ROBUST CONTROL OF CONTACT-RICH ROBOTS VIA NEURAL BAYESIAN INFERENCE

by

Nardos Ayele Ashenafi

A dissertation

submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

Boise State University

May 2023

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1:

BACKGROUND

1.1 Contact Modeling with Linear Complementarity Problem

1.2 Passivity-Based Control (PBC)

Let $x \in \mathcal{X} \subset \mathbb{R}^{2n}$ denote the state of the robot. The state x is represented in terms of the generalized positions and momenta $x = (q, p)$. With M denoting the symmetric, positive-definite mass matrix, the Hamiltonian H of the robot is expressed as

$$H(q,p) = \frac{1}{2} p^\top M^{-1}(q) p + V(q), \tag{1.1}$$

where $V(q)$ represents the potential energy. The system's equations of motion can then be expressed as

$$\begin{aligned} \begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} &= \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix} \begin{bmatrix} \nabla_q H \\ \nabla_p H \end{bmatrix} + \begin{bmatrix} 0 \\ G(q) \end{bmatrix} u, \\ y &= G^\top \dot{q}, \end{aligned} \tag{1.2}$$

where $G(q) \in \mathbb{R}^{n\times m}$ is the input matrix, I_n denotes the $n \times n$ identity matrix, and $u \in \mathbb{R}^m$ is the control input. The system (1.2) is *underactuated* if rank $G = m < n$.

The main idea of passivity-based control (PBC) [1] is to design the input u with the objective of imposing a desired storage function $H_d : \mathcal{X} \to \mathbb{R}$ on the closed-loop system, rendering it passive and consequently stable. In the standard formulation of PBC, the control comprises an energy shaping term and a damping injection term:

$$u = u_{es}(x) + u_{di}(x). \tag{1.3}$$

For mechanical systems, one solution to the PBC problem is of the form

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

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

$$G^{\perp}\left(\nabla_q H_d - \nabla_q H\right) = 0, \tag{1.4}$$

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

1.2.1 Neural PBC

CONTINUITY

However, the closed-form solution to the partial differential equations (PDEs) in (1.4) is intractable. The deterministic NeuralPbc framework presented in [2] solves the PDEs

given in (1.4) by rewriting the PBC problem as an optimization problem of the form

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \int_0^T \ell\left(\phi, u^\theta(\phi)\right)\, \mathrm{d}t, \\ \text{subject to} \quad & \dot{x} = \begin{bmatrix} \nabla_p H \\ -\nabla_q H \end{bmatrix} + \begin{bmatrix} 0 \\ G(q) \end{bmatrix} u^\theta, \\ & u^\theta = -G^\dagger \nabla_q H_d^\theta - K_v^\theta G^\top \nabla_p H_d^\theta, \end{aligned} \tag{1.5}$$

where $T > 0$ is the time horizon, $\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is a running cost function to be defined, and $\phi(t; x_0, u^\theta)$ is the flow of the dynamical system. The NeuralPbc technique adds three important features to the classical PBC framework.

1. The optimization problem finds an approximate solution to the PDEs in (1.4).

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

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

1.2.2 Neural Interconnection and Damping Assignment PBC

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

$$H_d(q,p) = \frac{1}{2} p^\top M_d^{-1}(q) p + V_d(q), \tag{1.6}$$

such that the control input must satisfy the PDEs given by

$$G^\perp \left\{ \nabla_q H - M_d M^{-1} \nabla_q H_d + J_2 M_d^{-1} p \right\} = 0. \tag{1.7}$$

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

$$\begin{aligned} u_{es} &= G^{\dagger}\left(\nabla_q H - M_d M^{-1}\nabla_q H_d + J_2 M_d^{-1} p\right), \\ u_{di} &= -K_v G^{\top}\nabla_p H_d, \end{aligned} \tag{1.8}$$

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

The closed-form solution to the PDEs in (1.7) is intractable. Hence, the deterministic NEURAL-IDAPBC framework introduced in [3] formulates the following optimization problem that finds an approximate solution to the PDEs.

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \|l_{\text{IDA}}(x)\|^2 = \left\|G^{\perp}\left\{\nabla_q H - M_d M^{-1}\nabla_q H_d + J_2 M_d^{-1} p\right\}\right\|^2, \\ \text{subject to} \quad & M_d^{\theta} = \left(M_d^{\theta}\right)^{\top} \succ 0, \\ & J_2^{\theta} = -\left(J_2^{\theta}\right)^{\top}, \\ & q^{\star} = \underset{q}{\text{argmin}}\ V_d^{\theta}. \end{aligned} \tag{1.9}$$

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

1.3 Bayesian Learning

The objective of Bayesian learning is to determine a stochastic model (target function) that best fits observed data $\mathcal{D}$ with inherent noise. Let this stochastic target function be represented by $F(x;\theta) : \mathcal{X} \to \mathbb{R}^t$, where $\theta \in \Theta \subset \mathcal{R}^v$ is a multivariate random variable that parameterizes the model. Given prior belief on the distribution of the parameters $p(\theta)$, Bayesian learning finds a posterior distribution $p(\theta \mid \mathcal{D})$ over θ that maximizes the likelihood of the target function generating the dataset $\mathcal{D}$ [4]. This can be expressed in terms of Bayes'

theorem as

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{\int_\theta p(\mathcal{D} \mid \theta')p(\theta')d\theta'}, \tag{1.10}$$

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

Posterior Distribution

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

1. Markov Chain Monte Carlo (MCMC) methods: learn the exact posterior distribution by collecting samples of θ either through random walk (e.g. Metropolis-Hastings) or following the gradient of the likelihood (e.g. Hamiltonian Monte Carlo). Metropolis-Hastings methods collect samples of θ from a conditional probability distribution until the samples converge to an equilibrium distribution per the properties of irreducible and aperiodic Markov chains [5]. Hamiltonian Monte Carlo (HMC) method, shown in detail in Algorithm 1, also finds the equilibrium distribution of the Markov chain, but unlike Metropolis-Hastings, it efficiently searches the parameter space through the gradient of the likelihood. In the case of HMC, the Markov chain is generated from two first-order differential equations shown in lines 5-7 of Algorithm 1. While HMC method learns the exact posterior distribution, it has slow convergence properties for high-dimensional parameters. In such cases, techniques such as variational inference

Algorithm 1 Hamiltonian Monte Carlo

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

compromise accuracy of the posterior distribution for speed of convergence.

2. Variational Inference (VI): this technique selects a posterior distribution $q(\theta; z)$ from the conjugate families of the likelihood and prior distributions. The goal is to learn the distribution parameters z that minimize the Kullback-Leibler divergence or equivalently maximize the evidence lower bound (Elbo). The Elbo, $\mathcal{L}$, is given by [6]

$$\begin{aligned} \mathcal{L}(\mathcal{D}, z) &= \mathbb{E}_{\theta \sim q} \left[\log(p(\theta, \mathcal{D}; z)) - \log(q(\theta; z)) \right], \\ p(\theta, \mathcal{D}; z) &= p(\mathcal{D} \mid \theta; z) p(\theta), \end{aligned} \tag{1.11}$$

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

Remark 1. *For continuous posterior distribution, the* ELBO *given in equation* (1.11) *is redefined using differential entropy, which expresses the prior and posterior in terms of their probability density functions. In this case, the likelihood* $p(\theta \mid \mathcal{D}; z)$ *is also a probability density function and the* ELBO *is not bounded by zero.*

The power of Bayesian learning lies in its ability to build a target function and make predictions that integrate over uncertainties [7]. These predictions can be found by marginalizing the model over the posterior as follows [8].

$$\hat{F}(x) = \frac{1}{N} \sum_{\theta \sim q} F(x, \theta), \tag{1.12}$$

where N is the number of samples drawn from the posterior. Moreover, Bayesian frameworks can quantify the confidence in the predictions through the variance of the predictive distribution, $p(F \mid x, \mathcal{D})$. The variance of $p(F|x, \mathcal{D})$ is given by [8]

$$\Sigma_{F|x,\mathcal{D}} = \frac{1}{N-1} \sum_{\theta \sim q} \left|\left| F(x, \theta) - \frac{1}{N} \sum_{\theta \sim q} F(x, \theta) \right|\right|^2. \tag{1.13}$$

CHAPTER 2:

SWITCHING CONTROL WITH DEEP-NET MIXTURE OF EXPERTS

2.1 Introduction

2.2 Learning Deep-Net Mixture of Expert Controllers

2.3 Experimental Results

2.3.1 Stable Switching Between Unstable Systems

2.3.2 Cartpole with Wall Contacts

2.4 Conclusion

CHAPTER 3:

UNCERTAINTY HANDLING VIA NEURAL BAYESIAN INFERENCE

3.1 Introduction

3.2 Theoretical Justification of Robustness

In this section, we demonstrate the improved robustness properties of Bayesian learning over point-estimates of a policy. This theoretical justification is given by a toy example, where closed-form calculation of the point-estimates and posterior distributions for the optimal controller is provided.

3.2.1 Optimal Control under Parameter Uncertainty

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

$$\begin{cases} \dot{x} = px + u, \ x(0) = x_0 \\ u(x) = \theta x. \end{cases} \tag{3.1}$$

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

$\theta \in \mathbb{R}$ to be determined through optimization. Without loss of generality, we will take the initial condition $x_0 = 1$. The performance index to be optimized for determining the best control parameter θ is

$$\mathcal{J} = \int_0^T \left(\frac{1}{2} q x^2 + \frac{1}{2} r u^2 \right) dt, \tag{3.2}$$

where T is the control horizon and $q \geq 0$ and $r > 0$ are design parameters. We solve the control system (3.1) to find $x(t) = e^{(p+\theta)t}$ and plug this into the performance index (3.2) along with the form selected for the controller. Performing the integration over time and letting $T \to \infty$, assuming that $p + \theta < 0$ then yields the infinite-horizon optimal cost functional

$$\mathcal{J}_\infty = -\frac{1}{4} \frac{q + r\theta^2}{p + \theta}. \tag{3.3}$$

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

$$\begin{aligned} \nabla_\theta \mathcal{J}_\infty &= -\frac{r}{4} \frac{(p+\theta)^2 - (p^2 + {}^q\!/_r)}{(p+\theta)^2} = 0, \\ \therefore \theta^\star &= g(p) := -p - \sqrt{p^2 + {}^q\!/_r}, \\ g^{-1}(\theta) &= \frac{q}{2r\theta} - \frac{\theta}{2}. \end{aligned} \tag{3.4}$$

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

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

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

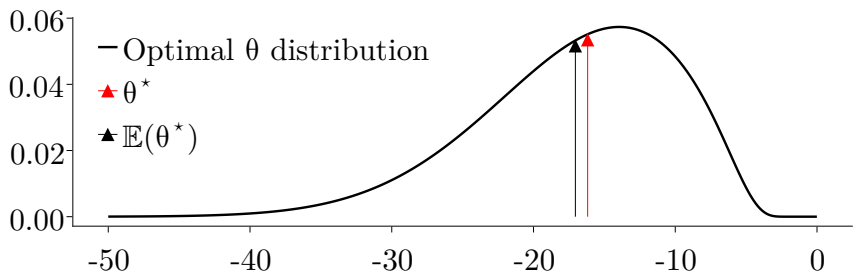

Figure 3.1: The optimal control parameter distribution given that the system parameter p is normally distributed with mean $\hat{p} = 5$ and $\sigma_p = 5$. The red and black arrows respectively indicate the optimal control parameter without considering the randomness of p, and the expected value of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

We can further eliminate the control parameter from the expression for the optimal cost function $\mathcal{J}_\infty$ by substituting for θ from equation (3.4), yielding

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

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

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

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

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

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

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

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

3.2.2 Optimal Control under Parameter Uncertainty and Measurement Noise

Consider the scenario in which the system (3.1) is also subject to measurement errors; that is, our measurement model for the state x is probabilistic and is distributed according to the Gaussian $\mathcal{N}(x, \sigma^2)$. Since the controller uses this measurement to determine its action,

the closed-loop system has to be modelled as a stochastic differential equation (SDE), given by

$$\begin{cases} dx(t) = (p+\theta)x(t)\ dt + \theta\sigma\ dW_t, \\ x(0) = 1, \end{cases} \tag{3.5}$$

where W denotes the Wiener process [9]. The initial state is assumed deterministic and is set to unity for simplicity. The unique solution to this SDE is given by

$$x(t) = e^{(p+\theta)t} + \theta\sigma \int_0^t e^{(p+\theta)(t-s)} dW_s. \tag{3.6}$$

Lemma 1. *The conditional expectation* $\mathbb{E}[\mathcal{J} \mid p]$ *of the performance index* (3.2) *given the system parameter* p *is*

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

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

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

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

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

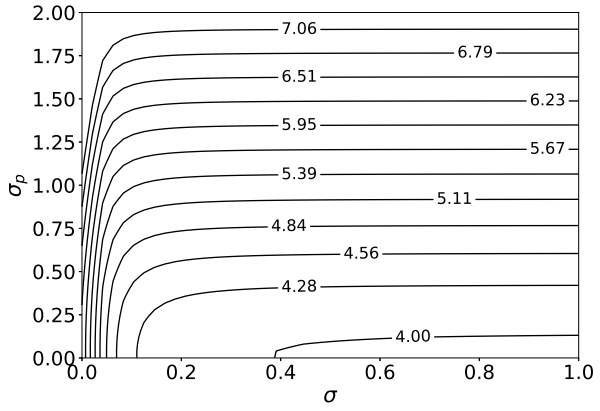

Figure 3.2: The optimal controller parameter magnitude $|\theta^\star|$.

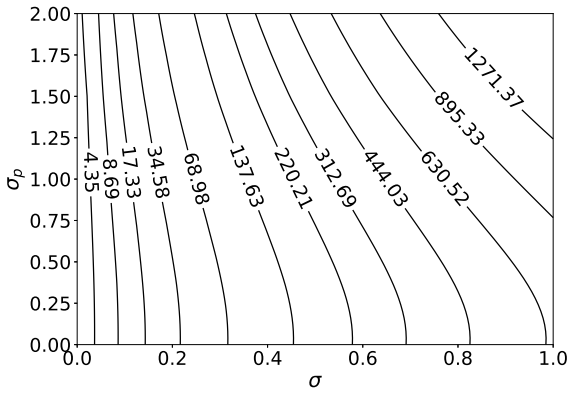

Figure 3.3: The minimal expected cost $\mathbb{E}[\mathcal{J}]$.

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

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

3.3.1 Simple Pendulum

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

$$\mathrm{d}x = \begin{bmatrix} \dot{\vartheta} \\ a \sin \vartheta + u^{\theta}(x) \end{bmatrix} \mathrm{d}t + \nabla_x u^{\theta}(x)\, \mathrm{d}W_t, \tag{3.7}$$

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

Training

Algorithm **??** is carried out with expert trajectories generated by the vanilla energy-shaping control (ESC) [10]. The ESC takes the general form

$$u^{\theta}(\vartheta, \dot{\vartheta}; \theta) = \theta_1 \dot{\vartheta} + \theta_2 \cos(\vartheta) \dot{\vartheta} + \theta_3 \dot{\vartheta}^3, \tag{3.8}$$

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

The running cost function is chosen to be the loss $J_{track}(\gamma_\perp)$ from homoclinic orbit $\gamma^\star$ provided in (**??**); the corresponding likelihood is given by (**??**). We collect dataset $\mathcal{D}$ of initial states sampled from the expert trajectory, per the state sampling technique discussed in Section **??**. For this particular experiment, we use Hamiltonian Monte Carlo outlined in Algorithm 1 to infer the exact posterior distribution $p(\theta|\mathcal{D})$. We compare the Bayesian policy inferred from the procedure in Algorithm **??** with the deterministic policy, which is simply given by the point-estimates of the expert vanilla ESC in (3.8) [11].

Simulation Tests

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

We demonstrate the robustness of the Bayesian policy by comparing ζ, as a function of the system parameter a, with that of the deterministic policy. We also investigate the effects of the prior distribution of θ on the performance of the controllers. In particular, we examine a uniform prior and a Gaussian prior centered around the deterministic solution of (3.8). The comparisons between the controllers from both cases are shown in Figure 3.4.

In addition to uncertainties in a, we test the deterministic and Bayesian policies with measurement noise, modelled as a Wiener process with variance 0.0005 rad in the ϑ- and 0.05 ${}^{\text{rad}}\!/_{\text{s}}$ in the $\dot{\vartheta}$-directions. These numbers are chosen to represent a typical error arising from an optical encoder with a resolution of 2048 pulses per revolution and its naive differentiation

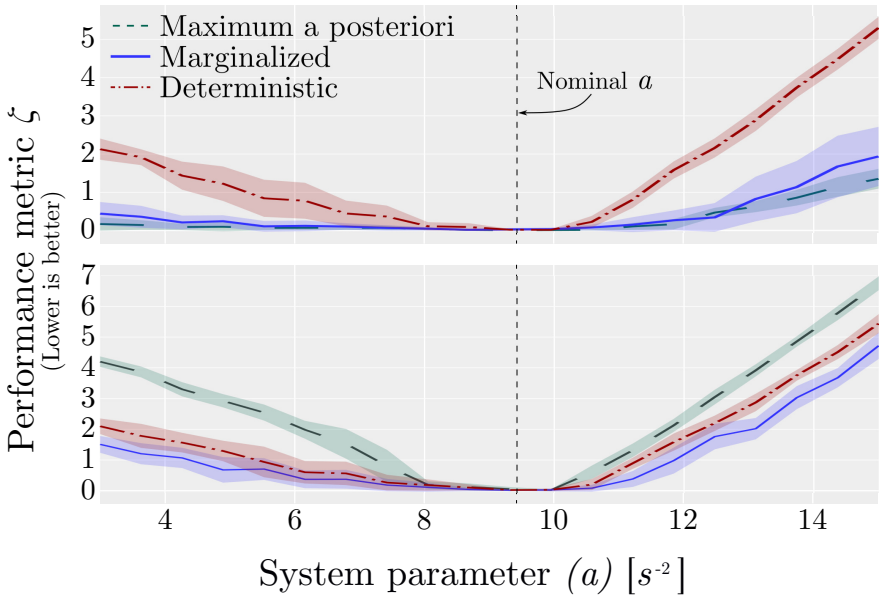

Figure 3.4: Performance comparisons between deterministic and Bayesian learning methods. The training is initialized with a Gaussian prior (top), and a uniform prior (bottom). The continuous error band is generated by computing ζ from 20 trajectories of (3.7), starting at the downward equilibrium with a small disturbance. The solid lines represent the mean of ζ. Best viewed in color.

via backward difference. To capture the influence of measurement noise, we generate 20 trajectories by integrating (3.7) from the same initial states. These trajectories are then used to compute ζ. The effects of noise on ζ are reflected by the error bands in Figure 3.4.

The results show that Bayesian learning yields controllers that outperform their deterministic counterpart throughout the whole range of a. The marginalization method in (1.12) for selecting θ from the learned distribution performs best when a uniform prior is used, while the MAP method performs best with the Gaussian prior. We emphasize that the error bands of the marginalized and MAP point estimates stay well below those of the deterministic curve in the top plot of Figure 3.4. 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.

3.3.2 Inertia Wheel Pendulum

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

The Hamiltonian of the IWP is given by Equation (1.1) with $n = 2$ and

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

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

$$\mathrm{d}x = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dfrac{mgl\sin(q_1) - u^\theta - b_1\dot{q}_1}{I_1} \\ \dfrac{u^\theta - b_2\dot{q}_2}{I_2} \end{bmatrix} \mathrm{d}t + \nabla_x u^\theta(x)\,\mathrm{d}W_t, \tag{3.9}$$

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.

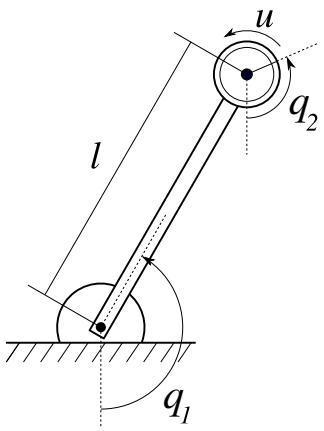

Figure 3.5: Schematic of the inertia wheel pendulum. Only the joint q_2 is actuated, and q_1 is not.

Training

The energy-like function H_d^θ is a fully-connected neural network with two hidden layers, each with the ELU activation function [12]. A uniform distribution in $[-2\pi, 2\pi] \times [-2\pi, 2\pi] \times [-10, 10] \times [-10, 10]$ is chosen as the probability distribution from which samples of initial states x_0 are drawn for the DAGGER strategy. In each gradient descent step, we sample a batch of 4 initial states $\{x_0\}$ from $x_0 \sim \mathcal{N}(x^\star, \Sigma_0)$ and DAGGER as discussed in Section **??**; these initial states are integrated forward with a time horizon of $t \in [0, 3]$ seconds. In the Bayesian framework, the standard deviations σ_ζ of system parameters $\zeta = [I_1, I_2, mgl]$ are chosen to be 10% of the nominal system parameters given in Section **??**. Moreover, we train on trajectories per the SDE in (**??**) with measurement error represented by Wiener process with standard deviation of 0.001 and 0.02 on the joint angles and velocities, respectively.

We use variational inference to estimate a Gaussian posterior distribution over uncorrelated parameters. The trainings are terminated when the loss function $J(\gamma) = J_{set}(\gamma) + J_T(\gamma)$ and the ELBO converge for the deterministic and Bayesian trainings, respectively. The hyperparameters for the deterministic and Bayesian NEURALPBC trainings are shown in Table 3.1.

Table 3.1: NeuralPBC training setup for deterministic and Bayesian frameworks

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

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

$$J^T = \frac{1}{2} \int_0^T \left(x^\top Q x + u^\top R u \right) dt. \tag{3.10}$$

The controller learned from the Bayesian training is marginalized over 10 parameters sampled from the posterior. As seen in Figure 3.6, the Bayesian training effectively collects less cost

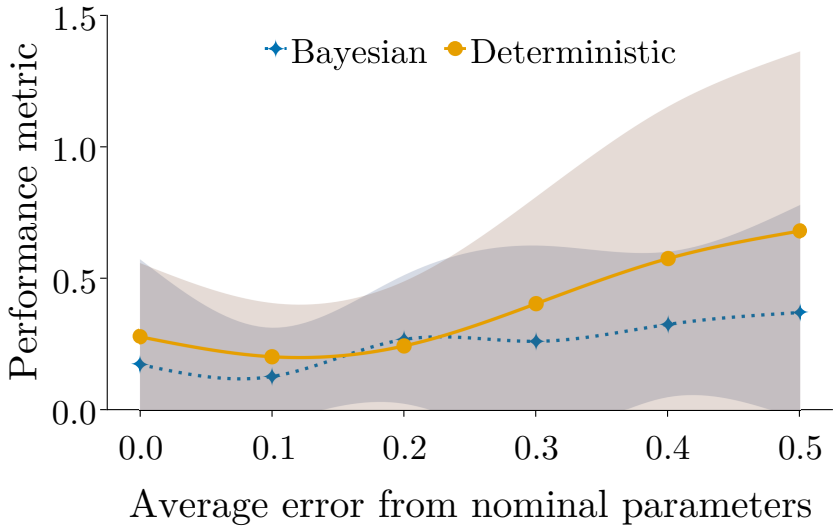

Figure 3.6: NeuralPBC Performance metric (J^T) for various error in system parameters. Measurement noise included as Wiener process with standard deviation of 0.001 **and** 0.02 **on joint angles and velocities, respectively**

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 hardware. We deliberately modify the hardware and test the controllers without any additional training. In particular, throughout the experiments, the inertia wheel attached to q_2 is replaced with parts (labelled A-C on Table 3.2) whose mass and inertia are different from the nominal values. The modified system parameters are summarized in Table 3.2.

The system starts from rest at the downward position. A small disturbance in the q_1 direction is introduced to start the swing-up. The state x is recorded and (3.10) is the performance metric used to evaluate the controllers. The results are summarized in

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

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

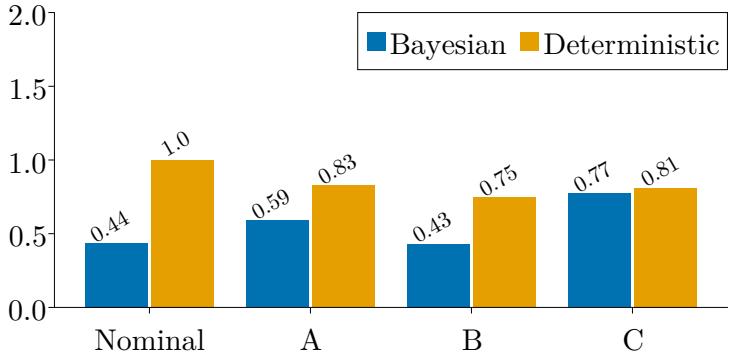

Figure 3.7: Controller performance for modified system parameters. The performance metric is given by Eq. (3.10). Lower values are better. These results show that controllers trained via Bayesian learning are consistently more robust to errors in system parameters.

Figure 3.7.

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

3.3.3 Rimless Wheel

3.4 Bayesian Neural Interconnection and Damping Assignment PBC

In this subsection, we formulate a Bayesian learning framework that tackles the adverse effects of system parameter uncertainties. We parametrize the function V_d^θ and the entries of L_θ and A_θ matrices with Bayesian neural networks. The goal is to learn the distribution parameters z of the posterior multivariate probability distribution $q(\theta; z)$ that maximize the ELBO given in (1.11).

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

$$p(\|l_{1k,\mathrm{IDA}}(q)\|^2 + \|l_{2,\mathrm{IDA}}(q)\|^2 \mid \theta) = \mathcal{N}(0, s)\,, \tag{3.11}$$

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 (3.11), maximizing the ELBO in (1.11) coaxes the loss $l_{\mathrm{IDA}}(x)$ to zero.

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

distribution is centered around the parameters learned from the deterministic technique discussed in Section **??**.

We update the distribution parameters z along the gradient $\partial\mathcal{L}/\partial z$ until the ELBO converges and the objective function of (**??**) reaches the threshold ϵ_{tol}. We invoke the reparameterization trick of the Automatic Differentiation Variational Inference(ADVI) [13] to compute the gradient of samples θ with respect to the distribution parameters z.

System parameter uncertainties can deteriorate the performance of controllers employed on real systems. Hence, in the Bayesian framework, we inject these uncertainties directly into the training loop in order to learn a controller that works for a wide range of system parameters. To model these uncertainties, we sample a set of system parameters ζ from a normal distribution $\mathcal{N}(\zeta_0, \sigma_\zeta)$ centered around the nominal parameter ζ_0, where σ_ζ represents the uncertainty in system parameters. Each time we compute the PDE loss l_{IDA} for a batch of discrete states sampled from the configuration space $\mathcal{Q}$, we draw a new sample of ζ.

3.4.1 Inertia Wheel Pendulum

Training

The optimization problem (1.9) is constructed as follows. The potential energy function V_d^θ is a fully-connected neural network with two hidden layers, each of which has the activation function ELU. The closed-loop mass matrix is constructed according to the Cholesky decomposition $M_d^\theta = L_\theta^\top L_\theta$, where the components of L_θ are part of the parameters θ to be optimized. We choose $J_2^\theta = 0$, as the mass matrix is independent of q for this system. The parameters of the surrogates are initialized according to the Glorot (Xavier) [14] scheme. The optimization problem is solved over a uniform discretization of $q = (q_1, q_2) \in [-2\pi, 2\pi] \times [-50, 50]$.

In the deterministic setting, the nominal system parameters reported in Table 3.2 are

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

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

used for $H(q,p)$ during training. In the Bayesian setting, the standard deviations σ_ζ of system parameters $\zeta = [I_1, I_2, mgl]$ are chosen to be 10% of the nominal system parameters given in Section **??**. 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 (1.8). A summary of the hyperparameters for both the deterministic and Bayesian methods are given in Table 3.3.

Simulation Tests

The performance of the controllers obtained from the deterministic and Bayesian trainings are compared as follows. Similar to the NEURALPBC simulation tests, we introduce system parameter uncertainties by moving the average system parameters by $\pm 10\%$ to $\pm 50\%$ with increments of 10%. For each average system parameter, we sample uniformly with a $\pm 5\%$ support around the average system parameters. This helps test the performance of the controller with various combinations of I_1, I_2 and m_3. Figure 3.8 shows the performance of the controllers. The policy learned from the Bayesian training is marginalized over 10 parameters sampled from the posterior per (1.12).

As seen in Figure 3.8, trajectories from the Bayesian controller incur much lower cost than the deterministic counterpart throughout a wide range of errors in system parameters.

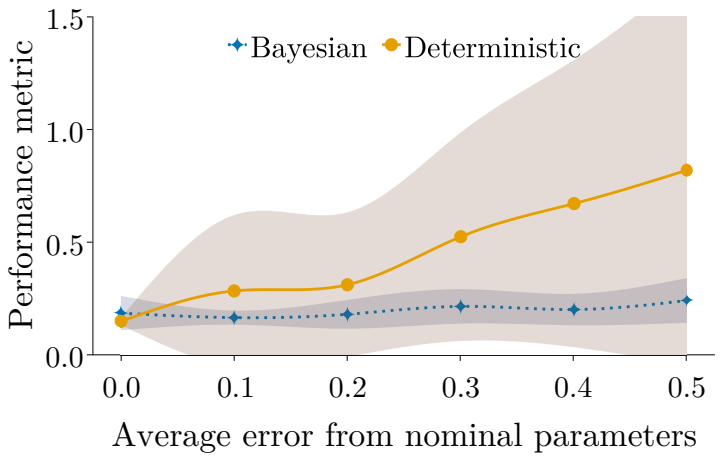

Figure 3.8: Accumulated quadratic cost (J^T) for a range of error in system parameters. Lower values correspond to better controller performance.

Moreover, we observe that the error band on the cost corresponding to Bayesian training is narrower. These results show that controllers trained via Bayesian learning are consistently more robust to errors in system parameters.

Hardware Tests

The hardware experiments are designed to further demonstrate the robustness of our controllers against model uncertainties, which include errors in the parameters, friction in the bearings, and any contribution to the dynamics from the belt-drive system. We deliberately modify the hardware to create large errors in the model parameters and test the controllers without any additional training. In particular, the inertia wheel attached to q_2 is replaced with parts whose mass and inertia values differ from the nominal values (see Table 3.2). The state x is recorded, and the performance metric (3.10) is used to evaluate the controllers. The results are summarized in Figure 3.9.

In all scenarios, we recorded a 100% success rate in the swing-up task despite the

errors introduced in the system parameters. Furthermore, we observe that the controller from Bayesian training consistently outperforms the deterministic counterpart, supporting the theoretical justification discussed in Section **??**.

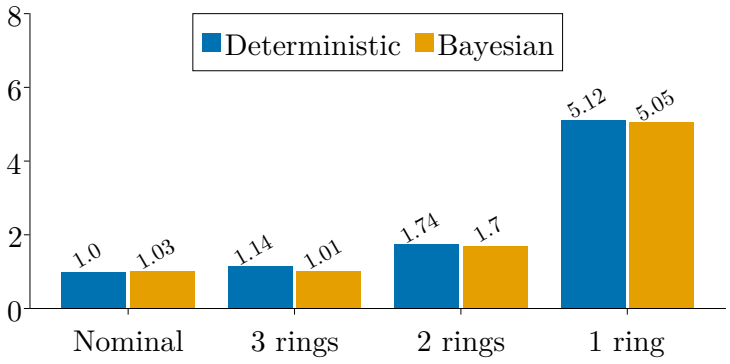

Figure 3.9: Normalized accumulated cost J_T (lower is better) for modified system parameters. The categories A-C correspond to the parameters shown in Table 3.2.

3.5 Conclusion

BIBLIOGRAPHY

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

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

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

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

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

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

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

[8] 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.

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

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

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

[12] 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.

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

[14] 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 (3.6) of the SDE (3.5) expression into the performance measure (3.2) yields

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

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

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

□