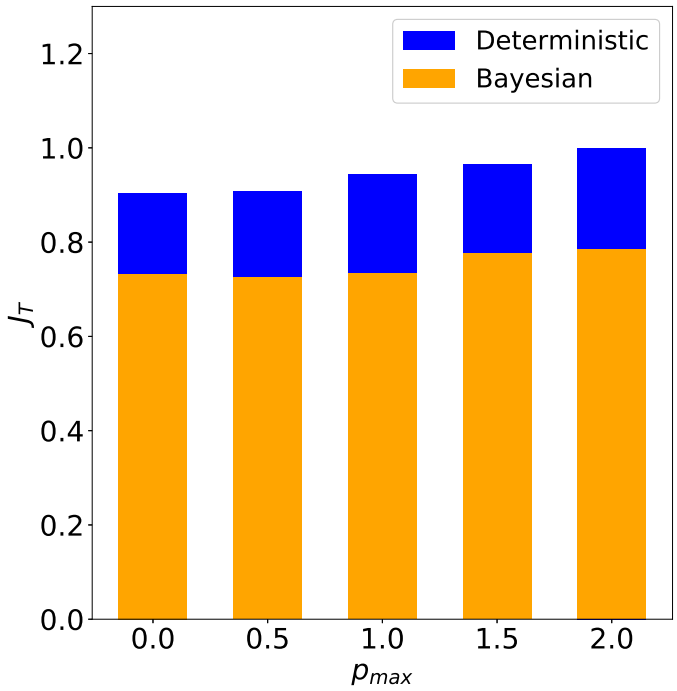


Figure 4.9: Comparison of deterministic and Bayesian control for the rimless wheel.

training incurs a relatively consistent cost across all values of p_{max}. This demonstrates the improved robustness properties of Bayesian NeuralPBC under uneven terrain.

Real-World Experiments

We test the performance of both the deterministic and Bayesian controllers on the hardware shown in Figures 4.10 and 4.11. 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 [48] 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.

Figures 4.12 and 4.13 show the horizontal hip speed achieved by the deterministic and Bayesian controllers, respectively, on the hardware. The real-world experiments exhibit very similar behavior to the simulation results shown in Figure 4.7. Both controllers are able to reach and maintain the desired hip speed of the walking machine. Akin to the simulation results, both controllers periodically apply large torque to react to the energy loss due to impacts. Compared to the deterministic controller, the Bayesian controller exhibits spikes in hip speed. This is due to the marginalizing scheme, which randomly samples parameters that result in a high torque in order to overcome large changes in elevation. This behavior is very beneficial in preventing the mechanism from stumbling due to impacts on uneven terrain.

Figure 4.10: Rimless Wheel Assembly

Figure 4.11: Rimless Wheel Hardware

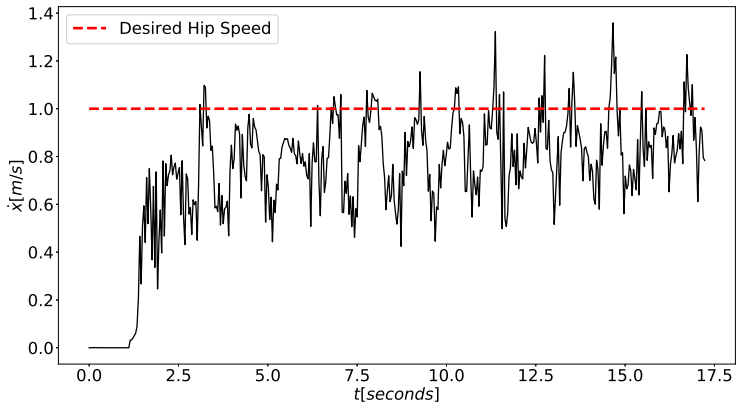


Figure 4.12: Trajectory generated by deterministic controller

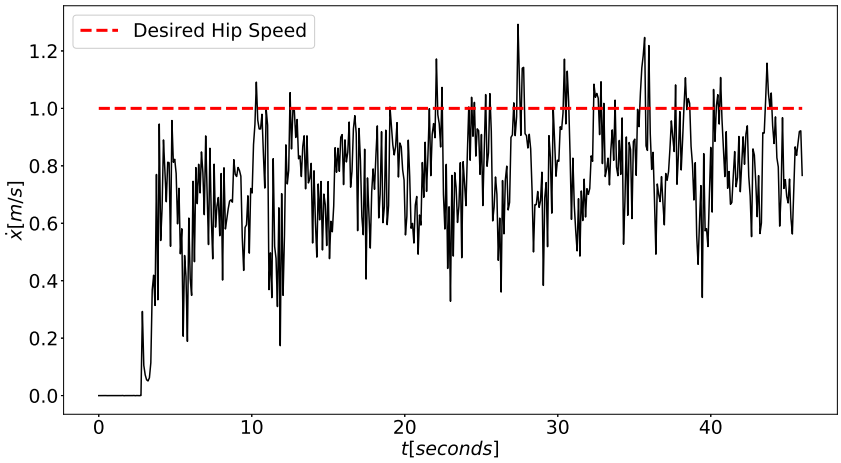


Figure 4.13: Trajectory generated by Bayesian controller

4.2 Bayesian Neural Interconnection and Damping Assignment PBC

In this section, we validate the Bayesian NEURAL-IDAPBC framework on the problem of swinging-up and stabilizing the inverted position of an inertia wheel pendulum (IWP). We

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

4.2.1 Inertia Wheel Pendulum

The objective of this case study is to learn the parameters of V_d, M_d, J_2 that render the closed-loop passive and thus stable at the desired equilibrium x^*, which corresponds to the upward position. We utilize the system model provided in Section 4.1.2 to learn the decision parameter θ from trajectories. The optimization problem (3.13) is constructed as follows.

Training

The potential energy function V_d^θ is a fully-connected neural network with two hidden layers, each of which has the activation function Elu. The closed-loop mass matrix is constructed according to the Cholesky decomposition $M_d^\theta = L_\theta^\top L_\theta$, where the components of L_θ are part of the parameters θ to be optimized. We choose $J_2^\theta = 0$, as the mass matrix is independent of q for this system. The parameters of the surrogates are initialized according to the Glorot (Xavier) [49] 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 4.2 are used for $H(q, p)$ during training. In the Bayesian setting, the standard deviations σ_p of system parameters $p_s = [I_1, I_2, mgl]$ are chosen to be 10% of the nominal system parameters given in Section 4.1.2. 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 4.3.

Table 4.3: 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 4.14 shows the performance of the controllers. The policy learned from the Bayesian training is marginalized over 10 parameters sampled from the posterior per (3.19).

As seen in Figure 4.14, 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

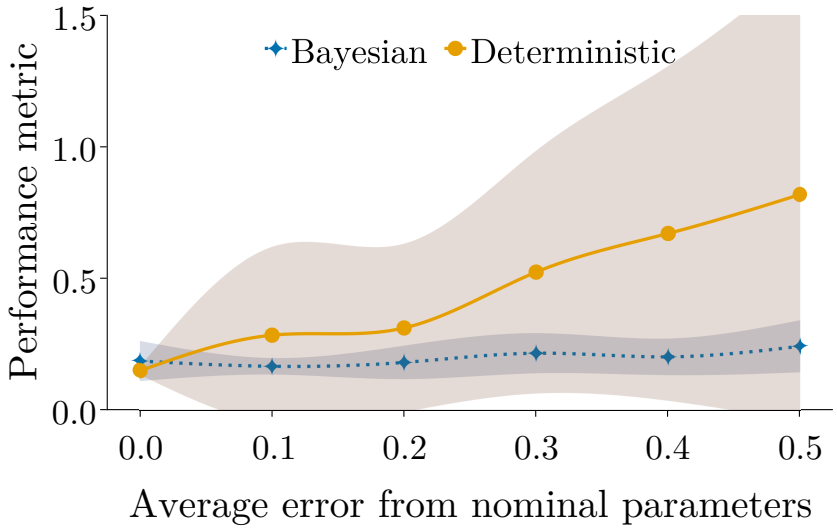


Figure 4.14: Accumulated quadratic cost (J^T) for a range of error in system parameters. Lower values correspond to better controller performance.

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 4.2). The state x is recorded, and the performance metric (4.4) is used to evaluate the controllers. The results are summarized in Figure 4.15.

In all scenarios, we recorded a 100% success rate in the swing-up task despite the errors introduced in the system parameters. Furthermore, we observe that the controller from Bayesian training consistently outperforms the deterministic counterpart, supporting the theoretical justification discussed in Section 3.2.

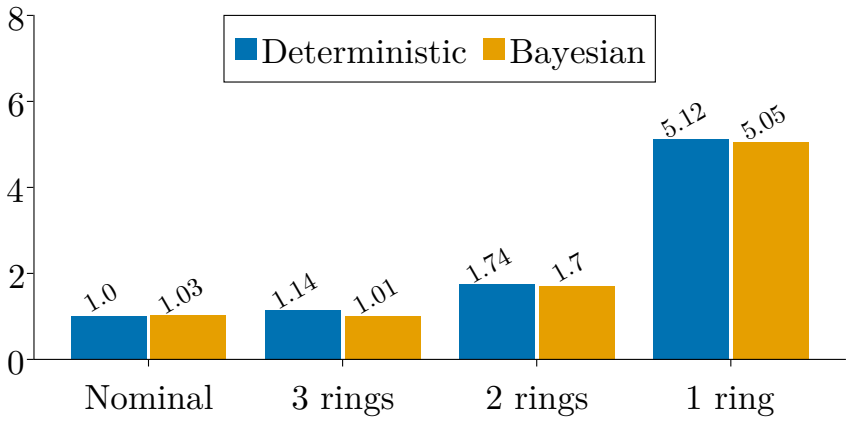


Figure 4.15: Normalized accumulated cost J_T (lower is better) for modified system parameters. The categories A-C correspond to the parameters shown in Table 4.2.

4.3 Conclusion

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

CHAPTER 5:

CONCLUSIONS AND FUTURE WORKS

We tackle two main difficulties in the control of contact-rich robotic systems. First, we address the computational complexities in designing multi-modal controllers for hybrid dynamical systems. In our framework, we use data-driven techniques to infer a mixture of expert controller that switches between several policies. The learning architecture also provides a gating network, which governs the control switching scheme based on the observed states. We use the linear complementarity formulation to accurately model contact-rich systems and to train contact-aware MoE controller in simulation. This data-driven technique finds controllers that react to the positive or negative effects of contacts and impacts. We demonstrate this behavior on the swing-up problem of the cartpole system. We modify the standard cartpole problem to a multi-modal contact-rich mechanism by introducing wall barriers on each sides of the system. Using the MoE architecture, we successfully swing up the cartpole in simulation and hardware. The case study shows that the contact-aware controller finds a way to leverage the impact from the wall barriers to catch the pendulum post-impact.

The second complication in the control of contact-rich systems involves designing robust controllers under uncertain environmental conditions. In particular, we raise the issue of operating a walking machine on uneven terrain, where the exact elevation of the runway is unknown. We characterize and tackle the effects of model uncertainties in the control

of dynamical system. We leverage the robustness properties of Bayesian learning to infer stochastic controllers that can achieve the desired performance under system parameter and measurement uncertainties. This technique is first demonstrated on smooth dynamical systems, such as the simple pendulum and the inertia wheel pendulum, in simulation and hardware. Then, we extend the Bayesian framework to contact-rich systems, such as the rimless wheel walking machine, and demonstrate the robustness properties of the Bayesian technique against a deterministic framework.

The control design techniques presented in this work have laid the foundation for numerous future research directions. The MoE architecture can be applied to address many interesting challenges in the control of contact-rich robots. For instance, we can utilize the MoE framework to select an optimal way to manipulate and maneuver an object in space. This framework can be used to select one of many object manipulation techniques, such as pushing, lifting or sliding an object, based on the obstacles present in the environment. We also intend to combine the multi-modal nature of the MoE framework with the robustness properties of Bayesian inference. This is most useful when the robotic system interacts with a dynamic environment. An example of such scenario can be found in the cartpole system with wall barriers. If these wall barriers are not static, the controller needs to reason about the uncertainty in the walls' positions, and learn a controller robust against the unexpected impacts from these barriers.

BIBLIOGRAPHY

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

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

[3] C. Kahraman, M. Deveci, E. Boltürk, and S. Türk, "Fuzzy controlled humanoid robots: A literature review," *Robotics and Autonomous Systems*, vol. 134, p. 103643, 2020.

[4] D. Katić and M. Vukobratović, "Survey of intelligent control techniques for humanoid robots," *Journal of Intelligent and Robotic Systems*, vol. 37, pp. 117–141, 2003.

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

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

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

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

[9] N. A. Ashenafi, W. Sirichotiyakul, and A. C. Satici, "Robust passivity-based control of underactuated systems via neural approximators and bayesian inference," *IEEE Control Systems Letters*, vol. 6, pp. 3457–3462, 2022.

[10] W. Sirichotiyakul and A. C. Satici, "Data-driven passivity-based control of underactuated mechanical systems via interconnection and damping assignment," *International Journal of Control*, pp. 1–9, 2022.

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

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

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

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

[15] S. Linderman, M. Johnson, A. Miller, R. Adams, D. Blei, and L. Paninski, "Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems,"

in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 914–922. [Online]. Available: https://proceedings.mlr.press/v54/linderman17a.html

[16] D. D. Fan, J. Nguyen, R. Thakker, N. Alatur, A.-a. Agha-mohammadi, and E. A. Theodorou, "Bayesian learning-based adaptive control for safety critical systems," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 4093–4099.

[17] F. E. Cellier, "Combined continuous/discrete simulation: applications, techniques and tools," in *Proceedings of the 18th conference on Winter simulation*, 1986, pp. 24–33.

[18] 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.

[19] J. J. Moreau, "Unilateral contact and dry friction in finite freedom dynamics," *Nonsmooth mechanics and Applications*, pp. 1–82, 1988.

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

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

[22] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.

[23] Z. Chen, Y. Deng, Y. Wu, Q. Gu, and Y. Li, "Towards understanding mixture of experts in deep learning," *arXiv preprint arXiv:2208.02813*, 2022.

[24] M. M. Zhang and S. A. Williamson, "Embarrassingly parallel inference for gaussian processes," *Journal of Machine Learning Research*, 2019.

[25] 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.

[26] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.

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

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

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

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

[31] 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.

[32] 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 (ACC)*. IEEE, 2022, pp. 3266–3272.

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

[34] H. K. Khalil, *Nonlinear control*. Pearson New York, 2015, vol. 406.

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

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

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

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

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

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

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

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

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

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

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

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

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

[48] 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.

[49] 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 (3.26) of the SDE (3.25) expression into the performance measure (3.22) yields

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

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

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

$\square$